

TC1796

32-Bit Single-Chip Microcontroller

Volume 1 (of 2): System Units

Volume 2 (of 2): Peripheral Units

32bit

Microcontrollers



Never stop thinking

Edition 2007-07

**Published by
Infineon Technologies AG
81726 Munich, Germany**

**© 2007 Infineon Technologies AG
All Rights Reserved.**

Legal Disclaimer

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics. With respect to any examples or hints given herein, any typical values stated herein and/or any information regarding the application of the device, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation, warranties of non-infringement of intellectual property rights of any third party.

Information

For further information on technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies Office (www.infineon.com).

Warnings

Due to technical requirements components may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies Office.

Infineon Technologies Components may only be used in life-support devices or systems with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system, or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body, or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

TC1796

32-Bit Single-Chip Microcontroller

Volume 1 (of 2): System Units

Volume 2 (of 2): Peripheral Units

Microcontrollers



Never stop thinking

TC1796 User's Manual
Volume 1 (of 2) System Units & Volume 2 (of 2) Peripheral Units
Revision History: V2.0, 2007-07

Previous Version: V1.0 2005-06

Page	Subjects (major changes since last revision)
------	--

TC1796 User's Manual Version 2.0 includes the contents of the TC1796 Documentation Addendum, V1.2, Apr. 2007. Furthermore, the complete document and especially the register description have been updated to allow document-automation processes based on XML technologies.

Volume 1: System Units

1-20	MSC baud rate feature item is updated.
1-25	Figure 1-7 is updated.
1-34	Figure 1-11 is updated.
1-37, 1-38	Footnote marking for EBU pins is removed; pad driver class for BFCLKI is added.
1-53	Pad driver classes for several System I/O pins are updated/added.
1-58	Table 1-6 is added.
2-10	CPU_ID is added.
2-12	MMU_CON long register name is updated.
2-13	CPS_ID is added.
2-14	CPU_SRCn.TOS bit description is updated.
2-21	Offset addresses in Table 2-6 are updated by absolute addresses.
2-26, 2-27	PMI_ID is added.
2-34, 2-35	DMI_ID is added.
3-18	Sequence of setting up the PLL after reset is updated.
4-1	Attention paragraph at the bottom is added.
4-10	Tuning Protection Section is removed.
4-24	Figure 4-6 is updated.
5-20, 5-22	Bit description of RENx is updated.
5-49	Figure 5-11 is updated.
5-61, 5-69	SCU_ID is added.
5-68	Bit description of PARAV is updated.
5-59	Bit description of ENON is updated.

TC1796 User's Manual
Volume 1 (of 2) System Units & Volume 2 (of 2) Peripheral Units
Revision History: V2.0, 2007-07

6-4	Figure 6-2 is updated.
6-7, 6-8	DBC _U _ID and PBC _U _ID are added.
6-9	Bit description of LEC is updated.
6-16, 6-17	LFI_ID is added.
6-21	Section 6.4.3 is updated.
6-22	Figure 6-7 is updated.
6-30	BCU breakpoint Trigger Combination Logic in Figure 6-12 is updated.
6-34, 6-36	SBC _U _ID and RBC _U _ID are added.
7-16	DFLASH addresses in Table 7-7 are updated
7-40, 7-41, 7-42	FLASH_ID and PMU_ID are added.
7-43	Bit description of FABUSY is updated.
7-58	Footnote is added to FLASH_CON register.
8-7	Section 8.5.2 is updated.
8-10, 8-11	DMU_ID is added.
9-5, 9-7, 9-9	Short name for SBRAM and SPRAM is updated.
11-52, 11-54	PCP_ID is added.
11-59	PCP_ES bit 5 is updated.
11-74	Counter Reload Value (COPY) in Table 11-13 is updated.
11-78	Figure 11-14 is updated.
11-103	Syntax description of ST.PI is updated.
12-9	Figure 12-5 is updated.
12-10	Figure 12-6 is updated.
12-16	Section 12.1.4.5 is updated.
12-29	Figure 12-20 is updated.
12-30	Figure 12-21 is updated.
12-34	The pattern detection description in Section 12.1.9 is updated.
12-42, 12-43, 12-45	DMA_ID is added.
12-52, 12-53	Bit descriptions of CH0x/CH1x and HTRE0x/HTRE1x are updated.

TC1796 User's Manual
Volume 1 (of 2) System Units & Volume 2 (of 2) Peripheral Units
Revision History: V2.0, 2007-07

12-99	Address range for AEN21 is updated.
12-111, 12-112	MCHK_ID is added.
13-3	Section 13.2.1 is updated.
13-24	CONF32BIT description is updated.
13-72	BFCLKO configurations in Figure 13-27 are updated.
13-75	Burst Flash Read Cycle diagram in Figure 13-29 is added.
13-79, 13-81	EBU_ID is added.
14-13	NMI trap handler in Figure 14.5 is updated.
15-7, 15-8, 15-10	STM_ID is added.
18-16	Access modes in Table 18-7 are updated.
18-122	DBCUC register addresses in Table 18-38 are updated.

Volume 2: Peripheral Units

19-19, 19-20	ASC0_ID and ASC1_ID register are added.
20-21	Note in Figure 20-11 corrected; TB write operation in Section 20.1.2.11 is updated.
20-24, 20-31, 20-33	SSC error interrupt control in Section 20.1.2.12 and bit description STIP, EN are updated.
20-27, 20-29	SSC0_ID and SSC1_ID registers are added.
20-37, 20-39,	Notes below register SSOC and SSOTC are updated.
21-26	Figure 21-18 is updated (sampling start).
21-36, 21-38	MSC0_ID and MSC1_ID registers are added.
21-42	Bit description of NDBH is updated.
21-21, 21-25, 21-66	Values for baud rate selections corrected at several locations and equations.
22-53	Usage of bit RXEN in Section 22.3.11.5 is updated.
22-54	First paragraph of Section 22.3.11.6 is updated.
22-59	Offset addresses for registers MSIMASK, PANCTR, MCR, and MITR in Table 22-5 corrected.

TC1796 User's Manual

Volume 1 (of 2) System Units & Volume 2 (of 2) Peripheral Units

Revision History: V2.0, 2007-07

22-58, 22-59, 22-61	CAN_ID register is added.
22-60	Figure 22-24 is updated.
22-79	CPU write to LEC in last row of Table 22-7 is updated.
22-90	CAN Bus State for of NoBit and NewBit in Table 22-9 is corrected.
22-201	Configuration of CAN_FDR.DM for Equation (22.1) and Equation (22.2) is updated.
Chapter 23	The functional description of the MLI , the module kernel description, and operation the MLI module is completely reworked.
23-75, 23-76, 23-78	MLI0_ID and MLI1_ID registers are added.
23-3	Programmable baud rate in Section 23.1.1.1 is updated.
23-115	Register description of RPxBAR (x = 0-3) is updated.
23-129	Equation (23.6) is added.
23-142	Selected address range for AEN21 corrected.
24-31, 24-32, 24-33, 24-34	Figure 24-20, Figure 24-21, Figure 24-22, and Figure 24-23 are updated.
24-61, 24-77	Cell Enabling on Event sections for GTC and LTC are updated.
24-76	Header of Table 24-4 is updated.
24-150, 24-151, 24-154	GPTA0_ID and GPTA1_ID registers are added.
24-233, 24-235	LTCA2_ID register is added.
24-125, 24-131	Section 24.2.6.3 "Check_Input()" and Section 24.2.6.6 "Manage_Mux()" are updated.
24-253	Table 24-26 is updated.
24-263	Figure 24-91 is updated.
24-266	Attention paragraph at the bottom is updated.
25-9	Equation (25.1) is updated.
25-37	The reference voltage Section 25.1.5 is updated.

TC1796 User's Manual
Volume 1 (of 2) System Units & Volume 2 (of 2) Peripheral Units
Revision History: V2.0, 2007-07

25-57, 25-58, 25-59	ADC0_ID register is added.
----------------------------	----------------------------

26-8	Equation (26.1) is updated.
-------------	------------------------------------

26-27, 26-28, 26-29	FADC_ID register is added.
----------------------------	----------------------------

Trademarks

TriCore® is a trademark of Infineon Technologies AG.

We Listen to Your Comments

Any information within this document that you feel is wrong, unclear or missing at all?
Your feedback will help us to continuously improve the quality of this document.

Please send your proposal (including a reference to this document) to:

mcdocu.comments@infineon.com



Table of Contents

This “Table of Contents” section refers to page numbers in both parts of the TC1796 User’s Manual, the “System Units” (volume 1 with marking “[1]”) and the “Peripheral Units” (volume 2 with marking “[2]”) parts.

System Units (Vol. 1 of 2)

1	Introduction	1-1 [1]
1.1	About this Document	1-1 [1]
1.1.1	Related Documentations	1-1 [1]
1.1.2	Text Conventions	1-1 [1]
1.1.3	Reserved, Undefined, and Unimplemented Terminology	1-3 [1]
1.1.4	Register Access Modes	1-4 [1]
1.1.5	Abbreviations and Acronyms	1-5 [1]
1.2	System Architecture of the TC1796	1-8 [1]
1.2.1	TC1796 Block Diagram	1-9 [1]
1.2.2	Features	1-10 [1]
1.3	On-Chip Peripheral Units of the TC1796	1-14 [1]
1.3.1	Serial Interfaces	1-14 [1]
1.3.1.1	Asynchronous/Synchronous Serial Interfaces	1-15 [1]
1.3.1.2	High-Speed Synchronous Serial Interfaces	1-17 [1]
1.3.1.3	Micro Second Channel Interfaces	1-19 [1]
1.3.1.4	MultiCAN Controller	1-21 [1]
1.3.1.5	Micro Link Serial Bus Interface	1-24 [1]
1.3.2	General Purpose Timer Array	1-26 [1]
1.3.2.1	Functionality of GPTA0 and GPTA1	1-27 [1]
1.3.2.2	Functionality of LTCA2	1-29 [1]
1.3.3	Analog-to-Digital Converters	1-30 [1]
1.3.3.1	Analog-to-Digital Converters (ADC0 and ADC1)	1-30 [1]
1.3.3.2	Fast Analog-to-Digital Converter Unit (FADC)	1-32 [1]
1.4	TC1796 Pin Definitions and Functions	1-34 [1]
1.4.1	TC1796 Pin Configuration	1-35 [1]
1.4.2	Pad Driver Classes Overview	1-57 [1]
1.4.3	Pull-Up/Pull-Down Behavior of the Pins	1-58 [1]
2	CPU Subsystem	2-1 [1]
2.1	TC1796 Processor Subsystem	2-1 [1]
2.2	Central Processing Unit Features	2-2 [1]
2.2.1	CPU Diagram	2-3 [1]
2.2.2	Instruction Fetch Unit	2-4 [1]
2.2.3	Execution Unit	2-5 [1]

Table of Contents

2.2.4	General Purpose Register File	2-6 [1]
2.3	Implementation-specific Features	2-7 [1]
2.3.1	Context Save Areas	2-7 [1]
2.3.2	Fast Context Switching	2-7 [1]
2.3.3	Reset System	2-7 [1]
2.3.4	Program Counter Register - PC	2-7 [1]
2.3.5	Interrupt System	2-8 [1]
2.3.6	Trap System	2-8 [1]
2.4	TC1796 CPU Subsystem Registers	2-9 [1]
2.4.1	Core Special Function Registers (CSFR)	2-10 [1]
2.4.1.1	Implementation-specific Core Special Function Registers	2-11 [1]
2.4.2	CPU Slave Interface (CPS) Registers	2-13 [1]
2.4.2.1	Implementation-specific CPU Slave Interface Registers	2-14 [1]
2.4.3	CPU General Purpose Registers	2-15 [1]
2.4.4	Core Debug Registers	2-17 [1]
2.4.4.1	Implementation-specific Core Debug Registers	2-18 [1]
2.4.5	Memory Protection Registers	2-20 [1]
2.4.5.1	Implementation-specific Memory Protection Registers	2-23 [1]
2.5	Program Memory Interface (PMI)	2-24 [1]
2.5.1	PMI Features	2-24 [1]
2.5.2	Parity Protection for PMI Memories	2-25 [1]
2.5.3	PMI Registers	2-26 [1]
2.5.3.1	PMI Module Identification Register	2-27 [1]
2.5.3.2	PMI Control Register 0	2-28 [1]
2.5.3.3	PMI Control Register 1	2-29 [1]
2.5.3.4	PMI Control Register 2	2-30 [1]
2.6	Data Memory Interface (DMI)	2-31 [1]
2.6.1	DMI Features	2-31 [1]
2.6.2	Dual-Ported Memory Operation	2-32 [1]
2.6.2.1	CPU Buffer Write Operation	2-32 [1]
2.6.3	Parity Protection for DMI Memories	2-33 [1]
2.6.4	DMI Registers	2-34 [1]
2.6.4.1	DMI Register Description	2-35 [1]
2.7	Instruction Timing	2-40 [1]
2.7.1	Integer-Pipeline Instructions	2-41 [1]
2.7.1.1	Simple Arithmetic Instruction Timings	2-41 [1]
2.7.1.2	Multiply Instruction Timings	2-45 [1]
2.7.1.3	MAC Instruction Timings	2-46 [1]
2.7.1.4	Control Flow Instruction Timing	2-47 [1]
2.7.2	Load-Store Pipeline Instructions	2-48 [1]
2.7.2.1	Address Arithmetic Timing	2-48 [1]
2.7.2.2	Control Flow Instruction Timing	2-49 [1]
2.7.2.3	Load Instruction Timing	2-50 [1]

Table of Contents

2.7.2.4	Store Instruction Timing	2-51 [1]
2.8	Floating Point Pipeline Timings	2-52 [1]
3	Clock System and Control	3-1 [1]
3.1	Overview	3-1 [1]
3.2	Clock Generation Unit	3-3 [1]
3.2.1	Main Oscillator Circuit	3-4 [1]
3.2.1.1	Oscillator Bypass Mode	3-5 [1]
3.2.1.2	Oscillator Run Detection	3-6 [1]
3.2.1.3	Oscillator Gain Control	3-7 [1]
3.2.1.4	Oscillator Control Register	3-8 [1]
3.2.2	Phase Locked Loop (PLL) Circuitry	3-10 [1]
3.2.2.1	Clock Source Control	3-10 [1]
3.2.2.2	PLL Parameters	3-12 [1]
3.2.2.3	PLL Clock Control and Status Register	3-16 [1]
3.2.2.4	Changing PLL Parameters	3-18 [1]
3.2.2.5	Setting up the PLL after Reset	3-18 [1]
3.2.2.6	Lock Detection	3-19 [1]
3.2.2.7	Loss-of-Lock Recovery	3-19 [1]
3.2.3	Power-on Startup Operation	3-21 [1]
3.3	Module Power Management and Clock Gating	3-22 [1]
3.3.1	Module Clock Generation	3-23 [1]
3.3.2	Clock Control Register CLC	3-24 [1]
3.3.3	Fractional Divider Operation	3-29 [1]
3.3.3.1	Overview	3-29 [1]
3.3.3.2	Fractional Divider Operating Modes	3-32 [1]
3.3.3.3	Fractional Divider Register	3-35 [1]
3.3.4	Module Clock Register Implementations	3-39 [1]
3.3.5	Fractional Divider Register Implementations	3-40 [1]
3.4	System Clock Output Control	3-41 [1]
3.4.1	System Clock Fractional Divider Register	3-42 [1]
4	Reset and Boot Operation	4-1 [1]
4.1	Reset and Boot Overview	4-1 [1]
4.1.1	Reset Status and Control Registers	4-3 [1]
4.1.1.1	Reset Status Register	4-3 [1]
4.1.1.2	Reset Request Register	4-5 [1]
4.2	Reset Operations	4-7 [1]
4.2.1	Power-On Reset	4-7 [1]
4.2.2	External Hardware Reset	4-7 [1]
4.2.3	Software Reset	4-8 [1]
4.2.4	Watchdog Timer Reset	4-8 [1]
4.2.5	Debug System Reset	4-10 [1]
4.2.6	Module Reset Behavior	4-10 [1]

Table of Contents

4.2.7	Bootling Scheme	4-12 [1]
4.2.7.1	Normal Boot Options	4-14 [1]
4.2.7.2	Debug Boot Options	4-15 [1]
4.3	Boot ROM	4-16 [1]
4.3.1	Addressing	4-16 [1]
4.3.2	Program Structure	4-16 [1]
4.3.3	Initial State after Boot ROM Exit	4-19 [1]
4.4	Bootstrap Loader (BSL)	4-20 [1]
4.4.1	Bootstrap Loader Mode 1 - ASC Boot via ASC0 Pins	4-21 [1]
4.4.2	Bootstrap Loader Mode 2 - CAN Boot via CAN Pins	4-24 [1]
4.4.3	Bootstrap Loader Mode 3 - ASC Boot via CAN Pins	4-28 [1]
4.4.4	Alternate Boot Modes	4-28 [1]
5	System Control Unit	5-1 [1]
5.1	Power Management	5-2 [1]
5.1.1	Power Management Overview	5-2 [1]
5.1.2	Power Management Control and Status Register, PMG_CSR	5-3 [1]
5.1.3	Power Management Modes	5-5 [1]
5.1.3.1	Idle Mode	5-5 [1]
5.1.3.2	Sleep Mode	5-6 [1]
5.1.3.3	States of TC1796 Units in Power Management Modes	5-7 [1]
5.2	Configuration Input Sampling	5-8 [1]
5.3	External Request Unit (ERU)	5-9 [1]
5.3.1	Input Channel	5-10 [1]
5.3.2	Output Channel	5-12 [1]
5.3.3	External Request Unit Implementation	5-14 [1]
5.3.4	External Request Unit Registers	5-16 [1]
5.4	Special System Interrupts	5-35 [1]
5.4.1	FPU Interrupts	5-35 [1]
5.4.2	Flash Interrupt	5-36 [1]
5.4.3	External Interrupts	5-36 [1]
5.5	SRAM Parity Control	5-37 [1]
5.5.1	Parity Error Trap Registers	5-39 [1]
5.6	Pad Driver Temperature Compensation Control	5-41 [1]
5.6.1	Functional Description	5-41 [1]
5.6.2	Temperature Compensation Registers	5-44 [1]
5.7	Die Temperature Sensor	5-48 [1]
5.8	GPTA1 Input IN1 Control	5-49 [1]
5.9	Pad Test Mode Control	5-50 [1]
5.9.1	Pad Test Mode Enabling	5-51 [1]
5.9.2	Pad Test Mode Registers	5-51 [1]
5.10	Emergency Stop Output Control for GPTA and MSC	5-57 [1]
5.10.1	GPTA Output Emergency Control in the GPIO Ports	5-58 [1]

Table of Contents

5.10.2	MSC Emergency Control Selection	5-58 [1]
5.10.3	Emergency Stop Register	5-59 [1]
5.11	Analog Input 7 Testmode	5-60 [1]
5.12	SCU Registers and Address Map	5-61 [1]
5.13	Miscellaneous SCU Registers	5-64 [1]
5.13.1	SCU Control Register	5-64 [1]
5.13.2	SCU Status Register	5-67 [1]
5.13.3	Device Identification Registers	5-69 [1]
6	On-Chip System Buses and Bus Bridges	6-1 [1]
6.1	Program and Data Local Memory Buses	6-2 [1]
6.1.1	Overview	6-2 [1]
6.1.2	Transaction Types	6-3 [1]
6.1.2.1	Single Transfers	6-3 [1]
6.1.2.2	Block Transfers	6-3 [1]
6.1.2.3	Atomic Transfers	6-3 [1]
6.1.3	Address Alignment Rules	6-3 [1]
6.1.4	Reaction of a Busy Slave	6-3 [1]
6.1.5	LMB Basic Operation	6-4 [1]
6.2	Local Memory Bus Controller Units	6-5 [1]
6.2.1	Basic Operation	6-5 [1]
6.2.2	LMB Bus Arbitration	6-5 [1]
6.2.2.1	LMB Bus Default Master	6-6 [1]
6.2.3	LMB Bus Error Handling	6-6 [1]
6.2.4	DLMB and PLMB Bus Registers	6-7 [1]
6.3	Local Memory to FPI Bus Interface (LFI Bridge)	6-15 [1]
6.3.1	Functional Overview	6-15 [1]
6.3.2	LFI Register	6-16 [1]
6.4	System and Remote Peripheral Bus	6-19 [1]
6.4.1	Overview	6-19 [1]
6.4.2	Bus Transaction Types	6-21 [1]
6.4.3	Reaction of a Busy Slave	6-21 [1]
6.4.4	Address Alignment Rules	6-22 [1]
6.4.5	FPI Bus Basic Operations	6-22 [1]
6.5	FPI Bus Control Units (SBCU and RBCU)	6-24 [1]
6.5.1	FPI Bus Arbitration	6-24 [1]
6.5.1.1	Arbitration on the System Peripheral Bus	6-24 [1]
6.5.1.2	Arbitration on the Remote Peripheral Bus	6-24 [1]
6.5.1.3	Starvation Prevention	6-25 [1]
6.5.2	FPI Bus Error Handling	6-25 [1]
6.5.3	Clock Management	6-27 [1]
6.5.4	BCU Debug Support	6-28 [1]
6.5.4.1	Address Triggers	6-28 [1]

Table of Contents

6.5.4.2	Signal Status Triggers	6-29 [1]
6.5.4.3	Grant Triggers	6-29 [1]
6.5.4.4	Combination of Triggers	6-30 [1]
6.5.4.5	BCU Breakpoint Generation Examples	6-31 [1]
6.5.5	SBCU and RBCU Registers	6-34 [1]
6.5.5.1	BCU Control Registers	6-36 [1]
6.5.5.2	BCU Error Registers	6-38 [1]
6.5.5.3	OCDS Registers	6-42 [1]
6.5.5.4	BCU Service Request Control Register	6-58 [1]
7	Program Memory Unit	7-1 [1]
7.1	Boot ROM	7-1 [1]
7.2	Program & Data Flash Memory	7-2 [1]
7.2.1	Program Flash Overview	7-4 [1]
7.2.2	Data Flash Overview	7-7 [1]
7.2.3	User Configuration Blocks Overview	7-9 [1]
7.2.4	Basic Flash Operating Modes	7-11 [1]
7.2.4.1	Read Mode	7-11 [1]
7.2.4.2	Command Mode	7-12 [1]
7.2.4.3	Page Mode	7-13 [1]
7.2.5	Command Sequence Definitions	7-14 [1]
7.2.5.1	Reset-to-Read Command	7-15 [1]
7.2.5.2	Enter Page Mode Command	7-15 [1]
7.2.5.3	Load Page Buffer Command	7-16 [1]
7.2.5.4	Write Page Command	7-18 [1]
7.2.5.5	Write User Configuration Page Command	7-19 [1]
7.2.5.6	Erase Sector Command	7-20 [1]
7.2.5.7	Erase User Configuration Block Command	7-22 [1]
7.2.5.8	Disable Write Protection Command	7-23 [1]
7.2.5.9	Disable Read Protection Command	7-24 [1]
7.2.5.10	Resume Protection Command	7-24 [1]
7.2.5.11	Clear Status Command	7-25 [1]
7.2.6	Data Flash and EEPROM Emulation	7-26 [1]
7.2.7	Read and Write Protection	7-27 [1]
7.2.7.1	User Configuration Block Definitions	7-27 [1]
7.2.7.2	Write and OTP Protection for PFLASH	7-30 [1]
7.2.7.3	Read Protection for PFLASH and DFLASH	7-32 [1]
7.2.7.4	Password Check Control	7-33 [1]
7.2.8	Error Correction and Margin Control	7-34 [1]
7.2.8.1	Dynamic Error Correction	7-34 [1]
7.2.8.2	Margin Check Control	7-35 [1]
7.2.9	Flash Interrupt Generation and Control	7-36 [1]
7.2.10	Flash Power Supply, Power Saving and Reset	7-38 [1]

Table of Contents

7.2.10.1	Power Supply	7-38 [1]
7.2.10.2	Sleep Mode	7-38 [1]
7.2.10.3	Shut-Down Mode	7-38 [1]
7.2.10.4	Reset Control	7-39 [1]
7.2.11	Flash Registers	7-40 [1]
7.2.11.1	Flash and PMU Module Identification Registers	7-41 [1]
7.2.11.2	Flash Status Register	7-43 [1]
7.2.11.3	Margin Control Registers	7-51 [1]
7.2.11.4	Flash Configuration Register	7-53 [1]
7.2.11.5	Protection Configuration Registers	7-59 [1]
7.3	Emulation Interface	7-61 [1]
8	Data Memory Unit	8-1 [1]
8.1	DLMB/PLMB Interfaces	8-2 [1]
8.2	SBRAM	8-2 [1]
8.3	SRAM	8-2 [1]
8.4	Parity Protection for DMU Memories	8-3 [1]
8.5	Data Access Overlay Functionality	8-3 [1]
8.5.1	Internal Overlay	8-7 [1]
8.5.2	Emulation Memory Overlay	8-7 [1]
8.5.3	Switching between Internal and Emulation Memory Overlay	8-7 [1]
8.5.4	Region Priority	8-7 [1]
8.5.5	Access Performance	8-7 [1]
8.6	Program Local Memory Bus Interface (LMI)	8-8 [1]
8.6.1	Data Read Buffer	8-8 [1]
8.7	DMU Registers	8-10 [1]
9	Memory Maps	9-1 [1]
9.1	How to Read the Address Maps	9-1 [1]
9.2	Contents of the Segments	9-4 [1]
9.3	Address Map of the FPI Bus System	9-6 [1]
9.3.1	Segments 0 to 14	9-6 [1]
9.3.2	Segment 15	9-10 [1]
9.4	Address Map of the Program Local Memory Bus (PLMB)	9-15 [1]
9.5	Address Map of the Data Local Memory Bus (DLMB)	9-19 [1]
9.6	Memory Module Access Restrictions	9-22 [1]
10	General Purpose I/O Ports and Peripheral I/O Lines	10-1 [1]
10.1	Basic Port Operation	10-2 [1]
10.2	Port Register Description	10-5 [1]
10.2.1	Port Input/Output Control Registers	10-7 [1]
10.2.2	Pad Driver Mode Register	10-10 [1]
10.2.3	Port Output Register	10-13 [1]
10.2.4	Port Output Modification Register	10-14 [1]

Table of Contents

10.2.5	Emergency Stop Register	10-16 [1]
10.2.6	Port Input Register	10-17 [1]
10.3	Port 0	10-18 [1]
10.3.1	Port 0 Configuration	10-18 [1]
10.3.2	Port 0 Function Table	10-19 [1]
10.3.3	Port 0 Registers	10-22 [1]
10.3.3.1	Port 0 Pad Driver Mode Register and Pad Classes	10-23 [1]
10.3.3.2	Port 0 Software Configuration Selection	10-24 [1]
10.3.3.3	Reserved SWOPT Bits of SCU_SCLIR Register	10-25 [1]
10.4	Port 1	10-26 [1]
10.4.1	Port 1 Configuration	10-26 [1]
10.4.2	Port 1 Function Table	10-27 [1]
10.4.3	Port 1 Registers	10-30 [1]
10.4.3.1	Port 1 Pad Driver Mode Register and Pad Classes	10-31 [1]
10.5	Port 2	10-32 [1]
10.5.1	Port 2 Configuration	10-32 [1]
10.5.2	Port 2 Function Table	10-33 [1]
10.5.3	Port 2 Registers	10-37 [1]
10.5.3.1	Port 2 Output Register	10-37 [1]
10.5.3.2	Port 2 Output Modification Register	10-37 [1]
10.5.3.3	Port 2 Input/Output Control Register 0	10-38 [1]
10.5.3.4	Port 2 Input Register	10-38 [1]
10.5.3.5	Port 2 Emergency Stop Register	10-38 [1]
10.5.3.6	Port 2 Pad Driver Mode Register and Pad Classes	10-39 [1]
10.6	Port 3	10-40 [1]
10.6.1	Port 3 Configuration	10-40 [1]
10.6.2	Port 3 Function Table	10-41 [1]
10.6.3	Port 3 Registers	10-45 [1]
10.6.3.1	Port 3 Pad Driver Mode Register and Pad Classes	10-46 [1]
10.7	Port 4	10-47 [1]
10.7.1	Port 4 Configuration	10-47 [1]
10.7.2	Port 4 Function Table	10-48 [1]
10.7.3	Port 4 Registers	10-51 [1]
10.7.3.1	Port 4 Pad Driver Mode Register and Pad Classes	10-53 [1]
10.8	Port 5	10-54 [1]
10.8.1	Port 5 Configuration	10-54 [1]
10.8.2	Port 5 Function Table	10-55 [1]
10.8.3	Port 5 Registers	10-57 [1]
10.8.3.1	Port 5 Output Register	10-57 [1]
10.8.3.2	Port 5 Output Modification Register	10-57 [1]
10.8.3.3	Port 5 Input Register	10-57 [1]
10.8.3.4	Port 5 Pad Driver Mode Register and Pad Classes	10-58 [1]
10.9	Port 6	10-59 [1]

Table of Contents

10.9.1	Port 6 Configuration	10-59 [1]
10.9.2	Port 6 Function Table	10-60 [1]
10.9.3	Port 6 Registers	10-63 [1]
10.9.3.1	Port 6 Output Register	10-63 [1]
10.9.3.2	Port 6 Output Modification Register	10-63 [1]
10.9.3.3	Port 6 Input Register	10-63 [1]
10.9.3.4	Port 6 Pad Driver Mode Register and Pad Classes	10-64 [1]
10.10	Port 7	10-65 [1]
10.10.1	Port 7 Configuration	10-65 [1]
10.10.2	Port 7 Function Table	10-66 [1]
10.10.3	Port 7 Registers	10-68 [1]
10.10.3.1	Port 7 Output Register	10-68 [1]
10.10.3.2	Port 7 Output Modification Register	10-68 [1]
10.10.3.3	Port 7 Input Register	10-68 [1]
10.10.3.4	Port 7 Pad Driver Mode Register and Pad Classes	10-69 [1]
10.11	Port 8	10-70 [1]
10.11.1	Port 8 Configuration	10-70 [1]
10.11.2	Port 8 Function Table	10-71 [1]
10.11.3	Port 8 Register	10-73 [1]
10.11.3.1	Port 8 Output Register	10-73 [1]
10.11.3.2	Port 8 Output Modification Register	10-73 [1]
10.11.3.3	Port 8 Input Register	10-73 [1]
10.11.3.4	Port 8 Emergency Stop Register	10-73 [1]
10.11.3.5	Port 8 Pad Driver Mode Register and Pad Classes	10-74 [1]
10.12	Port 9	10-75 [1]
10.12.1	Port 9 Configuration	10-75 [1]
10.12.2	Port 9 Function Table	10-76 [1]
10.12.3	Port 9 Register	10-78 [1]
10.12.3.1	Port 9 Output Register	10-78 [1]
10.12.3.2	Port 9 Output Modification Register	10-78 [1]
10.12.3.3	Port 9 Input/Output Control Register 8	10-79 [1]
10.12.3.4	Port 9 Input Register	10-79 [1]
10.12.3.5	Port 9 Emergency Stop Register	10-79 [1]
10.12.3.6	Port 9 Pad Driver Mode Register and Pad Classes	10-80 [1]
10.13	Port 10 (Hardware Select Inputs)	10-81 [1]
10.13.1	Port 10 Configuration	10-81 [1]
10.13.2	Port 10 Registers	10-82 [1]
10.13.2.1	Port 10 Input Register	10-82 [1]
10.14	Dedicated Peripheral I/O Lines	10-83 [1]
10.14.1	Dedicated I/O Lines for SSC0 and SSC1	10-83 [1]
10.14.2	LVDS Outputs of MSC0 and MSC1	10-85 [1]

Table of Contents

11	Peripheral Control Processor (PCP)	11-1 [1]
11.1	Peripheral Control Processor Overview	11-1 [1]
11.2	PCP Architecture	11-2 [1]
11.2.1	PCP Processor	11-3 [1]
11.2.2	PCP Code Memory	11-4 [1]
11.2.3	PCP Parameter RAM	11-4 [1]
11.2.4	FPI Bus Interface	11-4 [1]
11.2.5	PCP Interrupt Control Unit and Service Request Nodes	11-5 [1]
11.3	PCP Programming Model	11-6 [1]
11.3.1	General Purpose Register Set of the PCP	11-6 [1]
11.3.1.1	Register R0	11-7 [1]
11.3.1.2	Registers R1, R2, and R3	11-7 [1]
11.3.1.3	Registers R4 and R5	11-7 [1]
11.3.1.4	Register R6	11-8 [1]
11.3.1.5	Register R7	11-9 [1]
11.3.2	Contexts and Context Models	11-11 [1]
11.3.2.1	Context Models	11-11 [1]
11.3.2.2	Context Save Area	11-14 [1]
11.3.2.3	Context Restore Operation for CR6 and CR7	11-17 [1]
11.3.2.4	Context Save Operation for CR6 and CR7	11-21 [1]
11.3.2.5	Initialization of the Contexts	11-24 [1]
11.3.2.6	Context Save Optimization	11-24 [1]
11.3.3	Channel Programs	11-25 [1]
11.3.3.1	Channel Restart Mode	11-25 [1]
11.3.3.2	Channel Resume Mode	11-26 [1]
11.4	PCP Operation	11-28 [1]
11.4.1	PCP Initialization	11-28 [1]
11.4.2	Channel Invocation and Context Restore Operation	11-28 [1]
11.4.3	Channel Exit and Context Save Operation	11-29 [1]
11.4.3.1	Normal Exit	11-29 [1]
11.4.3.2	Error Condition Channel Exit	11-30 [1]
11.4.3.3	Debug Exit	11-31 [1]
11.5	PCP Interrupt Operation	11-32 [1]
11.5.1	Issuing Service Requests to CPU or PCP	11-32 [1]
11.5.2	PCP Interrupt Control Unit	11-33 [1]
11.5.3	PCP Service Request Nodes	11-33 [1]
11.5.4	Issuing PCP Service Requests	11-34 [1]
11.5.4.1	Service Request on EXIT Instruction	11-35 [1]
11.5.4.2	Service Request on Suspension of Interrupt	11-35 [1]
11.5.4.3	Service Request on Error	11-36 [1]
11.5.4.4	Queue Full Operation	11-36 [1]
11.6	PCP Error Handling	11-38 [1]
11.6.1	Enforced PRAM Partitioning	11-38 [1]

Table of Contents

11.6.2	Channel Watchdog	11-39 [1]
11.6.3	Invalid Opcode	11-39 [1]
11.6.4	Instruction Address Error	11-39 [1]
11.7	Instruction Set Overview	11-40 [1]
11.7.1	DMA Primitives	11-40 [1]
11.7.2	Load and Store	11-41 [1]
11.7.3	Arithmetic and Logical Instructions	11-42 [1]
11.7.4	Bit Manipulation	11-44 [1]
11.7.5	Flow Control	11-44 [1]
11.7.6	Addressing Modes	11-45 [1]
11.7.6.1	FPI Bus Addressing	11-45 [1]
11.7.6.2	PRAM Addressing	11-46 [1]
11.7.6.3	Bit Addressing	11-46 [1]
11.7.6.4	Flow Control Destination Addressing	11-46 [1]
11.8	Accessing PCP Resources from the FPI Bus	11-48 [1]
11.8.1	Access to the PCP Control Registers	11-48 [1]
11.8.2	Access to the PRAM	11-48 [1]
11.8.3	Access to the CMEM	11-49 [1]
11.9	Debugging the PCP	11-50 [1]
11.10	PCP Registers	11-52 [1]
11.10.1	Module Identification Register, PCP_ID	11-54 [1]
11.10.2	PCP Clock Control Register, PCP_CLC	11-55 [1]
11.10.3	PCP Control and Status Register, PCP_CS	11-56 [1]
11.10.4	PCP Error/Debug Status Register, PCP_ES	11-59 [1]
11.10.5	PCP Interrupt Control Register, PCP_ICR	11-61 [1]
11.10.6	PCP Interrupt Threshold Register, PCP_ITR	11-63 [1]
11.10.7	PCP Interrupt Configuration Register, PCP_ICON	11-64 [1]
11.10.8	PCP Stall Status Register, PCP_SSR	11-66 [1]
11.10.9	PCP Service Request Control Registers, PCP_SRC[1:0]	11-67 [1]
11.10.10	PCP Service Request Control Registers, PCP_SRC[3:2]	11-68 [1]
11.10.11	PCP Service Request Control Registers, PCP_SRC[8:4]	11-69 [1]
11.10.12	PCP Service Request Control Registers, PCP_SRC[11:9]	11-70 [1]
11.11	PCP Instruction Set Details	11-72 [1]
11.11.1	Instruction Codes and Fields	11-72 [1]
11.11.1.1	Conditional Codes	11-73 [1]
11.11.1.2	Instruction Fields	11-74 [1]
11.11.2	Counter Operation for COPY Instruction	11-77 [1]
11.11.3	Counter Operation for BCOPY Instruction	11-78 [1]
11.11.4	Divide and Multiply Instructions	11-79 [1]
11.11.5	ADD, 32-Bit Addition	11-80 [1]
11.11.6	BCOPY, DMA Operation	11-81 [1]
11.11.7	AND, 32-Bit Logical AND	11-82 [1]
11.11.8	CHKB, Check Bit	11-83 [1]

Table of Contents

11.11.9	CLR, Clear Bit	11-83 [1]
11.11.10	COMP, 32-Bit Compare	11-84 [1]
11.11.11	COPY, DMA Instruction	11-85 [1]
11.11.12	DEBUG, Debug Instruction	11-86 [1]
11.11.13	DINIT, Divide Initialization Instruction	11-87 [1]
11.11.14	DSTEP, Divide Instruction	11-88 [1]
11.11.15	INB, Insert Bit	11-88 [1]
11.11.16	EXIT, Exit Instruction	11-89 [1]
11.11.17	JC, Jump Conditionally	11-90 [1]
11.11.18	JL, Jump Long Unconditional	11-91 [1]
11.11.19	LD, Load	11-91 [1]
11.11.20	LDL, Load 16-bit Value	11-93 [1]
11.11.21	Multiply Initialization Instruction	11-93 [1]
11.11.22	MOV, Move Register to Register	11-94 [1]
11.11.23	Multiply Instructions	11-95 [1]
11.11.24	NEG, Negate	11-96 [1]
11.11.25	NOP, No Operation	11-96 [1]
11.11.26	NOT, Logical NOT	11-96 [1]
11.11.27	OR, Logical OR	11-97 [1]
11.11.28	PRI, Prioritize	11-98 [1]
11.11.29	PRAM Bit Operations	11-99 [1]
11.11.30	RL, Rotate Left	11-100 [1]
11.11.31	RR, Rotate Right	11-100 [1]
11.11.32	SET, Set Bit	11-101 [1]
11.11.33	SHL, Shift Left	11-101 [1]
11.11.34	SHR, Shift Right	11-102 [1]
11.11.35	ST, Store	11-103 [1]
11.11.36	SUB, 32-Bit Subtract	11-104 [1]
11.11.37	XCH, Exchange	11-105 [1]
11.11.38	XOR, 32-Bit Logical Exclusive OR	11-106 [1]
11.11.39	Flag Updates of Instructions	11-107 [1]
11.11.40	Instruction Timing	11-108 [1]
11.12	Programming of the PCP	11-112 [1]
11.12.1	Initial PC of a channel program	11-112 [1]
11.12.1.1	Channel Entry Table	11-112 [1]
11.12.1.2	Channel Resume	11-113 [1]
11.12.2	Channel Management for Small and Minimum Contexts	11-114 [1]
11.12.3	Unused Registers as Globals or Constants	11-114 [1]
11.12.4	Dispatch of Low Priority Tasks	11-115 [1]
11.12.5	Code Reuse Across Channels (Call and Return)	11-115 [1]
11.12.6	Case-like Code Switches (Computed Go-To)	11-116 [1]
11.12.7	Simple DMA Operation	11-116 [1]
11.12.7.1	COPY Instruction	11-116 [1]

Table of Contents

11.12.7.2	BCOPY Instruction (Burst Copy)	11-117 [1]
11.13	PCP Programming Notes and Tips	11-118 [1]
11.13.1	Notes on PCP Configuration	11-118 [1]
11.13.2	General Purpose Register Use	11-119 [1]
11.13.3	Use of Channel Interruption	11-120 [1]
11.13.3.1	Dynamic Interrupt Masking	11-120 [1]
11.13.3.2	Control of Channel Priority (CPPN)	11-120 [1]
11.13.4	Implementing Divide Algorithms	11-122 [1]
11.13.5	Implementing Multiply Algorithms	11-123 [1]
11.14	Implementation of the PCP in the TC1796	11-125 [1]
11.14.1	PCP Memories	11-125 [1]
11.14.2	Parity Protection for PCP Memories	11-125 [1]
11.14.3	PCP Reset Operation	11-126 [1]
11.14.4	BCOPY Instruction	11-127 [1]
12	Direct Memory Access Controller	12-1 [1]
12.1	DMA Controller Kernel Description	12-2 [1]
12.1.1	Features	12-3 [1]
12.1.2	Definition of Terms	12-4 [1]
12.1.3	DMA Principles	12-5 [1]
12.1.4	DMA Channel Functionality	12-6 [1]
12.1.4.1	Shadowed Source or Destination Address	12-6 [1]
12.1.4.2	DMA Channel Request Control	12-10 [1]
12.1.4.3	DMA Channel Operation Modes	12-11 [1]
12.1.4.4	Error Conditions	12-15 [1]
12.1.4.5	Channel Reset Operation	12-16 [1]
12.1.4.6	Transfer Count and Move Count	12-17 [1]
12.1.4.7	Circular Buffer	12-19 [1]
12.1.5	Transaction Control Engine	12-20 [1]
12.1.6	Bus Switch	12-21 [1]
12.1.7	On-Chip Debug Capabilities	12-23 [1]
12.1.7.1	Hard-suspend Mode	12-23 [1]
12.1.7.2	Soft-suspend Mode	12-23 [1]
12.1.7.3	Break Signal Generation	12-24 [1]
12.1.7.4	Trace Signal Generation	12-25 [1]
12.1.8	Interrupts	12-27 [1]
12.1.8.1	Channel Interrupts	12-27 [1]
12.1.8.2	Transaction Lost Interrupt	12-29 [1]
12.1.8.3	Move Engine Interrupts	12-30 [1]
12.1.8.4	Wrap Buffer Interrupts	12-32 [1]
12.1.8.5	Interrupt Request Compressor	12-33 [1]
12.1.9	Pattern Detection	12-34 [1]
12.1.9.1	Pattern Compare Logic	12-35 [1]

Table of Contents

12.1.9.2	Pattern Detection for 8-bit Data Width	12-36 [1]
12.1.9.3	Pattern Detection for 16-bit Data Width	12-37 [1]
12.1.9.4	Pattern Detection for 32-bit Data Width	12-39 [1]
12.1.9.5	Access Protection	12-40 [1]
12.2	DMA Module Kernel Registers	12-42 [1]
12.2.1	System Registers	12-45 [1]
12.2.2	General Control/Status Registers	12-51 [1]
12.2.3	Move Engine Registers	12-69 [1]
12.2.4	Channel Control/Status Registers	12-76 [1]
12.2.5	Channel Address Registers	12-88 [1]
12.3	DMA Module Implementation	12-91 [1]
12.3.1	DMA Request Wiring Matrix	12-92 [1]
12.3.2	Access Protection Assignment	12-98 [1]
12.3.3	Implementation-specific DMA Registers	12-103 [1]
12.3.3.1	Clock Control Register	12-105 [1]
12.3.3.2	DMA Service Request Control Registers	12-106 [1]
12.3.3.3	MLI Service Request Control Registers	12-107 [1]
12.3.3.4	System Interrupt Service Request Control Register	12-108 [1]
12.3.4	DMA Controller Address Map	12-109 [1]
12.4	Memory Checker Module	12-110 [1]
12.4.1	Functional Description	12-110 [1]
12.4.2	Registers	12-111 [1]
12.4.2.1	Memory Checker Registers	12-112 [1]
13	LMB External Bus Unit	13-1 [1]
13.1	Block Diagram	13-2 [1]
13.2	EBU Interface Signals	13-3 [1]
13.2.1	Data Bus, $D[31:0]$	13-3 [1]
13.2.2	Address Bus, $A[23:0]$	13-4 [1]
13.2.3	Chip Selects, $CS[3:0]$, $CSCOMB$	13-4 [1]
13.2.4	Burst Flash Clock Output/Input, $BFCLKO/BFCLKI$	13-4 [1]
13.2.5	Read/Write Control Lines, RD , RD/WR and MR/W	13-4 [1]
13.2.6	Address Valid, \overline{ADV}	13-5 [1]
13.2.7	Byte Controls, $BC[3:0]$	13-5 [1]
13.2.8	Wait Input, \overline{WAIT}	13-6 [1]
13.2.9	Burst Address Advance, \overline{BAA}	13-6 [1]
13.2.10	Bus Arbitration Signals, $HOLD$, $HLDA$, and $BREQ$	13-6 [1]
13.2.11	EBU Power Supply	13-6 [1]
13.3	External Bus Arbitration	13-7 [1]
13.3.1	External Bus Modes	13-7 [1]
13.3.2	Arbitration Signals and Parameters	13-7 [1]
13.3.3	Arbitration Modes	13-10 [1]
13.3.3.1	No Bus Arbitration Mode	13-10 [1]

Table of Contents

13.3.3.2	Sole Master Arbitration Mode	13-10 [1]
13.3.3.3	Arbiter Mode Arbitration Mode	13-10 [1]
13.3.3.4	“Participant Mode” Arbitration Mode	13-14 [1]
13.3.4	Arbitration Input Signal Sampling	13-17 [1]
13.3.5	Locking the External Bus	13-17 [1]
13.3.6	Reaction to an PLMB Access to the External Bus	13-18 [1]
13.3.6.1	Pending Access Time-Out	13-19 [1]
13.4	Start-Up/Boot Process	13-20 [1]
13.4.1	Disabled	13-20 [1]
13.4.2	Emulation Mode	13-20 [1]
13.4.3	External Boot Mode	13-21 [1]
13.4.3.1	Boot Process	13-21 [1]
13.4.3.2	Boot Configuration Value	13-23 [1]
13.5	Master Mode Operation	13-25 [1]
13.5.1	External Memory Regions	13-26 [1]
13.5.2	Chip Select Control	13-28 [1]
13.5.3	Address Comparison	13-30 [1]
13.5.4	Access Parameter Selection	13-35 [1]
13.5.5	Little-/Big-Endian Access Modes	13-36 [1]
13.5.6	PLMB Bus Width Translation	13-37 [1]
13.5.7	Address Alignment During Bus Accesses	13-38 [1]
13.6	PLMB Data Buffering	13-39 [1]
13.6.1	Data Read Buffer	13-39 [1]
13.6.2	Code Prefetch Buffer	13-40 [1]
13.6.3	Data Write Buffer	13-40 [1]
13.7	Standard Access Phases	13-41 [1]
13.7.1	Address Phase (AP)	13-41 [1]
13.7.2	Command Delay Phase (CD)	13-43 [1]
13.7.3	Command Phase (CP)	13-43 [1]
13.7.4	Data Hold Phase (DH)	13-45 [1]
13.7.5	Burst Phase (BP)	13-46 [1]
13.7.6	Recovery Phase (RP)	13-47 [1]
13.7.7	Multiplication Factor for Access Phase Length	13-49 [1]
13.8	Asynchronous Read/Write Accesses	13-50 [1]
13.8.1	Signal List	13-50 [1]
13.8.2	Demultiplexed Device Configurations	13-51 [1]
13.8.3	Standard Asynchronous Access Phases	13-53 [1]
13.8.4	Programmable Parameters	13-53 [1]
13.8.5	Accesses to Demultiplexed Devices	13-55 [1]
13.8.6	Dynamic Command Delay and Wait State Insertion	13-57 [1]
13.8.6.1	External Extension of the Command Phase by WAIT	13-57 [1]
13.8.6.2	Interfacing to INTEL-style Devices	13-61 [1]
13.8.6.3	Interfacing to Motorola-style Devices	13-63 [1]

Table of Contents

13.9	Burst Mode Read Accesses	13-65 [1]
13.9.1	Signal List	13-65 [1]
13.9.2	Burst Flash Memory Configurations	13-66 [1]
13.9.3	Burst Mode Access Phases	13-68 [1]
13.9.4	Programmable Parameters	13-68 [1]
13.9.5	Support for two Burst Flash Device Types	13-71 [1]
13.9.6	BFCLKO Output	13-71 [1]
13.9.7	BFCLKI Input and Burst Flash Clock Feedback	13-72 [1]
13.9.8	Burst Length Control	13-73 [1]
13.9.9	Control of ADV and BAA Delays During Burst Flash Access ...	13-73 [1]
13.9.10	Burst Flash Access Cycle	13-74 [1]
13.9.11	External Cycle Control via the WAIT Input	13-75 [1]
13.9.11.1	Wait for Page Load Mode	13-76 [1]
13.9.11.2	Terminate and Start New Burst Mode	13-77 [1]
13.9.12	Termination of a Burst Mode Read Access	13-78 [1]
13.10	EBU Registers	13-79 [1]
13.10.1	Identification Register, ID	13-81 [1]
13.10.2	Clock Control Register, CLC	13-82 [1]
13.10.3	Configuration Register, CON	13-83 [1]
13.10.4	Burst Flash Control Register, BFCON	13-86 [1]
13.10.5	Address Select Register, ADDRSELx	13-90 [1]
13.10.6	Bus Configuration Register, BUSCONx	13-92 [1]
13.10.7	Bus Access Parameter Register, BUSAPx	13-97 [1]
13.10.8	Emulator Address Select Register, EMUAS	13-101 [1]
13.10.9	Emulator Bus Configuration Register, EMUBC	13-102 [1]
13.10.10	Emulator Bus Access Parameter Register, EMUBAP	13-106 [1]
13.10.11	Emulator Overlay Register, EMUOVL	13-110 [1]
13.10.12	Test/Control Configuration Register, USERCON	13-111 [1]
14	Interrupt System	14-1 [1]
14.1	Overview	14-1 [1]
14.2	Service Request Nodes	14-3 [1]
14.2.1	Service Request Control Registers	14-3 [1]
14.2.1.1	General Service Request Control Register Layout	14-4 [1]
14.2.1.2	Request Set and Clear Bits (SETR, CLRR)	14-5 [1]
14.2.1.3	Enable Bit (SRE)	14-5 [1]
14.2.1.4	Service Request Flag (SRR)	14-6 [1]
14.2.1.5	Type-Of-Service Control (TOS)	14-6 [1]
14.2.1.6	Service Request Priority Number (SRPN)	14-7 [1]
14.3	Interrupt Control Units	14-8 [1]
14.3.1	Interrupt Control Unit (ICU)	14-8 [1]
14.3.1.1	ICU Interrupt Control Register (ICR)	14-8 [1]
14.3.1.2	Operation of the Interrupt Control Unit (ICU)	14-10 [1]

Table of Contents

14.3.2	PCP Interrupt Control Unit (PICU)	14-11 [1]
14.4	Arbitration Process	14-12 [1]
14.4.1	Controlling the Number of Arbitration Cycles	14-12 [1]
14.4.2	Controlling the Duration of Arbitration Cycles	14-13 [1]
14.5	Entering an Interrupt Service Routine	14-13 [1]
14.6	Exiting an Interrupt Service Routine	14-14 [1]
14.7	Interrupt Vector Table	14-15 [1]
14.8	Usage of the TC1796 Interrupt System	14-18 [1]
14.8.1	Spanning Interrupt Service Routines Across Vector Entries	14-18 [1]
14.8.2	Configuring Ordinary Interrupt Service Routines	14-19 [1]
14.8.3	Interrupt Priority Groups	14-19 [1]
14.8.4	Splitting Interrupt Service Across Different Priority Levels	14-20 [1]
14.8.5	Using different Priorities for the same Interrupt Source	14-21 [1]
14.8.6	Interrupt Priority 1	14-22 [1]
14.8.7	Software-Initiated Interrupts	14-22 [1]
14.8.8	External Interrupts	14-22 [1]
14.9	Service Request Node Table	14-23 [1]
14.10	Non-Maskable Interrupt	14-25 [1]
14.10.1	External NMI Input	14-25 [1]
14.10.2	Phase-Locked Loop NMI	14-25 [1]
14.10.3	Watchdog Timer NMI	14-25 [1]
14.10.4	SRAM Parity Error NMI	14-26 [1]
14.10.5	NMI Enable	14-27 [1]
14.10.6	NMI Status Register	14-27 [1]
15	System Timer	15-1 [1]
15.1	Overview	15-1 [1]
15.2	Operation	15-1 [1]
15.2.1	Resolution and Ranges	15-4 [1]
15.2.2	Compare Register Operation	15-5 [1]
15.2.3	Compare Match Interrupt Control	15-6 [1]
15.3	Kernel Registers	15-7 [1]
15.3.1	General Module Register	15-9 [1]
15.3.2	Timer/Capture Registers	15-11 [1]
15.3.3	Compare Registers	15-15 [1]
15.3.4	Interrupt Registers	15-18 [1]
16	Watchdog Timer	16-1 [1]
16.1	Watchdog Timer Overview	16-1 [1]
16.2	Features of the Watchdog Timer	16-2 [1]
16.3	The Endinit Function	16-3 [1]
16.4	Watchdog Timer Operation	16-5 [1]
16.4.1	WDT Register Overview	16-6 [1]
16.4.2	Operating Modes of the Watchdog Timer	16-7 [1]

Table of Contents

16.4.2.1	Time-Out Mode	16-8 [1]
16.4.2.2	Normal Mode	16-8 [1]
16.4.2.3	Disable Mode	16-8 [1]
16.4.2.4	Prewarning Mode	16-9 [1]
16.4.3	Password Access to WDT_CON0	16-10 [1]
16.4.4	Modify Access to WDT_CON0	16-11 [1]
16.4.5	Term Definitions for WDT_CON0 Accesses	16-12 [1]
16.4.6	Detailed Descriptions of the WDT Modes	16-13 [1]
16.4.6.1	Time-Out Mode Details	16-13 [1]
16.4.6.2	Normal Mode Details	16-14 [1]
16.4.6.3	Disable Mode Details	16-15 [1]
16.4.6.4	Prewarning Mode Details	16-16 [1]
16.4.6.5	WDT Operation During Power-Saving Modes	16-17 [1]
16.4.6.6	WDT Operation in OCDS Suspend Mode	16-17 [1]
16.4.7	Determining WDT Periods	16-18 [1]
16.4.7.1	Time-out Period	16-18 [1]
16.4.7.2	Normal Period	16-20 [1]
16.4.7.3	WDT Period During Power-Saving Modes	16-21 [1]
16.5	Handling the Watchdog Timer	16-22 [1]
16.5.1	System Initialization	16-22 [1]
16.5.2	Re-opening Access to Critical System Registers	16-23 [1]
16.5.3	Servicing the Watchdog Timer	16-23 [1]
16.5.4	Handling the User-Definable Password Field	16-24 [1]
16.5.5	Determining the Required Values for a WDT Access	16-27 [1]
16.6	Watchdog Timer Registers	16-28 [1]
16.6.1	Watchdog Timer Control Register 0	16-29 [1]
16.6.2	Watchdog Timer Control Register 1	16-31 [1]
16.6.3	Watchdog Timer Status Register	16-32 [1]
17	On-Chip Debug Support	17-1 [1]
17.1	Overview	17-1 [1]
17.2	OCDS Level 1	17-4 [1]
17.2.1	TriCore CPU Level 1 OCDS	17-4 [1]
17.2.1.1	Basic Concept	17-5 [1]
17.2.1.2	Debug Event Generation	17-6 [1]
17.2.1.3	Debug Actions	17-7 [1]
17.2.1.4	TriCore OCDS Registers	17-8 [1]
17.2.2	PCP OCDS Level 1	17-8 [1]
17.2.3	BCU OCDS Level 1	17-9 [1]
17.2.4	DMA OCDS Level 1	17-9 [1]
17.3	OCDS Level 2 Debugging via Trace Port	17-10 [1]
17.3.1	TriCore CPU and PCP OCDS Level 2 Trace	17-10 [1]
17.3.2	DMA OCDS Level 2 Trace	17-10 [1]

Table of Contents

17.3.3	Concurrent Debugging	17-11 [1]
17.4	Debug Interface (Cerberus)	17-12 [1]
17.4.1	RW Mode	17-12 [1]
17.4.2	Communication Mode	17-13 [1]
17.4.3	Triggered Transfers	17-13 [1]
17.4.4	Multi Core Break Switch	17-13 [1]
17.5	JTAG Interface	17-15 [1]
17.6	Cerberus and JTAG Registers	17-16 [1]
18	Register Overview	18-1 [1]
18.1	Address Map of Segment 15	18-2 [1]
18.2	Registers Tables	18-7 [1]

Peripheral Units (Vol. 2 of 2)

19	Asynchronous/Synchronous Serial Interface (ASC)	19-1 [2]
19.1	ASC Kernel Description	19-1 [2]
19.1.1	Overview	19-2 [2]
19.1.2	General Operation	19-3 [2]
19.1.3	Asynchronous Operation	19-4 [2]
19.1.3.1	Asynchronous Data Frames	19-5 [2]
19.1.3.2	Asynchronous Transmission	19-7 [2]
19.1.3.3	Asynchronous Reception	19-7 [2]
19.1.3.4	RXD/TXD Data Path Selection in Asynchronous Modes	19-8 [2]
19.1.4	Synchronous Operation	19-9 [2]
19.1.4.1	Synchronous Transmission	19-10 [2]
19.1.4.2	Synchronous Reception	19-10 [2]
19.1.4.3	Synchronous Timing	19-11 [2]
19.1.5	Baud Rate Generation	19-12 [2]
19.1.5.1	Baud Rates in Asynchronous Mode	19-13 [2]
19.1.5.2	Baud Rates in Synchronous Mode	19-16 [2]
19.1.6	Hardware Error Detection Capabilities	19-17 [2]
19.1.7	Interrupts	19-17 [2]
19.2	ASC Kernel Registers	19-19 [2]
19.2.1	Identification Register	19-20 [2]
19.2.2	Control Registers	19-21 [2]
19.2.3	Data Registers	19-28 [2]
19.3	ASC0/ASC1 Module Implementation	19-30 [2]
19.3.1	Interfaces of the ASC Modules	19-30 [2]
19.3.2	ASC0/ASC1 Module Related External Registers	19-31 [2]
19.3.2.1	ASC0 Clock Control Register	19-32 [2]
19.3.2.2	Peripheral Input Select Registers	19-34 [2]
19.3.2.3	Port Control Registers	19-36 [2]
19.3.2.4	Interrupt Control Registers	19-40 [2]
19.3.3	DMA Requests	19-42 [2]
20	Synchronous Serial Interface (SSC)	20-1 [2]
20.1	SSC Kernel Description	20-1 [2]
20.1.1	Overview	20-2 [2]
20.1.2	General Operation	20-3 [2]
20.1.2.1	Operating Mode Selection	20-5 [2]
20.1.2.2	Full-Duplex Operation	20-6 [2]
20.1.2.3	Half-Duplex Operation	20-9 [2]
20.1.2.4	Continuous Transfers	20-10 [2]
20.1.2.5	Port Control	20-11 [2]
20.1.2.6	Transmit FIFO Operation	20-12 [2]

Table of Contents

20.1.2.7	Receive FIFO Operation	20-14 [2]
20.1.2.8	FIFO Transparent Mode	20-16 [2]
20.1.2.9	Baud Rate Generation	20-18 [2]
20.1.2.10	Slave Select Input Operation	20-20 [2]
20.1.2.11	Slave Select Output Generation Unit	20-21 [2]
20.1.2.12	Error Detection Mechanisms	20-24 [2]
20.2	SSC Kernel Registers	20-27 [2]
20.2.1	Identification Register	20-29 [2]
20.2.2	Control Registers	20-30 [2]
20.2.3	Data Registers	20-45 [2]
20.3	SSC0/SSC1 Module Implementation	20-46 [2]
20.3.1	Interfaces of the SSC Modules	20-46 [2]
20.3.2	SSC0/SSC1 Module Related External Registers	20-47 [2]
20.3.2.1	Clock Control	20-48 [2]
20.3.2.2	Port Control	20-52 [2]
20.3.2.3	Interrupt Control Registers	20-59 [2]
20.3.3	DMA Requests	20-60 [2]
21	Micro Second Channel (MSC)	21-1 [2]
21.1	MSC Kernel Description	21-3 [2]
21.1.1	Overview	21-3 [2]
21.1.2	Downstream Channel	21-5 [2]
21.1.2.1	Frame Formats and Definitions	21-6 [2]
21.1.2.2	Shift Register Operation	21-12 [2]
21.1.2.3	Transmission Modes	21-14 [2]
21.1.2.4	Downstream Counter and Enable Signals	21-19 [2]
21.1.2.5	Baud Rate	21-20 [2]
21.1.2.6	Abort of Frames	21-20 [2]
21.1.3	Upstream Channel	21-21 [2]
21.1.3.1	Data Frames	21-22 [2]
21.1.3.2	Parity Checking	21-22 [2]
21.1.3.3	Data Reception	21-23 [2]
21.1.3.4	Baud Rate	21-25 [2]
21.1.3.5	Spike Filter	21-26 [2]
21.1.4	I/O Control	21-27 [2]
21.1.4.1	Downstream Channel Output Control	21-27 [2]
21.1.4.2	Upstream Channel	21-30 [2]
21.1.5	MSC Interrupts	21-31 [2]
21.1.5.1	Data Frame Interrupt	21-32 [2]
21.1.5.2	Command Frame Interrupt	21-32 [2]
21.1.5.3	Time Frame Finished Interrupt	21-33 [2]
21.1.5.4	Receive Data Interrupt	21-34 [2]
21.1.5.5	Interrupt Request Compressor	21-35 [2]

Table of Contents

21.2	MSC Kernel Registers	21-36 [2]
21.2.1	Identification Register	21-38 [2]
21.2.2	Status and Control Registers	21-39 [2]
21.2.3	Data Registers	21-59 [2]
21.3	MSC Module Implementation	21-62 [2]
21.3.1	Interface Connections of the MSC Modules	21-62 [2]
21.3.2	MSC0/MSC1 Module-Related External Registers	21-64 [2]
21.3.3	Clock Control	21-65 [2]
21.3.3.1	Clock Control Register	21-67 [2]
21.3.3.2	Fractional Divider Register	21-68 [2]
21.3.4	Port Control	21-70 [2]
21.3.4.1	Input/Output Function Selection	21-70 [2]
21.3.4.2	Pad Driver Mode Registers	21-74 [2]
21.3.5	On-Chip Connections	21-76 [2]
21.3.5.1	EMGSTOPMSC Signal (from SCU)	21-76 [2]
21.3.5.2	ALTINH and ALTINL Connections	21-76 [2]
21.3.5.3	DMA Controller Service Requests	21-77 [2]
21.3.6	Interrupt Control Registers	21-78 [2]
22	Controller Area Network (MultiCAN) Controller	22-1 [2]
22.1	CAN Basics	22-2 [2]
22.1.1	Addressing and Bus Arbitration	22-2 [2]
22.1.2	CAN Frame Formats	22-3 [2]
22.1.2.1	Data Frames	22-3 [2]
22.1.2.2	Remote Frames	22-5 [2]
22.1.2.3	Error Frames	22-7 [2]
22.1.3	The Nominal Bit Time	22-8 [2]
22.1.4	Error Detection and Error Handling	22-9 [2]
22.1.5	Different CAN Implementations	22-10 [2]
22.1.6	TTCAN Basics	22-11 [2]
22.1.6.1	Time Reference	22-11 [2]
22.1.6.2	Basic Cycle	22-11 [2]
22.1.6.3	Global System Time	22-12 [2]
22.1.6.4	The System Matrix	22-12 [2]
22.1.6.5	Generation of the Network Time Unit (NTU)	22-13 [2]
22.1.6.6	Global Time Generation and Drift Correction	22-13 [2]
22.2	Overview	22-14 [2]
22.2.1	MultiCAN Module	22-15 [2]
22.2.2	Time-Triggered Extension (TTCAN)	22-17 [2]
22.3	MultiCAN Kernel Functional Description	22-18 [2]
22.3.1	Module Structure	22-18 [2]
22.3.2	Clock Control	22-20 [2]
22.3.3	Port Input Control	22-22 [2]

Table of Contents

22.3.4	Suspend Mode	22-22 [2]
22.3.5	CAN Node Control	22-23 [2]
22.3.5.1	Bit Timing Unit	22-24 [2]
22.3.5.2	Bitstream Processor	22-25 [2]
22.3.5.3	Error Handling Unit	22-26 [2]
22.3.5.4	CAN Frame Counter	22-27 [2]
22.3.5.5	CAN Node Interrupts	22-27 [2]
22.3.6	Message Object List Structure	22-29 [2]
22.3.6.1	Basics	22-29 [2]
22.3.6.2	List of Unallocated Elements	22-30 [2]
22.3.6.3	Connection to the CAN Nodes	22-30 [2]
22.3.6.4	List Command Panel	22-31 [2]
22.3.7	CAN Node Analysis Features	22-34 [2]
22.3.7.1	Analyze Mode	22-34 [2]
22.3.7.2	Loop-Back Mode	22-34 [2]
22.3.7.3	Bit Timing Analysis	22-35 [2]
22.3.8	Message Acceptance Filtering	22-37 [2]
22.3.8.1	Receive Acceptance Filtering	22-37 [2]
22.3.8.2	Transmit Acceptance Filtering	22-38 [2]
22.3.9	Message Postprocessing	22-40 [2]
22.3.9.1	Message Object Interrupts	22-40 [2]
22.3.9.2	Pending Messages	22-42 [2]
22.3.10	Message Object Data Handling	22-44 [2]
22.3.10.1	Frame Reception	22-44 [2]
22.3.10.2	Frame Transmission	22-47 [2]
22.3.11	Message Object Functionality	22-50 [2]
22.3.11.1	Standard Message Object	22-50 [2]
22.3.11.2	Single Data Transfer Mode	22-50 [2]
22.3.11.3	Single Transmit Trial	22-50 [2]
22.3.11.4	Message Object FIFO Structure	22-51 [2]
22.3.11.5	Receive FIFO	22-53 [2]
22.3.11.6	Transmit FIFO	22-54 [2]
22.3.11.7	Gateway Mode	22-55 [2]
22.3.11.8	Foreign Remote Requests	22-57 [2]
22.4	MultiCAN Kernel Registers	22-58 [2]
22.4.1	Module Identification Register	22-61 [2]
22.4.2	Global Module Register	22-62 [2]
22.4.3	CAN Node Registers	22-73 [2]
22.4.4	Message Object Registers	22-91 [2]
22.5	Time-Triggered Extension (TTCAN Controller)	22-114 [2]
22.5.1	Generation of the Local Time	22-114 [2]
22.5.2	Automatic TUR Adjust	22-115 [2]
22.5.3	Cycle Time	22-115 [2]

Table of Contents

22.5.3.1	Local Time and Synchronization Marks	22-115 [2]
22.5.3.2	Time Marks	22-116 [2]
22.5.3.3	Watch Trigger	22-117 [2]
22.5.4	Master Reference Mark (Level 2 only)	22-117 [2]
22.5.5	Transmit Enable Window	22-117 [2]
22.5.6	Local Offset and Global Time	22-118 [2]
22.5.7	Transmit Trigger	22-119 [2]
22.5.8	Reference Message	22-120 [2]
22.5.8.1	Differences to Normal CAN Messages	22-120 [2]
22.5.8.2	Transmit Trigger for a Reference Message	22-121 [2]
22.6	TTCAN Scheduler	22-123 [2]
22.6.1	Overview	22-123 [2]
22.6.2	Scheduler Memory	22-124 [2]
22.6.2.1	Scheduler Entry Types	22-125 [2]
22.6.2.2	Scheduler Entry Type Description	22-126 [2]
22.6.2.3	End of Scheduler Memory Entry	22-141 [2]
22.6.3	Setup of the Scheduler Entries	22-142 [2]
22.6.4	Reading the Scheduler Entries	22-142 [2]
22.6.4.1	Instructions During a Basic Cycle	22-142 [2]
22.6.4.2	Instructions at the End of a Basic Cycle	22-143 [2]
22.6.5	Scheduler Instruction Sequence	22-145 [2]
22.6.5.1	BCC and CSM	22-145 [2]
22.6.5.2	General Instruction Sequence Rules	22-146 [2]
22.6.5.3	Scheduler Sequence Example	22-147 [2]
22.7	TTCAN Operation	22-148 [2]
22.7.1	Configuration	22-148 [2]
22.7.2	Configuration Error	22-148 [2]
22.7.3	Synchronization Phase	22-149 [2]
22.7.4	Time Masters	22-149 [2]
22.7.4.1	State of a Time Master	22-149 [2]
22.7.4.2	Strictly Time-Triggered Behavior	22-149 [2]
22.7.5	Error Handling	22-150 [2]
22.7.6	Application Watchdog	22-150 [2]
22.7.7	MSC Handling	22-151 [2]
22.7.8	TTCAN Interrupt Control	22-152 [2]
22.8	TTCAN Registers	22-154 [2]
22.8.1	TTCAN Timing Registers	22-157 [2]
22.8.2	TTCAN Control / Status / Configuration Registers	22-169 [2]
22.8.3	Scheduler Registers	22-192 [2]
22.9	MultiCAN Module Implementation	22-199 [2]
22.9.1	Interfaces of the MultiCAN Module	22-199 [2]
22.9.2	MultiCAN Module External Registers	22-200 [2]
22.9.3	Module Clock Generation	22-201 [2]

Table of Contents

22.9.3.1	Clock Control Registers	22-202 [2]
22.9.4	Port and I/O Line Control	22-205 [2]
22.9.4.1	Input/Output Function Selection in Ports	22-205 [2]
22.9.4.2	CAN Related Input/Output Control Registers	22-206 [2]
22.9.4.3	Node Receive Input Selection	22-207 [2]
22.9.4.4	Pad Output Driver Register	22-208 [2]
22.9.4.5	External CAN Time Trigger Inputs	22-209 [2]
22.9.4.6	DMA Request Outputs	22-209 [2]
22.9.5	Interrupt Control	22-210 [2]
22.9.5.1	Service Request Control Registers	22-212 [2]
22.9.6	Parity Protection for CAN Memories	22-213 [2]
22.9.6.1	CAN Module Register Map	22-214 [2]
23	Micro Link Interface (MLI)	23-1 [2]
23.1	Functional Description	23-2 [2]
23.1.1	General Introduction	23-2 [2]
23.1.1.1	MLI Overview	23-2 [2]
23.1.1.2	Naming Conventions	23-3 [2]
23.1.1.3	MLI Communication Principles	23-6 [2]
23.1.2	MLI Frame Structure	23-10 [2]
23.1.2.1	General Frame Layout	23-11 [2]
23.1.2.2	Copy Base Address Frame	23-12 [2]
23.1.2.3	Write Offset and Data Frame	23-13 [2]
23.1.2.4	Optimized Write Frame	23-14 [2]
23.1.2.5	Discrete Read Frame	23-15 [2]
23.1.2.6	Optimized Read Frame	23-16 [2]
23.1.2.7	Command Frame	23-17 [2]
23.1.2.8	Answer Frame	23-18 [2]
23.1.3	Handshake Description	23-19 [2]
23.1.3.1	Handshake Signals	23-21 [2]
23.1.3.2	Error-free Handshake	23-21 [2]
23.1.3.3	Ready Delay Time	23-22 [2]
23.1.3.4	Non-Acknowledge Error	23-23 [2]
23.1.4	Parity Generation	23-24 [2]
23.1.5	Address Prediction	23-24 [2]
23.2	Module Kernel Description	23-25 [2]
23.2.1	Frame Handling	23-25 [2]
23.2.1.1	Copy Base Address Frame	23-26 [2]
23.2.1.2	Write/Data Frames	23-28 [2]
23.2.1.3	Read Frames	23-32 [2]
23.2.1.4	Answer Frame	23-37 [2]
23.2.1.5	Command Frame	23-39 [2]
23.2.2	General MLI Features	23-42 [2]

Table of Contents

23.2.2.1	Parity Check and Parity Error Indication	23-42 [2]
23.2.2.2	Non-Acknowledge Error	23-45 [2]
23.2.2.3	Address Prediction	23-45 [2]
23.2.2.4	Automatic Data Mode	23-46 [2]
23.2.2.5	Memory Access Protection	23-47 [2]
23.2.2.6	Transmit Priority	23-48 [2]
23.2.2.7	Transmission Delay	23-48 [2]
23.2.3	Interface Description	23-49 [2]
23.2.3.1	Transmitter I/O Line Control	23-51 [2]
23.2.3.2	Receiver I/O Line Control	23-51 [2]
23.2.3.3	Connecting Several MLI Modules	23-53 [2]
23.2.4	MLI Service Request Generation	23-55 [2]
23.2.5	Transmitter Events	23-57 [2]
23.2.5.1	Parity/Time-out Error Event	23-58 [2]
23.2.5.2	Normal Frame Sent x Event	23-58 [2]
23.2.5.3	Command Frame Sent Events	23-59 [2]
23.2.6	Receiver Events	23-60 [2]
23.2.6.1	Discarded Read Answer Event	23-60 [2]
23.2.6.2	Memory Access Protection/Parity Error Event	23-61 [2]
23.2.6.3	Normal Frame Received/Move Engine Terminated Event	23-62 [2]
23.2.6.4	Interrupt Command Frame Event	23-63 [2]
23.2.6.5	Command Frame Received Event	23-64 [2]
23.2.7	Baud Rate Generation	23-65 [2]
23.2.8	Automatic Register Overwrite	23-66 [2]
23.3	Operating the MLI	23-67 [2]
23.3.1	Connection Setup	23-68 [2]
23.3.2	Local Transmitter and Pipe Setup	23-69 [2]
23.3.3	Remote Receiver Setup	23-69 [2]
23.3.4	Remote Transmitter and Local Receiver Setup	23-70 [2]
23.3.5	Delay Adjustment	23-71 [2]
23.3.6	Connection to DMA Mechanism	23-73 [2]
23.3.7	Connection of MLI to SPI	23-73 [2]
23.4	MLI Kernel Registers	23-75 [2]
23.4.1	General Module Registers	23-78 [2]
23.4.2	General Status/Control Registers	23-81 [2]
23.4.3	Access Protection Registers	23-88 [2]
23.4.4	Transmitter Status/Control Registers	23-90 [2]
23.4.5	Transmitter Data/Address Registers	23-101 [2]
23.4.6	Transmitter Interrupt Registers	23-105 [2]
23.4.7	Receiver Status/Control Registers	23-110 [2]
23.4.8	Receiver Data/Address Registers	23-114 [2]
23.4.9	Receiver Interrupt Registers	23-117 [2]
23.5	Implementation of the MLI0/MLI1 in TC1796	23-124 [2]

Table of Contents

23.5.1	Interfaces of the MLI Modules	23-124 [2]
23.5.2	MLI Module External Registers	23-127 [2]
23.5.2.1	Automatic Register Overwrite	23-127 [2]
23.5.3	Module Clock Generation	23-128 [2]
23.5.4	Port Control and Connections	23-130 [2]
23.5.4.1	Input/Output Function Selection	23-130 [2]
23.5.4.2	Input/Output Control Register	23-133 [2]
23.5.4.3	Pad Driver Characteristics Selection	23-137 [2]
23.5.5	On-Chip Connections	23-139 [2]
23.5.5.1	Service Request Output Connections	23-139 [2]
23.5.5.2	Break Signals	23-140 [2]
23.5.6	Access Protection	23-141 [2]
23.5.6.1	Fixed Address Range Definition	23-141 [2]
23.5.6.2	Programmable Address Sub-Range Definitions	23-143 [2]
23.5.7	MLI0/MLI1 Transfer Window Address Maps	23-146 [2]
24	General Purpose Timer Array (GPTA)	24-1 [2]
24.1	GPTA Overview	24-2 [2]
24.1.1	Functionality of GPTA0 and GPTA1	24-3 [2]
24.1.2	Functionality of LTCA2	24-5 [2]
24.2	GPTA0/GPTA1 Kernel Description	24-6 [2]
24.2.1	GTPA Units	24-7 [2]
24.2.2	Clock Generation Unit	24-8 [2]
24.2.2.1	Filter and Prescaler Cell (FPC)	24-10 [2]
24.2.2.2	Phase Discrimination Logic (PDL)	24-19 [2]
24.2.2.3	Duty Cycle Measurement Unit (DCM)	24-24 [2]
24.2.2.4	Digital Phase Locked Loop Cell (PLL)	24-28 [2]
24.2.2.5	Clock Distribution Unit (CDU)	24-35 [2]
24.2.3	Signal Generation Unit	24-37 [2]
24.2.3.1	Global Timers (GT)	24-37 [2]
24.2.3.2	Global Timer Cell (GTC)	24-55 [2]
24.2.3.3	Local Timer Cell (LTC00 to LTC62)	24-67 [2]
24.2.3.4	Local Timer Cell LTC63	24-79 [2]
24.2.3.5	LTC Application Examples	24-86 [2]
24.2.4	Input/Output Line Sharing Unit (IOLS)	24-90 [2]
24.2.4.1	FPC Input Line Selection	24-94 [2]
24.2.4.2	GTC and LTC Output Multiplexer Selection	24-95 [2]
24.2.4.3	GTC Input Multiplexer Selection	24-100 [2]
24.2.4.4	LTC Input Multiplexer Selection	24-105 [2]
24.2.4.5	Multiplexer Register Array Programming	24-110 [2]
24.2.5	Interrupt Sharing Unit (IS)	24-112 [2]
24.2.6	Pseudo Code Description of GPTA Kernel Functionality	24-115 [2]
24.2.6.1	FPC Algorithm	24-115 [2]

Table of Contents

24.2.6.2	PDL-Algorithm	24-121 [2]
24.2.6.3	DCM-Algorithm	24-125 [2]
24.2.6.4	PLL-Algorithm	24-128 [2]
24.2.6.5	GT-Algorithm	24-130 [2]
24.2.6.6	GTC-Algorithm	24-131 [2]
24.2.6.7	LTC-Algorithm for Cells 0 to 62	24-137 [2]
24.2.6.8	LTC Algorithm for Cell 63	24-144 [2]
24.2.7	Programming of a GPTA Module	24-148 [2]
24.3	GPTA0/1 Kernel Registers	24-150 [2]
24.3.1	Identification Register	24-154 [2]
24.3.2	FPC Registers	24-155 [2]
24.3.3	Phase Discriminator Registers	24-159 [2]
24.3.4	Duty Cycle Measurement Registers	24-161 [2]
24.3.5	Digital Phase Locked Loop Registers	24-165 [2]
24.3.6	Global Timer Registers	24-170 [2]
24.3.7	Clock Bus Register	24-173 [2]
24.3.8	Global Timer Cell Registers	24-175 [2]
24.3.9	Local Timer Cell Registers	24-180 [2]
24.3.10	I/O Sharing Unit Registers	24-191 [2]
24.3.11	Multiplexer Control Registers	24-194 [2]
24.3.12	Service Request Registers	24-206 [2]
24.4	LTCA2 Kernel Description	24-217 [2]
24.4.1	Local Timer Cell (LTC00 to LTC62)	24-218 [2]
24.4.2	Input/Output Line Sharing Unit (IOLS)	24-218 [2]
24.4.2.1	Output Multiplexer	24-221 [2]
24.4.2.2	LTC Input Multiplexing Scheme	24-225 [2]
24.4.2.3	Multiplexer Register Array Programming	24-229 [2]
24.4.3	Interrupt Sharing Unit (IS)	24-231 [2]
24.5	LTCA2 Kernel Registers	24-233 [2]
24.5.1	Bit Protection	24-234 [2]
24.5.2	Service Request Registers	24-234 [2]
24.5.3	Local Timer Cell Registers	24-234 [2]
24.5.4	Module Identification Register	24-235 [2]
24.5.5	I/O Sharing Unit Registers	24-236 [2]
24.5.6	Multiplexer Control Registers	24-239 [2]
24.5.6.1	Output Multiplexer Control Registers	24-239 [2]
24.5.6.2	LTC Input Multiplexer Control Registers	24-243 [2]
24.6	GPTA Module Implementation	24-247 [2]
24.6.1	Interconnections of the GPTA0/GPTA1/LTCA2 Modules	24-247 [2]
24.6.2	GPTA Module External Registers	24-249 [2]
24.6.3	Port Control and Connections	24-250 [2]
24.6.3.1	I/O Group to Port Line Assignment	24-250 [2]
24.6.3.2	Input/Output Function Selection	24-251 [2]

Table of Contents

24.6.3.3	Pad Driver Characteristics Selection	24-253 [2]
24.6.3.4	Emergency Control of GPTA Output Ports Lines	24-255 [2]
24.6.4	On-Chip Connections	24-256 [2]
24.6.4.1	Clock Bus Connections	24-256 [2]
24.6.4.2	MSC Controller Connections	24-257 [2]
24.6.4.3	GPTA Connections with SCU, MultiCAN, FADC, DMA, Ports	24-263 [2]
24.6.5	Module Clock Generation	24-265 [2]
24.6.5.1	Clock Control Registers	24-267 [2]
24.6.6	Limits of Cascading GTCs and LTCs	24-272 [2]
24.6.7	Interrupt Registers	24-273 [2]
24.6.8	GPTA Register Address Map	24-274 [2]
25	Analog-to-Digital Converter (ADC)	25-1 [2]
25.1	ADC Kernel Description	25-1 [2]
25.1.1	Analog Input Connections	25-4 [2]
25.1.2	Conversion Request Sources	25-5 [2]
25.1.2.1	Parallel Conversion Request Sources	25-5 [2]
25.1.2.2	Sequential Conversion Request Sources	25-6 [2]
25.1.2.3	Parallel Conversion Request Source "Timer"	25-8 [2]
25.1.2.4	Parallel Conversion Request Source "External Event"	25-11 [2]
25.1.2.5	Parallel Conversion Request Source "Software"	25-14 [2]
25.1.2.6	Parallel Conversion Request Source "Auto-Scan"	25-15 [2]
25.1.2.7	Sequential Conversion Request Source "Channel Injection"	25-21 [2]
25.1.2.8	Sequential Conversion Request Source "Queue"	25-25 [2]
25.1.3	Conversion Request Arbitration	25-29 [2]
25.1.3.1	Source Arbitration Level	25-30 [2]
25.1.3.2	Arbitration Participation Flags	25-30 [2]
25.1.3.3	Cancel Functionality	25-31 [2]
25.1.3.4	Clear of Pending Conversion Requests	25-31 [2]
25.1.3.5	Arbitration and Synchronized Injection	25-32 [2]
25.1.3.6	Arbitration Lock	25-32 [2]
25.1.4	Clock Circuit	25-33 [2]
25.1.4.1	Conversion Principles	25-34 [2]
25.1.4.2	Conversion Timing Control (CTC and CPS)	25-34 [2]
25.1.4.3	Sample Timing Control	25-35 [2]
25.1.4.4	Power-Up Calibration Time	25-36 [2]
25.1.5	Reference Voltages (V_{AREF} and V_{AGND})	25-37 [2]
25.1.6	Error through Overload Conditions	25-38 [2]
25.1.7	Limit Checking	25-39 [2]
25.1.8	Expansion of Analog Channels	25-41 [2]
25.1.8.1	Inverse Current Injection (Overload) Behavior	25-42 [2]
25.1.8.2	On Resistance of the External Multiplexer	25-42 [2]
25.1.8.3	Timing of the External Multiplexer	25-42 [2]

Table of Contents

25.1.8.4	Load Capacitance	25-42 [2]
25.1.9	Service Request Processing	25-43 [2]
25.1.9.1	Channel Request Source Control	25-45 [2]
25.1.9.2	Parallel/Serial Request Source Control	25-46 [2]
25.1.9.3	Service Request Compressor	25-47 [2]
25.1.9.4	Service Request Flag Control	25-48 [2]
25.1.10	Synchronization of Two ADC Modules	25-49 [2]
25.1.10.1	Synchronized Injection Mode	25-50 [2]
25.1.10.2	Status Information During Synchronized Conversion	25-51 [2]
25.1.10.3	Master-Slave Functionality for Synchronized Injection	25-51 [2]
25.1.10.4	Conversion Timing during Synchronized Conversion	25-54 [2]
25.1.10.5	Service Request Generation in Synchronized Injection	25-54 [2]
25.1.10.6	Example for Synchronized Injection	25-55 [2]
25.2	ADC0/ADC1 Kernel Registers	25-57 [2]
25.2.1	Identification Register	25-59 [2]
25.2.2	Channel Registers	25-60 [2]
25.2.3	Timer Registers	25-65 [2]
25.2.4	Queue Registers	25-69 [2]
25.2.5	External Trigger Registers	25-73 [2]
25.2.6	Auto-Scan Registers	25-75 [2]
25.2.7	Other Control/Status Registers	25-78 [2]
25.2.8	Channel Inject Register	25-90 [2]
25.2.9	Software Request Registers	25-92 [2]
25.2.10	Service Request Registers	25-94 [2]
25.3	Implementation of ADC0/ADC1	25-100 [2]
25.3.1	Interface Connections of the ADC Modules	25-100 [2]
25.3.2	ADC0/ADC1 Module Related External Registers	25-102 [2]
25.3.3	Clock Control	25-103 [2]
25.3.3.1	ADC Clock Control Register	25-105 [2]
25.3.3.2	ADC Fractional Divider Register	25-106 [2]
25.3.4	Port Control	25-108 [2]
25.3.4.1	Input/Output Control Register	25-109 [2]
25.3.4.2	Pad Output Driver Characteristics Selection	25-110 [2]
25.3.5	Analog Input Line to Analog Input Channel Connections	25-111 [2]
25.3.6	On-Chip Connections	25-113 [2]
25.3.6.1	Reference Voltage Selection	25-113 [2]
25.3.6.2	Request/Gating Input Signal Connections	25-114 [2]
25.3.6.3	Service Request Output Lines	25-119 [2]
25.3.6.4	Service Request Control Registers	25-120 [2]
25.3.6.5	Die Temperature Sensor	25-120 [2]
26	Fast Analog-to-Digital Converter (FADC)	26-1 [2]
26.1	FADC Kernel Description	26-2 [2]

Table of Contents

26.1.1	Analog Inputs	26-4 [2]
26.1.1.1	Analog Input Stage Configurations	26-5 [2]
26.1.2	Conversion Timing	26-8 [2]
26.1.3	Channel Triggers	26-9 [2]
26.1.4	Channel Timer	26-11 [2]
26.1.5	Control Logic	26-12 [2]
26.1.5.1	Conversion Control	26-12 [2]
26.1.5.2	Static Channel Priority	26-12 [2]
26.1.5.3	Dynamic Priority Assignment	26-12 [2]
26.1.5.4	Clock Generation	26-13 [2]
26.1.5.5	Suspend Mode Behavior	26-13 [2]
26.1.6	Data Reduction Unit	26-13 [2]
26.1.6.1	Filter Block Structure	26-15 [2]
26.1.6.2	Filter Block Operation	26-16 [2]
26.1.6.3	Filter Concatenation	26-19 [2]
26.1.6.4	Width of Result Registers	26-20 [2]
26.1.7	Neighbor Channel Trigger	26-21 [2]
26.1.8	Offset and Gain Calibration	26-22 [2]
26.1.8.1	Offset Calibration	26-23 [2]
26.1.8.2	Gain Calibration	26-23 [2]
26.1.9	Interrupt Generation	26-24 [2]
26.2	FADC Kernel Registers	26-27 [2]
26.2.1	Identification Register	26-29 [2]
26.2.2	Global Registers	26-30 [2]
26.2.3	Channel Registers	26-41 [2]
26.2.4	Filter Registers	26-48 [2]
26.3	Implementation of FADC	26-56 [2]
26.3.1	Interfaces of the FADC Module	26-56 [2]
26.3.2	FADC Module Related External Registers	26-57 [2]
26.3.3	Clock Control	26-58 [2]
26.3.3.1	Clock Control Registers	26-59 [2]
26.3.4	Port Control	26-61 [2]
26.3.5	Interrupt Control	26-63 [2]
26.3.6	On-Chip Connections	26-64 [2]
26.3.6.1	Analog Input Lines	26-64 [2]
26.3.6.2	Trigger/Gating Source Input Connections	26-64 [2]
26.3.6.3	Service Request Lines	26-65 [2]
	Keyword Index	L-1 [1+2]
	Register Index	L-16 [1+2]

1 Introduction

This User's Manual describes the Infineon TC1796, a 32-bit microcontroller DSP based on the Infineon TriCore Architecture.

1.1 About this Document

This document is designed to be read primarily by design engineers and software engineers who need a detailed description of the interactions of the TC1796 functional units, registers, instructions, and exceptions.

This TC1796 User's Manual describes the features of the TC1796 with respect to the TriCore Architecture. Where the TC1796 directly implements TriCore architectural functions, this manual simply refers to those functions as features of the TC1796. In all cases where this manual describes a TC1796 feature without referring to the TriCore Architecture, this means that the TC1796 is a direct implementation of the TriCore Architecture.

Where the TC1796 implements a subset of TriCore architectural features, this manual describes the TC1796 implementation, and then describes how it differs from the TriCore Architecture. For example, where the TriCore Architecture specifies up to four Memory Protection Register Sets, the TC1796 implements but two. Such differences between the TC1796 and the TriCore Architecture are documented in the text covering each such subject.

1.1.1 Related Documentations

A complete description of the TriCore architecture is found in the document entitled "TriCore Architecture Manual". The architecture of the TC1796 is described separately this way because of the configurable nature of the TriCore specification: Different versions of the architecture may contain a different mix of systems components. The TriCore architecture, however, remains constant across all derivative designs in order to preserve compatibility.

In addition to this "TC1796 System Units (Vol. 1 of 2) User's Manual", a second document, the "TC1796 Peripheral Units (Vol. 2 of 2) User's Manual", is available. These two User's Manuals together with the "TriCore Architecture Manual" are required to understand the complete TC1796 microcontroller functionality.

Implementation-specific details of the TC1796 such as electrical characteristics and timing parameters are defined in the "TC1796 Data Sheet".

1.1.2 Text Conventions

This document uses the following text conventions for named components of the TC1796:

- Functional units of the TC1796 are given in plain UPPER CASE. For example: “The EBU provides an interface to external peripherals”.
- Pins using negative logic are indicated by an overline. For example: “The external reset pin, $\overline{\text{HDRST}}$, has a dual function.”.
- Bit fields and bits in registers are in general referenced as “Register name.Bit field” or “Register name.Bit”. For example: “The Current CPU Priority Number bit field ICR.CCPN is cleared”. Most of the register names contain a module name prefix, separated by a underscore character “_” from the real register name (for example, “ASC0_CON”, where “ASC0” is the module name prefix, and “CON” is the kernel register name). In chapters describing the kernels of the peripheral modules, the registers are mainly referenced with their kernel register names. The peripheral module implementation sections mainly refer to the real register names with module prefixes.
- Variables used to describe sets of processing units or registers appear in mixed upper-and-lower-case font. For example, register name “MSGCFGn” refers to multiple “MSGCFG” registers with variable n. The bounds of the variables are always given where the register expression is first used (for example, “n = 31-0”), and are repeated as needed in the rest of the text.
- The default radix is decimal. Hexadecimal constants are suffixed with a subscript letter “H”, as in 100_H. Binary constants are suffixed with a subscript letter “B”, as in 111_B.
- When the extent of register fields, groups of signals, or groups of pins are collectively named in the body of the document, they are given as “NAME[A:B]”, which defines a range for the named group from B to A. Individual bits, signals, or pins are given as “NAME[C]” where the range of the variable C is given in the text. For example: CFG[2:0], and SRPN[0].
- Units are abbreviated as follows:
 - **MHz** = Megahertz
 - **μs** = Microseconds
 - **kBaud, kbit** = 1000 characters/bits per second
 - **MBaud, Mbit** = 1,000,000 characters/bits per second
 - **Kbyte** = 1024 bytes of memory
 - **Mbyte** = 1048576 bytes of memoryIn general, the k prefix scales a unit by 1000 whereas the K prefix scales a unit by 1024. Hence, the Kbyte unit scales the expression preceding it by 1024. The kBaud unit scales the expression preceding it by 1000. The M prefix scales by 1,000,000 or 1048576, and μ scales by .000001. For example, 1 Kbyte is 1024 bytes, 1 Mbyte is 1024 x 1024 bytes, 1 kBaud/kbit are 1000 characters/bits per second, 1 MBaud/Mbit are 1000000 characters/bits per second, and 1 MHz is 1,000,000 Hz.
- Data format quantities are defined as follows:
 - **Byte** = 8-bit quantity
 - **Half-word** = 16-bit quantity

- **Word** = 32-bit quantity
- **Double-word** = 64-bit quantity

1.1.3 Reserved, Undefined, and Unimplemented Terminology

In tables where register bit fields are defined, the following conventions are used to indicate undefined and unimplemented functions. Furthermore, types of bits and bit fields are defined using the abbreviations as shown in [Table 1-1](#).

Table 1-1 Bit Function Terminology

Function of Bits	Description
Unimplemented, Reserved	<p>Register bit fields named 0 indicate unimplemented functions with the following behavior.</p> <ul style="list-style-type: none"> • Reading these bit fields returns 0. • Writing these bit fields has no effect. <p>These bit fields are reserved. When writing, software should always set such bit fields to 0 in order to preserve compatibility with future products.</p>
Undefined, Reserved	<p>Certain bit combinations in a bit field can be marked “Reserved”, indicating that the behavior of the TC1796 is undefined for that combination of bits. Setting the register to such undefined bit or bit field combinations may lead to unpredictable results. Such bit combinations are reserved. When writing, software must always set such bit fields to legal values as defined for it.</p>
rw	The bit or bit field can be read and written.
rwh	As rw, but bit or bit field can be also set or cleared by hardware.
r	The bit or bit field can only be read (read-only).
w	The bit or bit field can only be written (write-only).
rh	The bit or bit field can only be read. It can be also set or cleared by hardware (typical example: status flags).
s	Bits with this attribute are “sticky” in one direction. If their reset value is once overwritten by software, they can be switched again into their reset state only by a reset operation. Software cannot switch this type of bit into its reset state by writing the register. This attribute can be combined to ‘rws’ or ‘rwhs’.
f	Bits with this attribute are readable only when they are accessed by an instruction fetch. Normal data read operations will return other values.

1.1.4 Register Access Modes

Read and write access to registers and memory locations are sometimes restricted. In memory and register access tables, the terms as defined in [Table 1-2](#) are used.

Table 1-2 Access Terms

Symbol	Description
U	Access Mode: Access permitted in User Mode 0 or 1.
	Reset Value: Value or bit is not changed by a reset operation.
SV	Access permitted in Supervisor Mode.
R	Read-only register.
32	Only 32-bit word accesses are permitted to this register/address range.
E	Endinit-protected register/address.
PW	Password-protected register/address.
NC	No change, indicated register is not changed.
BE	Indicates that an access to this address range generates a Bus Error.
nBE	Indicates that no Bus Error is generated when accessing this address range, even though it is either an access to an undefined address or the access does not follow the given rules.
nE	Indicates that no Error is generated when accessing this address or address range, even though the access is to an undefined address or address range. True for CPU accesses (MTCR/MFCR) to undefined addresses in the CSFR range.
X	Undefined value or bit.

1.1.5 Abbreviations and Acronyms

The following acronyms and termini are used in this document:

ABM	Alternate Boot Mode
ADC	Analog-to-Digital Converter
AGPR	Address General Purpose Register
ALU	Arithmetic and Logic Unit
ASC	Asynchronous/Synchronous Serial Controller
BCU	Bus Control Unit
BROM	Boot ROM & Test ROM
BSL	Bootstrap Loader
CAN	Controller Area Network
CMEM	PCP Code Memory
CISC	Complex Instruction Set Computing
CPS	CPU Slave Interface
CPU	Central Processing Unit
CSA	Context Save Area
CSFR	Core Special Function Register
DBCUC	Data Local Memory Bus Control Unit
DFLASH	Data Flash Memory
DGPR	Data General Purpose Register
DMA	Direct Memory Access
DMI	Data Memory Interface
DMU	Data Memory Unit
DPRAM	Dual-Port RAM
EBU	External Bus Unit
EMI	Electro-Magnetic Interference
FADC	Fast Analog-to-Digital Converter
FAM	Flash Array Module
FCS	Flash Command State Machine
FIM	Flash Interface and Control Module
FPI	Flexible Peripheral Interconnect (Bus)

FPU	Floating Point Unit
GPIO	General Purpose Input/Output
GPR	General Purpose Register
GPTA	General Purpose Timer Array
ICACHE	Instruction Cache
I/O	Input / Output
LDRAM	Local Data RAM
LFI	Local Memory-to-FPI Bus Interface
LMB	Local Memory Bus
LMI	Local Memory Interface
LSB	Least Significant Bit
LTCA	Local Timer Cell Array
MLI	Micro Link Interface
MMU	Memory Management Unit
MSB	Most Significant Bit
MSC	Micro Second Channel
NMI	Non-Maskable Interrupt
OCDS	On-Chip Debug Support
OVRAM	Code Overlay Memory
PBCU	Program Local Memory Bus Control Unit
PCP	Peripheral Control Processor
PFLASH	Program Flash Memory
PMI	Program Memory Interface
PMU	Program Memory Unit
PLL	Phase Locked Loop
PLMB	Program Local Memory Bus
PMU	Program Memory Unit
PRAM	PCP Parameter RAM
RAM	Random Access Memory
RBCU	Remote Peripheral Bus Control Unit
RISC	Reduced Instruction Set Computing
RPB	Remote Peripheral Bus

SBCU	System Peripheral Bus Control Unit
SBRAM	Stand-by Data Memory
SCU	System Control Unit
SFR	Special Function Register
SPB	System Peripheral Bus
SPRAM	Scratch-Pad RAM
SRAM	Static Data Memory
SRN	Service Request Node
SSC	Synchronous Serial Controller
STM	System Timer
WDT	Watchdog Timer

1.2 System Architecture of the TC1796

The TC1796 combines three powerful technologies within one silicon die, achieving new levels of power, speed, and economy for embedded applications:

- Reduced Instruction Set Computing (RISC) processor architecture
- Digital Signal Processing (DSP) operations and addressing modes
- On-chip memories and peripherals

DSP operations and addressing modes provide the computational power necessary to efficiently analyze complex real-world signals. The RISC load/store architecture provides high computational bandwidth with low system cost. On-chip memory and peripherals are designed to support even the most demanding high-bandwidth real-time embedded control-systems tasks.

Additional high-level features of the TC1796 include:

- Program Memory Unit – instruction memory and instruction cache
- Data Memory Unit – data memory
- Serial communication interfaces – flexible synchronous and asynchronous modes
- Peripheral Control Processor – standalone data operations and interrupt servicing
- DMA Controller – DMA operations and interrupt servicing
- General-purpose timers
- High-performance on-chip buses
- On-chip debugging and emulation facilities
- Flexible interconnections to external components
- Flexible power-management

The TC1796 is a high-performance microcontroller with TriCore CPU, program and data memories, buses, bus arbitration, an interrupt controller, a peripheral control processor and a DMA controller, several on-chip peripherals, and an external bus interface. The TC1796 is designed to meet the needs of the most demanding embedded control systems applications where the competing issues of price/performance, real-time responsiveness, computational power, data bandwidth, and power consumption are key design elements.

The TC1796 offers several versatile on-chip peripheral units such as serial controllers, timer units, and Analog-to-Digital converters. Within the TC1796, all these peripheral units are connected to the TriCore CPU/system via two Flexible Peripheral Interconnect (FPI) Buses. Several I/O lines on the TC1796 ports are reserved for these peripheral units to communicate with the external world.

1.2.1 TC1796 Block Diagram

Figure 1-1 shows the block diagram of the TC1796.

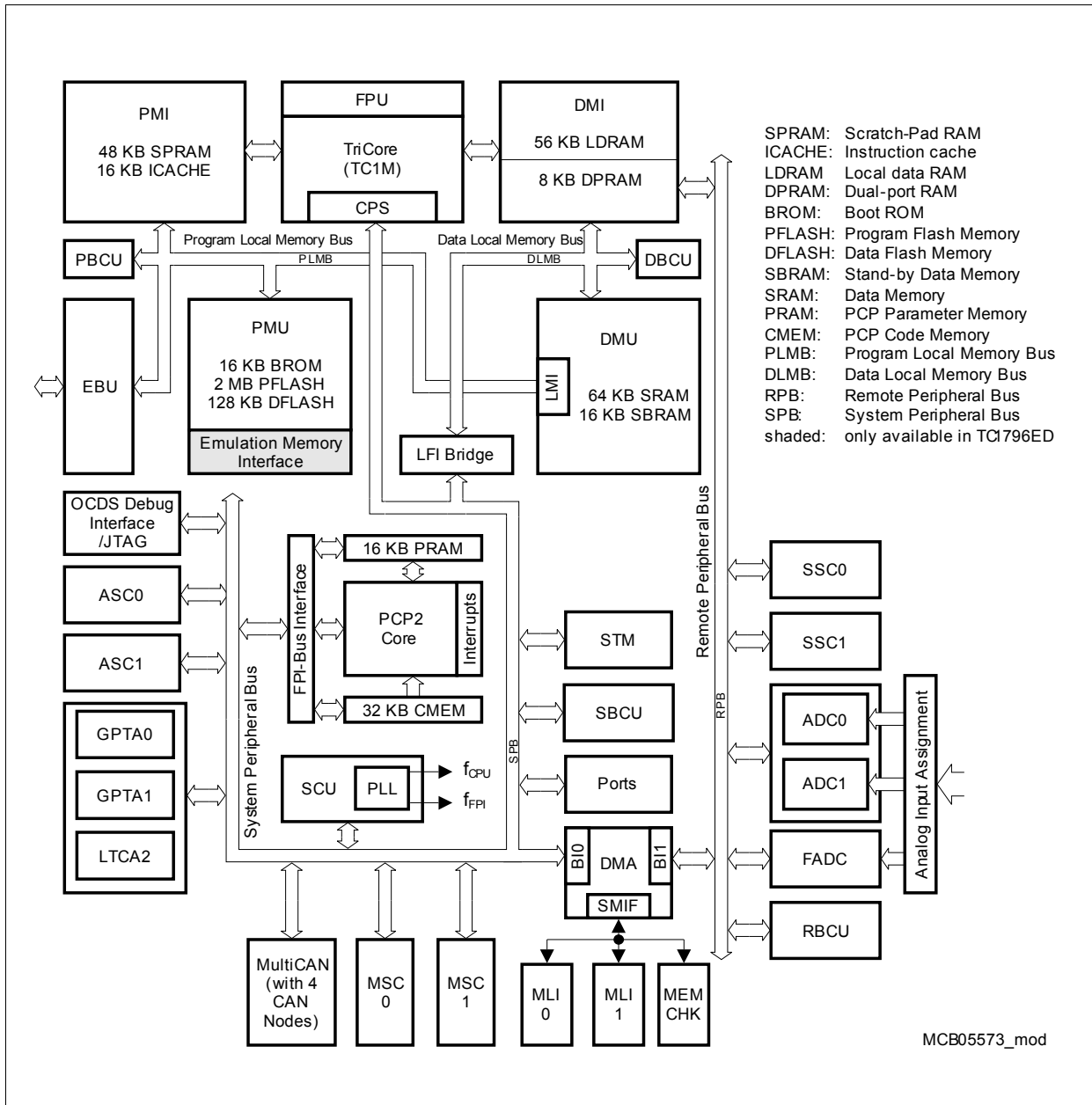


Figure 1-1 TC1796 Block Diagram

1.2.2 Features

The TC1796 has the following features:

High-performance 32-Bit CPU

- 32-bit architecture with 4 Gbyte unified data, program, and input/output address space
- Fast automatic context-switching
- Multiply-accumulate unit
- Floating point unit
- Saturating integer arithmetic
- Two high-performance on-chip peripheral buses (FPI Bus)
- Register-based design with multiple variable register banks
- Bit handling
- Packed data operations
- Zero overhead loop
- Precise exceptions
- Flexible power management

High-efficiency Instruction Set

- 16/32-bit instructions for reduced code size
- Data types include: Boolean, array of bits, character, signed and unsigned integer, integer with saturation, signed fraction, double-word integers, and IEEE-754 single-precision floating point
- Data formats include: Bit, 8-bit byte, 16-bit half-word, 32-bit word, and 64-bit double-word data formats
- Powerful instruction set
- Flexible and efficient addressing mode for high code density

External Bus Interface

- Programmable external bus interface for low-cost system implementation
- Glueless interface to a wide selection of external memories
- 8-/16-/32-bit data transfers
- Intel-style and Motorola-style peripheral/device support.
- Burst flash memory support
- Flexible address generation and access timing

Integrated On-Chip Memories

- Code memory
 - 2 Mbyte on-chip Program Flash (PFLASH)
 - 48 Kbyte Scratch-Pad RAM (SPRAM)

- 16 Kbyte Instruction Cache (ICACHE)
- 16 Kbyte Boot ROM (BROM)
- Data memory
 - 64 Kbyte Data Memory (SRAM)
 - 16 Kbyte data memory (SBRAM) for standby operation during power-down
 - 56 Kbyte Local Data RAM (LDRAM)
 - 8 Kbyte Dual-port RAM (DPRAM)
 - 128 Kbyte on-chip Data Flash (DFLASH)
- PCP
 - 32 Kbyte PCP Code Memory (CMEM)
 - 16 Kbyte PCP Data Memory (PRAM)
- On-chip SRAMs with parity error detection

Interrupt System

- Total of 181 Service Request Nodes (SRNs)
- Flexible interrupt-prioritizing scheme with 255 interrupt priority levels
- Fast interrupt response
- Service requests are serviced by CPU or PCP

Peripheral Control Processor (PCP)

- Data move between any two memory or I/O locations
- Data move until predefined limit reached supported
- Read-Modify-Write capabilities
- Full computation capabilities including basic MUL/DIV
- Read/move data and accumulate it to previously read data
- Read two data values and perform arithmetic or logical operation and store result
- Bit-handling capabilities (testing, setting, clearing)
- Flow-control instructions (conditional/unconditional jumps, breakpoint)

DMA Controller

- 16 independent DMA channels
- Programmable priority of the DMA sub-blocks on the bus interfaces
- Buffer capability for move actions on the buses (minimum of 1 move per bus is buffered).
- Individually programmable operation modes for each DMA channel
- Full 32-bit addressing capability of each DMA channel
- Programmable data width of DMA transfer/transaction: 8-bit, 16-bit, or 32-bit
- Micro Link bus interface support
- One register set for each DMA channel
- Flexible interrupt generation

- DMA Controller operates as bus bridge between System Peripheral Bus and Remote Peripheral Bus

Parallel I/O Ports

- 127 digital General-Purpose Input/Output (GPIO) port lines
- Input/output functionality individually programmable for each port line
- Programmable input characteristics (pull-up, pull-down, no pull device)
- Programmable output driver strength for EMI minimization (weak, medium, strong)
- Programmable output characteristics (push-pull, open drain)
- Programmable alternate output functions
- Output lines of each port can be updated port-wise or set/cleared/toggled bit-wise

On-Chip Peripheral Units

- Two Asynchronous/Synchronous Serial Channels (ASC) with baud-rate generator, parity, framing and overrun error detection
- Two Synchronous Serial Channels (SSC) with programmable data length and shift direction
- Two Micro Second Channel Interfaces (MSC) for serial communication
- One CAN Module with four CAN nodes (MultiCAN) for high-efficiency data handling via FIFO buffering and gateway data transfer
- Two Micro Link Serial Bus Interfaces (MLI) for serial multiprocessor communication
- Two General Purpose Timer Arrays (GPTA) with a powerful set of digital signal filtering and timer functionality to accomplish autonomous and complex input/output management
- One Local Timer Cell Array (LTCA) for signal generation purposes
- Two medium-speed Analog-to-Digital Converter Units (ADC) with 8-bit, 10-bit, or 12-bit resolution and sixteen analog inputs each
- One fast Analog-to-Digital Converter Unit (FADC)

Package

- P-BGA-416 package, 1 mm pitch

Temperature Ranges

- Ambient temperature: -40 ° to +125 °C
- Maximum junction temperature: +150 °C

Clock Frequencies

- Maximum CPU clock frequency: 150 MHz
- Maximum system clock frequency: 75 MHz

Complete Development Support

A variety of software and hardware development tools for the 32-bit microcontroller TC1796 are available from experienced international tool suppliers. The development environment for the Infineon 32-bit microcontroller includes the following tools:

- Embedded Development Environment for TriCore Products
- The TC1796 On-chip Debug Support (OCDS) provides a JTAG port for communication between external hardware and the system.
- Two Flexible Peripheral Interconnect Buses (FPI Bus) for on-chip interconnections and its FPI Bus control units (SBCU, RBCU).
- The System Timer (STM) with high-precision, long-range timing capabilities.
- The TC1796 includes a power management system, a watchdog timer as well as reset logic.

1.3 On-Chip Peripheral Units of the TC1796

The TC1796 microcontroller offers several versatile on-chip peripheral units such as serial controllers, timer units, and Analog-to-Digital converters. Several I/O lines on the TC1796 ports are reserved for these peripheral units to communicate with the external world.

The peripherals mentioned in this overview section are all described in detail in the chapters of the “TC1796 Peripheral Units (Vol. 2 of 2) User’s Manual” part.

1.3.1 Serial Interfaces

The TC1796 includes eight + four (CAN) serial peripheral interface units:

- Two Asynchronous/Synchronous Serial Interfaces (ASC0 and ASC1)
- Two high-speed Synchronous Serial Interfaces (SSC0 and SSC1)
- Two Micro Second Channel Interfaces (MSC0 and MSC1)
- Two Micro Link Serial Bus Interfaces (MLI0 and MLI1) dedicated for the communication between two controllers of the TriCore family.
- One MultiCAN Interface with 4 CAN nodes

1.3.1.1 Asynchronous/Synchronous Serial Interfaces

Figure 1-2 shows a global view of the Asynchronous/Synchronous Serial Interface (ASC)

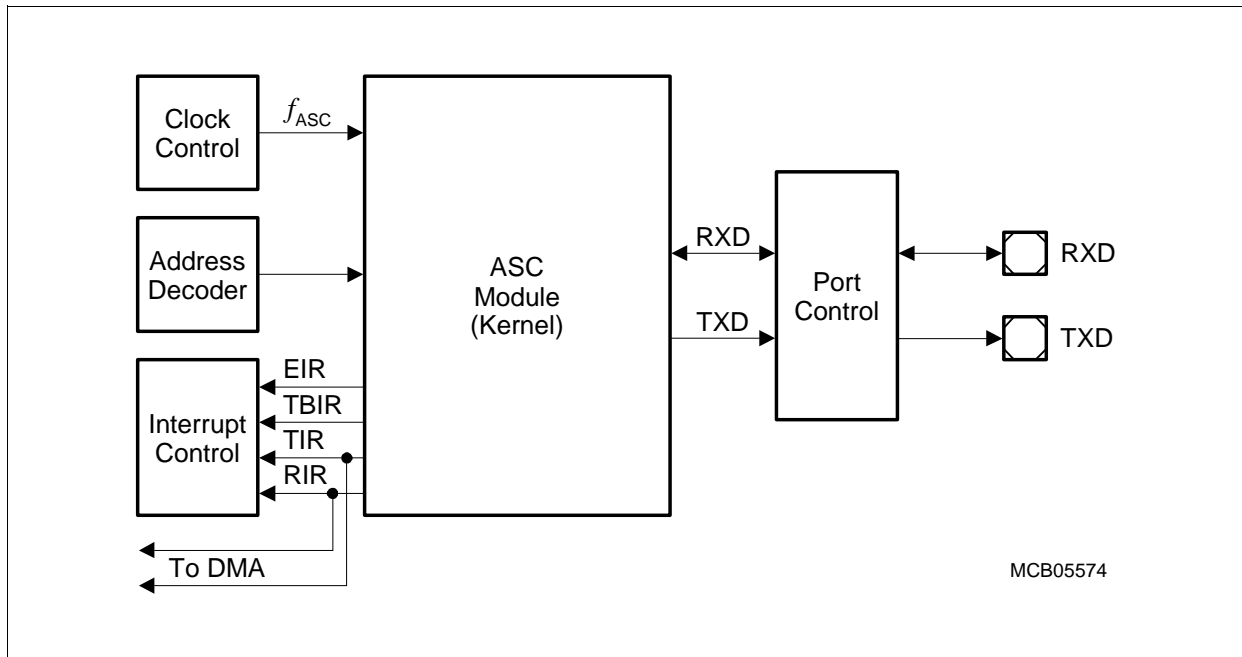


Figure 1-2 General Block Diagram of the ASC Interface

The ASC provides serial communication between the TC1796 and other microcontrollers, microprocessors, or external peripherals.

The ASC supports full-duplex asynchronous communication and half-duplex synchronous communication. In Synchronous Mode, data is transmitted or received synchronous to a shift clock that is generated by the ASC internally. In Asynchronous Mode, 8-bit or 9-bit data transfer, parity generation, and the number of stop bits can be selected. Parity, framing, and overrun error detection are provided to increase the reliability of data transfers. Transmission and reception of data is double-buffered. For multiprocessor communication, a mechanism is included to distinguish address bytes from data bytes. Testing is supported by a loop-back option. A 13-bit baud rate generator provides the ASC with a separate serial clock signal which can be very accurately adjusted by a prescaler implemented as a fractional divider.

Each ASC module, ASC0 and ASC1, communicates with the external world via two I/O lines. The RXD line is the receive data input signal (in Synchronous Mode also output). TXD is the transmit output signal. Clock control, address decoding, and interrupt service request control are managed outside the ASC module kernel.

Features

- Full-duplex asynchronous operating modes
 - 8-bit or 9-bit data frames, LSB first
 - Parity-bit generation/checking
 - One or two stop bits
 - Baud rate from 4.69 MBaud to 1.12 Baud (@ 75 MHz clock)
 - Multiprocessor mode for automatic address/data byte detection
 - Loop-back capability
- Half-duplex 8-bit synchronous operating mode
 - Baud rate from 9.38 Mbit/s to 763 Bit/s (@ 75 MHz clock)
- Double-buffered transmitter/receiver
- Interrupt generation
 - On a transmit buffer empty condition
 - On a transmit last bit of a frame condition
 - On a receive buffer full condition
 - On an error condition (frame, parity, overrun error)

1.3.1.2 High-Speed Synchronous Serial Interfaces

Figure 1-3 shows a global view of the Synchronous Serial Interface (SSC).

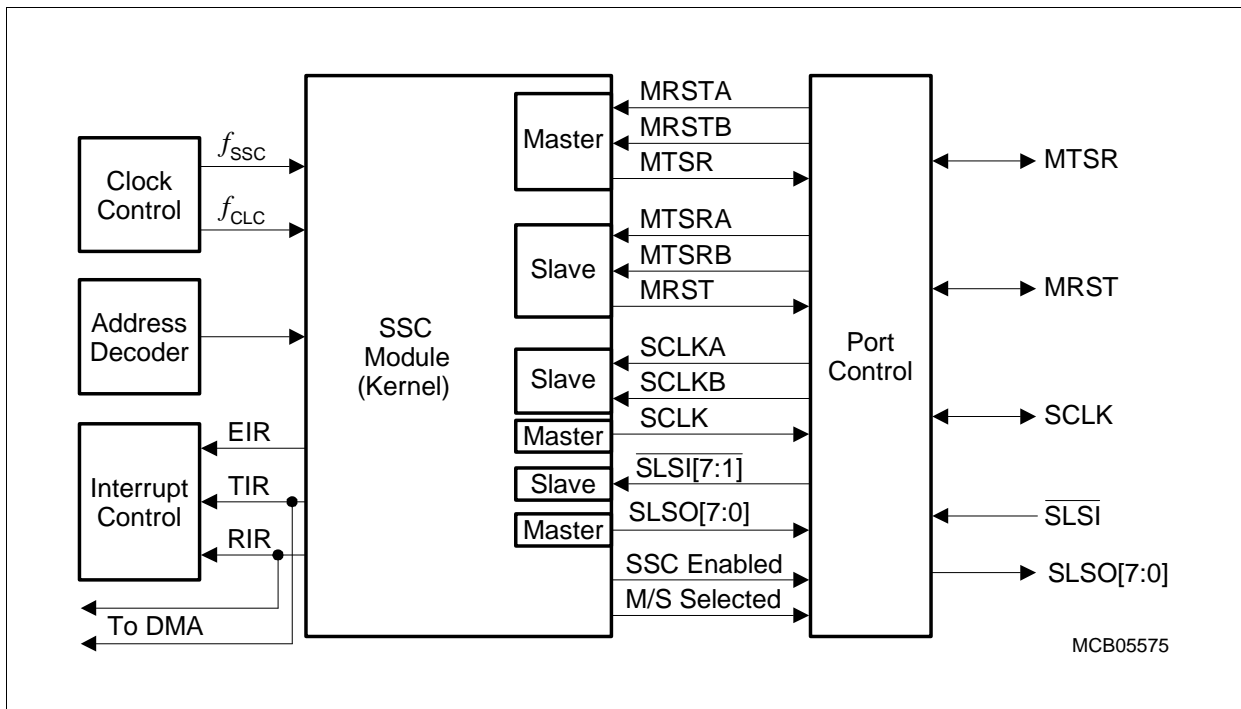


Figure 1-3 General Block Diagram of the SSC Interface

The SSC supports full-duplex and half-duplex serial synchronous communication up to 37.5 Mbaud (@ 75 MHz module clock) with Receive and Transmit FIFO support. The serial clock signal can be generated by the SSC itself (Master Mode) or can be received from an external master (Slave Mode). Data width, shift direction, clock polarity and phase are programmable. This allows communication with SPI-compatible devices. Transmission and reception of data are double-buffered. A shift clock generator provides the SSC with a separate serial clock signal. One slave select input are available for Slave Mode operation. Eight programmable slave select outputs (chip selects) are supported in Master Mode.

Note: The SSC0 contains an 8-stage Receive and Transmit FIFO. The SSC1 does not provide any FIFO functionality.

Features

- Master and Slave Mode operation
 - Full-duplex or half-duplex operation
 - Automatic pad control possible
- Flexible data format
 - Programmable number of data bits: 2 to 16 bits
 - Programmable shift direction: LSB or MSB shift first

Introduction

- Programmable clock polarity: Idle low or idle high state for the shift clock
- Programmable clock/data phase: Data shift with leading or trailing edge of the shift clock
- Baud rate generation from 37.5 MBaud to 572.2 Baud (@ 75 MHz module clock)
- Interrupt generation
 - On a transmitter empty condition
 - On a receiver full condition
 - On an error condition (receive, phase, baud rate, transmit error)
- Flexible SSC pin configuration
- Seven slave select inputs $\overline{\text{SLSI}}[7:1]$ in Slave Mode
- Eight programmable slave select outputs $\text{SLSO}[7:0]$ in Master Mode
 - Automatic SLSO generation with programmable timing
 - Programmable active level and enable control
- SSC0 only: 8-stage Receive FIFO (RXFIFO) and 8-stage Transmit FIFO (TXFIFO)
 - Independent control of RXFIFO and TXFIFO
 - 2- to 16-bit FIFO data width
 - Programmable receive/transmit interrupt trigger level
 - Receive and Transmit FIFO filling level indication
 - Overrun error generation
 - Underflow error generation

1.3.1.3 Micro Second Channel Interfaces

The Micro Second Channel (MSC) interfaces provide serial communication links typically used to connect power switches or other peripheral devices. The serial communication link includes a fast synchronous downstream channel and a slow asynchronous upstream channel. **Figure 1-4** shows a global view of the interface signals of the MSC interface.

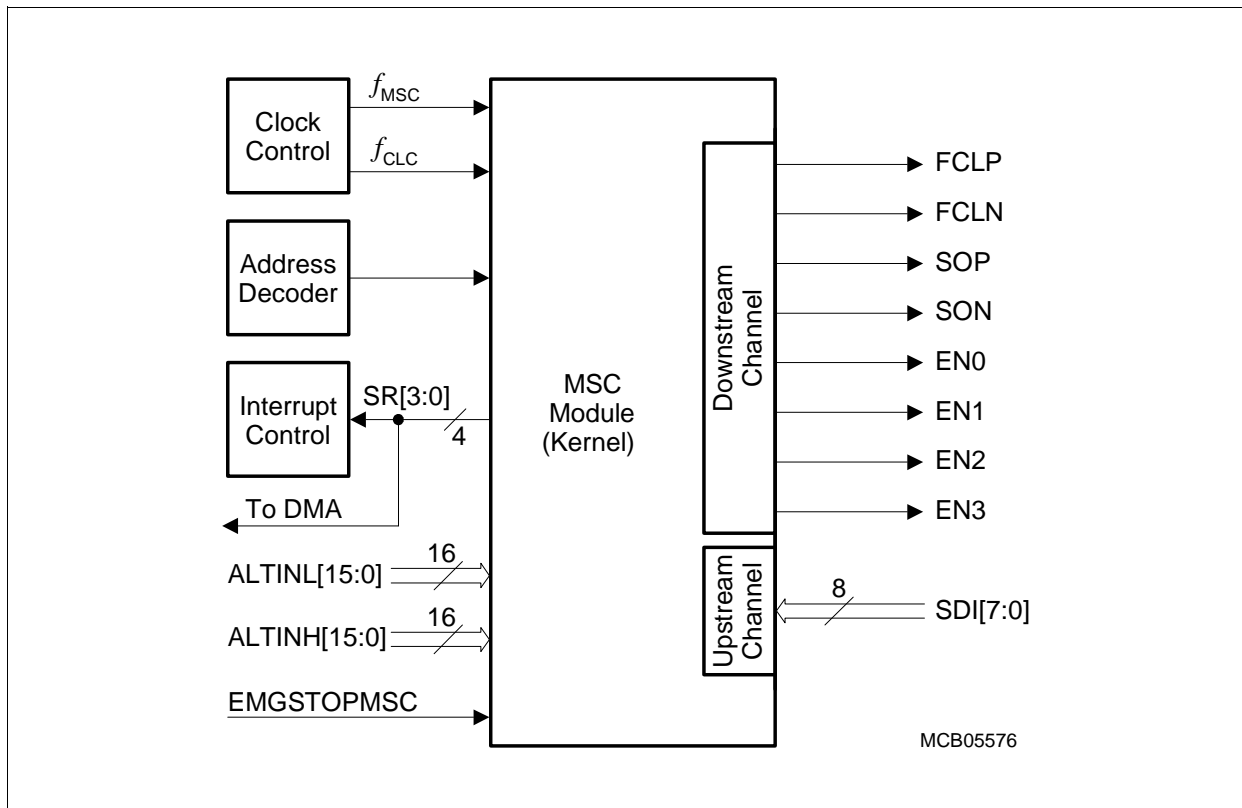


Figure 1-4 General Block Diagram of the MSC Interface

The downstream and upstream channels of the MSC module communicate with the external world via nine I/O lines. Eight output lines are required for the serial communication of the downstream channel (clock, data, and enable signals). One out of eight input lines $SDI[7:0]$ is used as serial data input signal for the upstream channel. The source of the serial data to be transmitted by the downstream channel can be MSC register contents or data that is provided on the $ALTINL/ALTINH$ input lines. These input lines are typically connected to other on-chip peripheral units (for example with a timer unit such as the GPTA). An emergency stop input signal makes it possible to set bits of the serial data stream to dedicated values in an emergency case.

Clock control, address decoding, and interrupt service request control are managed outside the MSC module kernel. Service request outputs are able to trigger an interrupt or a DMA request.

Features

- Fast synchronous serial interface to connect power switches in particular, or other peripheral devices via serial buses
- High-speed synchronous serial transmission on downstream channel
 - Serial output clock frequency: $f_{FCL} = f_{MSC}/2$
 - Fractional clock divider for precise frequency control of serial clock f_{MSC}
 - Command, data, and passive frame types
 - Start of serial frame: Software-controlled, timer-controlled, or free-running
 - Programmable upstream data frame length (16 or 12 bits)
 - Transmission with or without SEL bit
 - Flexible chip select generation indicates status during serial frame transmission
 - Emergency stop without CPU intervention
- Low-speed asynchronous serial reception on upstream channel
 - Baud rate: f_{MSC} divided by 4, 8, 16, 32, 64, 128, or 256
 - Standard asynchronous serial frames
 - Parity error checker
 - 8-to-1 input multiplexer for SDI lines
 - Built-in spike filter on SDI lines

1.3.1.4 MultiCAN Controller

The MultiCAN module contains four independent CAN nodes, representing four serial communication interfaces.

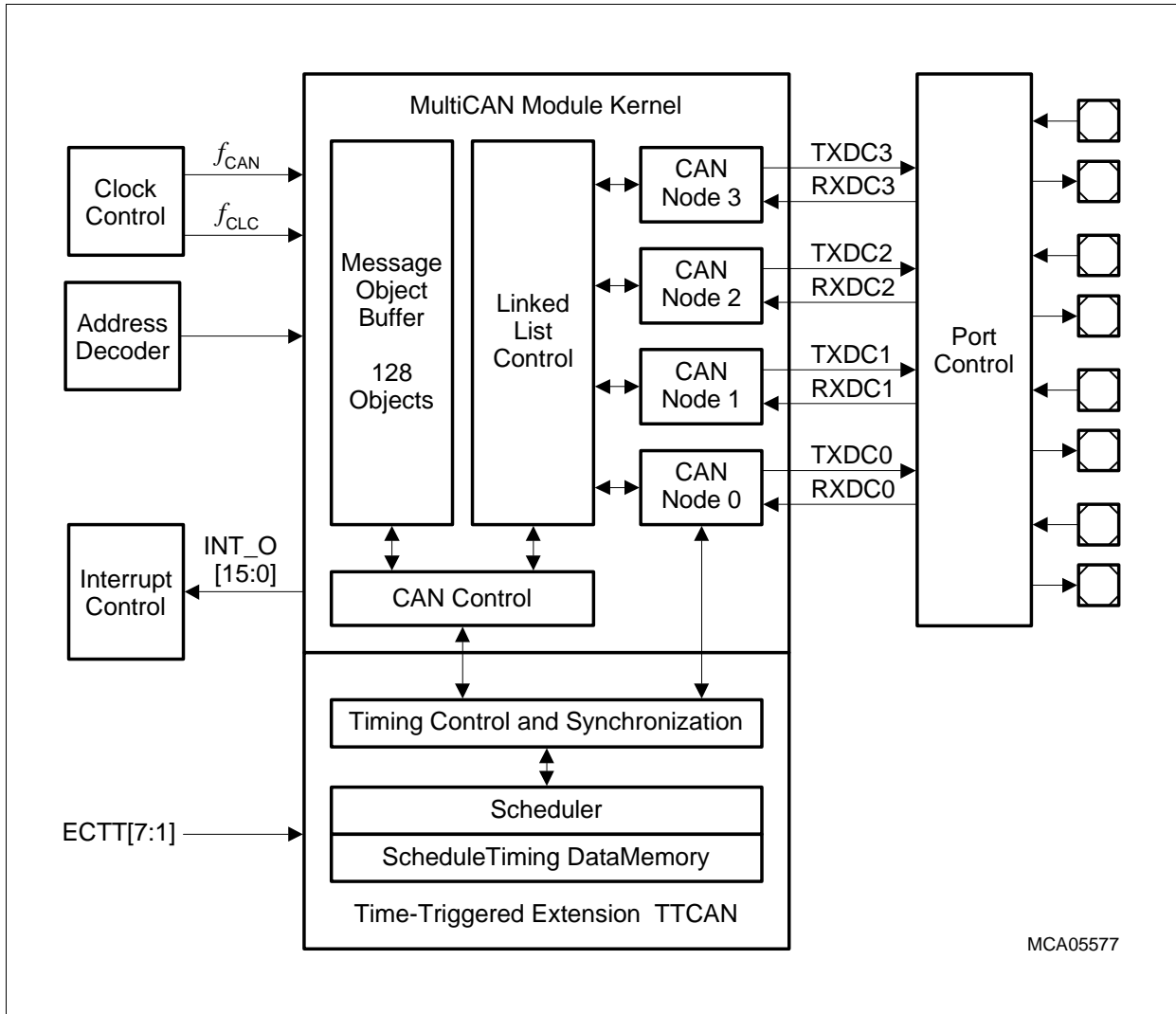


Figure 1-5 Overview of the MultiCAN Module with Time-Triggered Extension

The MultiCAN module contains four independently operating CAN nodes with Full-CAN functionality that are able to exchange Data and Remote Frames via a gateway function. Transmission and reception of CAN frames is handled in accordance with CAN specification V2.0 B (active). Each CAN node can receive and transmit standard frames with 11-bit identifiers as well as extended frames with 29-bit identifiers.

All four CAN nodes share a common set of message objects. Each message object can be individually allocated to one of the CAN nodes. Besides serving as a storage container for incoming and outgoing frames, message objects can be combined to build gateways between the CAN nodes or to set up a FIFO buffer.

The message objects are organized in double-chained linked lists, where each CAN node has its own list of message objects. A CAN node stores frames only into message objects that are allocated to the message object list of the CAN node, and it transmits only messages belonging to this message object list. A powerful, command-driven list controller performs all message object list operations.

The bit timings for the CAN nodes are derived from the module timer clock (f_{CAN}), and are programmable up to a data rate of 1 Mbit/s. External bus transceivers are connected to a CAN node via a pair of receive and transmit pins.

CAN Features

- Compliant with ISO 11898
- CAN functionality according to CAN specification V2.0 B active
- Dedicated control registers for each CAN node
- Data transfer rates up to 1 Mbit/s
- Flexible and powerful message transfer control and error handling capabilities
- Advanced CAN bus bit timing analysis and baud rate detection for each CAN node via a frame counter
- Full-CAN functionality: A set of 128 message objects can be individually
 - Allocated (assigned) to any CAN node
 - Configured as transmit or receive object
 - Set up to handle frames with 11-bit or 29-bit identifier
 - Identified by a timestamp via a frame counter
 - Configured to remote monitoring mode
- Advanced acceptance filtering
 - Each message object provides an individual acceptance mask to filter incoming frames.
 - A message object can be configured to accept standard or extended frames or to accept both standard and extended frames.
 - Message objects can be grouped into four priority classes for transmission and reception.
 - The selection of the message to be transmitted first can be based on frame identifier, IDE bit and RTR bit according to CAN arbitration rules, or on its order in the list.
- Advanced message object functionality
 - Message objects can be combined to build FIFO message buffers of arbitrary size, limited only by the total number of message objects.
 - Message objects can be linked to form a gateway that automatically transfers frames between 2 different CAN buses. A single gateway can link any two CAN nodes. An arbitrary number of gateways can be defined.
- Advanced data management
 - The message objects are organized in double-chained lists.

Introduction

- List reorganizations can be performed at any time, even during full operation of the CAN nodes.
- A powerful, command-driven list controller manages the organization of the list structure and ensures consistency of the list.
- Message FIFOs are based on the list structure and can easily be scaled in size during CAN operation.
- Static allocation commands offer compatibility with TwinCAN applications that are not list-based.
- Advanced interrupt handling
 - Up to 16 interrupt output lines are available. Interrupt requests can be routed individually to one of the 16 interrupt output lines.
 - Message post-processing notifications can be combined flexibly into a dedicated register field of 256 notification bits.

Time-Triggered Extension (TTCAN)

In addition to the event-driven CAN functionality, a deterministic behavior can be achieved for CAN node 0 by an extension module that supports TTCAN functionality. The TTCAN protocol is compliant with the confirmed standardization proposal for ISO 11898-4 and fully conforms to the existing CAN protocol.

The time-triggered functionality is added as higher-layer extension (session layer) to the CAN protocol in order to be able to operate in safety-critical applications. The new features allow for deterministic behavior of a CAN network and the synchronization of networks. Global time information is available. The time-triggered extension is based on a scheduler mechanism with a timing control unit and a dedicated timing data part.

TTCAN Features

- Full support of basic cycle and system matrix functionality
- Support of reference messages level 1 and level 2
- Usable as time master
- Arbitration windows supported in time-triggered mode
- Global time information available
- CAN node 0 can be configured either for event-driven or time-triggered mode
- Built-in scheduler mechanism and a timing synchronization unit
- Write protection for scheduler timing data memory
- Timing-related interrupt functionality

1.3.1.5 Micro Link Serial Bus Interface

The Micro Link Interface is a fast synchronous serial interface that makes it possible to exchange data between microcontrollers of the 32-bit AUDDO microcontroller family without intervention of a CPU or other bus masters. **Figure 1-6** shows how two microcontrollers are typically connected together via their MLI interfaces. The MLI operates in both microcontrollers as a bus master on the system bus.

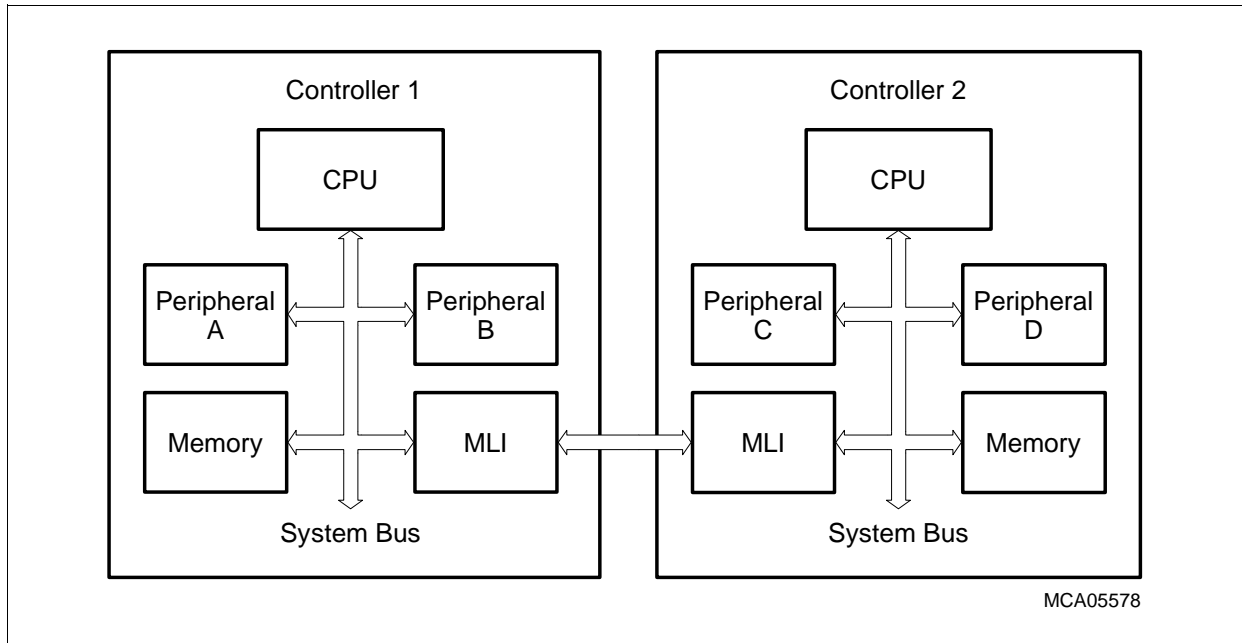


Figure 1-6 Typical Micro Link Interface Connection

Features

- Synchronous serial communication between MLI transmitters and MLI receivers located on the same or on different microcontroller devices
- Automatic data transfer/request transactions between local/remote controller
- Fully transparent read/write access supported (= remote programming)
- Complete address range of remote controller available
- Specific frame protocol to transfer commands, addresses and data
- Error control by parity bit
- 32-bit, 16-bit, and 8-bit data transfers
- Programmable baud rate: max. $f_{MLI}/2$ (= 37.5 Mbit/s @ 75 MHz module clock)
- Multiple remote (slave) controllers supported

Figure 1-7 shows a general block diagram of the MLI Module.

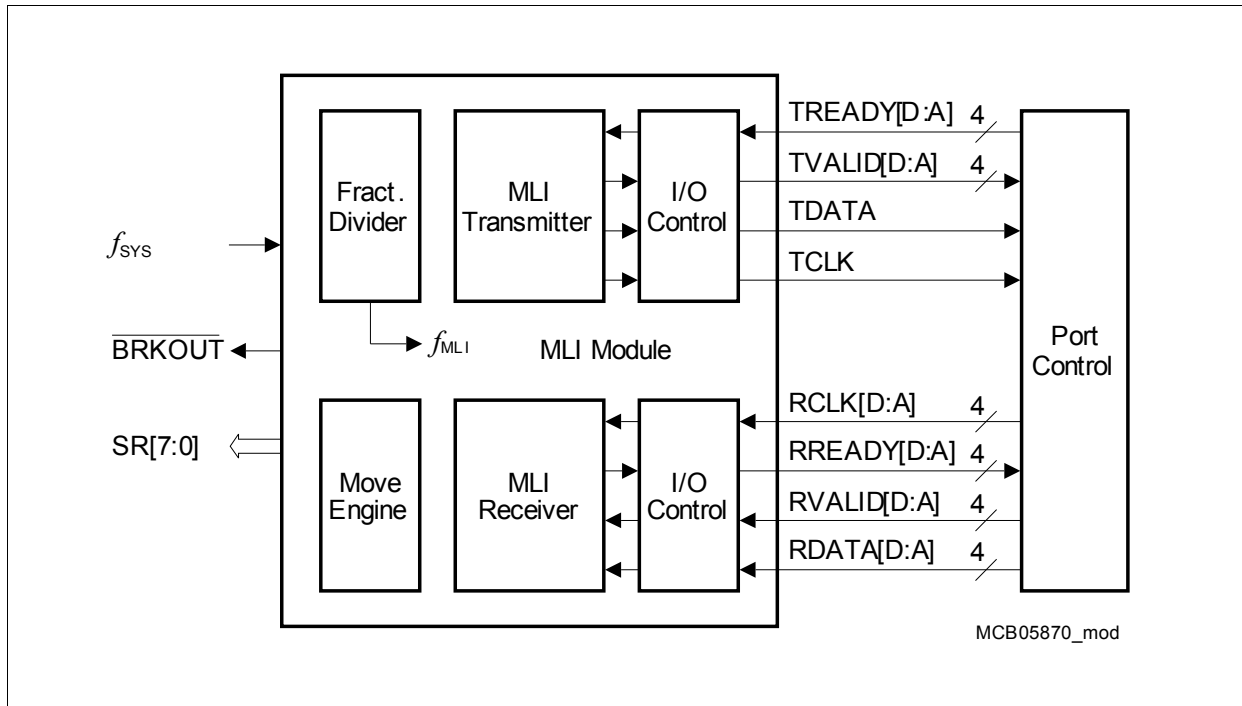


Figure 1-7 General Block Diagram of the MLI Module

The MLI transmitter and MLI receiver communicate with other off-chip MLI receivers and MLI transmitters via a four-line serial I/O bus each. Several I/O lines of these I/O buses are available outside the MLI module kernel as four-line output or input buses with index numbering A, B, C and D. The transmitter signals are named with the prefix “T” and the receiver signals are named with the prefix “R”.

Data read and write operations from/to remote window areas can be handled by a Move Engine that is able to operate as a bus master. Clock control, address decoding, and interrupt service request control are managed outside the MLI module kernel. Eight service request outputs can be used to trigger an interrupt or a DMA request.

1.3.2 General Purpose Timer Array

The TC1796 contains two General Purpose Timer Arrays (GPTA0 and GPTA1) with identical functionality, plus an additional Local Timer Cell Array (LTCA2). **Figure 1-8** shows a global view of the GPTA modules.

The GPTA provides a set of timer, compare, and capture functionalities that can be flexibly combined to form signal measurement and signal generation units. They are optimized for tasks typical of engine, gearbox, and electrical motor control applications, but can also be used to generate simple and complex signal waveforms needed in other industrial applications.

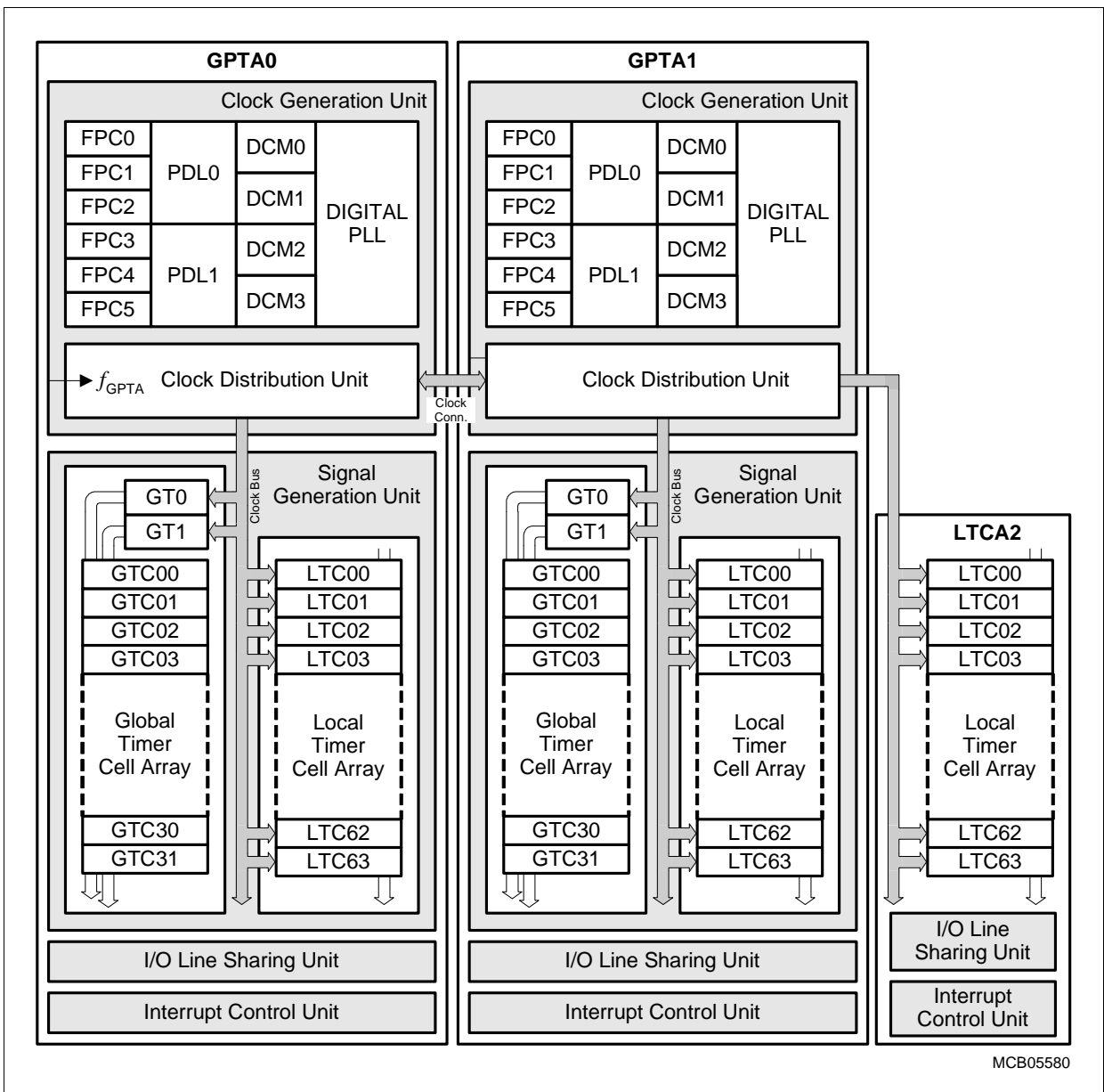


Figure 1-8 General Block Diagram of the GPTA Modules

1.3.2.1 Functionality of GPTA0 and GPTA1

Each of the General Purpose Timer Arrays (GPTA0 and GPTA1) provides a set of hardware modules required for high-speed digital signal processing:

- Filter and Prescaler Cells (FPC) support input noise filtering and prescaler operation.
- Phase Discrimination Logic units (PDL) decode the direction information output by a rotation tracking system.
- Duty Cycle Measurement Cells (DCM) provide pulse-width measurement capabilities.
- A Digital Phase Locked Loop unit (PLL) generates a programmable number of GPTA module clock ticks during an input signal's period.
- Global Timer units (GT) driven by various clock sources are implemented to operate as a time base for the associated Global Timer Cells.
- Global Timer Cells (GTC) can be programmed to capture the contents of a Global Timer on an external or internal event. A GTC may also be used to control an external port pin depending on the result of an internal compare operation. GTCs can be logically concatenated to provide a common external port pin with a complex signal waveform.
- Local Timer Cells (LTC) operating in Timer, Capture, or Compare Mode may also be logically tied together to drive a common external port pin with a complex signal waveform. LTCs – enabled in Timer Mode or Capture Mode – can be clocked or triggered by various external or internal events.

Input lines can be shared by an LTC and a GTC to trigger their programmed operation simultaneously.

The following list summarizes the specific features of the GPTA units.

Clock Generation Unit

- Filter and Prescaler Cell (FPC)
 - Six independent units
 - Three basic operating modes:
Prescaler, Delayed Debounce Filter, Immediate Debounce Filter
 - Selectable input sources:
Port lines, GPTA module clock, FPC output of preceding FPC cell
 - Selectable input clocks:
GPTA module clock, prescaled GPTA module clock, DCM clock, compensated or uncompensated PLL clock.
 - $f_{\text{GPTA}}/2$ maximum input signal frequency in Filter Modes
- Phase Discriminator Logic (PDL)
 - Two independent units
 - Two operating modes (2- and 3-sensor signals)
 - $f_{\text{GPTA}}/4$ maximum input signal frequency in 2-sensor Mode, $f_{\text{GPTA}}/6$ maximum input signal frequency in 3-sensor Mode

- Duty Cycle Measurement (DCM)
 - Four independent units
 - 0 - 100% margin and time-out handling
 - f_{GPTA} maximum resolution
 - $f_{\text{GPTA}}/2$ maximum input signal frequency
- Digital Phase Locked Loop (PLL)
 - One unit
 - Arbitrary multiplication factor between 1 and 65535
 - f_{GPTA} maximum resolution
 - $f_{\text{GPTA}}/2$ maximum input signal frequency
- Clock Distribution Unit (CDU)
 - One unit
 - Provides eight clock output signals (clock bus):
 f_{GPTA} , divided f_{GPTA} clocks, FPC1/FPC4 outputs, DCM clock, LTC prescaler clock

Signal Generation Unit

- Global Timers (GT)
 - Two independent units
 - Two operating modes (Free-Running Timer and Reload Timer)
 - 24-bit data width
 - f_{GPTA} maximum resolution
 - $f_{\text{GPTA}}/2$ maximum input signal frequency
- Global Timer Cell (GTC)
 - 32 units related to the Global Timers
 - Two operating modes (Capture, Compare and Capture after Compare)
 - 24-bit data width
 - f_{GPTA} maximum resolution
 - $f_{\text{GPTA}}/2$ maximum input signal frequency
- Local Timer Cell (LTC)
 - 64 independent units
 - Three basic operating modes (Timer, Capture and Compare) for 63 units
 - Special compare modes for one unit
 - 16-bit data width
 - f_{GPTA} maximum resolution
 - $f_{\text{GPTA}}/2$ maximum input signal frequency

Interrupt Sharing Unit

- 286 interrupt sources, generating up to 92 service requests.

I/O Sharing Unit

- Interconnecting inputs and outputs from internal clocks, FPC, GTC, LTC, ports, and MSC interface

1.3.2.2 Functionality of LTCA2

- 64 Local Timer Cells (LTCs)
 - Three basic operating modes (Timer, Capture and Compare) for 63 units
 - Special compare modes for one unit
 - 16-bit data width
 - f_{GPTA} maximum resolution
 - $f_{GPTA}/2$ maximum input signal frequency

1.3.3 Analog-to-Digital Converters

The TC1796 contains two medium-speed Analog-to-Digital Converters (ADC0 and ADC1) with identical functionality and a third fast Analog-to-Digital Converter (FADC). ADC0 and ADC1 provide 2-3 μs conversion time @ 10-bit resolution and are intended mainly for single-ended signals. They offer a very flexible and comprehensive control for monitoring a large number of relatively slow signals, including synchronous conversion of both ADCs.

The FADC offers very fast conversion rates (280 ns @ 10-bit resolution) thus allowing sampling of high-frequency signals. For slow and mid-range frequency signal heavy oversampling can be performed and thus expensive filters can be avoided.

1.3.3.1 Analog-to-Digital Converters (ADC0 and ADC1)

The two on-chip ADC modules of the TC1796 are analog-to-digital converters with 8-bit, 10-bit, or 12-bit resolution including sample & hold functionality. The A/D converters operate by the method of successive approximation. A multiplexer selects up to 32 analog inputs that can be connected to the 16 conversion channels in each ADC module. An automatic self-calibration adjusts the ADC modules to changing temperatures or process variations.

Features

- 8-bit, 10-bit, 12-bit A/D conversion
- Conversion time below 2.5 μs @ 10-bit resolution
- Extended channel status information on request source
- Successive approximation conversion method
- Total Unadjusted Error (TUE) of ± 2 LSB @ 10-bit resolution
- Integrated sample & hold functionality
- Direct control of up to 16(32) analog input channels per ADC
- Dedicated control and status registers for each analog channel
- Powerful conversion request sources
- Selectable reference voltages for each channel
- Programmable sample and conversion timing schemes
- Limit checking
- Flexible ADC module service request control unit
- Synchronization of the two on-chip A/D Converters
- Automatic control of external analog multiplexers
- Equidistant samples initiated by timer
- External trigger and gating inputs for conversion requests
- Power reduction and clock control feature

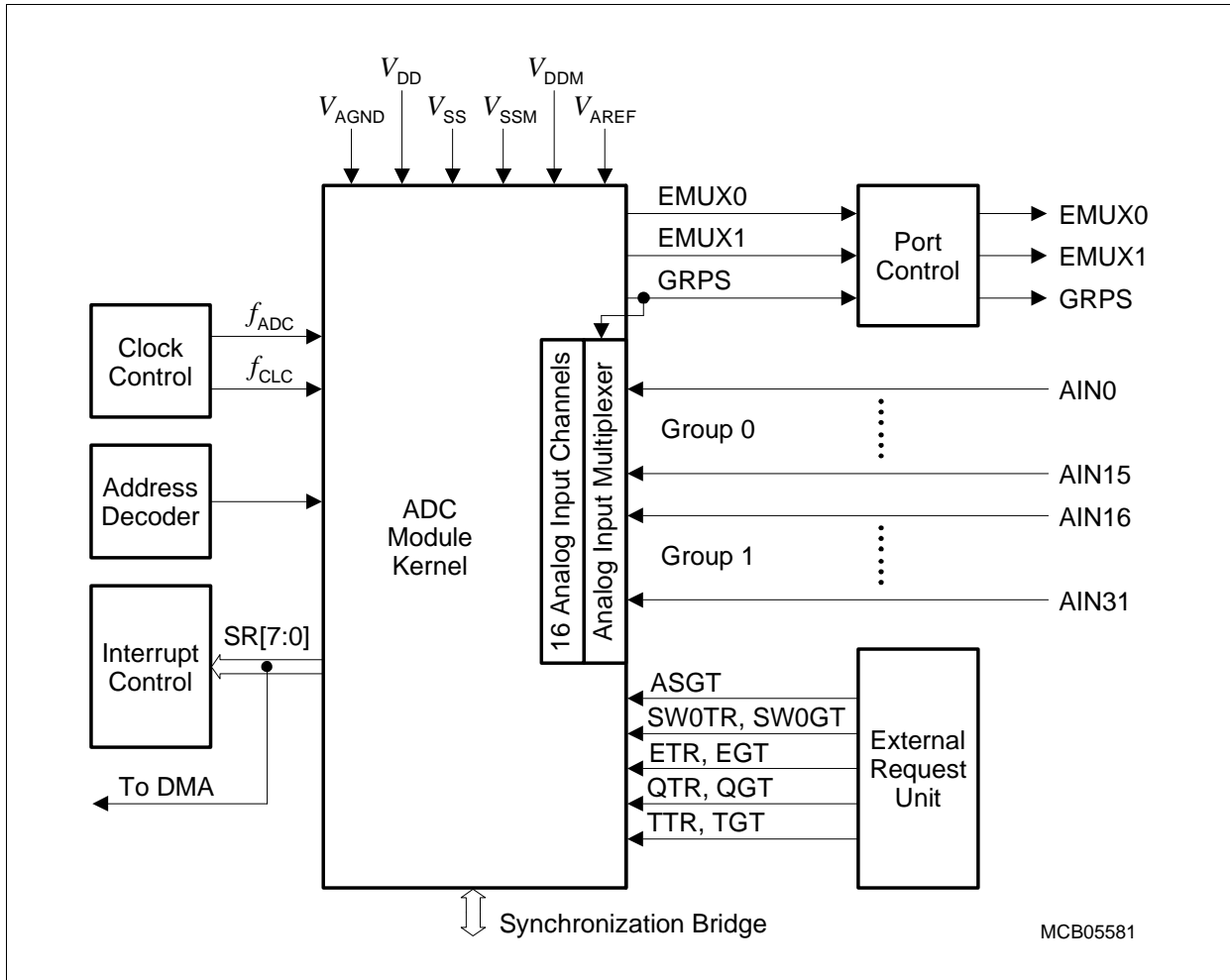


Figure 1-9 General Block Diagram of the ADC Module

Each of the two ADC modules, ADC0 and ADC1 has 16 analog input channels. An analog multiplexer selects the input line for the analog input channels from among 32 analog inputs. Additionally, an external analog multiplexer can be used for analog input extension. External Clock control, address decoding, and service request (interrupt) control are managed outside the ADC module kernel. A synchronization bridge is used for synchronization of two ADC modules. External trigger conditions are controlled by an External Request Unit. This unit generates the control signals for auto-scan control (ASGT), software trigger control (SW0TR, SW0GT), the event trigger control (ETR, EGT), queue control (QTR, QGT), and timer trigger control (TTR, TGT).

1.3.3.2 Fast Analog-to-Digital Converter Unit (FADC)

The on-chip FADC module of the TC1796 is primarily a four-channel A/D converter with 10-bit resolution that operates by the method of successive approximation.

Features

- Extreme fast conversion, 21 cycles of f_{FADC} (280 ns @ $f_{FADC} = 75$ MHz)
- 10-bit A/D conversion
 - Higher resolution by averaging of consecutive conversions is supported
- Successive approximation conversion method
- Four differential input channels
- Offset and gain calibration support for each channel
- Differential input amplifier with programmable gain of 1, 2, 4 and 8 for each channel
- Free-running (Channel Timers) or triggered conversion modes
- Trigger and gating control for external signals
- Built-in Channel Timers for internal triggering
- Channel timer request periods independently selectable for each channel
- Selectable, programmable antialiasing and data reduction filter block

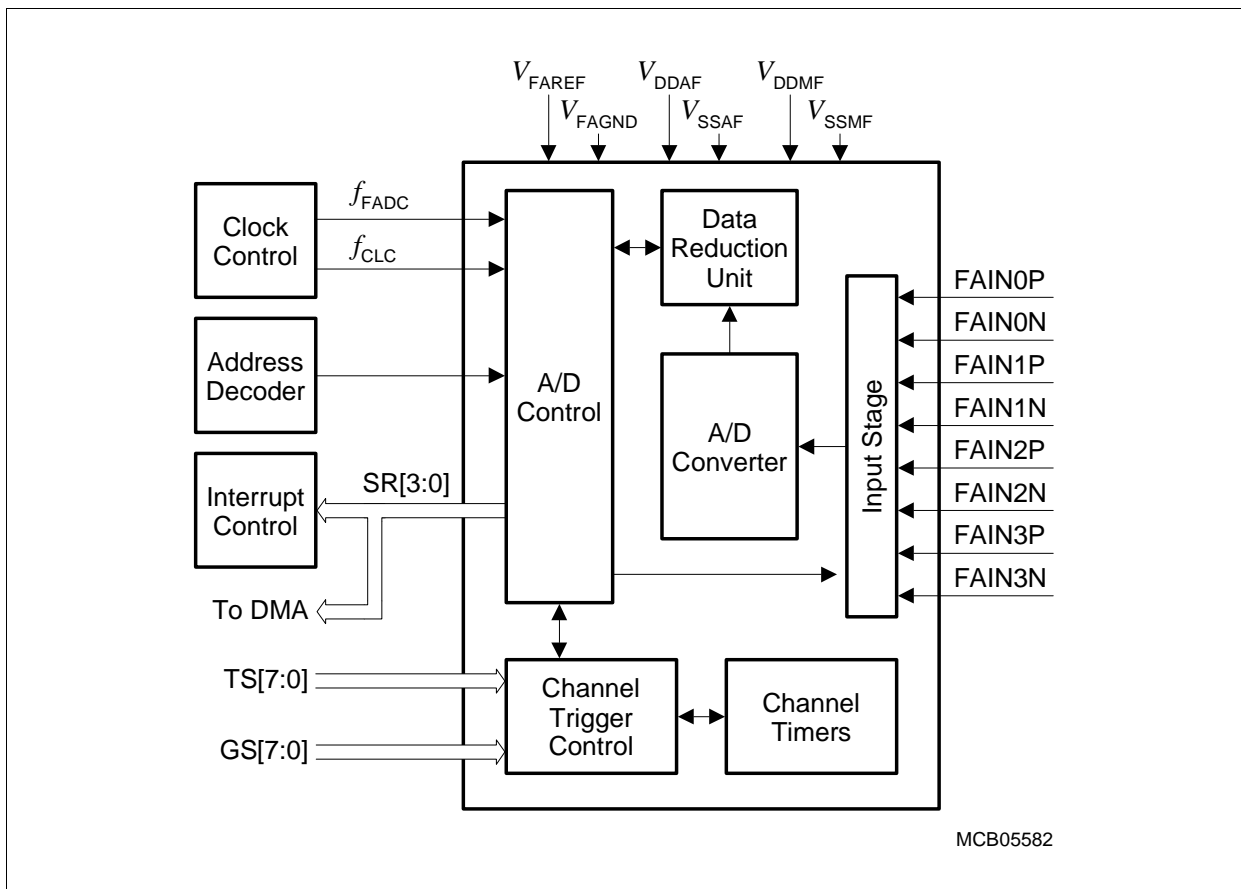


Figure 1-10 Block Diagram of the FADC Module

As shown in **Figure 1-10**, the main FADC functional blocks are:

- The Input Stage – contains the differential inputs and the programmable amplifier
- The A/D Converter – is responsible for the analog-to-digital conversion
- The Data Reduction Unit – contains programmable antialiasing and data reduction filters
- The Channel Trigger Control block – determines the trigger and gating conditions for the four FADC channels.
- The Channel Timers – can independently trigger the conversion of each FADC channel
- The A/D Control block is responsible for the overall FADC functionality.

The FADC module is supplied by the following power supply and reference voltage lines:

- V_{DDMF}/V_{DDMF} : FADC Analog Part Power Supply (3.3 V)
- V_{DDAF}/V_{DDAF} : FADC Analog Part Logic Power Supply (1.5 V)
- V_{FAREF}/V_{FAGND} : FADC Reference Voltage (3.3 V)/FADC Reference Ground

1.4 TC1796 Pin Definitions and Functions

Figure 1-11 shows the logic symbol of the TC1796.

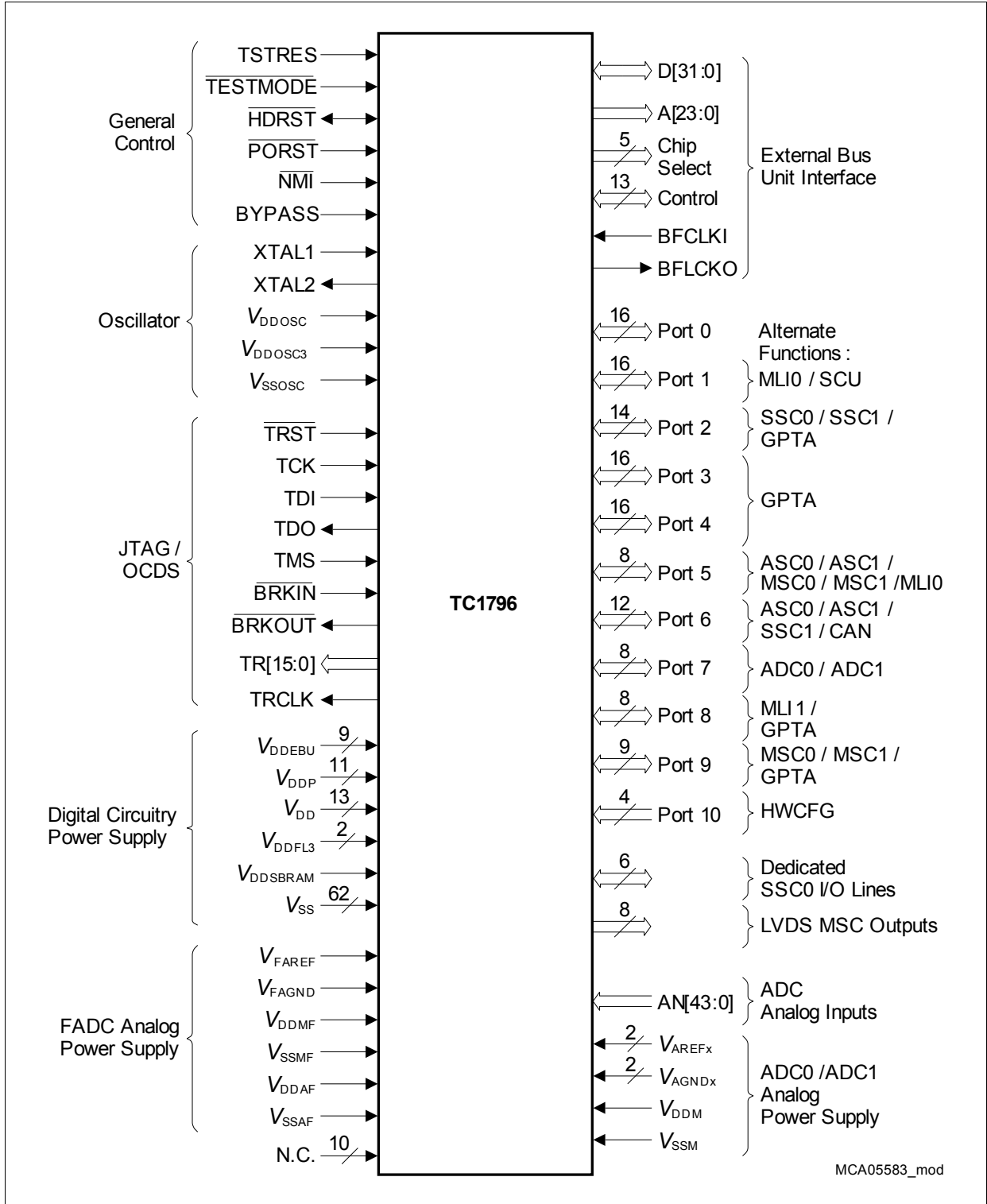


Figure 1-11 TC1796 Logic Symbol

1.4.1 TC1796 Pin Configuration

Figure 1-11 shows the logic symbol of the TC1796.

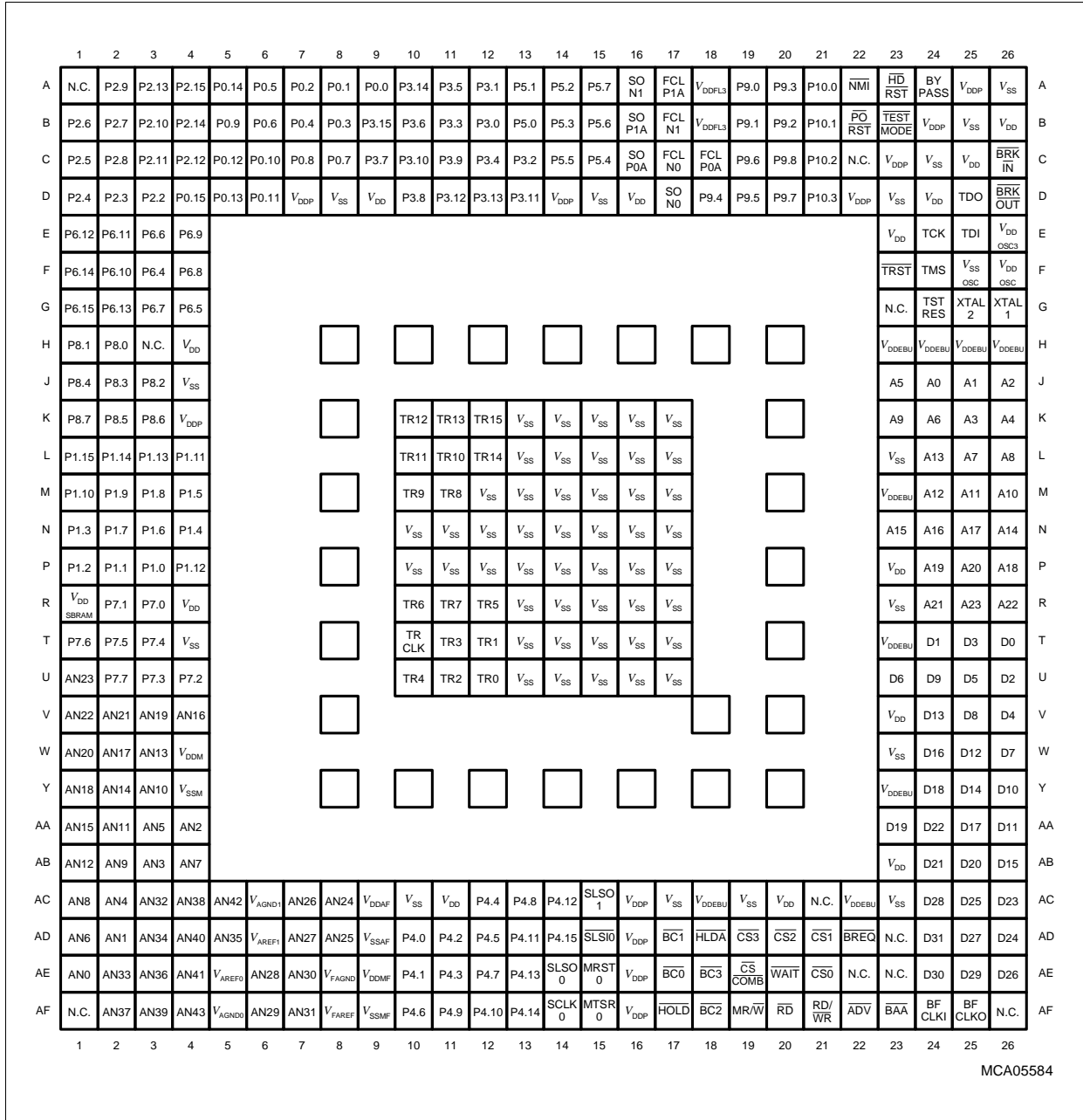


Figure 1-12 TC1796 Pinning for P-BGA-416 Package (top view)

Table 1-3 Pin Definitions and Functions

Symbol	Pins	I/O	Pad Driver Class	Power Supply	Functions
External Bus Interface (EBU)					
D[31:0]		I/O	B1	V_{DDEBU}	EBU Data Bus Lines The EBU Data Bus Lines D[31:0] serve as external data bus.
D0	T26	I/O			Data bus line 0
D1	T24	I/O			Data bus line 1
D2	U26	I/O			Data bus line 2
D3	T25	I/O			Data bus line 3
D4	V26	I/O			Data bus line 4
D5	U25	I/O			Data bus line 5
D6	U23	I/O			Data bus line 6
D7	W26	I/O			Data bus line 7
D8	V25	I/O			Data bus line 8
D9	U24	I/O			Data bus line 9
D10	Y26	I/O			Data bus line 10
D11	AA26	I/O			Data bus line 11
D12	W25	I/O			Data bus line 12
D13	V24	I/O			Data bus line 13
D14	Y25	I/O			Data bus line 14
D15	AB26	I/O			Data bus line 15
D16	W24	I/O			Data bus line 16
D17	AA25	I/O			Data bus line 17
D18	Y24	I/O			Data bus line 18
D19	AA23	I/O			Data bus line 19
D20	AB25	I/O			Data bus line 20
D21	AB24	I/O			Data bus line 21
D22	AA24	I/O			Data bus line 22
D23	AC26	I/O			Data bus line 23
D24	AD26	I/O			Data bus line 24
D25	AC25	I/O			Data bus line 25
D26	AE26	I/O			Data bus line 26
D27	AD25	I/O			Data bus line 27
D28	AC24	I/O			Data bus line 28
D29	AE25	I/O			Data bus line 29
D30	AE24	I/O			Data bus line 30
D31	AD24	I/O			Data bus line 31

Table 1-3 Pin Definitions and Functions (cont'd)

Symbol	Pins	I/O	Pad Driver Class	Power Supply	Functions
A[23:0]		O	B1	V_{DDEBU}	EBU Address Bus Lines A[23:0] The EBU Address Bus Lines serve as external address bus.
A0	J24	O			Address bus line 0
A1	J25	O			Address bus line 1
A2	J26	O			Address bus line 2
A3	K25	O			Address bus line 3
A4	K26	O			Address bus line 4
A5	J23	O			Address bus line 5
A6	K24	O			Address bus line 6
A7	L25	O			Address bus line 7
A8	L26	O			Address bus line 8
A9	K23	O			Address bus line 9
A10	M26	O			Address bus line 10
A11	M25	O			Address bus line 11
A12	M24	O			Address bus line 12
A13	L24	O			Address bus line 13
A14	N26	O			Address bus line 14
A15	N23	O			Address bus line 15
A16	N24	O			Address bus line 16
A17	N25	O			Address bus line 17
A18	P26	O			Address bus line 18
A19	P24	O			Address bus line 19
A20	P25	O			Address bus line 20
A21	R24	O			Address bus line 21
A22	R26	O			Address bus line 22
A23	R25	O			Address bus line 23
CS0	AE21	O	B1	V_{DDEBU}	Chip Select Output Lines Chip select output line 0
CS1	AD21	O			Chip select output line 1
CS2	AD20	O			Chip select output line 2
CS3	AD19	O			Chip select output line 3
CS COMB	AE19	O	B1	V_{DDEBU}	Combined Chip Select Output for Global Select / Emulator Memory Region / Emulator Overlay Memory

Table 1-3 Pin Definitions and Functions (cont'd)

Symbol	Pins	I/O	Pad Driver Class	Power Supply	Functions
BFCLKO	AF25	O	B2	V_{DDEBU}	Burst Mode Flash Clock Output (non-differential)
BFCLKI	AF24	I	B1		Burst Mode Flash Clock Input (feedback clock)
RD	AF20	O	B1		Read Control Line
RD/WR	AF21	O	B1		Write Control Line
ADV	AF22	O	B1		Address Valid Output
BC0	AE17	O	B1		Byte Control Lines Byte control line 0
BC1	AD17	O			Byte control line 1
BC2	AF18	O			Byte control line 2
BC3	AE18	O			Byte control line 3
MR/W	AF19	O	B1		Motorola-style Read / Write Control Signal
WAIT	AE20	I	–		Wait Input for inserting Wait States
BAA	AF23	O	B1		Burst Address Advance Output
HOLD	AF17	I	–		Hold Request Input
HLDA	AD18	O	B1		Hold Acknowledge Output
BREQ	AD22	O	B1	Bus Request Output	

Table 1-3 Pin Definitions and Functions (cont'd)

Symbol	Pins	I/O	Pad Driver Class	Power Supply	Functions
Parallel Ports					
P0		I/O	A1		<p>Port 0 Port 0 is a 16-bit bi-directional general-purpose I/O port.</p> <p>P0.0 A9 I/O Port 0 I/O line 0 P0.1 A8 I/O Port 0 I/O line 1 P0.2 A7 I/O Port 0 I/O line 2 P0.3 B8 I/O Port 0 I/O line 3 P0.4 B7 I/O Port 0 I/O line 4 P0.5 A6 I/O Port 0 I/O line 5 P0.6 B6 I/O Port 0 I/O line 6 P0.7 C8 I/O Port 0 I/O line 7 P0.8 C7 I/O Port 0 I/O line 8 P0.9 B5 I/O Port 0 I/O line 9 P0.10 C6 I/O Port 0 I/O line 10 P0.11 D6 I/O Port 0 I/O line 11 P0.12 C5 I/O Port 0 I/O line 12 P0.13 D5 I/O Port 0 I/O line 13 P0.14 A5 I/O Port 0 I/O line 14 P0.15 D4 I/O Port 0 I/O line 15</p> <p>The states of the Port 0 pins are latched into the software configuration input register SCU_SCLIR at the rising edge of $\overline{\text{HDRST}}$. In the different TC1796 device versions several Port 0 pins are reserved for device configuration purposes (see Page 10-25).</p>

Table 1-3 Pin Definitions and Functions (cont'd)

Symbol	Pins	I/O	Pad Driver Class	Power Supply	Functions
P1		I/O	A1/A2	V_{DDP}	Port 1 Port 1 is a 16-bit bi-directional general-purpose I/O port which can be alternatively used for the MLI0 interface or as external trigger input lines.
P1.0	P3	I	A1		REQ0 External trigger input 0
P1.1	P2	I	A1		REQ1 External trigger input 1
P1.2	P1	I	A1		REQ2 External trigger input 3
P1.3	N1	I	A1		REQ3 External trigger input 2
		I	A1		TREADY0B MLI0 transmit channel ready input B
P1.4	N4	O	A2		TCLK0 MLI0 transmit channel clock output
P1.5	M4	I	A1		TREADY0A MLI0 transmit channel ready input A
P1.6	N3	O	A2		TVALID0A MLI0 transmit channel valid output A
P1.7	N2	O	A2		TDATA0 MLI0 transmit channel data output
P1.8	M3	I	A1		RCLK0A MLI0 receive channel clock input A
P1.9	M2	O	A2		RREADY0A MLI0 receive channel ready output A
P1.10	M1	I	A1		RVALID0A MLI0 receive channel valid input A
P1.11	L4	I	A1		RDATA0A MLI0 receive channel data input A
P1.12	P4	O	A2		SYSCLK System clock output
P1.13	L3	I	A1		RCLK0B MLI0 receive channel clock input B
P1.14	L2	I	A1		RVALID0B MLI0 receive channel valid input B
P1.15	L1	I	A1		RDATA0B MLI0 receive channel data input B

Table 1-3 Pin Definitions and Functions (cont'd)

Symbol	Pins	I/O	Pad Driver Class	Power Supply	Functions
P2		I/O	A1/A2	V_{DDP}	Port 2 Port 2 is a 14-bit bi-directional general-purpose I/O port which can be used alternatively for the six upper SSC slave select outputs or for GPTA I/O lines.
P2.2	D3	O	A2		SLSO2 Slave select output line 2
P2.3	D2	O	A2		SLSO3 Slave select output line 3
P2.4	D1	O	A2		SLSO4 Slave select output line 4
P2.5	C1	O	A2		SLSO5 Slave select output line 5
P2.6	B1	O	A2		SLSO6 Slave select output line 6
P2.7	B2	O	A2		SLSO7 Slave select output line 7
P2.8	C2	I/O	A1		IN0 / OUT0 line of GPTA
P2.9	A2	I/O	A1		IN1 / OUT1 line of GPTA
P2.10	B3	I/O	A1		IN2 / OUT2 line of GPTA
P2.11	C3	I/O	A1		IN3 / OUT3 line of GPTA
P2.12	C4	I/O	A1		IN4 / OUT4 line of GPTA
P2.13	A3	I/O	A1		IN5 / OUT5 line of GPTA
P2.14	B4	I/O	A1		IN6 / OUT6 line of GPTA
P2.15	A4	I/O	A1		IN7 / OUT7 line of GPTA

Table 1-3 Pin Definitions and Functions (cont'd)

Symbol	Pins	I/O	Pad Driver Class	Power Supply	Functions
P3		I/O	A1	V_{DDP}	Port 3 Port 3 is a 16-bit bi-directional general-purpose I/O port which can be alternatively used for GPTA I/O lines.
P3.0	B12	I/O			IN8 / OUT8 line of GPTA
P3.1	A12	I/O			IN9 / OUT9 line of GPTA
P3.2	C13	I/O			IN10 / OUT10 line of GPTA
P3.3	B11	I/O			IN11 / OUT11 line of GPTA
P3.4	C12	I/O			IN12 / OUT12 line of GPTA
P3.5	A11	I/O			IN13 / OUT13 line of GPTA
P3.6	B10	I/O			IN14 / OUT14 line of GPTA
P3.7	C9	I/O			IN15 / OUT15 line of GPTA
P3.8	D10	I/O			IN16 / OUT16 line of GPTA
P3.9	C11	I/O			IN17 / OUT17 line of GPTA
P3.10	C10	I/O			IN18 / OUT18 line of GPTA
P3.11	D13	I/O			IN19 / OUT19 line of GPTA
P3.12	D11	I/O			IN20 / OUT20 line of GPTA
P3.13	D12	I/O			IN21 / OUT21 line of GPTA
P3.14	A10	I/O			IN22 / OUT22 line of GPTA
P3.15.	B9	I/O			IN23 / OUT23 line of GPTA

Table 1-3 Pin Definitions and Functions (cont'd)

Symbol	Pins	I/O	Pad Driver Class	Power Supply	Functions
P4		I/O	A1/A2	V_{DDP}	Port 4 Port 4 is a 16-bit bi-directional general-purpose I/O port which can be alternatively used for GPTA I/O lines.
P4.0	AD10	I/O	A2 ¹⁾		IN24 / OUT24 line of GPTA
P4.1	AE10	I/O	A2 ¹⁾		IN25 / OUT25 line of GPTA
P4.2	AD11	I/O	A2 ¹⁾		IN26 / OUT26 line of GPTA
P4.3	AE11	I/O	A2 ¹⁾		IN27 / OUT27 line of GPTA
P4.4	AC12	I/O	A2 ¹⁾		IN28 / OUT28 line of GPTA
P4.5	AD12	I/O	A2 ¹⁾		IN29 / OUT29 line of GPTA
P4.6	AF10	I/O	A2 ¹⁾		IN30 / OUT30 line of GPTA
P4.7	AE12	I/O	A2 ¹⁾		IN31 / OUT31 line of GPTA
P4.8	AC13	I/O	A1		IN32 / OUT32 line of GPTA
P4.9	AF11	I/O	A1		IN33 / OUT33 line of GPTA
P4.10	AF12	I/O	A1		IN34 / OUT34 line of GPTA
P4.11	AD13	I/O	A1		IN35 / OUT35 line of GPTA
P4.12	AC14	I/O	A1		IN36 / OUT36 line of GPTA
P4.13	AE13	I/O	A1		IN37 / OUT37 line of GPTA
P4.14	AF13	I/O	A1		IN38 / OUT38 line of GPTA
P4.15	AD14	I/O	A1		IN39 / OUT39 line of GPTA

Table 1-3 Pin Definitions and Functions (cont'd)

Symbol	Pins	I/O	Pad Driver Class	Power Supply	Functions
P5		I/O	A2	V_{DDP}	Port 5 Port 5 is an 8-bit bi-directional general-purpose I/O port which can be alternatively used for ASC0/1 or MSC0/1 lines.
P5.0	B13	I/O			RXD0A ASC0 receiver input / output A
P5.1	A13	O			TXD0A ASC0 transmitter output A
P5.2	A14	I/O			RXD1A ASC1 receiver input / output A
P5.3	B14	O			TXD1A ASC1 transmitter output A. P5.3 is latched with the rising edge of \overline{PORST} if BYPASS = 1 and stored in inverted state as bit OSC_CON.MOSC.
P5.4	C15	O			EN00 MSC0 device select output 0
		O			RREADY0B MLI0 receive channel ready output B
P5.5	C14	I			SDI0 MSC0 serial data input
P5.6	B15	O			EN10 MSC1 device select output 0
		O			TVALID0B MLI0 transmit channel valid output B
P5.7	A15	I			SDI1 MSC1 serial data input

Table 1-3 Pin Definitions and Functions (cont'd)

Symbol	Pins	I/O	Pad Driver Class	Power Supply	Functions
P6		I/O	A2	V_{DDP}	Port 6 Port 6 is a 12-bit bi-directional general-purpose I/O port which can be alternatively used for SSC1, ASC0/1, and CAN I/O lines.
P6.4	F3	O I			MTSR1 SSC1 master transmit output / SSC1 slave receive input
P6.5	G4	I O			MRST1 SSC1 master receive input / SSC1 slave transmit output
P6.6	E3	I/O			<u>SCLK1</u> SSC1 clock input/output
P6.7	G3	I			SLSI1 SSC1 slave select input
P6.8	F4	I I/O			RXDCAN0 CAN node 0 receiver input RXD0B ASC0 receiver input / output B
P6.9	E4	O			TXDCAN0 CAN node 0 transmitter output
P6.10	F2	O I I/O			TXD0B ASC0 transmitter output B RXDCAN1 CAN node 1 receiver input RXD1B ASC1 receiver input / output B
P6.11	E2	O			TXDCAN1 CAN node 1 transmitter output
P6.12	E1	O I			TXD1B ASC1 transmitter output B
P6.13	G2	O			RXDCAN2 CAN node 2 receiver input TXDCAN2 CAN node 2 transmitter output
P6.14	F1	I			RXDCAN3 CAN node 3 receiver input
P6.15	G1	O			TXDCAN3 CAN node 3 transmitter output

Table 1-3 Pin Definitions and Functions (cont'd)

Symbol	Pins	I/O	Pad Driver Class	Power Supply	Functions
P7		I/O	A1	V_{DDP}	Port 7 Port 7 is a 8-bit bi-directional general-purpose I/O port which can be alternatively used as external trigger input lines and for ADC0/1 external multiplexer control.
P7.0	R3	I			REQ4 External trigger input 4
P7.1	R2	I			REQ5 External trigger input 5
		O			AD0EMUX2 ADC0 external multiplexer control output 2
P7.2	U4	O			AD0EMUX0 ADC0 external multiplexer control output 0
P7.3	U3	O			AD0EMUX1 ADC0 external multiplexer control output 1
P7.4	T3	I			REQ6 External trigger input 6
P7.5	T2	I			REQ7 External trigger input 7
P7.6	T1	O			AD1EMUX0 ADC1 external multiplexer control output 0
P7.7	U2	O			AD1EMUX1 ADC1 external multiplexer control output 1

Table 1-3 Pin Definitions and Functions (cont'd)

Symbol	Pins	I/O	Pad Driver Class	Power Supply	Functions
P8		I/O	A1/A2	V_{DDP}	Port 8 Port 8 is an 8-bit bi-directional general-purpose I/O port which can be alternatively used for the MLI1 interface or as GPTA I/O lines.
P8.0	H2	O	A2		TCLK1 MLI1 transmit channel clock output
P8.1	H1	I/O	A2		IN40 / OUT40 line of GPTA
		I	A1		TREADY1A MLI1 transmit channel ready input A
P8.2	J3	I/O	A1		IN41 / OUT41 line of GPTA
		O	A2		TVALID1A MLI1 transmit channel valid output A
P8.3	J2	I/O	A2		IN42 / OUT42 line of GPTA
		O	A2		TDATA1 MLI1 transmit channel data output A
P8.4	J1	I/O	A2		IN43 / OUT43 line of GPTA
		I	A1		RCLK1A MLI1 receive channel clock input A
P8.5	K2	I/O	A1		IN44 / OUT44 line of GPTA
		O	A2		RREADY1A MLI1 receive channel ready output A
P8.6	K3	I/O	A2		IN45 / OUT45 line of GPTA
		I	A1		RVALID1A MLI1 receive channel valid input A
P8.7	K1	I/O	A1		IN46 / OUT46 line of GPTA
		I	A1		RDATA1A MLI1 receive channel data input A
		I/O	A1		IN47 / OUT47 line of GPTA

Table 1-3 Pin Definitions and Functions (cont'd)

Symbol	Pins	I/O	Pad Driver Class	Power Supply	Functions
P9		I/O	A2	V_{DDP}	Port 9 Port 9 is a 9-bit bi-directional general-purpose I/O port which can be alternatively used as GPTA or MSC0/1 I/O lines.
P9.0	A19	I/O O			IN48 / OUT48 line of GPTA EN12 MSC1 device select output 2
P9.1	B19	I/O O			IN49 / OUT49 line of GPTA EN11 MSC1 device select output 1
P9.2	B20	I/O O			IN50 / OUT50 line of GPTA SOP1B MSC1 serial data output
P9.3	A20	I/O O			IN51 / OUT51 line of GPTA FCLP1B MSC1 clock output
P9.4	D18	I/O O			IN52 / OUT52 line of GPTA EN03 MSC0 device select output 3
P9.5	D19	I/O O			IN53 / OUT53 line of GPTA EN02 MSC0 device select output 2
P9.6	C19	I/O O			IN54 / OUT54 line of GPTA EN01 MSC0 device select output 1
P9.7	D20	I/O O			IN55 / OUT55 line of GPTA SOP0B MSC0 serial data output
P9.8	C20	O			FCLP0B MSC0 clock output

Table 1-3 Pin Definitions and Functions (cont'd)

Symbol	Pins	I/O	Pad Driver Class	Power Supply	Functions
P10		I	–	V_{DDP}	Hardware Configuration Inputs / Port 10 These inputs are boot mode (hardware configuration) control inputs. <u>They are</u> latched with the rising edge of $\overline{\text{HDRST}}$.
P10.0	A21	I			Port 10 input line 0 / HWCFG0
P10.1	B21	I			Port 10 input line 1 / HWCFG1
P10.2	C21	I			Port 10 input line 2 / HWCFG2
P10.3	D21	I			Port 10 input line 3 / HWCFG3
					After reset ($\overline{\text{HDRST}} = 1$), the state of the Port 10 input pins may be modified from the reset configuration state. Their actual state can be read via software (P10_IN register). During normal operation, input HWCFG1 serves as emergency shut-off control input for certain I/O lines (e.g. GPTA-related outputs).

Dedicated Peripheral I/Os

SLSO0	AE14	O	A2	V_{DDP}	SSC0 Slave Select Output Line 0	
SLSO1	AC15	O			SSC0 Slave Select Output Line 1	
MTR0	AF15	O			I	SSC0 Master Transmit Output / SSC0 Slave Receive Input
		I				
MRST0	AE15	I			O	SSC0 Master Receive Input / SSC0 Slave Transmit Output
		O				
SCLK0	AF14	I/O			SSC0 Clock Input/Output	
SLSI0	AD15	I			SSC0 Slave Select Input	

Table 1-3 Pin Definitions and Functions (cont'd)

Symbol	Pins	I/O	Pad Driver Class	Power Supply	Functions
MSC Outputs					
FCLP0A	C18	O	C	V_{DDP}	LVDS MSC Clock and Data Outputs²⁾ MSC0 differential driver clock output positive A MSC0 differential driver clock output negative MSC0 differential driver serial data output positive A MSC0 differential driver serial data output negative MSC1 differential driver clock output positive A MSC1 differential driver clock output negative MSC1 differential driver serial data output positive A MSC1 differential driver serial data output negative
FCLN0	C17	O			
SOP0A	C16	O			
SON0	D17	O			
FCLP1A	A17	O			
FCLN1	B17	O			
SOP1A	B16	O			
SON1	A16	O			

Table 1-3 Pin Definitions and Functions (cont'd)

Symbol	Pins	I/O	Pad Driver Class	Power Supply	Functions
Analog Inputs					
AN[43:0]		I	D	–	ADC Analog Input Port The ADC Analog Input Port provides 44 analog input lines for the A/D converters ADC0, ADC1, and FADC.
AN0	AE1	I			Analog input 0
AN1	AD2	I			Analog input 1
AN2	AA4	I			Analog input 2
AN3	AB3	I			Analog input 3
AN4	AC2	I			Analog input 4
AN5	AA3	I			Analog input 5
AN6	AD1	I			Analog input 6
AN7	AB4	I			Analog input 7
AN8	AC1	I			Analog input 8
AN9	AB2	I			Analog input 9
AN10	Y3	I			Analog input 10
AN11	AA2	I			Analog input 11
AN12	AB1	I			Analog input 12
AN13	W3	I			Analog input 13
AN14	Y2	I			Analog input 14
AN15	AA1	I			Analog input 15
AN16	V4	I			Analog input 16
AN17	W2	I			Analog input 17
AN18	Y1	I			Analog input 18
AN19	V3	I			Analog input 19
AN20	W1	I			Analog input 20
AN21	V2	I			Analog input 21
AN22	V1	I			Analog input 22
AN23	U1	I			Analog input 23
AN24	AC8	I			Analog input 24
AN25	AD8	I			Analog input 25
AN26	AC7	I			Analog input 26
AN27	AD7	I			Analog input 27
AN28	AE6	I			Analog input 28
AN29	AF6	I			Analog input 29
AN30	AE7	I			Analog input 30
AN31	AF7	I			Analog input 31

Table 1-3 Pin Definitions and Functions (cont'd)

Symbol	Pins	I/O	Pad Driver Class	Power Supply	Functions
			D	–	ADC Analog Input Port (cont'd)
AN32	AC3	I			Analog input 32
AN33	AE2	I			Analog input 33
AN34	AD3	I			Analog input 34
AN35	AD5	I			Analog input 35
AN36	AE3	I			Analog input 36
AN37	AF2	I			Analog input 37
AN38	AC4	I			Analog input 38
AN39	AF3	I			Analog input 39
AN40	AD4	I			Analog input 40
AN41	AE4	I			Analog input 41
AN42	AC5	I			Analog input 42
AN43	AF4	I			Analog input 43
TR[15:0]		O	A3	V_{DDP}	OCDS Level 2 Debug Trace Lines²⁾ (located on center balls)
TR0	U12	O			Trace output line 0
TR1	T12	O			Trace output line 1
TR2	U11	O			Trace output line 2
TR3	T11	O			Trace output line 3
TR4	U10	O			Trace output line 4
TR5	R12	O			Trace output line 5
TR6	R10	O			Trace output line 4
TR7	R11	O			Trace output line 7
TR8	M11	O			Trace output line 8
TR9	M10	O			Trace output line 9
TR10	L11	O			Trace output line 10
TR11	L10	O			Trace output line 11
TR12	K10	O			Trace output line 12
TR13	K11	O			Trace output line 13
TR14	L12	O			Trace output line 14
TR15	K12	O			Trace output line 15
TRCLK	T10	O	A4	V_{DDP}	Trace Clock for OCDS Level 2 Debug Trace Lines²⁾ (located on a center ball)

Table 1-3 Pin Definitions and Functions (cont'd)

Symbol	Pins	I/O	Pad Driver Class	Power Supply	Functions
System I/O					
<u>TRST</u>	F23	I	A2	V_{DDP}	JTAG Module Reset/Enable Input³⁾
<u>TCK</u>	E24	I	A2		JTAG Module Clock Input³⁾
<u>TDI</u>	E25	I	A1		JTAG Module Serial Data Input
<u>TDO</u>	D25	O	A2		JTAG Module Serial Data Output
<u>TMS</u>	F24	I	A1		JTAG Module State Machine Control Input
<u>BRKIN</u>	C26	I/O	A3		OCDS Break Input (Alternate Output)³⁾
<u>BRK OUT</u>	D26	I/O	A3		OCDS Break Output (Alternate Input)³⁾
<u>NMI</u>	A22	I	– ⁴⁾		Non-Maskable Interrupt Input
<u>HDRST</u>	A23	I/O	A2 ⁵⁾		Hardware Reset Input/ Reset Indication Output
<u>PORST</u>	B22	I	– ⁴⁾		Power-on Reset Input
<u>BYPASS</u>	A24	I	–		PLL Bypass Select Input This input has to be held stable between two power-on resets. With <u>BYPASS</u> = 1 the spike filters in the <u>HDRST</u> , <u>PORST</u> , and <u>NMI</u> inputs are switched off.
<u>TEST MODE</u>	B23	I	– ⁶⁾		Test Mode Select Input For normal operation of the TC1796, this pin should be connected to high level.
<u>TSTRES</u>	G24	I	– ⁶⁾		Test Reset Input For normal operation of the TC1796, this pin must be connected to low level. Otherwise an unpredictable reset behavior may occur.
<u>XTAL1</u> <u>XTAL2</u>	G26 G25	I O	n.a.	V_{DD}	Oscillator / PLL / Clock Generator Input / Output Pins³⁾

Table 1-3 Pin Definitions and Functions (cont'd)

Symbol	Pins	I/O	Pad Driver Class	Power Supply	Functions
N.C.	A1 C22 G23 H3 AF1 AF26 AC21 AD23 AE22 AE23	–	–	–	Not Connected These pins are reserved for future extension and should not be connected externally.
Power Supplies					
V_{DDM}	W4	–	–	–	ADC0/1 Analog Part Power Supply (3.3 V)
V_{SSM}	Y4	–	–	–	ADC0/1 Analog Part Ground for V_{DDM}
V_{DDMF}	AE9	–	–	–	FADC Analog Part Power Supply (3.3 V)
V_{SSMF}	AF9	–	–	–	FADC Analog Part Ground for V_{DDAF}
V_{DDAF}	AC9	–	–	–	FADC Analog Part Logic Power Supply (1.5 V)
V_{SSAF}	AD9	–	–	–	FADC Analog Part Log Ground for V_{DDAF}
V_{AREF0}	AE5	–	–	–	ADC0 Reference Voltage
V_{AGND0}	AF5	–	–	–	ADC0 Reference Ground
V_{AREF1}	AD6	–	–	–	ADC1 Reference Voltage
V_{AGND1}	AC6	–	–	–	ADC1 Reference Ground
V_{FAREF}	AF8	–	–	–	FADC Reference Voltage
V_{FAGND}	AE8	–	–	–	FADC Reference Ground
V_{DDOSC}	F26	–	–	–	Main Oscillator Power Supply (1.5 V)
V_{DDOSC3}	E26	–	–	–	Main Oscillator Power Supply (3.3 V)
V_{SSOSC}	F25	–	–	–	Main Oscillator Ground
V_{DDFL3}	A18 B18	–	–	–	Power Supply for Flash (3.3 V)
V_{DD} SBRAM	R1	–	–	–	Power Supply for Stand-by SRAM (1.5 V)

Table 1-3 Pin Definitions and Functions (cont'd)

Symbol	Pins	I/O	Pad Driver Class	Power Supply	Functions
V_{DDEBU}	H23 H24 H25 H26 M23 T23 Y23 AC18 AC22	–	–	–	EBU Power Supply (2.3 - 3.3 V)
V_{DD}	B26 C25 D9 D16 D24 E23 H4 P23 R4 V23 AB23 AC11 AC20	–	–	–	Core Power Supply (1.5 V)
V_{DDP}	A25 B24 C23 D7 D14 D22 K4 AC16 AD16 AE16 AF16	–	–	–	Port Power Supply (3.3 V) (also for OCDS)
V_{SS}	see Table 1-4	–	–	–	Ground 15 V_{SS} lines are located on outer balls. 47 V_{SS} lines are located on center balls.

Introduction

- 1) In order to minimize noise coupling to the on-chip A/D converters, it is recommended to use these pins as less as possible in strong driver mode.
- 2) In case of a power-fail condition (one or more power supply voltages drop below the specified voltage range), an undefined output driving level may occur at these pins.
- 3) In case of a power-fail condition (one or more power supply voltages drop below the specified voltage range), an undefined output driving level may occur at these pins even if they are input-only pins.
- 4) Input pad with input spike-filter.
- 5) Open drain pad with input spike-filter.
- 6) Input pad, test function only, without input spike-filter.

Table 1-4 V_{SS} Balls

V_{SS} Outer Balls	V_{SS} Center Balls
A26, B25, C24, D8, D15, D23, J4, L23, R23, T4, W23, AC10, AC17, AC19, AC23	K[17:13], L[17:13], M[17:12], N[17:10], P[17:10], R[17:13], T[17:13], U[17:13]

1.4.2 Pad Driver Classes Overview

Table 1-5 gives an overview of the pad driver classes of the TC1796. Further details can be found in the “TC1796 Data Sheet”.

Table 1-5 Pad Driver and Input Classes Overview

Class	Power Supply	Type	Sub Class	Speed Grade	Load	Termination
A	3.3 V	LVTTL I/O, LVTTL outputs	A1 (e.g. GPIO)	6 MHz	100 pF	No
			A2 (e.g. serial I/Os)	40 MHz	50 pF	Series termination recommended
			A3 (e.g. Trace outputs, serial I/Os)	75 MHz	50 pF	Series termination recommended (for $f > 25$ MHz)
			A4 (e.g. Trace Clock)	150 MHz	25 pF	Series termination recommended
B	2.375 / 3.6 V ¹⁾	LVTTL I/O	B1 (e.g. Bus Interface)	40 MHz	50 pF	No
			B2 (e.g. Bus Clock)	75 MHz	35 pF	Series termination recommended (for $f > 25$ MHz)
C	3.3 V	LVDS	–	50 MHz	–	Parallel termination ²⁾ , 100 $\Omega \pm 10\%$
D	–	Analog input, reference voltage inputs.				

1) AC characteristics for EBU pins are valid for 2.5 V \pm 5% and 3.3 V \pm 5%.

2) In applications where the LVDS pins are not used (disabled), these pins must be either left unconnected or properly terminated with the differential parallel termination of 100 $\Omega \pm 10\%$.

1.4.3 Pull-Up/Pull-Down Behavior of the Pins

Table 1-6 List of Pull-Up/Pull-Down Reset Behavior of the Pins

Pins	$\overline{\text{PORST}} = 0$	$\overline{\text{PORST}} = 1$
$\overline{\text{TSTRES}}$, $\overline{\text{TDI}}$, $\overline{\text{TMS}}$, $\overline{\text{TESTMODE}}$, $\overline{\text{BRKOUT}}$, $\overline{\text{BRKIN}}$, all GPIOs, $\overline{\text{RD}}$, $\overline{\text{RD/WR}}$, $\overline{\text{ADV}}$, $\overline{\text{BC}[3:0]}$, $\overline{\text{MR/W}}$, $\overline{\text{WAIT}}$, $\overline{\text{BAA}}$, $\overline{\text{HOLD}}$, $\overline{\text{HLDA}}$, $\overline{\text{BREQ}}$, $\overline{\text{D}[31:0]}$, $\overline{\text{A}[23,0]}$, $\overline{\text{CS}[3:0]}$, $\overline{\text{CSCOMB}}$	Weak pull-up device active	
$\overline{\text{NMI}}$, $\overline{\text{PORST}}$	Weak pull-down device active	
$\overline{\text{BYPASS}}$, $\overline{\text{SLSO0}}$, $\overline{\text{SLSO1}}$, $\overline{\text{MTSR0}}$, $\overline{\text{MRST0}}$, $\overline{\text{SCLK0}}$, $\overline{\text{SLSI0}}$, $\overline{\text{TDO}}$, $\overline{\text{BFCLKI}}$	Weak pull-up device active	High-impedance
$\overline{\text{BFCLKO}}$	Weak pull-up device active	Push-pull driver active
$\overline{\text{HDRST}}$	Open-drain device drives 0 (strong pull-down)	Weak pull-up device active Open-drain device active
$\overline{\text{TRST}}$, $\overline{\text{TCK}}$	High-impedance	Weak pull-down device active

2 CPU Subsystem

The TC1796 processor contains a TriCore 1 V1.3 CPU. This chapter describes the implementation-specific options of the CPU, and should be read in conjunction with the TriCore 1 Architecture Manual, which describes the complete TriCore Architecture including the register and instruction set description.

2.1 TC1796 Processor Subsystem

The diagram below shows the block diagram of the TC1796 Processor subsystem. It also shows the on-chip bus systems.

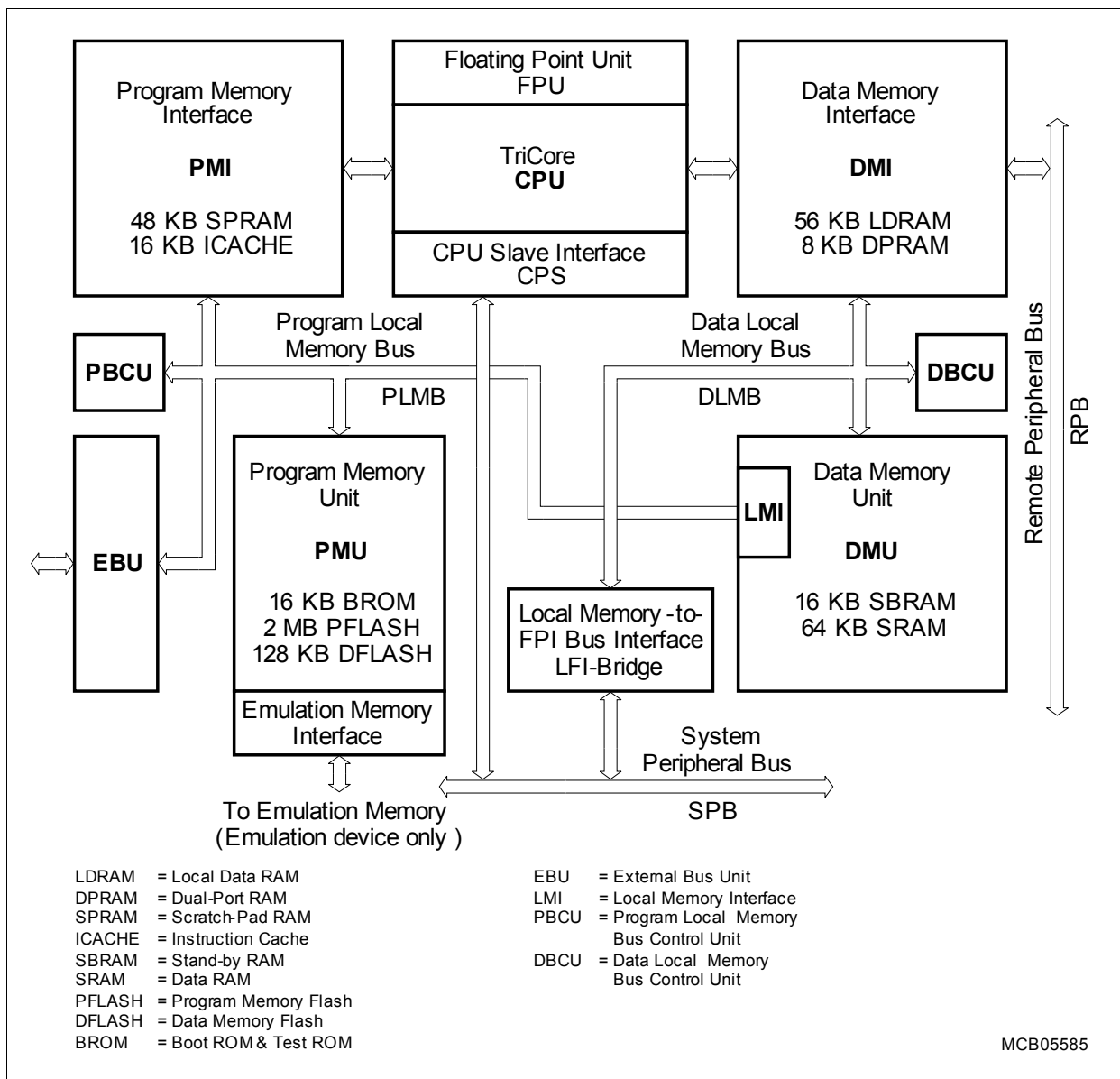


Figure 2-1 Processor Subsystem Block Diagram

2.2 Central Processing Unit Features

The 150 MHz TriCore TC1796 CPU includes:

Architecture

- 32-bit load/store architecture
- 4 Gbyte address range (2^{32})
- 16-bit and 32-bit instructions for reduced code size
- Data types:
 - Boolean, integer with saturation, bit array, signed fraction, character, double-word integers, signed integer, unsigned integer, IEEE-754 single-precision floating point
- Data formats:
 - Bit, byte (8-bit), half-word (16-bit), word (32-bit), double-word (64-bit)
- Byte and bit addressing
- Little-endian byte ordering for data, memory and CPU registers
- Multiply and Accumulate (MAC) instructions:
 - Dual 16 x 16, 16 x 32, 32 x 32
- Saturation integer arithmetic
- Packed data
- Addressing modes:
 - Absolute, circular, bit reverse, long + short, base + offset with pre and post-update
- Instruction types:
 - Arithmetic, address arithmetic, comparison, address comparison, logical, MAC, shift, coprocessor, bit logical, branch, bit field, load/store, packed data, system
- General Purpose Register Set (GPRS):
 - Sixteen 32-bit data registers
 - Sixteen 32-bit address registers
 - Three 32-bit status and program counter registers (PSW, PC, PCXI)
- Core Debug support (OCDS):
 - Level 1 and 2, supported in conjunction with the CPS block
 - Level 3, supported

Implementation

- Most instructions executed in 1 cycle
- Branch instructions in 1, 2 or 3 cycles (using branch prediction)
- Shadow registers for fast context switch
- Automatic context save-on-entry and restore-on-exit for: subroutine, interrupt, trap
- Two memory protection register sets
- Dual instruction issuing (in parallel into Integer Pipeline and Load/Store Pipeline)
- Third pipeline for loop instruction only (zero overhead loop)
- Optional floating point instruction set implemented

- Optional Memory Management instruction set not implemented
(Memory management configuration registers are always read as MMU not present)

2.2.1 CPU Diagram

The Central Processing Unit (CPU) is comprised of an Instruction Fetch Unit, an Execution Unit, a General Purpose Register File (GPR), a CPU Slave interface (CPS), and optional Floating Point Unit (FPU).

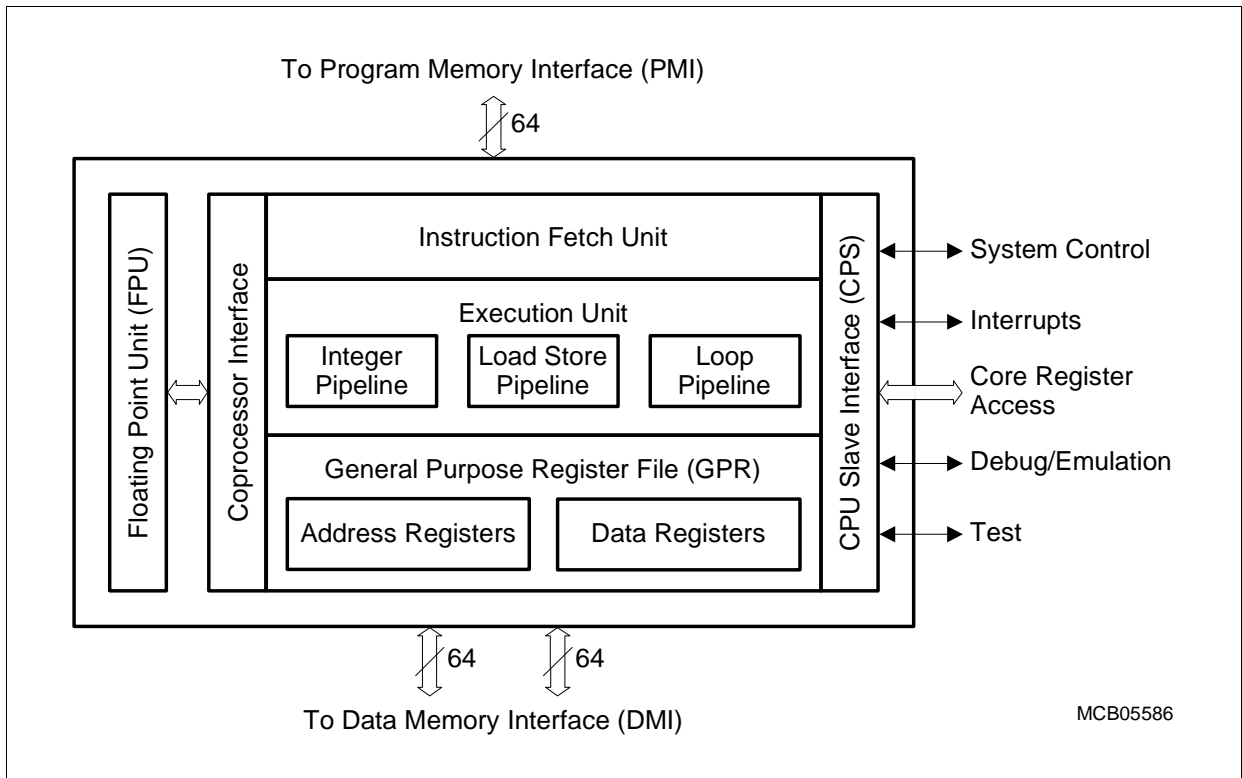


Figure 2-2 CPU Block Diagram

2.2.2 Instruction Fetch Unit

The Instruction Fetch Unit pre-fetches and aligns incoming instructions from the 64-bit wide Program Memory Interface (PMI). The Issue Unit directs the instruction to the appropriate pipeline. The Instruction Protection Unit checks the validity of accesses to the PMI and also checks for instruction breakpoint conditions. The PC Unit is responsible for updating the program counters.

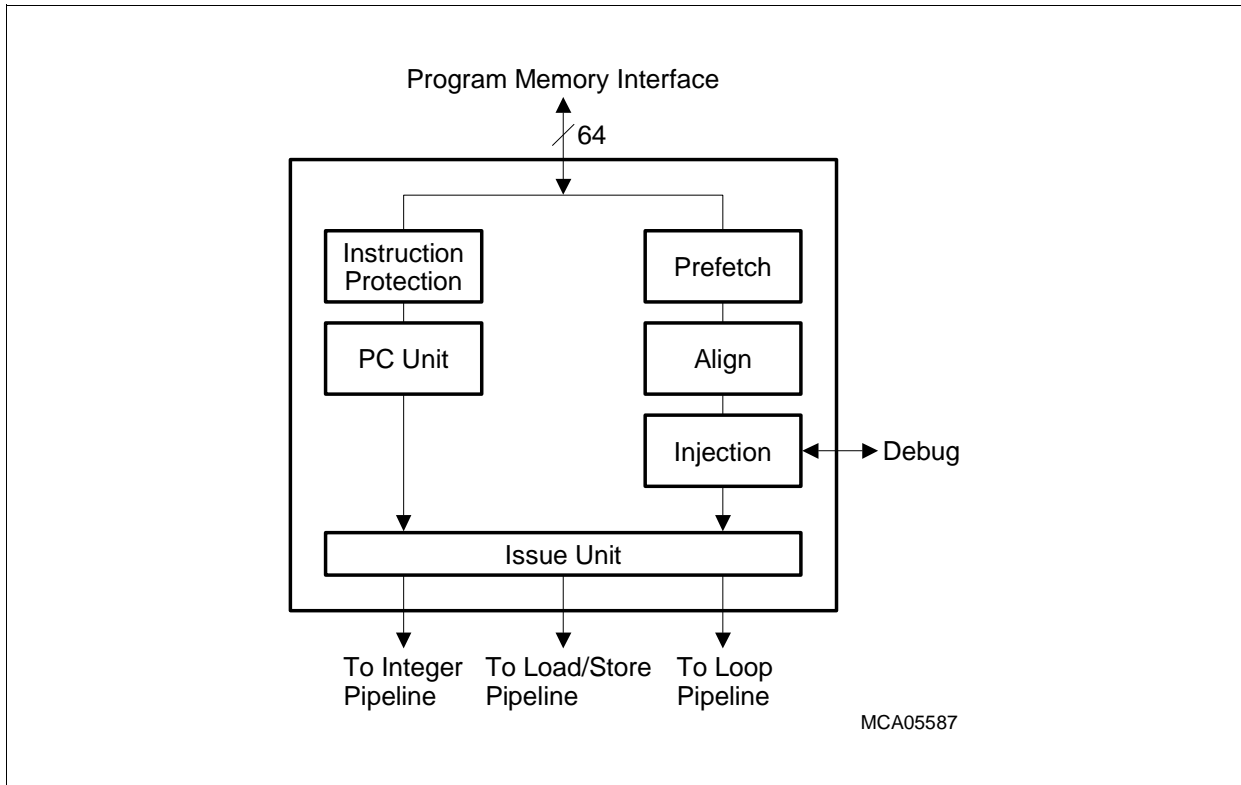


Figure 2-3 Instruction Fetch Unit

2.2.3 Execution Unit

The Execution Unit contains the Integer Pipeline, the Load/Store Pipeline and the Loop Pipeline.

The Integer Pipeline and Load/Store Pipeline have four stages: Fetch, Decode, Execute, and Write-back. The Execute stage may extend beyond one cycle to accommodate multi-cycle operations such as load instructions.

The Loop Pipeline has two stages: Decode and Write-back.

All three pipelines operate in parallel, permitting up to three instructions to be executed in one clock cycle.

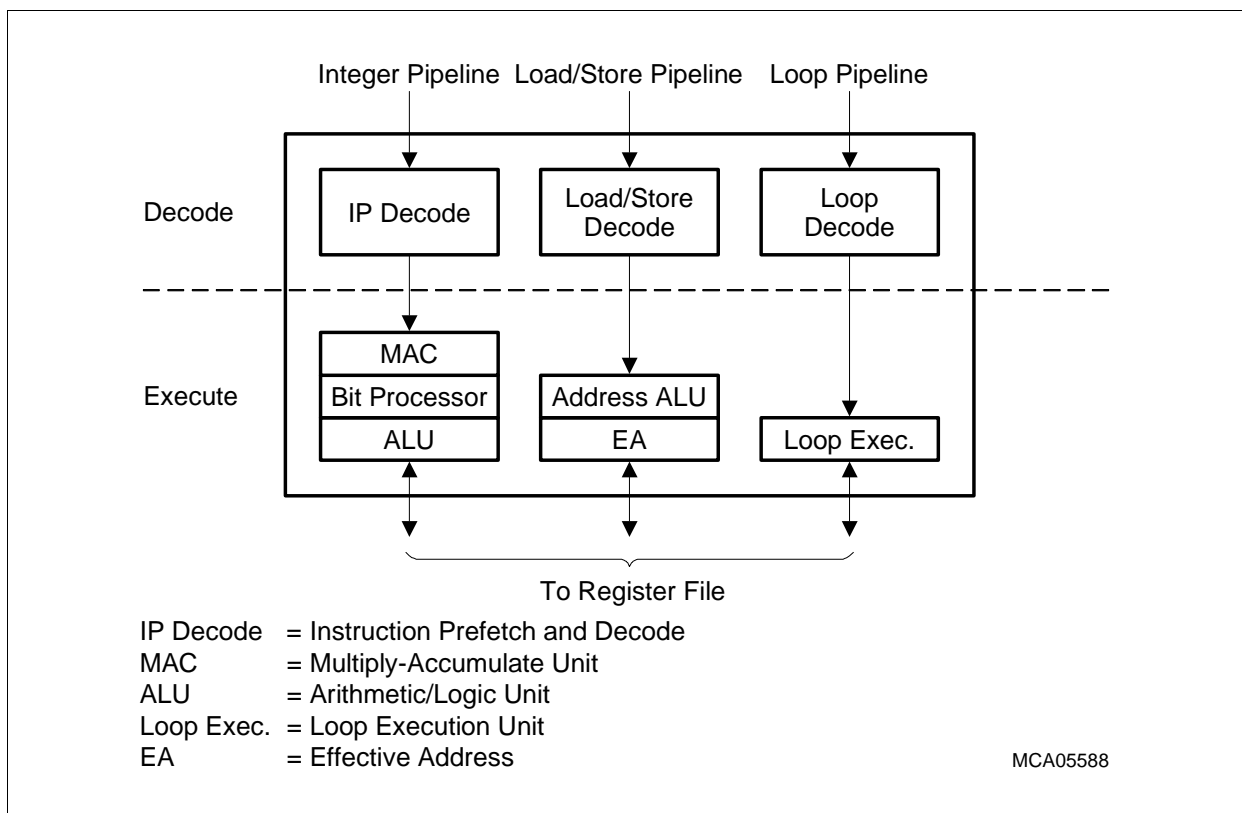


Figure 2-4 Execution Unit

2.2.4 General Purpose Register File

The CPU has a General Purpose Register (GPR) file, divided into an Address Register File (registers A0 through A15) and a Data Register File (registers D0 through D15).

The data flow for instructions issued to the Load/Store Pipeline is steered through the Address Register File.

The data flow for instructions issued to/from the Integer Pipeline and for data load/store instructions issued to the Load/Store Pipeline is steered through the Data Register File.

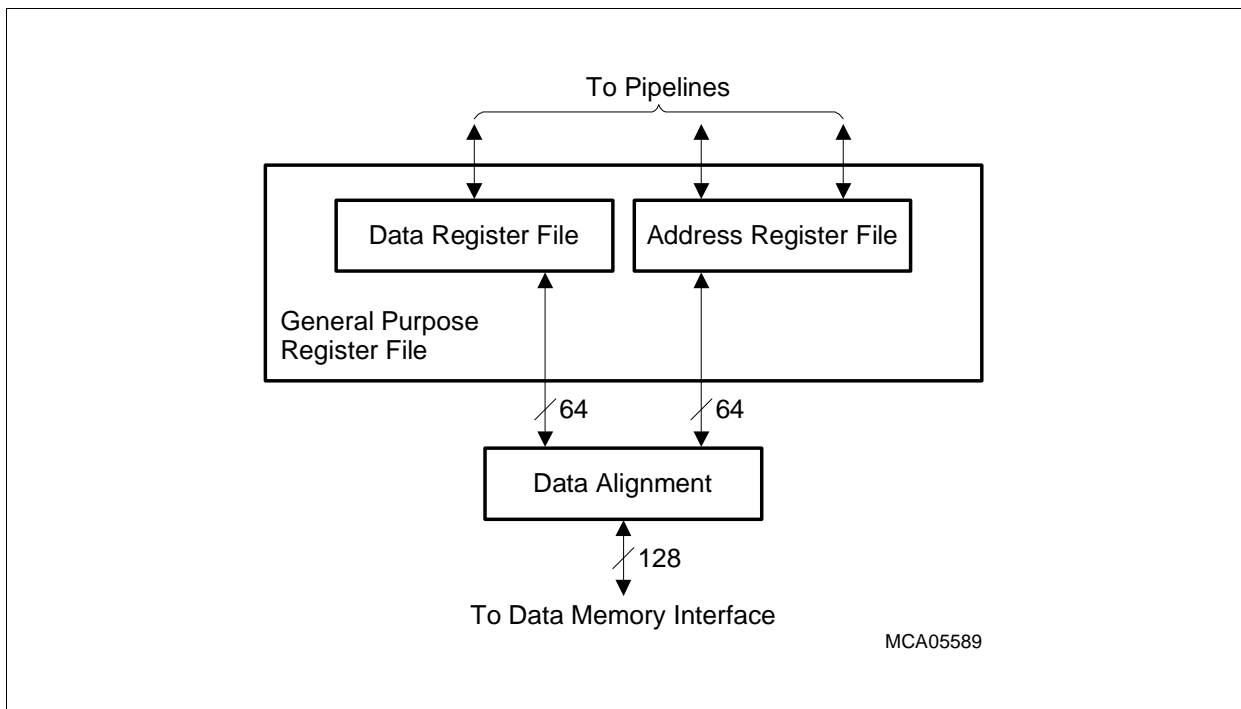


Figure 2-5 General Purpose Register File

2.3 Implementation-specific Features

This section describes the implementation-specific features of the TC1796 CPU. For a full description of the TriCore architecture refer to the TriCore 1 Architecture Manual.

2.3.1 Context Save Areas

In the TC1796, Context Save Areas (CSA) must be placed in LDRAM. CSAs should not be located in the DPRAM (dual-ported RAM).

Refer to the TriCore 1 Architecture Manual Chapter 5 - Tasks and Functions.

2.3.2 Fast Context Switching

The TC1796 uses a uniform context-switching method for function calls, interrupts and traps. In all cases, the Upper Context of the task is automatically saved and restored by hardware. Saving and restoring of the Lower Context may be performed optionally by software.

Fast context switching is enhanced by the unique memory subsystem design. When they are not full, the shadow registers allow a complete Upper Context to be saved in as few as two clock cycles. When the shadow registers are full, the upper context save takes up to five cycles. On the average, an upper context save takes 2.7 cycles. Shadow registers are automatically restored from memory when required.

2.3.3 Reset System

Several events can cause the TC1796 system to be reset. The CPU does not differ in its behavior on reset. The status register RST_SR allows the CPU to determine which event caused the reset. Refer to [Chapter 4](#) of this TC1796 User's Manual.

2.3.4 Program Counter Register - PC

The Program Counter (PC) holds the address of the instruction that is currently fetched and forwarded to the CPU pipelines. The CPU handles updates of the PC automatically. Software can use the current value of the PC for various tasks, such as performing code address calculations. Reading the PC through software executed by the CPU must only be done with an MFCR instruction. Explicit writes to the PC through an MTCR instruction must not be done due to possible unexpected behavior of the CPU.

The CPU must not perform Load/Store instructions to the mapped address of the PC in Segment 15. A MEM trap will be generated in such a case.

Bit 0 of the PC register is read-only and hard-wired to 0.

Refer to the TriCore 1 Architecture Manual - Core Registers.

2.3.5 Interrupt System

An interrupt request can be generated by the TC1796 on-chip peripheral units, or it can be generated by external events. Requests can be targeted to either the CPU, or to the Peripheral Control Processor (PCP).

The TC1796 interrupt system evaluates service requests for priority and to identify whether the CPU or PCP should receive the request. The highest-priority service request is then presented to the CPU (or PCP) by way of an interrupt.

The term “interrupt” is used generally to mean an event directed to the CPU, while the term “service request” describes an event that can be directed to either the CPU or the PCP. For more information, refer to [Chapter 11](#) of this TC1796 User’s Manual.

2.3.6 Trap System

The following traps have implementation-specific properties. For a complete description of the trap system, refer to the TriCore 1 Architecture Manual - Trap System.

UOPC - Unimplemented Opcode (TIN 2)

The TC1796 UOPC trap is raised on optional MMU instructions, coprocessor two and coprocessor three instructions.

OPD - Invalid Operand (TIN 3)

The TC1796 CPU does not raise OPD traps.

DSE - Data Access Synchronous Error (TIN 2)

The Data Access Synchronous Bus Error (DSE) trap is generated by the DMI on a DMI control register access error, LMB Bus access error, or a DMI scratch memory range error. The exact cause of the error can be read in the DMI Synchronous Trap Flag Register, DMI_STR. DSE traps occur on load accesses.

DAE - Data Access Asynchronous Error (TIN 3)

The Data Access Asynchronous Error Trap (DAE) is generated by the DMI on either a DMI control register access error, LMB Bus access error, or a DMI scratch memory range error. The exact cause of the error can be read via the DMI Asynchronous Trap Flag Register, DMI_ATR. DAE traps occur on store accesses.

2.4 TC1796 CPU Subsystem Registers

This section only describes the implementation-specific features of the registers listed in [Table 2-1](#). For complete descriptions of all registers, please refer to the TriCore 1 Architecture Manual.

TC1796 implementation-specific CPU registers are referred directly in this section.

Table 2-1 CPU and Processor Subsystem Registers

Registers	Purpose	Description	Address Map
Core Special Function Registers (CSFRs)	Program state information, context and stack management, interrupt and trap control, system control	Architecture Manual	see Page 18-108
CPU Slave Interface Registers (CPSs)	Software break control and software service request control		see Page 18-104
Core General Purpose Registers (GPRs)	Address and data		see Page 18-105
Core Debug Registers (OCDS)	Debug control		see Page 18-108
Memory Protection Registers	Memory protection control and mode selection		see Page 18-105
Program Memory Interface Registers (PMI)	PMI instruction cache control and status	see Page 2-26	see Page 18-124
Data Memory Interface Registers (DMI)	DMI status and trap flags	see Page 2-34	see Page 18-123

The complete and detailed address map of the CPU and processor subsystem registers shown in [Table 2-1](#) above is located in [Chapter 18](#). The entries in column “Address Map” of [Table 2-1](#) directly point to the corresponding pages.

2.4.1 Core Special Function Registers (CSFR)

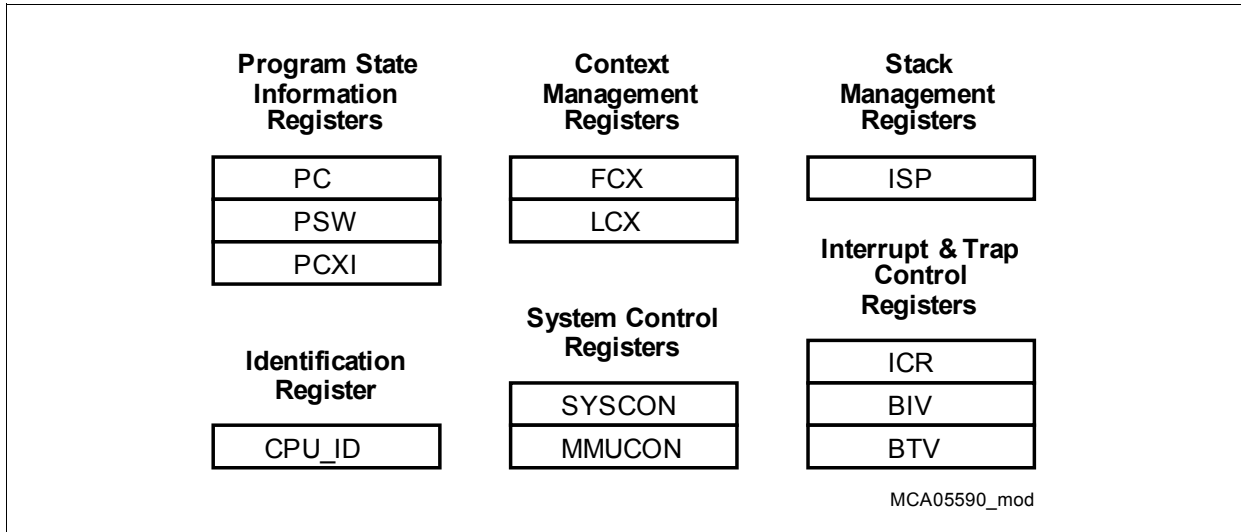


Figure 2-6 CSFR Registers

Table 2-2 Core Special Function Registers

Register Short Name	Register Long Name	Address
MMU_CON	MMU Configuration Register	F7E1 8000 _H
PCXI	Previous Context Information Register	F7E1 FE00 _H
PSW	Program Status Word	F7E1 FE04 _H
PC	Program Counter	F7E1 FE08 _H
SYSCON	System Configuration Register	F7E1 FE14 _H
CPU_ID	CPU Identification Register	F7E1 FE18 _H
BIV	Interrupt Vector Table Pointer	F7E1 FE20 _H
BTV	Trap Vector Table Pointer	F7E1 FE24 _H
ISP	Interrupt Stack Pointer	F7E1 FE28 _H
ICR	ICU Interrupt Control Register	F7E1 FE2C _H
FCX	Free Context List Head Pointer	F7E1 FE38 _H
LCX	Free Context List Limit Pointer	F7E1 FE3C _H

2.4.1.1 Implementation-specific Core Special Function Registers

This section describes the implementation-specific Program Status Word (PSW) which is an extension of the PSW description in the TriCore 1 Architecture Manual.

The PSW status flags used for FPU operations overlay the status flags used for Arithmetic Logic Unit (ALU) operations of the CPU. The non-shaded areas in the PSW register description define the implementation-specific bits/bit fields.

PSW

Program Status Word

(F7E1FE04_H)

Reset Value: 0000 0B80_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C	V	SV	AV	SAV	FX	RM		0							
or	or	or	or	or											
FS	FI	FV	FZ	FU											
rwh	rwh	rwh	rwh	rwh	rwh	rw		r							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		PRS		IO		IS	GW	CDE	CDC						
r		rwh		rwh		rwh	rwh	rwh	rwh						

Field	Bits	Type	Description
RM	[25:24]	rw	FPU Rounding Mode Selection
FX	26	rwh	FPU Inexact Flag
SAV	27	rh	Sticky Advance Overflow Flag
FU		rwh	FPU Underflow Flag
AV	28	rwh	Advance Overflow Flag
FZ			FPU Divide by Zero Flag
SV	29	rwh	Sticky Overflow Flag
FV			FPU Overflow Flag
V	30	rwh	Overflow Flag
FI			FPU Invalid Operation Flag
C	31	rwh	Carry Flag
FS			FPU Some Exception Flag

The Interrupt Control Register is also an implementation-specific CSFR. Its Arbitration Cycle Control implementation-specific details are described on [Page 14-8](#).

CPU Subsystem

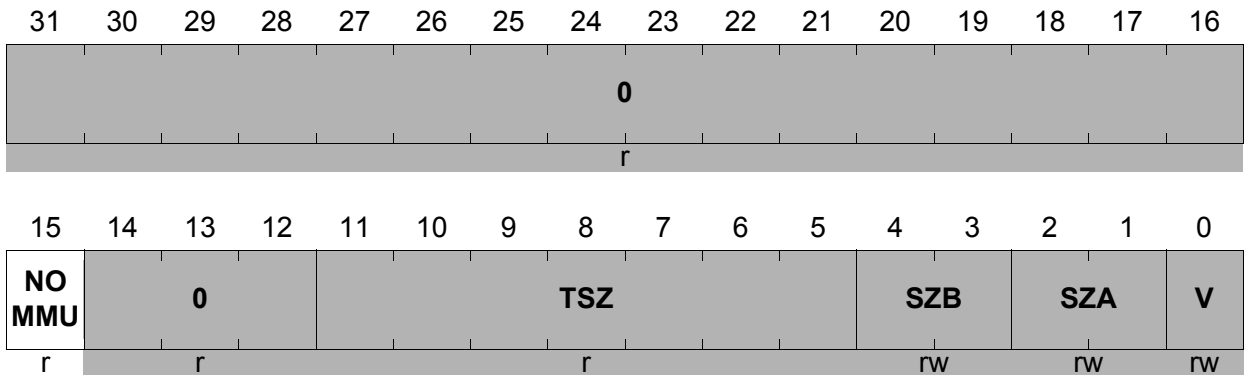
In the TC1796, the MMU_CON register indicates the non-availability of the TriCore 1's memory management unit (bit MXT is always set).

MMU_CON

MMU Configuration Register

(F7E18000_H)

Reset Value: 0000 8000_H



Field	Bits	Type	Description
NOMMU	15	r	No MMU Available 0 _B MMU is available 1 _B MMU is not available. All other bits of MMU_CON are undefined.
0	[14:0], [31:16]	r	Reserved Read as 0; should be written with 0.

2.4.2 CPU Slave Interface (CPS) Registers

In the TC1796, the CPU Slave Interface (CPS) of the TriCore CPU directly accesses the interrupt service request registers in the CPU from the TC1796 System Peripheral Bus. The CPS registers are described in detail in the TriCore 1 Architecture Manual - Core Registers.



Figure 2-7 CPS Registers

The absolute register addresses are calculated by adding the offset addresses from [Table 2-3](#) to the CPS Base Address. The registers CPU_SBSRC0 and CPU_SRC[3:0] are not bit-addressable.

Table 2-3 CPS Registers

Register Short Name	Register Long Name	Address
CPS_ID	CPS Module Identification Register	F7E0 FF08 _H
CPU_SBSRC0 ¹⁾	CPU Software Breakpoint Service Request Control Register 0	F7E0 FFBC _H
CPU_SRC3	CPU Service Request Control Register 3	F7E0 FFF0 _H
CPU_SRC2	CPU Service Request Control Register 2	F7E0 FFF4 _H
CPU_SRC1	CPU Service Request Control Register 1	F7E0 FFF8 _H
CPU_SRC0	CPU Service Request Control Register 0	F7E0 FFFC _H

1) CPU_SBSRC[3:1] are not implemented in the TC1796. Implementation-specific details see [Page 2-19](#).

2.4.2.1 Implementation-specific CPU Slave Interface Registers

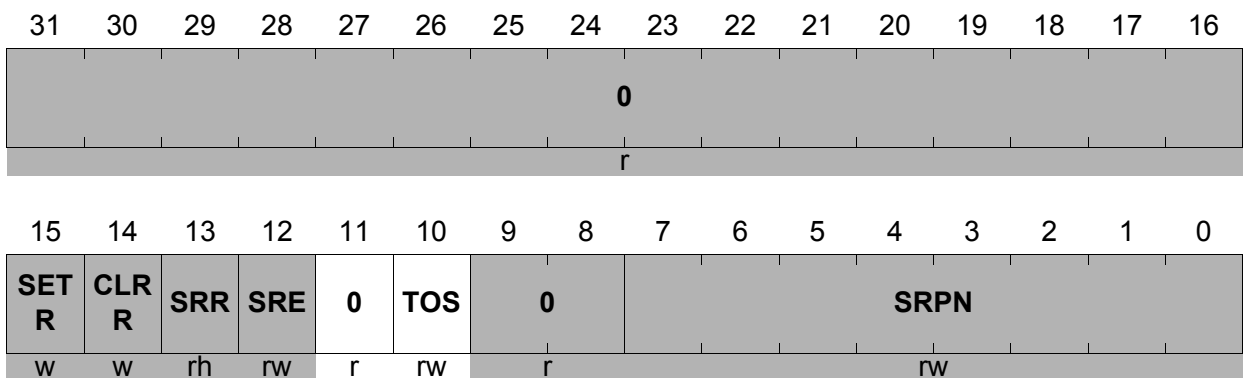
All registers from [Table 2-3](#) have a TC1796-specific implementation detail, the Type of Service Control (TOS) bit/bit field. CPU_SBSRC0 is described on [Page 2-19](#). The non-shaded areas in the CPU_SRCn register description defines the implementation-specific bits/bit fields.

CPU_SRCn (n = 0-3)

CPU Service Request Control Register n

(F7E0FFFC_{H-n*4_H})

Reset Value: 0000 0000_H



Field	Bits	Type	Description
TOS	10	rw	Type of Service Control 0 _B Service Provider = CPU 1 _B Service Provider = PCP
0	11	r	Reserved Read as 0; should be written with 0.

2.4.3 CPU General Purpose Registers

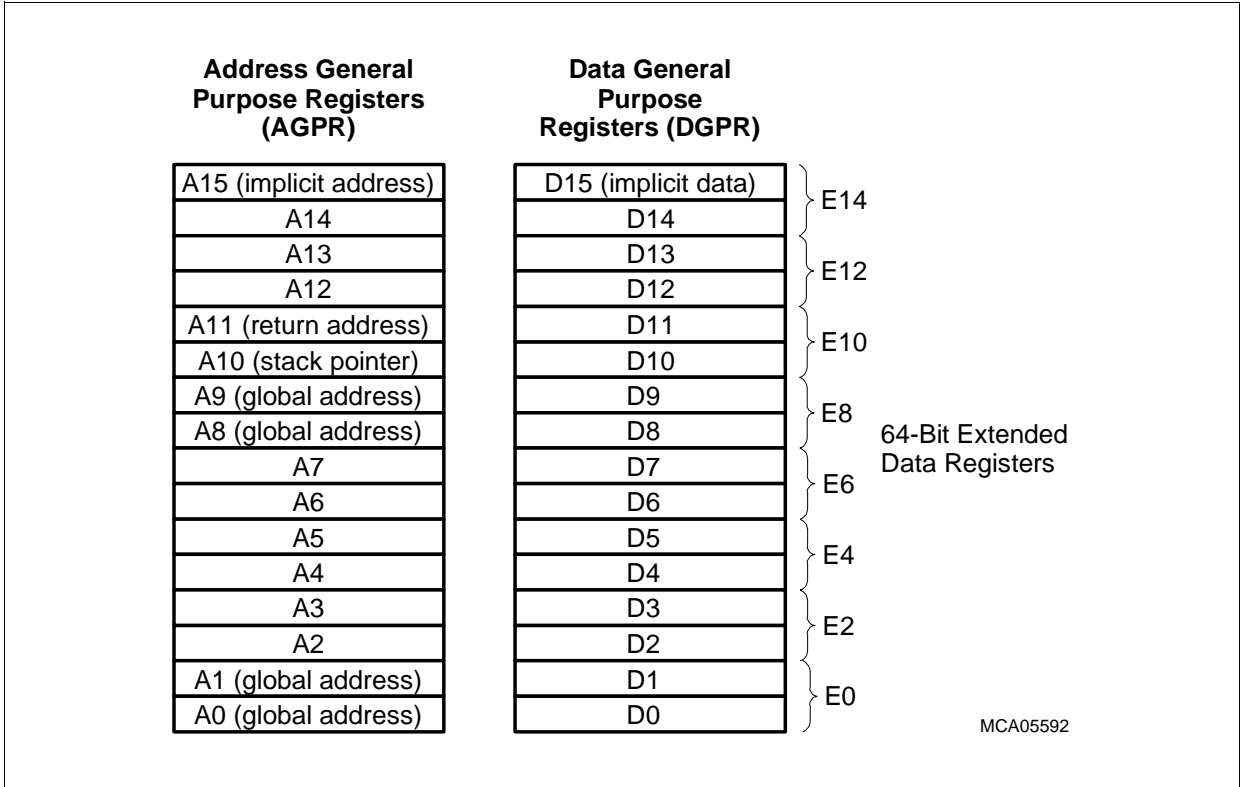


Figure 2-8 GPR Registers

Table 2-4 GPR Registers

Register Short Name	Register Long Name	Address
D0	Data Register 0	F7E1 FF00 _H
D1	Data Register 1	F7E1 FF04 _H
D2	Data Register 2	F7E1 FF08 _H
D3	Data Register 3	F7E1 FF0C _H
D4	Data Register 4	F7E1 FF10 _H
D5	Data Register 5	F7E1 FF14 _H
D6	Data Register 6	F7E1 FF18 _H
D7	Data Register 7	F7E1 FF1C _H
D8	Data Register 8	F7E1 FF20 _H
D9	Data Register 9	F7E1 FF24 _H
D10	Data Register 10	F7E1 FF28 _H

Table 2-4 GPR Registers (cont'd)

Register Short Name	Register Long Name	Address
D11	Data Register 11	F7E1 FF2C _H
D12	Data Register 12	F7E1 FF30 _H
D13	Data Register 13	F7E1 FF34 _H
D14	Data Register 14	F7E1 FF38 _H
D15	Data Register 15	F7E1 FF3C _H
A0	Address Register 0 (Global Address Register)	F7E1 FF80 _H
A1	Address Register 1 (Global Address Register)	F7E1 FF84 _H
A2	Address Register 2	F7E1 FF88 _H
A3	Address Register 3	F7E1 FF8C _H
A4	Address Register 4	F7E1 FF90 _H
A5	Address Register 5	F7E1 FF94 _H
A6	Address Register 6	F7E1 FF98 _H
A7	Address Register 7	F7E1 FF9C _H
A8	Address Register 8 (Global Address Register)	F7E1 FFA0 _H
A9	Address Register 9 (Global Address Register)	F7E1 FFA4 _H
A10	Address Register 10 (Stack Pointer)	F7E1 FFA8 _H
A11	Address Register 11 (Return Address)	F7E1 FFAC _H
A12	Address Register 12	F7E1 FFB0 _H
A13	Address Register 13	F7E1 FFB4 _H
A14	Address Register 14	F7E1 FFB8 _H
A15	Address Register 15	F7E1 FFBC _H

2.4.4 Core Debug Registers

In the TC1796, several Core Debug registers are available for debug purposes. These Core Debug registers are described in detail in the TriCore 1 Architecture Manual chapter - Core Debug Controller.

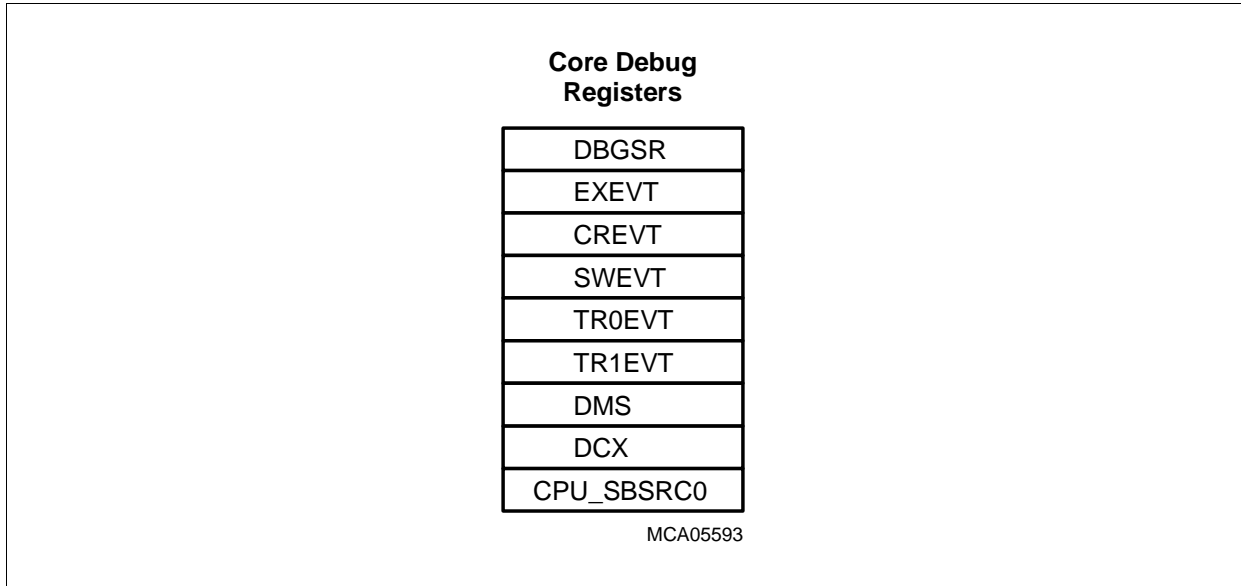


Figure 2-9 Core Debug Registers

Table 2-5 Core Debug Registers

Register Short Name	Register Long Name	Address
DBGSR	Debug Status Register	F7E1 FD00 _H
EXEVT	External Break Input Event Specifier Register	F7E1 FD08 _H
CREVT	Core SFR Access Break Event Specifier Register	F7E1 FD0C _H
SWEVT	Software Break Event Specifier Register	F7E1 FD10 _H
TR0EVT	Trigger Event 0 Specifier Register	F7E1 FD20 _H
TR1EVT	Trigger Event 1 Specifier Register	F7E1 FD24 _H
DMS	Debug Monitor Start Address Register	F7E1 FD40 _H
DCX	Debug Context Save Area Pointer	F7E1 FD44 _H
CPU_SBSCR0	CPU Software Breakpoint Service Request Control Register 0	see Table 2-3 ¹⁾

1) Located in the CPU slave (CPS) interface register area.

2.4.4.1 Implementation-specific Core Debug Registers

This section describes the implementation-specific Core Debug Registers which differ from the description in the TriCore 1 Architecture Manual. These are:

- Trigger Event 0 Specifier Register
- Trigger Event 1 Specifier Register
- Software Breakpoint Service Request Control Register 0

The non-shaded areas in the register description define the implementation-specific bits/bit fields.

TR0EVT

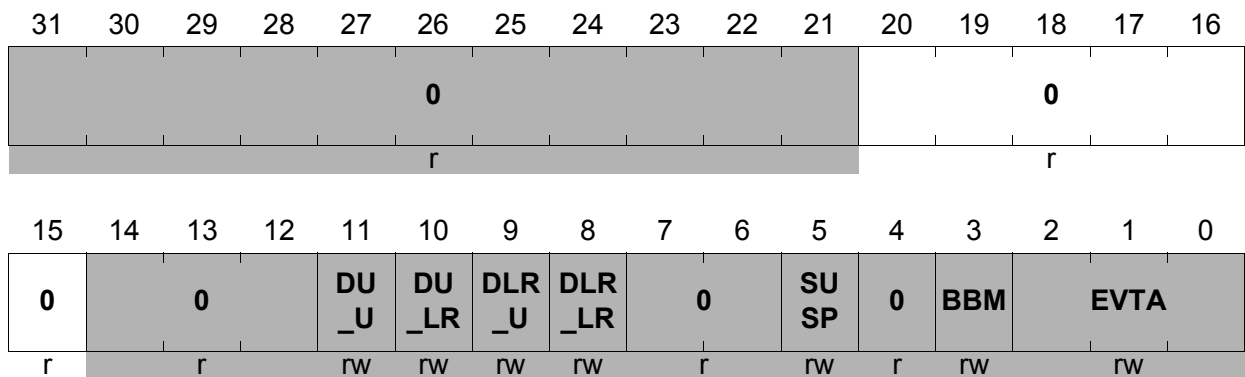
Trigger Event 0 Specifier Register (F7E1 FD20_H)

Reset Value: 0000 0000_H

TR1EVT

Trigger Event 1 Specifier Register (F7E1 FD20_H)

Reset Value: 0000 0000_H



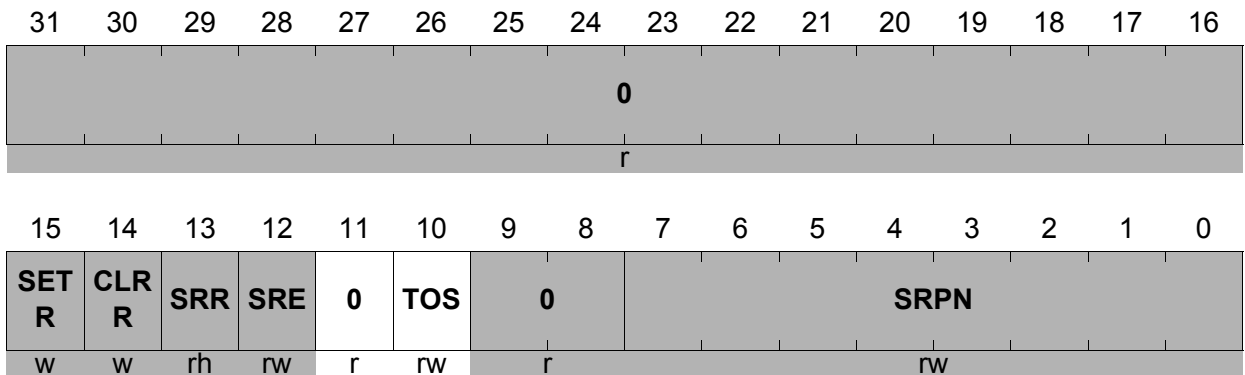
Field	Bits	Type	Description
0	[20:15]	r	Reserved Read as 0; should be written with 0.

CPU_SBSRC0

CPU Software Breakpoint Service Request Control Register 0

(F7E0 FFBC_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
TOS	10	rw	Type of Service Control 0 _B Service Provider = CPU 1 _B Reserved
0	11	r	Reserved Read as 0; should be written with 0.

2.4.5 Memory Protection Registers

As shown in **Figure 2-10**, there are two Memory Protection Register Sets in the TC1796, Set 0 and Set 1, which specify memory protection ranges and permissions for code and data. Set 2 and Set 3 are not implemented in the TC1796. The PSW.PRS bit field determines which of these sets is currently in use by the CPU.

The Memory Protection Registers are described in detail in the TriCore 1 Architecture Manual chapter - Memory Protection System.

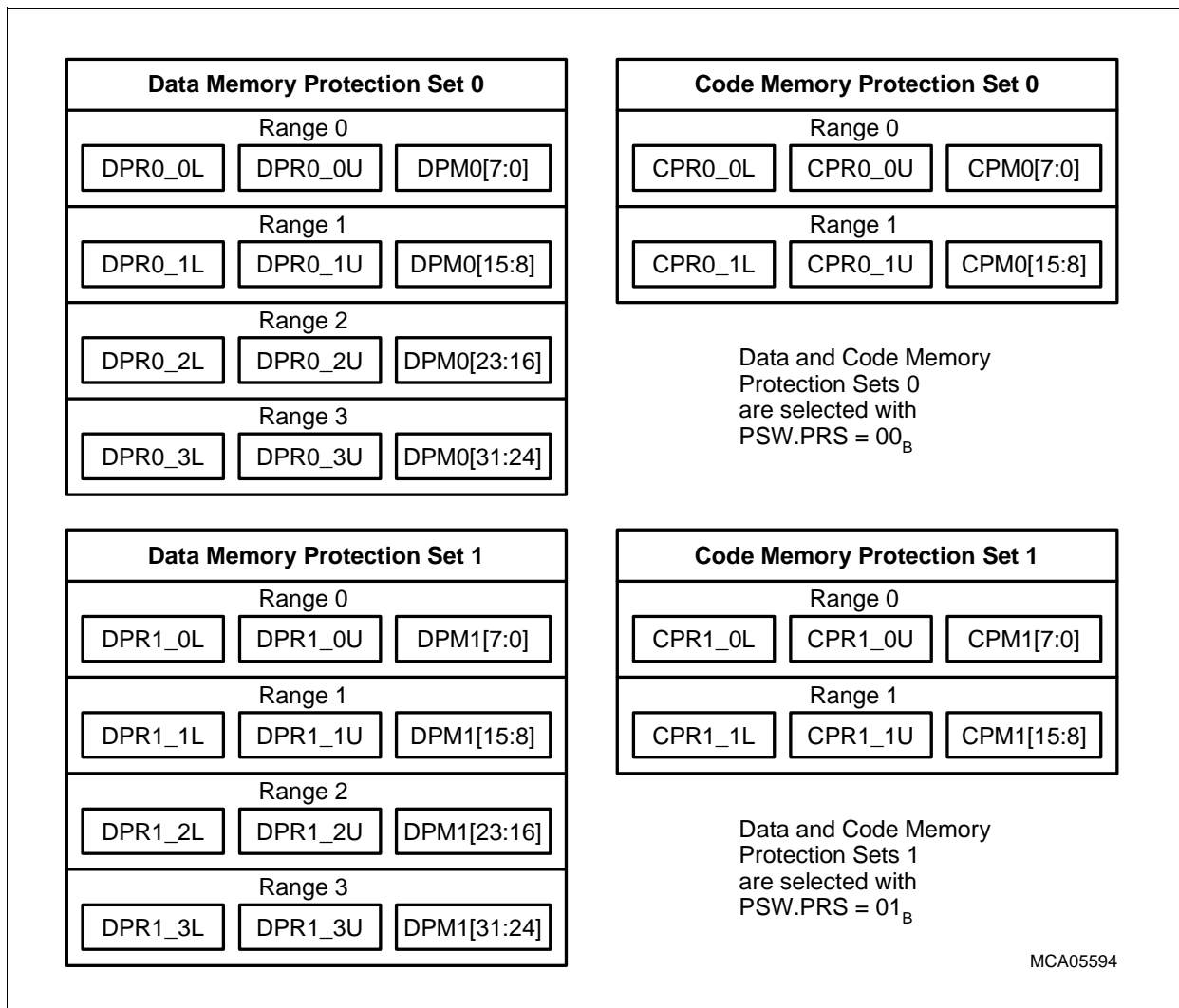


Figure 2-10 Memory Protection Register Sets of the TC1796

Table 2-6 Memory Protection Registers

Register Short Name	Register Long Name	Address
DPR0_0L	Data Segment Protection Register Set 0, Range 0, Lower Boundary	F7E1 C000 _H
DPR0_0U	Data Segment Protection Register Set 0, Range 0, Upper Boundary	F7E1 C004 _H
DPR0_1L	Data Segment Protection Register Set 0, Range 1, Lower Boundary	F7E1 C008 _H
DPR0_1U	Data Segment Protection Register Set 0, Range 1, Upper Boundary	F7E1 C00C _H
DPR0_2L	Data Segment Protection Register Set 0, Range 2, Lower Boundary	F7E1 C010 _H
DPR0_2U	Data Segment Protection Register Set 0, Range 2, Upper Boundary	F7E1 C014 _H
DPR0_3L	Data Segment Protection Register Set 0, Range 3, Lower Boundary	F7E1 C018 _H
DPR0_3U	Data Segment Protection Register Set 0, Range 3, Upper Boundary	F7E1 C01C _H
DPR1_0L	Data Segment Protection Register Set 1, Range 0, Lower Boundary	F7E1 C400 _H
DPR1_0U	Data Segment Protection Register Set 1, Range 0, Upper Boundary	F7E1 C404 _H
DPR1_1L	Data Segment Protection Register Set 1, Range 1, Lower Boundary	F7E1 C408 _H
DPR1_1U	Data Segment Protection Register Set 1, Range 1, Upper Boundary	F7E1 C40C _H
DPR1_2L	Data Segment Protection Register Set 1, Range 2, Lower Boundary	F7E1 C410 _H
DPR1_2U	Data Segment Protection Register Set 1, Range 2, Upper Boundary	F7E1 C414 _H
DPR1_3L	Data Segment Protection Register Set 1, Range 3, Lower Boundary	F7E1 C418 _H
DPR1_3U	Data Segment Protection Register Set 1, Range 3, Upper Boundary	F7E1 C41C _H

Table 2-6 Memory Protection Registers (cont'd)

Register Short Name	Register Long Name	Address
CPR0_0L	Code Segment Protection Register Set 0, Range 0, Lower Boundary	F7E1 D000 _H
CPR0_0U	Code Segment Protection Register Set 0, Range 0, Upper Boundary	F7E1 D004 _H
CPR0_1L	Code Segment Protection Register Set 0, Range 1, Lower Boundary	F7E1 D008 _H
CPR0_1U	Code Segment Protection Register Set 0, Range 1, Upper Boundary	F7E1 D00C _H
CPR1_0L	Code Segment Protection Register Set 1, Range 0, Lower Boundary	F7E1 D400 _H
CPR1_0U	Code Segment Protection Register Set 1, Range 0, Upper Boundary	F7E1 D404 _H
CPR1_1L	Code Segment Protection Register Set 1, Range 1, Lower Boundary	F7E1 D408 _H
CPR1_1U	Code Segment Protection Register Set 1, Range 1, Upper Boundary	F7E1 D40C _H
DPM0	Data Protection Mode Register Set 0	F7E1 E000 _H
DPM1	Data Protection Mode Register Set 1	F7E1 E080 _H
CPM0	Code Protection Mode Register Set 0	F7E1 E200 _H
CPM1	Code Protection Mode Register Set 1	F7E1 E280 _H

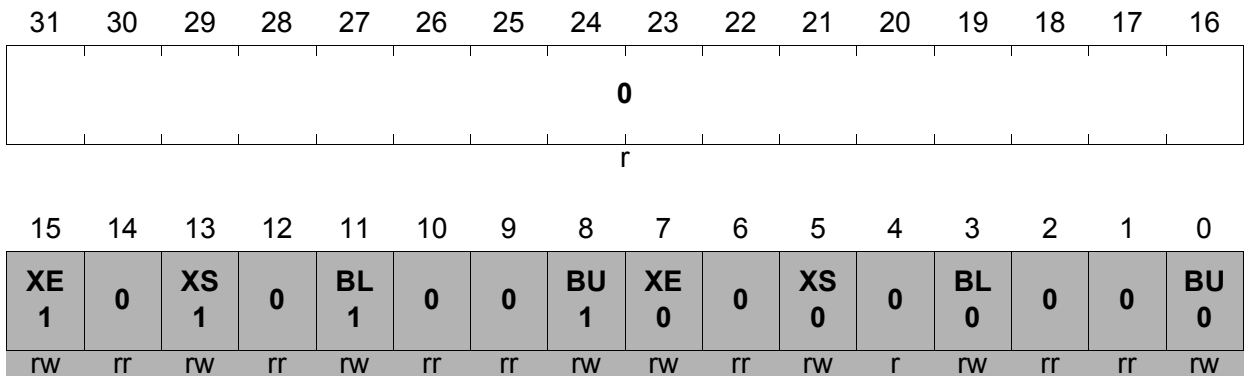
2.4.5.1 Implementation-specific Memory Protection Registers

This section describes the implementation-specific Code Protection Mode Registers that differ from the description in the TriCore 1 Architecture Manual. The non-shaded areas in the CPMx register descriptions define the implementation-specific bits/bit fields. Therefore, the uppermost 16 bits of CPMx are of type “0, r”.

CPMx (x = 0, 1)

Code Protection Mode Register Set x

Reset Value: 0000 0000_H



Field	Bits	Type	Description
0	[31:16]	r	Reserved Read as 0; should be written with 0. These bits refer to code memory ranges 2 and 3, which are not available in the TC1796.

2.5 Program Memory Interface (PMI)

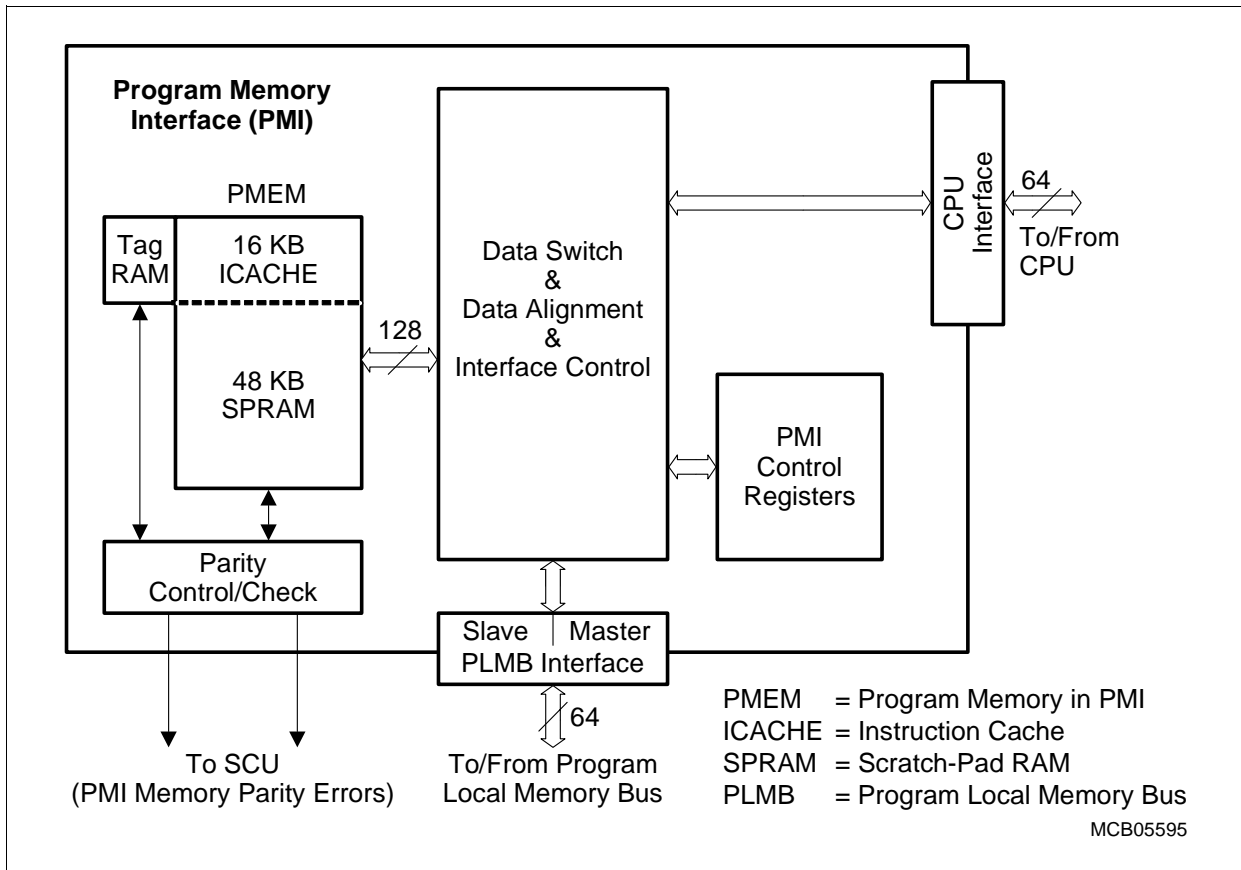


Figure 2-11 PMI Block Diagram

2.5.1 PMI Features

The Program Memory Interface (PMI) has the following features:

- 64 Kbyte memory as:
 - 16 Kbyte instruction cache (ICACHE)
 - 48 Kbyte scratch-pad RAM (SPRAM)
- ICACHE operation features:
 - Two-way set associative cache
 - LRU (Least-Recently Used) replacement algorithm
 - Cache line size = 256 bits (4 double-words)
 - Validity granularity (4 double-words per cache line)
 - ICACHE can be globally invalidated to provide support for software cache coherency (to be handled by the programmer).
 - ICACHE can be bypassed to provide a direct fetch from the CPU to on-chip and off-chip resources.

- ICACHE refill mechanism:
critical double-word first, no line wrap around, streaming to CPU
- CPU interface
 - Supporting unaligned accesses (16-bit aligned) with a penalty of one cycle for unaligned accesses crossing two lines (SPRAM or ICACHE lines)
- Program Local Memory Bus (PLMB) interface to PMU/EBU
- PMI memory cannot be accessed by the PCP using the BCOPY instruction (burst transfers)
- PLMB slave interface cannot be byte-accessed but can be accessed by half-word, word or double-word aligned functions only.
- PMI SRAMs (SPRAM, ICACHE, and Tag SRAM) are parity protected

2.5.2 Parity Protection for PMI Memories

In the TC1796, the PMI memory blocks SPRAM, ICACHE, and Tag RAM are equipped with a parity error detection logic that makes it possible to detect parity errors separately for SPRAM/ICACHE or the ICACHE Tag RAM. In case of a parity error a NMI is generated.

Note that before using parity protection for PMI memory blocks the first time after a power-on reset operation (before setting the corresponding parity error enable bit), the SPRAM memory must be completely initialized by a user program that writes every memory location of it once.

More details about the parity control for on-chip memories are described in [Section 5.5](#) on [Page 5-37](#).

2.5.3 PMI Registers

Three control registers are implemented in the Program Memory Interface. These registers and their bits are described in this section.

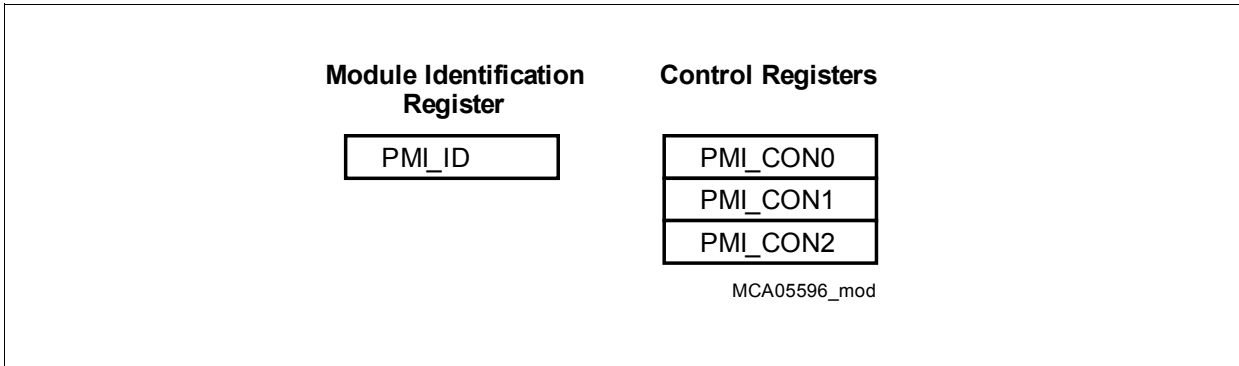


Figure 2-12 PMI Registers

Table 2-7 PMI Registers

Register Short Name	Register Long Name	Address	Description
PMI_ID	PMI Module Identification Register	F87F FD08 _H	Page 2-27
PMI_CON0	PMI Control Register 0	F87F FD10 _H	Page 2-28
PMI_CON1	PMI Control Register 1	F87F FD14 _H	Page 2-29
PMI_CON2	PMI Control Register 2	F87F FD18 _H	Page 2-30

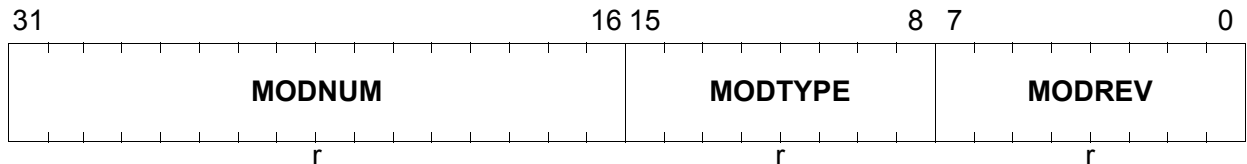
2.5.3.1 PMI Module Identification Register

PMI_ID

PMI Module Identification Register

(F87FFD08_H)

Reset Value: 000B C0XX_H



Field	Bits	Type	Description
MODREV	[7:0]	r	Module Revision Number MODREV defines the module revision number. The value of a module revision starts with 01 _H (first revision).
MODTYPE	[15:8]	r	Module Type This bit field defines the module as a 32-bit module: C0 _H
MODNUM	[31:16]	r	Module Number Value This bit field defines the module identification number for the PMI: 000B _H

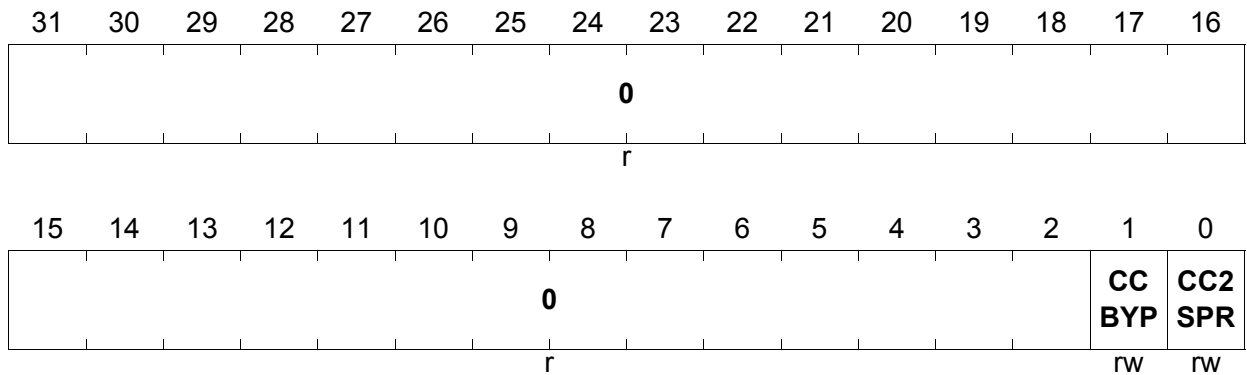
2.5.3.2 PMI Control Register 0

PMI_CON0

PMI Control Register 0

(F87FFD10_H)

Reset Value: 0000 0002_H



Field	Bits	Type	Description
CC2SPR	0	rw	Code Cache Memory to SPR This bit is used for cache test mode purposes. CC2PR must be written with 0. Setting it to 1 may lead to unpredictable program behavior.
CCBYP	1	rw	Code Cache Bypass 0 _B Cache enabled 1 _B Cache bypassed (disabled)
0	[31:2]	r	Reserved Returns 0 when read; should be written with 0.

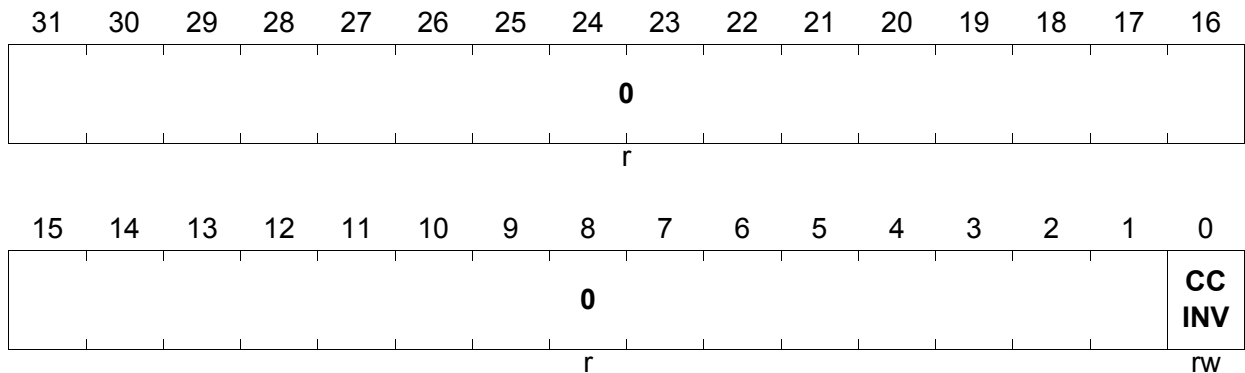
2.5.3.3 PMI Control Register 1

PMI_CON1

PMI Control Register 1

(F87FFD14_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CCINV	0	rw	Code Cache Invalidate 0 _B Normal code cache (ICACHE) operation 1 _B All cache lines are invalidated As long as CCINV is set, all instruction fetch accesses generate a cache refill. It is recommended that CCINV be kept set until ICACHE coherency is guaranteed.
0	[31:1]	r	Reserved Returns 0 when read; should be written with 0.

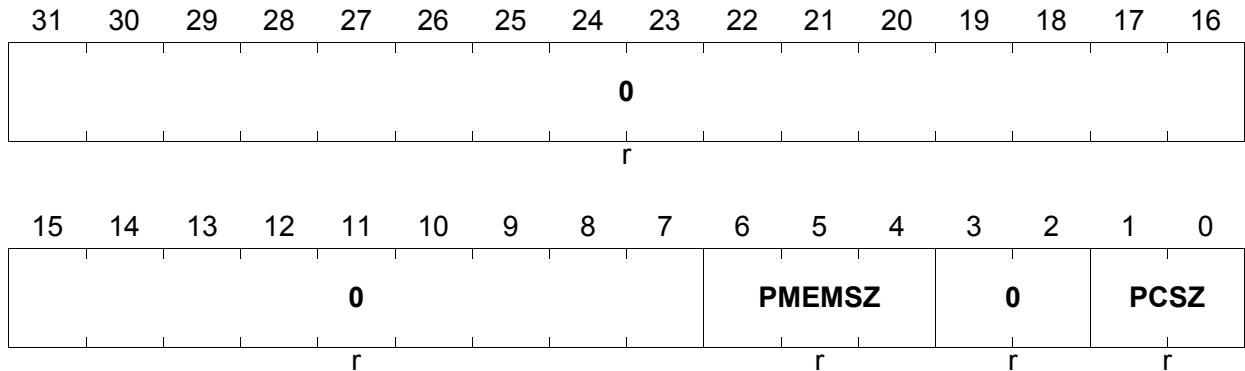
2.5.3.4 PMI Control Register 2

PMI_CON2

PMI Control Register 2

(F87FFD18_H)

Reset Value: 0000 0073_H



Field	Bits	Type	Description
PCSZ	[1:0]	r	Program Cache Size This bit field indicates the ICACHE size and TAGRAM configuration of the PMI. The TC1796 has a fixed ICACHE size of 16 Kbyte. Therefore, PCSZ is always read as 11 _B . 11 _B 16 Kbyte cache
PMEMSZ	[6:4]	r	Program Memory Size (ICACHE + SPRAM) This bit field indicates the ICACHE plus SPRAM size of the PMI program memory. The TC1796 has a fixed ICACHE and SPRAM size of 64 KB. PMEMSZ is always read as 111 _B . 111 _B 64 Kbyte program memory
0	[3:2], [31:7]	r	Reserved Returns 0 when read; should be written with 0.

2.6 Data Memory Interface (DMI)

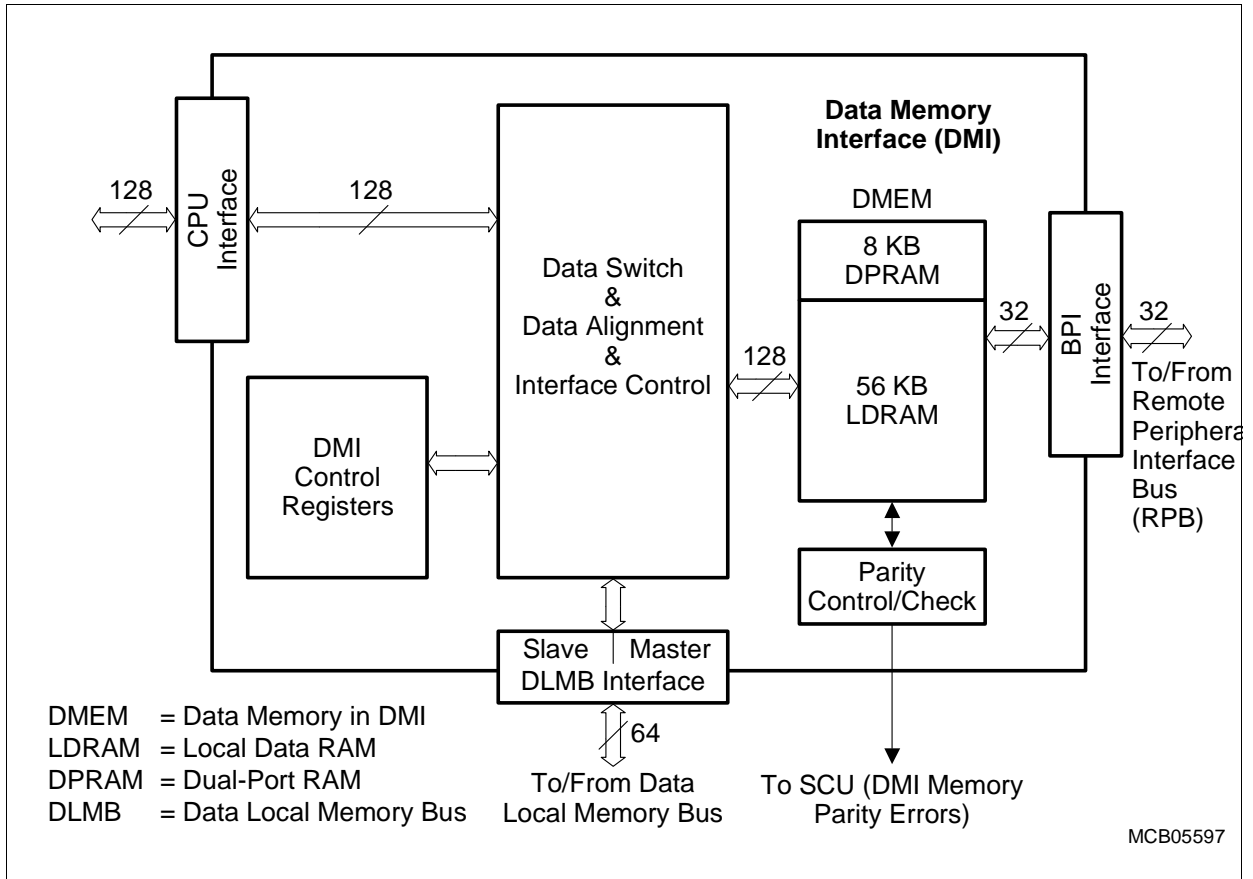


Figure 2-13 DMI Block Diagrams

2.6.1 DMI Features

The Data Memory Interface (DMI) has the following features:

- 64 Kbyte data memory:
 - 8 Kbyte Dual-Port data memory (DPRAM), accessible from CPU and Remote Peripheral Bus
 - 56 Kbyte local data memory (LDRAM)
 - DPRAM and LDRAM are parity protected
- CPU interface:
 - Supporting unaligned accesses (16-bit aligned) with a minimum penalty of one cycle for unaligned accesses crossing 2 lines
- Data Local Memory Bus (DLMB) interface. Allows access to the rest of the system.

2.6.2 Dual-Ported Memory Operation

The dual-ported memory (DPRAM) allows the CPU and Remote Peripheral Bus (RPB) masters to access the same memory locations simultaneously, reading or writing invalid data without conflict. In the case of simultaneous accesses to the same address, the CPU has a higher priority than the RPB master. [Table 2-8](#) defines DPRAM operations.

Table 2-8 DPRAM Access Conflict Handling

Actions		CPU Access	
		Read	Write
RPB Master Access	Read	A simultaneous read access of CPU and RPB master occurs. Both read the same memory content.	The CPU write operation is executed first and the RPB master read access is delayed. The data read is the value that has been written by the CPU.
	Write	Bus data is transferred to CPU and memory in parallel	The CPU write operation is executed. The RPB master write data is not written.

2.6.2.1 CPU Buffer Write Operation

Write accesses from the CPU to DMI memories using ST (store) or LDMST (load-modify-store) instructions first store the write data in a buffer. The content of this buffer is written to the memory when the CPU interface and DLMB (Data Local Memory Bus) are idle, or when another CPU write operation to DMI memory follows. A read operation to the same address (LD instruction) immediately following the write operation will read the data out of the buffer and not from the DPRAM location. The same is true if a continuous sequence of read operations to other DPRAM locations keep the CPU interface busy in the meantime.

For DPRAM operations, such as semaphore handling, it is necessary to ensure that at least two non-DMI-memory related instructions or another ST instruction are executed between the ST/LDMST instruction and the following LD instruction to the semaphore location.

If the data written to a DPRAM location has to be available immediately on the RPB bus side of the DPRAM, two non-DMI-memory related instructions or another ST instruction to the DMI memory must follow the critical store instruction.

The smallest memory unit that can be read from or written to the dual-ported memory is a byte. The largest unit is a double-word. On the remote peripheral bus double-word accesses are performed as two 32-bit RPB transfers. The conflict handling also operates properly when CPU access width and RPB access width are different.

Note: There is no support for locked read-modify-write operations in the dual ported memory.

2.6.3 Parity Protection for DMI Memories

In the TC1796, the LDRAM and DPRAM memory blocks of the DMI are equipped with a parity error detection logic that makes it possible to detect parity errors separately for LDRAM or DPRAM. In case of a parity error a NMI is generated.

Note that before using parity protection for DMI memory blocks the first time after a power-on reset operation (before setting the corresponding parity error enable bit), the LDRAM and DPRAM memories must be completely initialized by a user program that writes every memory location of it once.

More details about the parity control for on-chip memories are described in [Section 5.5](#) on [Page 5-37](#).

2.6.4 DMI Registers

Two control registers and two trap flag registers are implemented in the DMI. These registers and their bits are described in this section.

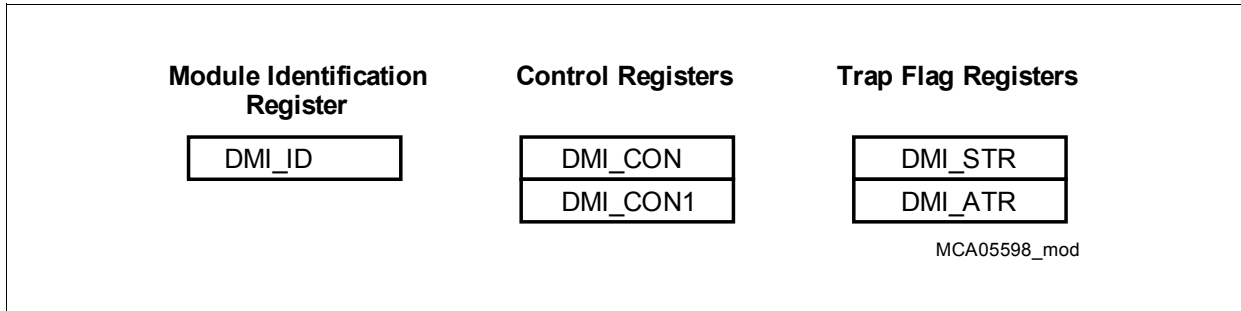


Figure 2-14 DMI Registers

Table 2-9 DMI Registers

Register Short Name	Register Long Name	Address	Description
DMI_ID	DMI Module Identification Register	F87F FC08 _H	Page 2-35
DMI_CON	DMI Control Register	F87F FC10 _H	Page 2-36
DMI_STR	DMI Synchronous Trap Flag Register	F87F FC18 _H	Page 2-38
DMI_ATR	DMI Asynchronous Trap Flag Register	F87F FC20 _H	Page 2-39
DMI_CON1	DMI Control Register 1	F87F FC28 _H	Page 2-37

Access to DMI control registers must only be made with double-word aligned word accesses. An access not conforming to this rule, or an access that does not follow the specified privilege mode (Supervisor mode, Endinit-protection), or a write access to a read-only register, will lead to a bus error if the access was from the LMB Bus, or to a trap, flagged in DMI_STR/DMI_ATR register in case of a CPU load/store access.

2.6.4.1 DMI Register Description

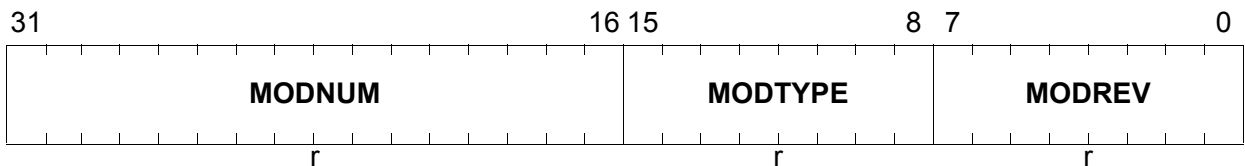
The DMI Module Identification Register ID contains read-only information about the DMI module version.

DMI_ID

DMI Module Identification Register

(F87FFC08_H)

Reset Value: 0008 C0XX_H



Field	Bits	Type	Description
MODREV	[7:0]	r	Module Revision Number MODREV defines the module revision number. The value of a module revision starts with 01 _H (first revision).
MODTYPE	[15:8]	r	Module Type This bit field defines the module as a 32-bit module: C0 _H
MODNUM	[31:16]	r	Module Number Value This bit field defines the module identification number for the DMI: 0008 _H

CPU Subsystem

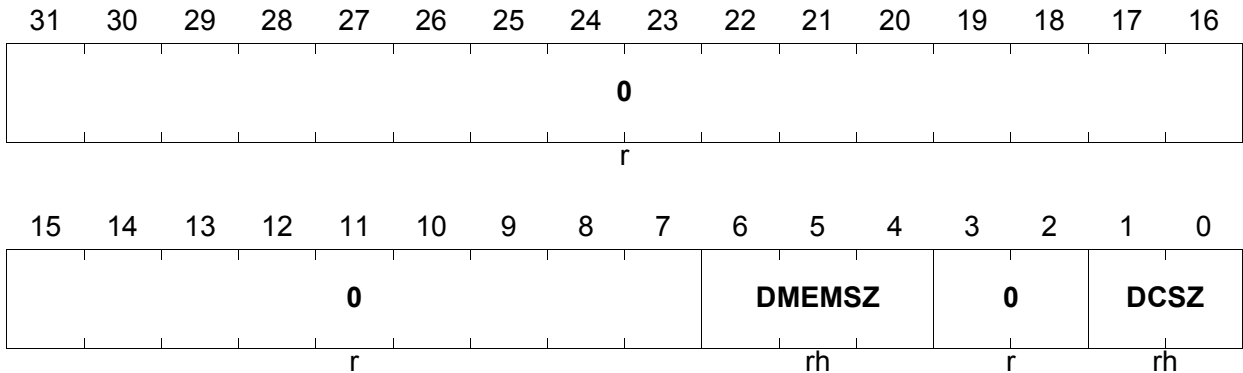
The DMI control register indicates the DMI data memory size and data cache availability.

DMI_CON

DMI Control Register

(F87FFC10_H)

Reset Value: 0000 0070_H



Field	Bits	Type	Description
DCSZ	[1:0]	r	Data Cache Size This bit field indicates the DMI data cache configuration. In the TC1796 no data cache is available, therefore DCSZ is always read as 00 _B . 00 _B No cache available
DMEMSZ	[6:4]	r	Data Memory Size This bit field indicates the DMI data memory size. In the TC1796 DMEMSZ is always read as 111 _B . 111 _B 64 Kbyte DMI data memory
0	[3:2], [31:7]	r	Reserved Returns 0 when read.

CPU Subsystem

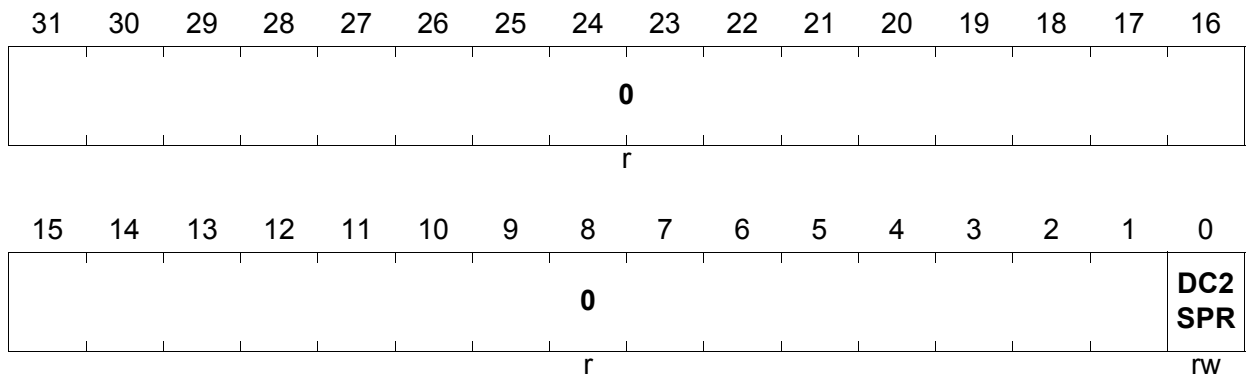
The DMI control register 1 is required for data cache test purposes. It is noted here for the sake of completeness.

DMI_CON1

DMI Control Register 1

(F87FFC28_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
DC2SPR	0	rw	Cache Test Mode Enable This bit must always be written with 0. Setting to 1 will have no effect in TC1796.
0	[31:1]	r	Reserved Returns 0 when read; should be written with 0.

CPU Subsystem

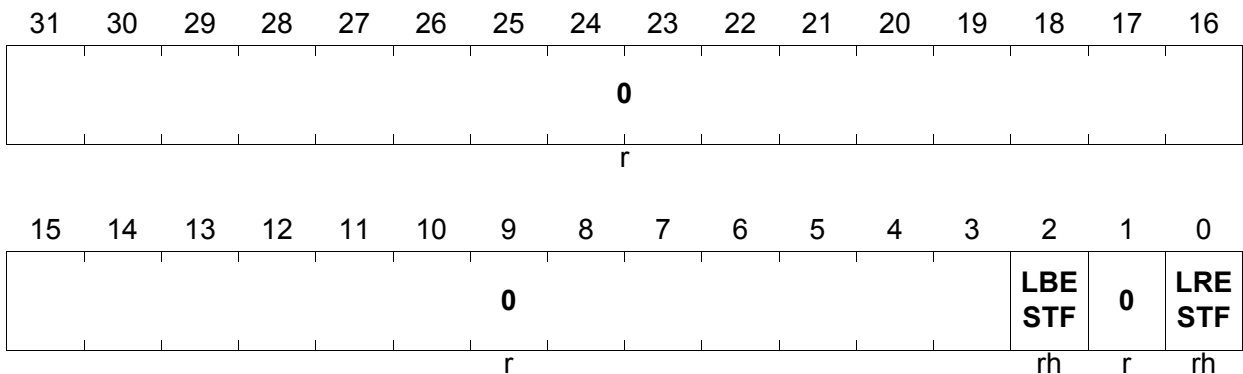
The DMI Synchronous Trap Flag Register, DMI_STR, holds the flags that identify the root cause of a Data-access Synchronous Bus Error (DSE). Reading DMI_STR in supervisor mode returns the register contents and then clears its contents. Reading DMI_STR in user mode returns the contents of the register but does not clear its contents.

DMI_STR

DMI Synchronous Trap Flag Register

(F87FFC18_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
LRESTF	0	rh	Load Range Synchronous Error 0 _B No error 1 _B Load range synchronous error has occurred
LBESTF	2	rh	Bus Load Synchronous Error 0 _B No error 1 _B Bus load synchronous error has occurred
0	1, [31:3]	r	Reserved Returns 0 when read.

CPU Subsystem

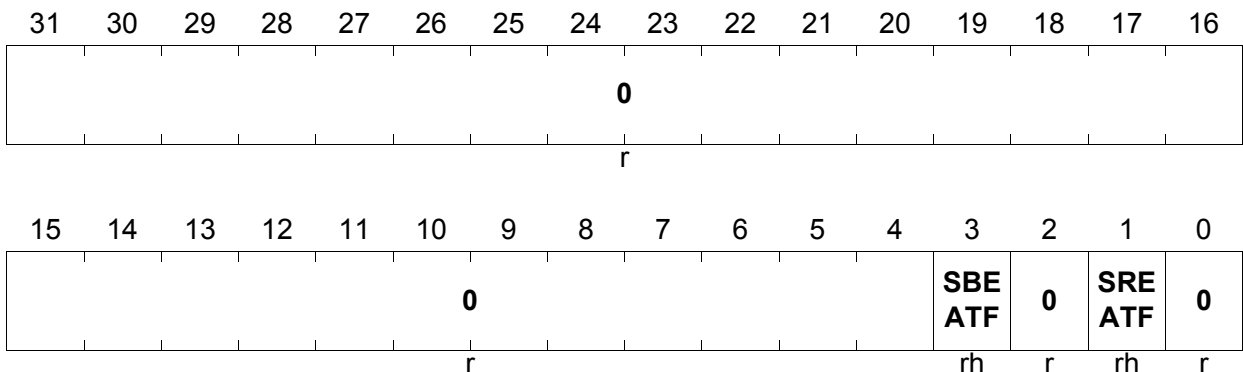
The DMI Asynchronous Trap Flag Register, DMI_ATR, holds the flags that inform about the root cause of a Data Access Asynchronous Bus Error (ASE). Reading DMI_ATR in supervisor mode returns the register contents and then clears its contents. Reading DMI_ATR in user mode returns the contents of the register but does not clear its contents.

DMI_ATR

DMI Asynchronous Trap Flag Register

(F87FFC20_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SREATF	1	rh	Store Range Asynchronous Error 0 _B No error 1 _B Store range asynchronous error has occurred
SBEATF	3	rh	LMB Bus Store Asynchronous Error 0 _B No error 1 _B Bus store asynchronous error has occurred
0	0, 2, [31:4]	r	Reserved Returns 0 when read.

2.7 Instruction Timing

This section gives information on instruction timing by execution unit. The Integer Pipeline and Load/Store Pipeline are always present, and the Floating Point Unit (FPU) is optional. The Load/Store unit implements the optional TLB instructions.

Definition of Terms:

- **Repeat Rate**
Assuming the same instruction is being issued sequentially, repeat is the minimum number of clock cycles between two consecutive issues. There may be additional delays described elsewhere due to internal pipeline effects when issuing a different subsequent instruction.
- **Result Latency**
The number of clock cycles from the cycle when the instruction is issued to the cycle when the result value is available to be used as an operand to a subsequent instruction or written into a GPR. Result latency is not meaningful for instructions that do not write a value into a GPR.
- **Address Latency**
The number of clock cycles from the cycle when the instruction is issued to the cycle when the addressing mode updated value is available as an operand to a subsequent instruction or written into an Address Register.
- **Flow Latency**
The number of clock cycles from the cycle when the instruction is issued to the cycle when the next instruction (located at the target location or the next sequential instruction if the control change is conditional) is issued.

2.7.1 Integer-Pipeline Instructions

2.7.1.1 Simple Arithmetic Instruction Timings

Each instruction is single issued.

Table 2-10 Simple Arithmetic Instruction Timing

Instruction	Result Latency	Repeat Rate	Instruction	Result Latency	Repeat Rate
Integer Pipeline Arithmetic Instructions					
ABS	1	1	MAX.H	1	1
ABS.B	1	1	MAX.HU	1	1
ABS.H	1	1	MAX.U	1	1
ABSDIF	1	1	MIN	1	1
ABSDIF.B	1	1	MIN.B	1	1
ABSDIF.H	1	1	MIN.BU	1	1
ABSDIFS	1	1	MIN.H	1	1
ABSDIFS.H	1	1	MIN.HU	1	1
ABSS	1	1	MIN.U	1	1
ABSS.H	1	1	RSUB	1	1
ADD	1	1	RSUBS	1	1
ADD.B	1	1	RSUBS.U	1	1
ADD.H	1	1	SAT.B	1	1
ADDC	1	1	SAT.BU	1	1
ADDI	1	1	SAT.H	1	1
ADDIH	1	1	SAT.HU	1	1
ADDS	1	1	SEL	1	1
ADDS.H	1	1	SELN	1	1
ADDS.HU	1	1	SUB	1	1
ADDS.U	1	1	SUB.B	1	1
ADDX	1	1	SUB.H	1	1
CADD	1	1	SUBC	1	1
CADDN	1	1	SUBS	1	1

Table 2-10 Simple Arithmetic Instruction Timing (cont'd)

Instruction	Result Latency	Repeat Rate	Instruction	Result Latency	Repeat Rate
CSUB	1	1	SUBS.H	1	1
CSUBN	1	1	SUBS.HU	1	1
MAX	1	1	SUBS.U	1	1
MAX.B	1	1	SUBX	1	1
MAX.BU	1	1			
Compare Instructions					
EQ	1	1	LT.B	1	1
EQ.B	1	1	LT.BU	1	1
EQ.H	1	1	LT.H	1	1
EQ.W	1	1	LT.HU	1	1
EQANY.B	1	1	LT.U	1	1
EQANY.H	1	1	LT.W	1	1
GE	1	1	LT.WU	1	1
GE.U	1	1	NE	1	1
LT	1	1			
Count Instructions					
CLO	1	1	CLS.H	1	1
CLO.H	1	1	CLZ	1	1
CLS	1	1	CLZ.H	1	1
Extract Instructions					
DEXTR	1	1	INS.T	1	1
EXTR	1	1	INSN.T	1	1
EXTR.U	1	1	INSERT	1	1
IMASK	1	1			
Logical Instructions					
AND	1	1	OR.EQ	1	1
AND.AND.T	1	1	OR.GE	1	1
AND.ANDN.T	1	1	OR.GE.U	1	1
AND.EQ	1	1	OR.LT	1	1
AND.GE	1	1	OR.LT.U	1	1

Table 2-10 Simple Arithmetic Instruction Timing (cont'd)

Instruction	Result Latency	Repeat Rate	Instruction	Result Latency	Repeat Rate
AND.GE.U	1	1	OR.NE	1	1
AND.LT	1	1	OR.NOR.T	1	1
AND.LT.U	1	1	OR.OR.T	1	1
AND.NE	1	1	OR.T	1	1
AND.NOR.T	1	1	ORN	1	1
AND.OR.T	1	1	ORN.T	1	1
AND.T	1	1	XNOR	1	1
ANDN	1	1	XNOR.T	1	1
ANDN.T	1	1	XOR	1	1
NAND	1	1	XOR.EQ	1	1
NAND.T	1	1	XOR.GE	1	1
NOR	1	1	XOR.GE.U	1	1
NOR.T	1	1	XOR.LT	1	1
OR	1	1	XOR.LT.U	1	1
OR.AND.T	1	1	XOR.NE	1	1
OR.ANDN.T	1	1	XOR.T	1	1
Move Instructions					
CMOV	1	1	MOV.U	1	1
CMOVN	1	1	MOVH	1	1
MOV	1	1			
Shift Instructions					
SH	1	1	SH.NE	1	1
SH.AND.T	1	1	SH.NOR.T	1	1
SH.ANDN.T	1	1	SH.OR.T	1	1
SH.EQ	1	1	SH.ORN.T	1	1
SH.GE	1	1	SH.XNOR.T	1	1
SH.GE.U	1	1	SH.XOR.T	1	1
SH.H	1	1	SHA	1	1
SH.LT	1	1	SHA.H	1	1
SH.LT.U	1	1	SHAS	1	1

Table 2-10 Simple Arithmetic Instruction Timing (cont'd)

Instruction	Result Latency	Repeat Rate	Instruction	Result Latency	Repeat Rate
SH.NAND.T	1	1			
Coprocessor 0 Instructions					
BMERGE	1	1	DVINIT.WS	1	1
BSPLIT	1	1	DVINIT.WU	1	1
DVADJ	1	1	DVSTEP.S	4	4
DVINIT	1	1	DVSTEP.U	4	4
DVINIT.U	1	1	IXMAX	1	1
DVINIT.B	1	1	IXMAX.U	1	1
DVINIT.H	1	1	IXMIN	1	1
DVINIT.BS	1	1	IXMIN.U	1	1
DVINIT.BU	1	1	PACK	1	1
DVINIT.HS	1	1	PARITY	1	1
DVINIT.HU	1	1	UNPACK	1	1

2.7.1.2 Multiply Instruction Timings

Each instruction is single issued.

Table 2-11 Multiple Instruction Timing

Instruction	Result Latency	Repeat Rate	Instruction	Result Latency	Repeat Rate
IP Arithmetic Instructions					
MUL	3	2	MUL.H	2	1
MUL.U	3	2	MUL.Q	1/2/3	1/1/2
MULS	3	2	MULM.H	2	1
MULS.U	3	2	MULR.H	2	1
			MULR.Q	2	1

For MUL.Q Instruction:

	Result Latency	Repeat Rate
16 × 16	1	1
16 × 32	2	1
32 × 32	3	2

2.7.1.3 MAC Instruction Timings

Each instruction is single issued.

Table 2-12 MAC Instruction Timing

Instruction	Result Latency	Repeat Rate	Instruction	Result Latency	Repeat Rate
IP Arithmetic Instructions					
MADD	3	2	MSUB	3	2
MADD.U	3	2	MSUB.U	3	2
MADDS	3	2	MSUBS	3	2
MADDS.U	3	2	MSUBS.U	3	2
MADD.H	2	1	MSUB.H	2	1
MADD.Q	2/3	1/2	MSUB.Q	2/3	1/2
MADDM.H	2	1	MSUBM.H	2	1
MADDMS.H	2	1	MSUBMS.H	2	1
MADDR.H	2	1	MSUBR.H	2	1
MADDR.Q	2	1	MSUBR.Q	2	1
MADDRS.H	2	1	MSUBRS.H	2	1
MADDRS.Q	2	1	MSUBRS.Q	2	1
MADDS.H	2	1	MSUBS.H	2	1
MADDS.Q	2/3	1/2	MSUBS.Q	2/3	1/2
MADDSU.H	2	1	MSUBAD.H	2	1
MADDSUM.H	2	1	MSUBADM.H	2	1
MADDSUMS.H	2	1	MSUBADMS.H	2	1
MADDSUR.H	2	1	MSUBADR.H	2	1
MADDSURS.H	2	1	MSUBADRS.H	2	1
MADDSUS.H	2	1	MSUBADS.H	2	1

For MADD.Q, MADDS.Q, MSUB.Q, MSUBS.Q Instructions:

	Result Latency	Repeat Rate
16 × 16	2	1
16 × 32	2	1
32 × 32	3	2

2.7.1.4 Control Flow Instruction Timing

Note all Integer Pipeline Control flow instructions are conditional.

- Each instruction is single issued.
- All target locations yield a full instruction in one access (i.e. not 16-bits of a 32-bit instruction).
- All code fetches take a single cycle.
- Timing is best case; no cache misses for context operations, no pending stores.

Table 2-13 Integer Pipeline Control Flow Instruction Timing

Instruction	Flow Latency	Repeat Rate	Instruction	Flow Latency	Repeat Rate
Branch Instructions					
JEQ	1/2/3	1/2/3	JLTZ	1/2/3	1/2/3
JGE	1/2/3	1/2/3	JNE	1/2/3	1/2/3
JGE.U	1/2/3	1/2/3	JNED	1/2/3	1/2/3
JGEZ	1/2/3	1/2/3	JNEI	1/2/3	1/2/3
JGTZ	1/2/3	1/2/3	JNZ	1/2/3	1/2/3
JLEZ	1/2/3	1/2/3	JNZ.T	1/2/3	1/2/3
JLT	1/2/3	1/2/3	JZ	1/2/3	1/2/3
JLT.U	1/2/3	1/2/3	JZ.T	1/2/3	1/2/3

For All Control Flow Instructions:

	Flow Latency	Repeat Rate
Correctly predicted, not taken	1	1
Correctly predicted, taken	2	2
Wrongly predicted	3	3

2.7.2 Load-Store Pipeline Instructions

2.7.2.1 Address Arithmetic Timing

Each instruction is single issued.

Table 2-14 Address Arithmetic Instruction Timing

Instruction	Result Latency	Repeat Rate	Instruction	Result Latency	Repeat Rate
LS Arithmetic Instructions					
ADD.A	1	1	GE.A	1	1
ADDIH.A	1	1	LT.A	1	1
ADDSC.A	1	1	NE.A	1	1
ADDSC.AT	1	1	NEZ.A	1	1
EQ.A	1	1	SUB.A	1	1
EQZ.A	1	1	NOP	1	1
Trap and Interrupt Instructions					
DEBUG	–	1	TRAPSV ¹⁾	–	1
DISABLE	–	1	TRAPV ³⁾	–	1
ENABLE	–	1	RSTV	–	1
Move Instructions					
MFCR	1	1	MOV.A	1	1
MTCR	–	1	MOV.AA	1	1
MOVH.A	1	1	MOV.D	1	1
Sync Instructions					
DSYNC ²⁾	–	1	ISYNC ³⁾	–	1

1) Execution cycles when no TRAP is taken. The execution timing in the case of raising these TRAPs is the same as other TRAPs such as SYSCALL.

2) Repeat rate assumes that no shadow register write-back is pending, other wise the repeat rate will depend upon the time for all delayed memory operation to occur.

3) Repeat rate assumes that code refetch takes a single cycle.

2.7.2.2 Control Flow Instruction Timing

- Each instruction is single issued.
- All targets yield a full instruction in one access (not 16-bits of a 32-bit instruction).
- All code fetches take a single cycle. Timing is best case; no cache misses for context operations, no pending stores.

Table 2-15 Load/Store Control Flow Instruction Timing

Instruction	Flow Latency	Repeat Rate	Instruction	Flow Latency	Repeat Rate
Branch Instructions					
J	2	2	JLI	2	2
JA	2	2	JEQ.A	1/2/3	1/2/3
JI	2	2	JNE.A	1/2/3	1/2/3
JL	2	2	JNZ.A	1/2/3	1/2/3
JLA	2	2	JZ.A	1/2/3	1/2/3
CSA Instructions					
CALL¹⁾	2-5	2-5	SYSCALL¹⁾	–	2-5
CALLA¹⁾	2-5	2-5	SVLCX¹⁾	–	4-9
CALLI¹⁾	2-5	2-5	RSLCX¹⁾	–	4
RET¹⁾	–	2-5	RFE¹⁾	–	2-5
BISR¹⁾	–	4-9	RFM²⁾	–	–
Loop Instructions					
LOOP³⁾	2/1/3	2/1/3	LOOPU³⁾	2/1/3	2/1/3

- 1) Latency of CSA related instructions varies according to preceding instruction and status of the shadow register file. Average latency is ~2.7 cycles
- 2) Not strictly a CSA operation, but retrieves from memory a subset of context information and changes control flow in a similar manner.
- 3) First time encountered executed in LS pipeline: Flow latency = 2, Repeat rate = 2
Successive time executed in Loop Pipeline: Flow latency = 1: Repeat rate = 1 (nested up to 2 deep)
Last time encountered: Flow latency = 3: Repeat rate = 3

For JLI, JEQ.A, JNE.A JNZ.A, JZ.A Instructions:

	Flow Latency	Repeat Rate
Correctly predicted, not taken	1	1
Correctly predicted, taken	2	2
Wrongly predicted	3	2

2.7.2.3 Load Instruction Timing

Load instructions can produce two results if they use the pre-increment, post-increment, circular or bit-reverse addressing modes. Hence, in those cases there are two latencies that must be specified, the result latency for the value loaded from memory and the address latency for using the updated address register result.

- Each instruction is single issued.
- The memory references is naturally aligned.
- The memory accessed takes a single cycle to return a data item.
- Timing is best case; no cache misses, no pending stores.

Table 2-16 Load Instruction Timing

Instruction	Addr. Latency	Result Latency	Repeat Rate	Instruction	Addr. Latency	Result Latency	Repeat Rate
Load Instructions							
LD.A	1	2	1	LD.Q	1	1	1
LD.B	1	1	1	LD.W	1	1	1
LD.BU	1	1	1	LDLCX	4	4	4
LD.D	1	1	1	LDUCX	4	4	4
LD.DA	1	2	1	SWAP.W	2	2	2
LD.H	1	1	1	LEA ¹⁾	–	1	1
LD.HU	1	1	1				

1) The addressing mode returning an updated address is not relevant for this instruction.

2.7.2.4 Store Instruction Timing

Cache and Store instructions similar to Load instructions will have a result for the pre-increment, post-increment, circular or bit-reverse addressing modes, but do not produce a 'memory' result.

- Each instruction is single issued.
- The memory references is naturally aligned.
- The memory accessed takes a single cycle to accept a data item.
- Timing is best case; no cache misses, no pending stores.

Table 2-17 Cache and Store Instruction Timing

Instruction	Addr. Latency	Repeat Rate	Instruction	Address Latency	Repeat Rate
Cache Instructions					
CACHEA.I	1	1	CACHEA.WI ¹⁾	1	1
CACHEA.W ¹⁾	1	1			
Store Instructions					
ST.A	1	1	ST.T	2	2
ST.B	1	1	ST.W	1	1
ST.D	1	1	STLCX	4	4
ST.DA	1	1	STUCX	4	4
ST.H	1	1			
ST.Q	1	1	LDMST	2	2

1) Repeat rate assumes that no memory writeback operation occurs. Otherwise the repeat rate will depend upon the time for the castout buffers to clear.

2.8 Floating Point Pipeline Timings

These instructions are only valid if the optional Floating Point Unit is implemented.

- Each instruction is single issued.

Table 2-18 Floating Point Instruction Timing

Instruction	Result Latency	Repeat Rate	Instruction	Result Latency	Repeat Rate
Floating Point Instructions					
ADDF	2	2	MSUB.F	3	3
CMP.F	1	1	MUL.F	2	2
DIV.F	15	15	Q31TOF	2	2
FTOI	2	2	QSEED.F	1	1
FTOQ31	2	2	SUBF	2	2
FTOU	2	2	UPDFL	–	1
ITOF	2	2	UTOF	2	2
MADD.F	3	3			

3 Clock System and Control

3.1 Overview

This chapter describes the TC1796 clock system. Topics covered include clock gating, clock domains, clock generation, the operation of clock circuitry, boot-time operation, fail-safe operation, and clock control registers.

The TC1796 clock system performs the following functions:

- Acquires and buffers incoming clock signals to create a master clock frequency
- Distributes in-phase synchronized clock signals throughout the TC1796's entire clock tree
- Divides a system master clock frequency into lower frequencies required by the different modules for operation
- Dynamically reduces power consumption during operation of functional units
- Statically reduces power consumption through programmable power-saving modes
- Reduces electromagnetic interference (EMI) by switching off unused modules

The clock system must be operational before the TC1796 can function, so it contains special logic to handle power-up and reset operations. Its services are fundamental to the operation of the entire system, so it contains special fail-safe logic.

Figure 3-1 shows the structure of the TC1796 clock system. The system clock f_{SYS} is generated by the oscillator circuit and the PLL (phase-locked loop) unit. Each peripheral module operates with its module clocks f_{CLC} and f_{MOD} . Suffix "MOD" is a place holder for the module name, e.g. f_{ASC0} .

The functionality of the control blocks shown in **Figure 3-1** varies depending on the functional unit being controlled. Some functional units such as the watchdog timer, are directly driven by the system clock. The implemented clock control register options are described for each unit on **Table 3-9** on **Page 3-39**.

All clock control registers CLC and the fractional divider registers FDR are Endinit-protected.

Features of the TC1796 Clock System

- PLL operation for multiplying clock source by different factors
- Direct drive capability for direct clocking
- Comfortable state machine for secure switching between basic PLL, direct or prescaler operation
- Sleep and Power-Down Mode support

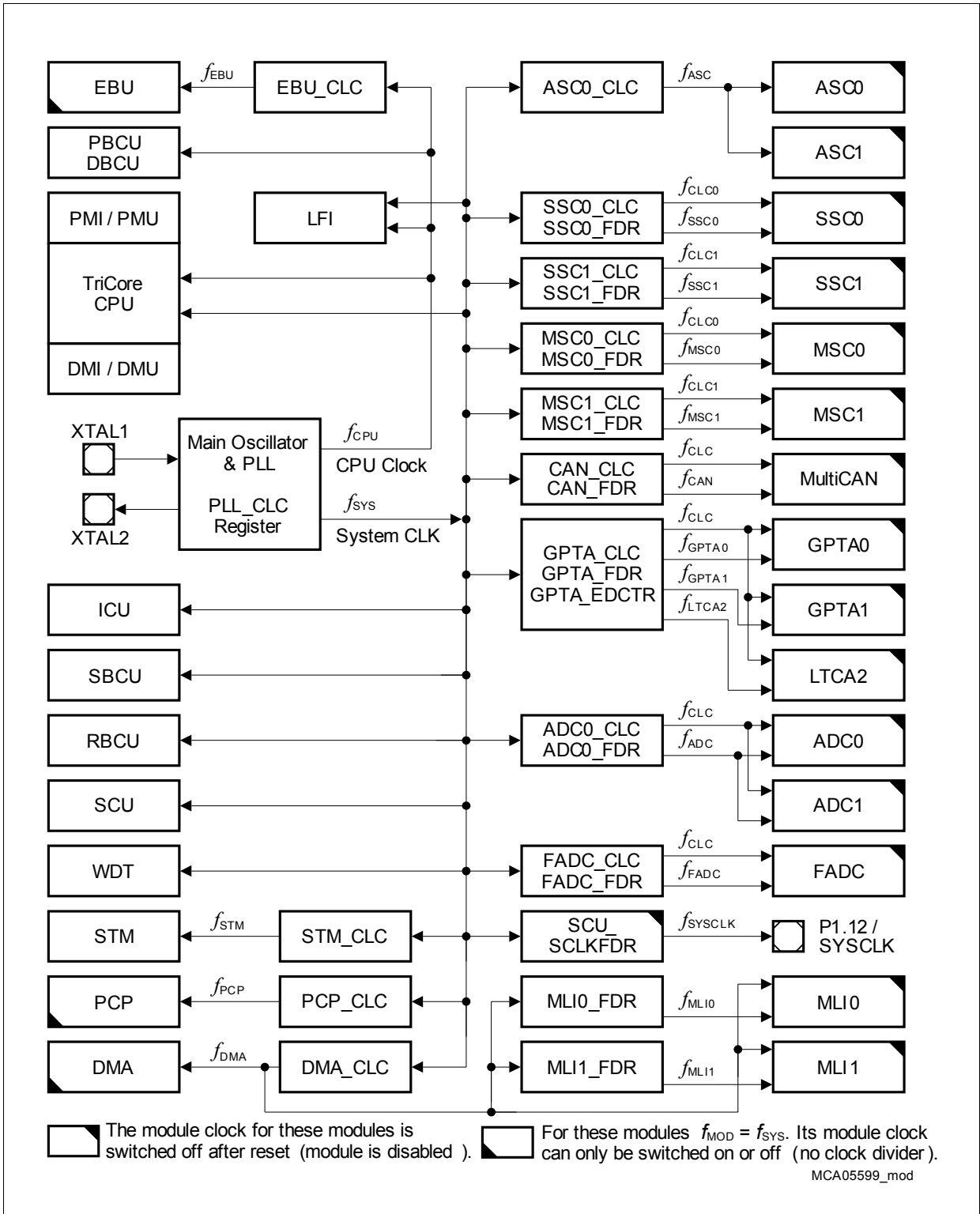


Figure 3-1 TC1796 Clocking System

3.2 Clock Generation Unit

The PLL can convert a low-frequency external clock signal to a high-speed internal clock for maximum performance. The PLL also has fail-safe logic that detects degenerate external clock behavior such as abnormal frequency deviations or a total loss of the external clock. It can execute emergency actions if it loses its lock on the external clock.

The Clock Generation Unit (CGU) in the TC1796, shown in **Figure 3-2**, consists basically of an oscillator circuit and a Phase-Locked Loop (PLL). The operation of the CGU is controlled by two registers, OSC_CON and PLL_CLC, which are located in the System Control Unit (SCU).

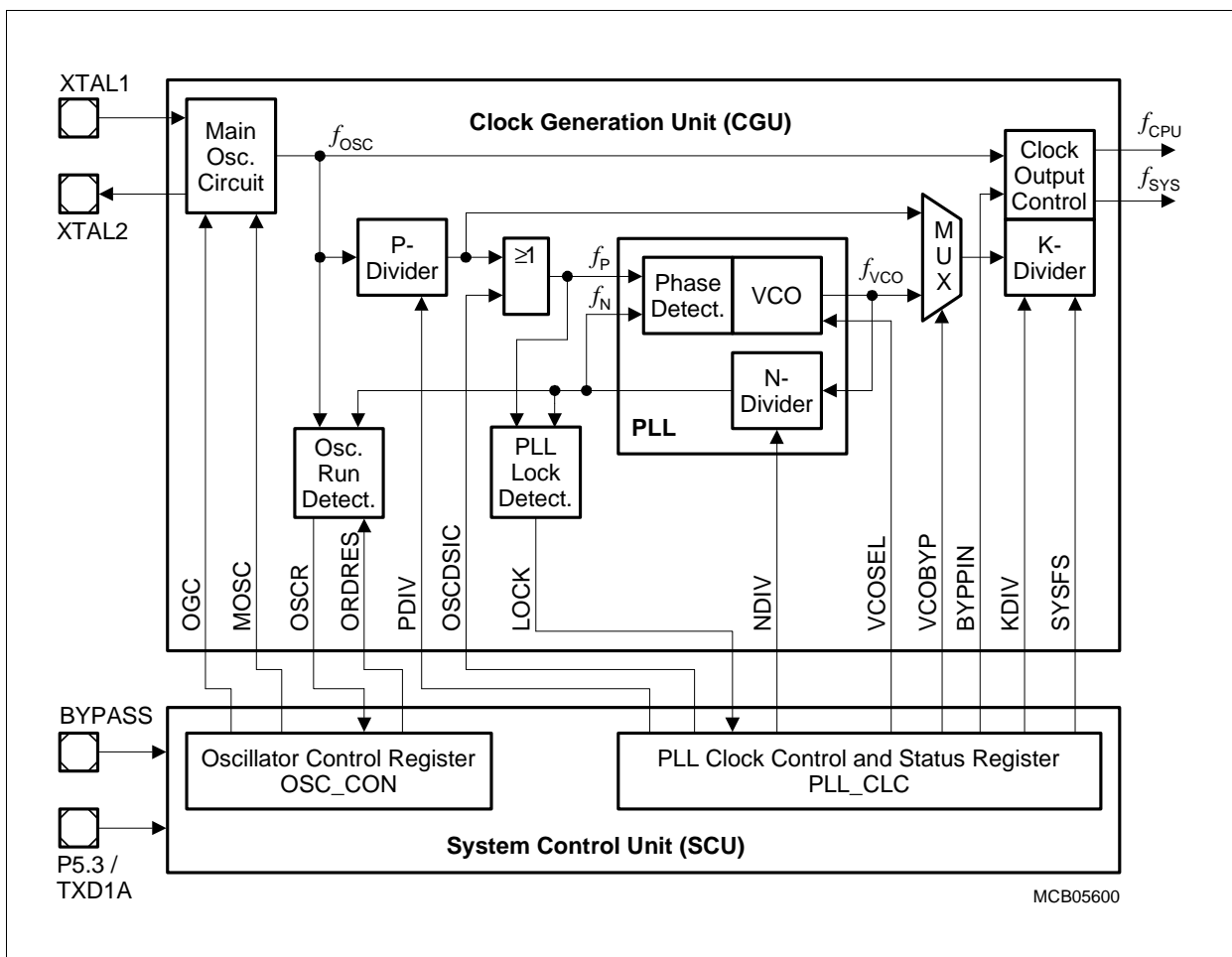


Figure 3-2 CGU Detailed Block Diagram

3.2.1 Main Oscillator Circuit

The oscillator circuit, a Pierce oscillator, is designed to work with both, an external crystal oscillator or an external stable clock source. It basically consists of an inverting amplifier and a feedback element with XTAL1 as input, and XTAL2 as output.

When using a crystal, a proper external oscillator circuitry must be connected to both pins, XTAL1 and XTAL2. The crystal frequency can be within the range of 4 MHz to 25 MHz. Additionally are necessary, two load capacitances C_{X1} and C_{X2} , and depending on the crystal type, a series resistor R_{X2} to limit the current. A test resistor R_Q may be temporarily inserted to measure the oscillation allowance (negative resistance) of the oscillator circuitry. R_Q values are typically specified by the crystal vendor. The C_{X1} and C_{X2} values shown in **Figure 3-3** can be used as starting points for the negative resistance evaluation and for non-productive systems. The exact values and related operating range are dependent on the crystal frequency and have to be determined and optimized together with the crystal vendor using the negative resistance method. Oscillation measurement with the final target system is strongly recommended to verify the input amplitude at XTAL1 and to determine the actual oscillation allowance (margin negative resistance) for the oscillator-crystal system.

When using an external clock signal, the signal must be connected to XTAL1. XTAL2 is left open (unconnected). The external clock frequency can be in the range of 0 - 40 MHz if the PLL is bypassed, and 4 - 40 MHz if the PLL is used.

The oscillator can also be used in combination with a ceramic resonator. The final circuitry must be also verified by the resonator vendor.

Figure 3-3 shows the recommended external oscillator circuits for both operating modes, external crystal mode and external input clock mode.

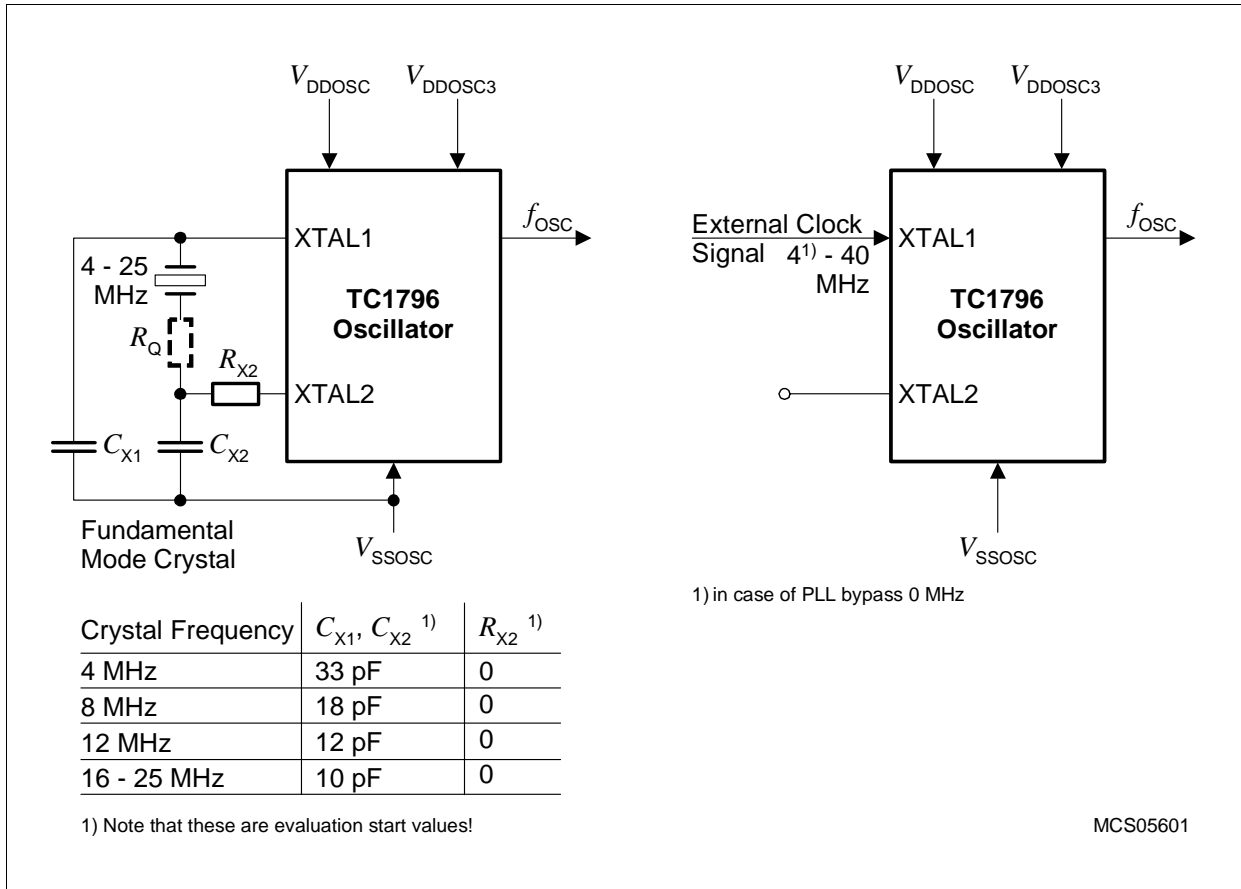


Figure 3-3 Main Oscillator Circuitry

3.2.1.1 Oscillator Bypass Mode

Especially for test purposes, the oscillator circuitry can be bypassed (disabled). In this oscillator bypass mode, the clock line f_{OSC} is disconnected from the oscillator circuitry and directly connected to the XTAL1 pin. The oscillator bypass mode is controlled by bit OSC_CON.MOSC.

- **Normal Mode** (OSC_CON.MOSC = 0):
 f_{OSC} is derived from the crystal or from an external clock signal
- **Oscillator Bypass Mode** (OSC_CON.MOSC = 1):
The oscillator is bypassed and f_{OSC} is directly derived from an external clock signal which is applied to XTAL1.

The MOSC bit can be set in two ways:

- By software: Writing a 1 to bit MOSC of register OSC_CON.
- By hardware at a power-on reset operation: If pin BYPASS = 1, the state of the pin P5.3/TXD1A is latched with the rising edge of \overline{PORST} and determines the inverted state of the MOSC bit. If P5.3 = 0 at the rising edge of \overline{PORST} , MOSC is set and the oscillator bypass mode is enabled (see also [Figure 3-1](#)).

3.2.1.2 Oscillator Run Detection

The oscillator run detection logic indicates during oscillator start-up after a power-on operation whether the oscillator is already running or if an emergency operation with PLL base frequency has to be started. When the oscillator run condition is met, bit OSC_CON.OSCR is set and the output clock f_{OSC} is enabled to supply the clock signal to the rest of the system. **Figure 3-4** shows the oscillator run detection logic.

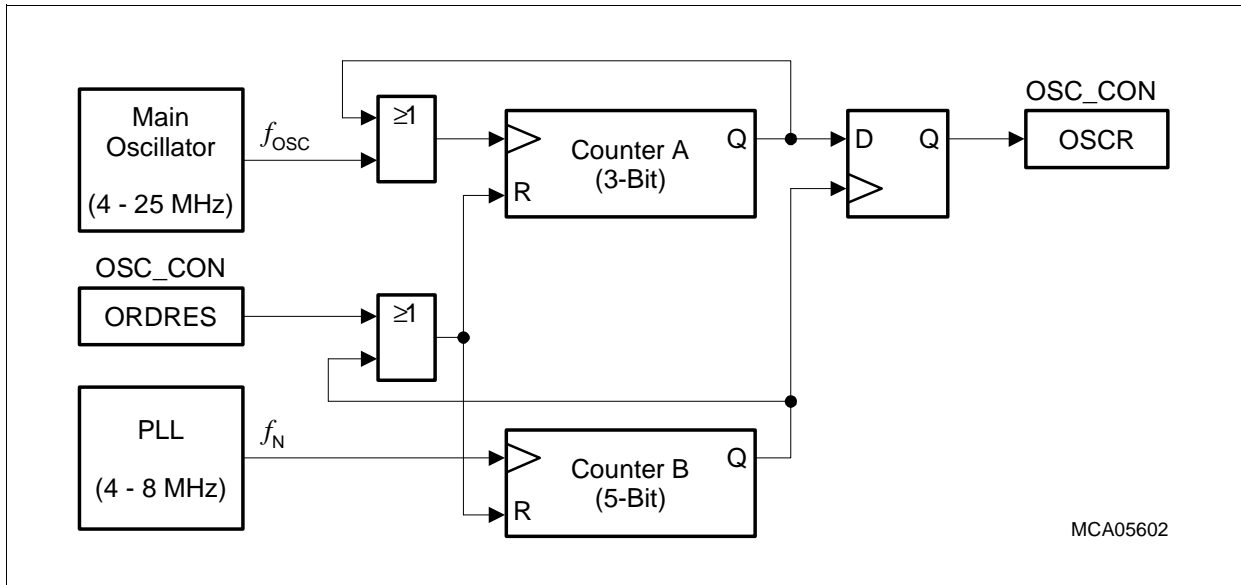


Figure 3-4 Oscillator Run Detection Circuitry

The oscillator run detection consists of two counters, Counter A and B. The 3-bit Counter A is running with the oscillator frequency and stops at its terminal count value. The 5-bit Counter B is running at f_N , the divided (N-Divider) VCO clock frequency. Always at the terminal count of Counter B the state of Counter A is latched in a flip-flop, bit OSCR is updated, and both counters are reset. This means, if Counter A does not reach its terminal count value ($8 f_{OSC}$ clock periods) within a counter period of Counter B ($32 f_N$ clock periods), the oscillator is designated as “not running” by OSCR = 0. If Counter A is at its terminal count value at the end of the counter period of Counter B, the “running” state (OSCR = 1) is detected.

The circuit can start without a reset and becomes defined after at least 32 pulses of counter B. Setting bit OSC_CON.ORDRES makes it possible to start the oscillator run detection during normal operation of the TC1796, e.g. in case of a PLL loss-of-lock condition.

3.2.1.3 Oscillator Gain Control

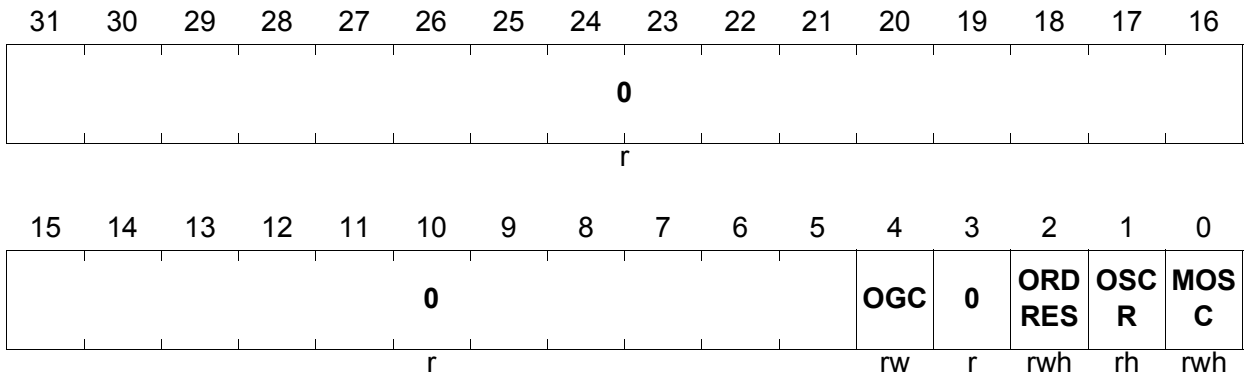
The oscillator starts with a high drive level (gain) during and after a power-on reset to ensure safe start-up behavior in the beginning (force the crystal oscillation). When a stable oscillation has been reached after oscillation start-up, the gain of the oscillator can be reduced. This reduces the oscillator's power consumption, which is especially important in the power saving modes. This gain reduction is selected by `OSC_CON.OGC = 1`.

Note: Oscillator measurement (margin or negative resistance and XTAL1 input amplitude) for the oscillator crystal system must be executed also with reduced gain.

3.2.1.4 Oscillator Control Register

Note: Register OSC_CON is Endinit-protected.

OSC_CON
Oscillator Control Register (F0000018_H) Reset Value: see Table 3-1



Field	Bits	Type	Description
MOSC	0	rwh	<p>Main Oscillator Test Mode</p> <p>This bit determines the mode of the main oscillator.</p> <p>0_B The oscillator is running. The oscillator signal of the main oscillator or an external clock input signal is used as f_{OSC}.</p> <p>1_B The oscillator circuitry is bypassed. An external clock input signal at XTAL1 must be provided and is used as f_{OSC}.</p> <p>It is 1 if the BYPASS pin is high and TXD1A is zero. Its state is latched with the rising edge of \overline{PORST}.</p>
OSCR	1	rh	<p>Oscillator Run Status Bit</p> <p>This bit shows the state of the oscillator run state.</p> <p>0_B The oscillator is not running.</p> <p>1_B The oscillator is running.</p> <p>The OSCR bit is valid always after a power-on reset operation. After an oscillator run detection reset (setting OSC_CON.ORDRES), OSCR is invalid up to a maximum of $64 f_N$ clock cycles.</p>

Clock System and Control

Field	Bits	Type	Description
ORDRES	2	rwh	<p>Oscillator Run Detection Reset This bit allows the oscillator run detection to start during normal operating mode.</p> <p>0_B No operation 1_B The oscillator run detection logic is reset and restarted.</p> <p>After ORDRES has been set, it becomes automatically reset by hardware.</p>
OGC	4	rw	<p>Oscillator Gain Control This bit determines the main oscillator gain.</p> <p>0_B High gain is selected (default after reset). 1_B Low gain is selected.</p>
0	3, [31:5]	r	<p>Reserved Read as 0; should be written with 0.</p>

Table 3-1 Reset Values of Register OSC_CON

Condition		Function	Reset Values
BYPASS	TXD1A		
0	X	The system is driven by the PLL clock which is derived from the oscillator clock.	0000 0000 _H
1	0	The system is driven directly by the external clock which is applied to XTAL1.	0000 0001 _H
	1	The system is driven by the crystal oscillator clock or the external clock passed through the oscillator.	0000 0000 _H

3.2.2 Phase Locked Loop (PLL) Circuitry

The PLL is a main component of the CGU that is dedicated to generate the CPU and system clock inside the TC1796. The PLL basically converts a low-frequency external clock signal into high-speed internal CPU and system clocks for maximum performance.

The PLL consists of a Voltage Controlled Oscillator (VCO) with a feedback path. A divider in the feedback path (N-Divider) divides the VCO frequency down. The resulting frequency is then compared with the externally provided and divided frequency (P-Divider). The phase detection logic determines the difference between the two clock signals and accordingly controls the frequency of the VCO (f_{VCO}). During start-up, the VCO increases its frequency until the divided feedback clock matches the external clock frequency. A PLL lock detection unit monitors and signals this condition. The phase detection logic continues to monitor the two clock signals and adjusts the VCO clock if required. The CGU output clocks f_{CPU} and f_{SYS} are derived from the VCO clock by the K-Divider.

3.2.2.1 Clock Source Control

The CPU clock f_{CPU} and the system clock f_{SYS} are generated from f_{OSC} in one of four hardware/software selectable modes:

- Direct Drive Mode (PLL Bypass Mode)
- VCO Bypass Mode (Prescaler Mode)
- PLL Mode
- PLL Base Mode

Direct Drive Mode (PLL Bypass Operation)

In Direct Drive Mode, the PLL is bypassed and the CGU clock outputs are directly fed from the clock signal f_{OSC} , i.e. $f_{CPU} = f_{OSC}$ and $f_{SYS} = f_{OSC}/2$ or f_{OSC} . This mode can be only selected by hardware when pin `BYPASS` = 1 during the rising edge of `PORST`.

VCO Bypass Mode (Prescaler Mode)

In VCO Bypass Mode, f_{CPU} and f_{SYS} are derived from f_{OSC} by the two divider stages, P-Divider and K-Divider. This mode is selected by setting `PLL_CLC.VCOBYP` = 1. The system clock f_{SYS} can be equal to f_{CPU} (`PLL_CLC.SYSFS` = 1) or equal to $f_{CPU}/2$ (`PLL_CLC.SYSFS` = 0).

$$f_{CPU} = \frac{1}{P \times K} \times f_{OSC} \tag{3.1}$$

$$f_{SYS} = f_{CPU} \text{ or } f_{CPU}/2$$

PLL Mode

In PLL Mode, the PLL is running. The VCO clock f_{VCO} is derived from f_{OSC} , divided by the P factor, multiplied by the PLL (N-Divider). This mode is selected by setting `PLL_CLC.VCOBYP = 0`. Further, f_{CPU} and f_{SYS} are derived from f_{VCO} by the K-Divider. The system clock f_{SYS} can be equal to f_{CPU} (`PLL_CLC.SYSFS = 1`) or equal to $f_{CPU}/2$ (`PLL_CLC.SYSFS = 0`).

$$f_{CPU} = \frac{N}{P \times K} \times f_{OSC} \quad (3.2)$$

$$f_{SYS} = f_{CPU} \text{ or } f_{CPU}/2$$

PLL Base Mode

In PLL Base Mode, the PLL is running at its VCO base frequency and f_{CPU} and f_{SYS} are derived from f_{VCO} only by the K-Divider. In this mode, `PLL_CLC.VCOBYP` must be 0 and pin `BYPASS` must have been latched as 0 at the end of the last power-on reset operation. In this mode, the system clock f_{SYS} can be equal to f_{CPU} (`PLL_CLC.SYSFS = 1`) or equal to $f_{CPU}/2$ (`PLL_CLC.SYSFS = 0`).

$$f_{CPU} = \frac{1}{K} \times f_{VCObase} \quad (3.3)$$

$$f_{SYS} = f_{CPU} \text{ or } f_{CPU}/2$$

3.2.2.2 PLL Parameters

As shown in [Equation \(3.2\)](#) to [Equation \(3.3\)](#), the PLL operation depends on the setup of up to four main PLL parameters: P-Divider, N-Divider, K-Divider, and VCO range selection.

P-Divider

The P-Divider divides the oscillator clock f_{OSC} by factor P for the PLL input clock f_P . [Table 3-2](#) shows the P factor values of the P-Divider which are selected by programming the PLL_CLC.PDIV bit field. It also lists the resulting f_P frequency for some dedicated values of f_{OSC} but the complete range of 4 to 40 MHz can be applied and used for f_{OSC} . Note that the P-Divider factor is always PLL_CLC.PDIV+1.

Table 3-2 P-Divider Selections

PLL_CLC. PDIV	P-Divider: P = PDIV+1	Resulting f_P Frequency (in MHz) for				
		$f_{osc} =$ 4 MHz	$f_{osc} =$ 10 MHz	$f_{osc} =$ 20 MHz	$f_{osc} =$ 30 MHz	$f_{osc} =$ 40 MHz
0	1	4	10	20	30	40
1	2	2	5	10	15	20
2	3	1.33	3.33	6.67	10	13.3
...
6	7	0.57	1.43	2.857	4.286	5.7
7	8	0.5	1.25	2.5	3.75	5

N-Divider

The N-Divider in the feedback path of the PLL divides the VCO clock f_{VCO} by factor N for the N-divider output clock f_N . This feedback clock is used as input clock for the PLL phase detection unit, which compares it with the PLL input clock f_P . The phase detector determines the difference between its two input clocks f_P and f_N and accordingly regulates the frequency of the VCO output clock f_{VCO} .

Table 3-3 shows the N factor values of the N-Divider which are selected by programming the PLL_CLC.NDIV bit field. It also lists the resulting N-divider output clock f_N depending on N and dedicated VCO frequencies. Note that the N-Divider factor is always PLL_CLC.NDIV+1. For proper operation of the PLL, only N-Divider values of 20 to 100 are allowed.

Table 3-3 N-Divider Selections

PLL_CLC. NDIV ¹⁾	N-Divider: N = NDIV+1 ¹⁾	Resulting f_N Frequency (in MHz) for			
		$f_{VCO} =$ 400 MHz	$f_{VCO} =$ 500 MHz	$f_{VCO} =$ 600 MHz	$f_{VCO} =$ 700 MHz
≤ 18	≤ 19	not allowed			
19	20	20	25	30	35
20	21	19.05	23.81	28.57	33,33
21	22	18.18	22.73	27.27	31.82
...
97	98	4.08	5.10	6.12	7.14
98	99	4.04	5.05	6.06	7.07
99 ²⁾	100	4	5	6	7
≥ 100	≥ 101	not allowed			

1) These columns include decimal values.

2) This is the default value after a power-on reset.

K-Divider

The K-Divider divides the VCO clock f_{VCO} by factor K for the CGU output clocks f_{CPU} (and f_{SYS}). **Table 3-4** shows the K factor values of the K-Divider that are selected by programming the PLL_CLC.KDIV bit field. It also lists the resulting output clock f_{CPU} depending on K and dedicated VCO frequencies. Note that the K-Divider factor is always PLL_CLC.KDIV+1. For odd K-Divider factors some restrictions for f_{CPUmax} must be regarded.

Table 3-4 K-Divider Selections

PLL_CLC. KDIV	K-Divider: K = KDIV+1	Resulting f_{CPU} Frequency (in MHz) for				f_{CPU}	
		$f_{VCO} = 400$ MHz	$f_{VCO} =$ 500 MHz	$f_{VCO} =$ 600 MHz	$f_{VCO} =$ 700 MHz	Duty Cycle [%]	Max. Value [MHz]
0	1	400	500	600	700	1)	1)
1	2	200	250	300	350		
2	3	133.33	166.67	200	233.33	33.33	100 ²⁾
3	4	100	125	150	175	50	150
4	5	80	100	120	140	40	120 ²⁾
5	6	66.67	83.33	100	116.67	50	150
6	7	57.14	71.43	85.71	100	42.86	130 ²⁾
7	8	50	62.5	75	87.5	50	150
8	9	44	55.56	66.67	77.78	44.44	130 ²⁾
9	10	40	50	60	70	50	150
10	11	36.36	45.45	54.55	63.64	45.45	140 ²⁾
11	12	33.33	41.67	50	58.33	50	150
12	13	30.77	38.46	46.15	53.85	46.15	140 ²⁾
13	14	28.57	35.71	42.86	50	50	150
14	15	26.67	33.33	40	46.67	46.67	145 ²⁾
15 ³⁾	16	25	31.25	37.5	43.75	50	150

1) These KDIV selections are not allowed in PLL Mode of the TC1796.

2) This is a restriction in f_{CPUmax} for odd K-divider factors.

3) This is the default value after a power-on reset.

Note: The shaded selections cannot be used because the maximum TC1796 f_{CPU} frequency of 150 MHz is exceeded.

VCO Operating Range

The VCO can be selected for three operating ranges by programming the PLL_CLC.VCOSEL bit field. **Table 3-5** defines the min./max. f_{VCO} frequency as well as the VCO base frequency $f_{VCObase}$. This is the base VCO frequency when no PLL input clock f_P is connected.

Table 3-5 VCO Operating Range Selection

Bit PLL_CLC.VCOSEL	f_{VCOmin}	f_{VCOmax}	$f_{VCObase}^{1)}$	$f_{OSCmin}^{2)}$	Unit
00 _B	400	500	approx. 140 - 320	1.5	MHz
01 _B	500	600	approx. 150 - 400	1.75	MHz
10 _B ³⁾	600	700	approx. 200 - 480	2	MHz
11 _B	Reserved, do not use this combination				

- 1) $f_{VCObase}$ is the free-running operation frequency of the PLL when no input clock is available.
- 2) This is the minimum oscillator frequency to allow oscillator run detection to work properly.
- 3) This is the default value after a power-on reset.

3.2.2.3 PLL Clock Control and Status Register

The PLL Clock Control and Status Register PLL_CLC is located in the address range of the System Control Unit (see [Page 5-62](#)). It holds the hardware configuration bits of the PLL and provides the control for the N, P and K-Dividers as well as the PLL lock status bit. Register PLL_CLC is Endinit-protected.

PLL_CLC

PLL Clock Control Register

(F000040_H)

Reset Values: see [Table 3-6](#)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16																
0		BYP PIN		0		OSC DISC		0		NDIV																					
r		rh		r		rwh		r		rw																					
15		14		13		12		11		10		9		8		7		6		5		4		3		2		1		0	
PDIV				0		KDIV				VCOSEL		VCO BYP		0		SYS FS		RES LD		LO CK											
rw				r		rw				rw		rw		r		rw		rwh		rh											

Field	Bits	Type	Description
LOCK	0	rh	PLL Lock Status Flag 0 _B PLL is not locked (default after reset) 1 _B PLL is locked
RESLD	1	rwh	Restart Lock Detection Writing a 1 to this bit will clear the LOCK flag and restart the PLL lock detection. After written with a 1, this bit is reset automatically to 0 and therefore always read back as 0. This bit becomes activated with a power-on reset operation. 0 _B No operation 1 _B PLL lock detection is restarted.
SYSFS	2	rw	System Frequency Select Selects the f_{CPU} -to- f_{SYS} clock ratio. 0 _B Ratio f_{CPU}/f_{SYS} is 2/1 (default after reset). 1 _B The ratio f_{CPU}/f_{SYS} is 1/1.
VCOBYP	5	rw	VCO Bypass Mode Selection 0 _B Normal operation (default after reset) 1 _B VCO Bypass Mode selected

Clock System and Control

Field	Bits	Type	Description
VCOSEL	[7:6]	rw	VCO Range Selection This bit field selects operating range of the VCO. The coding is defined in Table 3-5 .
KDIV	[11:8]	rw	PLL K-Divider Selection This bit field selects the K-Divider value. The coding is defined in Table 3-4 .
PDIV	[15:13]	rw	PLL P-Divider Selection This bit field selects the P-Divider value. The coding is defined in Table 3-2 .
NDIV	[22:16]	rw	PLL N-Divider This bit field selects the N-Divider value. The coding is defined in Table 3-3 . Note that only NDIV values between 19 _D and 99 _D (means N-Divider values of 20 _D and 100 _D) are allowed.
OSCDISC	24	rwh	Oscillator Disconnect This bit is used to disconnect the divided f_{OSC} clock from the PLL in order to avoid unstable operation due to noise or sporadic clock pulses coming from the oscillator circuit while the PLL is still trying to lock to invalid clock pulses. 0 _B Oscillator clock f_{OSC} is connected to the PLL. 1 _B Oscillator clock f_{OSC} is disconnected from the PLL (default after reset) This bit is set by hardware if a PLL loss-of-lock failure is detected.
BYPPIN	29	rh	Bypass Pin Status Flag This bit indicates the state of the BYPASS input pin as sampled with the last rising edge of PORST.
0	3, 4, 12, 23, [28:25], [31:30]	r	Reserved Read as 0; should be written with 0.

Table 3-6 Reset Values of Register PLL_CLC

Reset	BYPASS	Function	Reset Values
Power-on reset	0	The system is driven by the PLL clock based on the VCO base frequency.	0163 0F80 _H
	1	The system is driven directly by the oscillator clock output.	2163 0F80 _H
Other resets	X	Register content remains unmodified	UUUU UUUU _H ¹⁾

1) U = unchanged to previously programmed values.

3.2.2.4 Changing PLL Parameters

There are some restrictions that must be regarded when PLL parameters in register PLL_CLC are modified.

- Only one parameter should be changed with a PLL_CLC register write operation
- SYSFS can be changed without precautions
- VCOBYP can be changed without precautions
- PDIV and KDIV can be switched at any time in VCO Bypass Mode. However, the maximum operating frequency of the TC1796 must not be exceeded.
- Before changing VCOSEL, the VCO Bypass Mode must be selected.
- Before deselecting the VCO Bypass Mode, PLL lock detection must be restarted (RESLD bit set) and then the LOCK flag must be checked for the PLL lock condition. When LOCK is set, the VCO Bypass Mode can be deselected again.

Note: PDIV and NDIV can also be switched in PLL Mode. When changing NDIV, the VCO clock f_{VCO} may exceed the target frequency until the PLL becomes locked. After changing PDIV or NDIV, wait for the PLL lock condition. This procedure is typically used for increasing the VCO clock step-by-step.

3.2.2.5 Setting up the PLL after Reset

After reset, the system clock will be running at the VCO base frequency $f_{VCObase}$ divided by factor K. Now the following actions must be executed next:

1. Wait until the oscillator is running (OSC_CON.OSCR = 1)
2. Selection of the VCO Bypass Mode (PLL_CLC.VCOBYP = 1)
3. Selecting the VCO band by programming PLL_CON.VCOSEL
4. Program the desired P, N and K values (PDIV, NDIV, and KDIV bit fields of register PLL_CLC)
5. Connect the oscillator to the PLL, default after reset (PLL_CLC.OSCDISC = 0)
6. Wait until the PLL becomes locked (PLL_CLC.LOCK = 1)
7. Disable the VCO Bypass Mode (PLL_CLC.VCOBYP = 0)

After this procedure, the device is operating on the PLL target frequency. The note in [Section 3.2.2.4](#) is also valid for this procedure.

3.2.2.6 Lock Detection

The PLL has the capability to detect a failure of its input clock f_P and to bring the TC1796 into a safe state in such a case. This clock failure detection is done by the PLL lock detection unit. This unit indicates whether the PLL has reached its target frequency properly or not.

The lock detection unit operates as follows:

Two counters, A and B, count the clock pulses of the N-divider clock output f_N and the PLL reference clock f_P . When the counter values differ by more than 2 during counting, the counters are reset (meaning PLL is still unlocked). When the counter values reach the end of a counting session (224 clock pulses) with a counting difference which is 2 or less than 2, the PLL is locked (PLL_CLC.LOCK = 1).

When the PLL is locked, the two counters proceed to count clock pulses. After every fourth clock pulse, the counter values are checked, and when the counter values differ by more than 2, the unlocked state is entered (fast unlock check). If no unlock condition is detected (counter difference less than or equal 2), the counters are further incremented up to a maximum counter value of 232 clock pulses. After 232 clock pulses with no unlock condition of the counter values, the two counters are reset (slow unlock check).

The PLL may become unlocked, caused by a break of the crystal or the external clock line. In such a case, an NMI trap is generated by setting the NMISR.PLLNMI flag. Additionally, the PLL clock input f_P is disconnected from the PLL to avoid unstable operation due to noise or sporadic clock pulses coming from the oscillator circuit. Without a clock input f_P , the PLL gradually slows down to its VCO base frequency and remains there. The TC1796 remains in this state until the next power-on reset through pin $\overline{\text{PORST}}$, after that the PLL tries to restart and lock to the external clock again. No other reset cause can terminate this loss-of-clock state. This is done to avoid unstable operation due to the PLL trying to lock again. The TC1796 remains in the PLL unlocked state until the next power-on reset or a successful lock recovery occurs.

Note that the PLL unlock state can also be entered when the oscillator disconnect bit PLL_CLC.DISC becomes set by software.

3.2.2.7 Loss-of-Lock Recovery

If PLL has lost the lock condition, user software can try to re-lock the PLL again by executing the following sequence:

1. Restart the oscillator run detection by setting bit OSC_CON.ORDRES
2. Wait until OSCR is set

Clock System and Control

3. If bit OSCR is set, then
 - Select the VCO Bypass Mode (PLL_CLC.VCOBYP = 1)
 - Re-connect the oscillator to the PLL (PLL_CLC.OSCDISC = 0)
 - Set the restart lock detection bit PLL_CLC.RESLD = 1
 - Wait until the PLL becomes locked (PLL_CLC.LOCK = 1)
 - When the PLL_CLC.LOCK is set again, the VCO Bypass Mode can be deselected (PLL_CLC.VCOBYP = 0) and normal PLL operation is resumed.

The note in [Section 3.2.2.4](#) is also valid for this procedure.

3.2.3 Power-on Startup Operation

In order to support a wide range of input clock frequencies, the TC1796 requires a generic procedure to start-up the system clock.

When the TC1796 is powered up, a low level (0) must be applied to the power-on reset pin, $\overline{\text{PORST}}$. While $\overline{\text{PORST}}$ is active (at low level), the device is asynchronously held in reset and the state of the BYPASS pin controls the operation of the clock circuitry. Therefore, BYPASS must be held at constant level during $\overline{\text{PORST}}$ active.

If the BYPASS pin is at a high level during power-on reset, Direct Drive Mode is selected (see [Page 3-10](#)). The low level at pin $\overline{\text{PORST}}$ has to be held long enough to make sure that a stable clock is provided to the TC1796. In case of an external crystal oscillator, it can take several ms until the oscillator has started up and is stable. If the clock input is provided by another clock source with faster startup characteristics, the $\overline{\text{PORST}}$ low level can be released earlier.

If the BYPASS pin is at low level during power-on reset, PLL Mode is selected (see [Page 3-11](#)) and the procedure to start-up the PLL is as described in the next paragraph.

With $\overline{\text{PORST}} = 0$, the oscillator is disconnected from the PLL. The PLL starts running at the VCO base frequency f_{VCObase} . After deactivation of the $\overline{\text{PORST}}$, the CPU and system clock are provided internally with a frequency that is equal to f_{VCObase}/K ($K = 16$ after reset). This means that in the TC1796, the Boot ROM code execution is started with the VCO base frequency. When the oscillator is running properly, bit OSCR becomes set and the user software can setup the PLL by programming its parameters as described in [Section 3.2.2.4](#). If OSCR becomes not set, the user software has the possibility to run an emergency program using the base frequency of the PLL divided by K . In this case, K can be set to the minimum value, which results in the maximum possible CPU and system frequency.

Note: See also [Page 4-7](#) for further details on the power-on reset operation.

3.3 Module Power Management and Clock Gating

Because power dissipation is related to the frequency of gate transitions, the TC1796 performs power management principally by clock gating – that is, controlling whether the clock is supplied to its various functional units. Gating off the clock to unused functional modules also reduces electro-magnetic interference (EMI) since EMI is related to both the frequency and the number of gate transitions.

Clock gating is done either dynamically or statically. Dynamic clock gating in this context means that the TC1796 itself enables or disables clock signals within some functional modules to conserve power. Static gating means that software must enable or disable clock signals to functional modules. Clock gating is performed differently at different levels of system scope: Dynamic gating is generally performed at the lowest levels, either within a small region of logic, or at functional-unit boundaries for uncomplicated functions where hardware can dynamically determine whether that functionality is required, and can enable or disable it appropriately without software intervention. Static gating - which requires software intervention - is used to enable or disable clock delivery to individual high-level functional units, or to disable clock delivery globally at the clock's source. When the clock to individual functional units is gated off, they are said to be in Sleep Mode.

The TC1796 implements three levels of clock gating:

1. **Gated dynamically at the register:**

The clock is shut off to a particular local resource in a functional module when this resource is not being used in that clock cycle. This operation is done primarily in the CPU and the PCP data paths, where unused resources are easily identified and controlled in each clock cycle.

2. **Gated dynamically at the functional unit (Idle Mode):**

The clock is shut off at the functional unit boundary when the unit has nothing useful to do. This operation is done primarily in the CPU and the PCP. For the CPU, idle mode is controlled via software. The PCP disables its own clock when no program is running.

3. **Gated statically at each functional unit (Sleep Mode):**

Software can send a global sleep request to individual functional units requesting that they enter Sleep Mode. Software must determine when conditions are such that entering Sleep Mode is appropriate. The individual units can be programmed to ignore or respond to this signal. If programmed to respond, units will first complete pending operations, then will shut off their own clocks according to their own criteria.

3.3.1 Module Clock Generation

As shown in [Figure 3-5](#), module clock generation of the TC1796 on-chip modules have two registers implemented:

- Clock Control Register CLC
- Fractional Divider Register FDR

The following sections describes the general functionality of CLC and FDR. The module-specific implementation details are described in the corresponding module chapters.

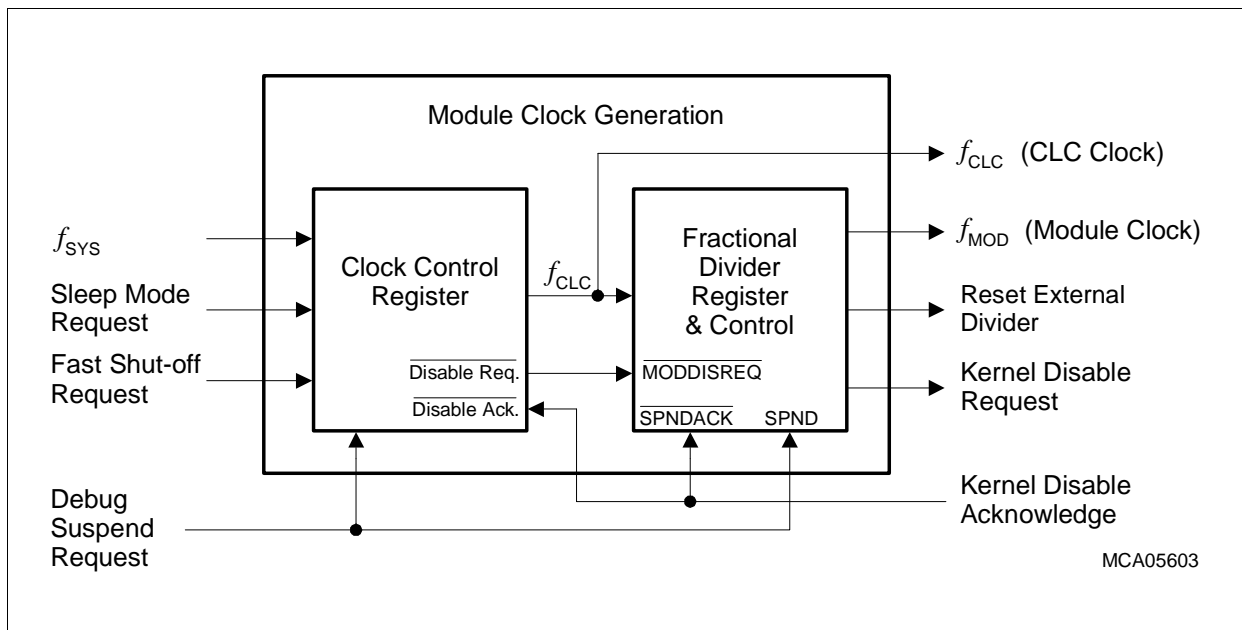


Figure 3-5 Module Clock Generation

Module clock and CLC clock are both derived from the system clock f_{SYS} . The CLC register provides the f_{CLC} clock which acts as clock input for the fractional divider and control logic. The CLC clock f_{CLC} is typically used by a peripheral module for clocking its FPI Bus interface and registers, while the module clock f_{MOD} is dedicated for kernel operation or timer clocks. The output signal RST_EXT_DIV makes it possible to enable/disable external divider stages which are connected to the module clock f_{MOD} . The fractional divider divides the f_{CLC} either by the factor $1/n$ or by a fraction of $n/1024$ for any value of n from 0 to 1023.

Furthermore, the module clock generation unit handles the sleep mode request signal, the fast shut-off request signal, and the debug suspend request signal.

3.3.2 Clock Control Register CLC

All CLC registers have basically the same bit and bit field layout. However, not all CLC register functions are implemented for each peripheral unit. [Table 3-9](#) defines in detail which bits and bit fields of the CLC registers are implemented for each clock control register.

The CLC register controls the generation of the peripheral module clock which is derived from the system clock. The following functions for the module are associated with the CLC register:

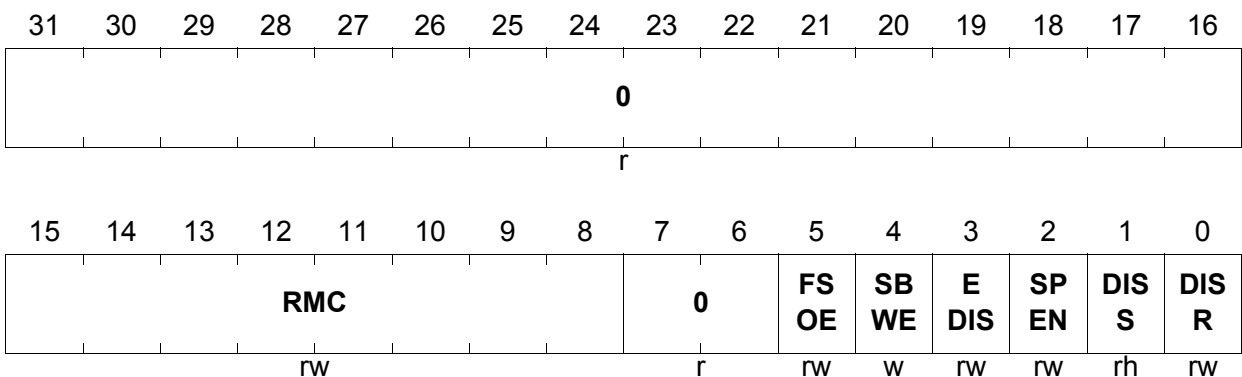
- Peripheral clock static on/off control
- Module clock behavior in Sleep Mode
- Operation during Debug Suspend Mode
- Fast Shut-off Mode control

MOD_CLC

Clock Control Register

(00_H)

Reset Value: Module-specific



Field	Bits	Type	Description
DISR	0	rw	Module Disable Request Bit Used for enable/disable control of the module. 0 _B Module disable is not requested 1 _B Module disable is requested
DISS	1	rh	Module Disable Status Bit Bit indicates the current status of the module 0 _B Module is enabled 1 _B Module is disabled If the RMC field is implemented and if it is 0, DISS is set to 1 automatically.

Clock System and Control

Field	Bits	Type	Description
SPEN	2	rw	<p>Module Suspend Enable Used for enabling the suspend mode.</p> <p>0_B Module cannot be suspended (suspend is disabled).</p> <p>1_B Module can be suspended (suspend is enabled). This bit can be written only if SBWE is set to 1 during the same write operation.</p>
EDIS	3	rw	<p>Sleep Mode Enable Control Used for module sleep mode control.</p> <p>0_B Sleep mode request is regarded. Module is enabled to go into sleep mode.</p> <p>1_B Sleep mode request is disregarded: Sleep mode cannot be entered on a request.</p>
SBWE	4	w	<p>Module Suspend Bit Write Enable for OCDS Determines whether SPEN and FSOE are write-protected.</p> <p>0_B Bits SPEN and FSOE are write-protected</p> <p>1_B Bits SPEN and FSOE are overwritten by respective value of SPEN or FSOE</p> <p>This bit is a write-only bit. The value written to this bit is not stored. Reading this bit returns always 0.</p>
FSOE	5	rw	<p>Fast Switch Off Enable Used for fast clock switch-off in OCDS suspend mode.</p> <p>0_B Clock switch-off in OCDS suspend mode via Disable Control Feature (Secure Clock Switch Off) selected</p> <p>01_B Fast clock switch off in OCDS suspend mode selected</p> <p>This bit can be written only if SBWE is set to 1 during the same write operation.</p>
RMC	[15:8]	rw	<p>8-Bit Clock Divider Value in RUN Mode This is a maximum 8-bit divider value for clock f_{SYS}. If RMC is set to 0 the module is disabled.</p>
0	7, 6, [31:16]	r	<p>Reserved Read as 0; should be written with 0.</p>

Module Enable/Disable Control

If a module is not used at all by an application, it can be completely shut off by setting bit DISR in its CLC register. For peripheral modules with a run mode clock divider field RMC, a second option to completely switch off the module is to set bit field RMC to 00_H. This also disables the module's operation.

The status bit DISS always indicates whether a module is currently switched off (DISS = 1) or switched on (DISS = 0). With a few exceptions (e.g. EBU_CLC), the default state of a peripheral module after reset is "module disabled" with DISS set (see [Table 3-9](#)).

Write operations to the registers of disabled modules are not allowed. However, the CLC of a disabled module can be written. An attempt to write to any of the other writable registers of a disabled module except CLC will cause the corresponding Bus Control Unit (BCU) to generate a bus error.

A read operation of registers of a disabled module (except CAN) is allowed and does not generate a bus error.

When a disabled module is switched on by writing an appropriate value to its MOD_CLC register (DISR = 0 and RMC (if implemented) > 0), status bit DISS changes from 1 to 0. During the phase in which the module becomes active, any write access to corresponding module registers (when DISS is still set) will generate a bus error. Therefore, when enabling a disabled module, application software should check after activation of the module once (read back of the CLC register) to find out whether DISS is already reset, before a module register (including the CLC register) will be written to.

Note: A read access occurring while a module is disabled is treated as a normal read access. This means, if a module register or a bit of it is cleared as a side-effect of a read access of an enabled module, it will not be cleared by this read access while the module is disabled.

Sleep Mode Control

The EDIS bit in the CLC register controls whether or not a module is stopped during sleep mode. If EDIS is 0 (default after reset), a sleep mode request can be recognized by the module and, when received, its clock is shut off.

If EDIS is set to 1, a sleep mode request is disregarded by the module and the module continues its operation.

Debug Suspend Mode Control

During emulation and debugging of TC1796 applications, the execution of an application program can be suspended. When an application is suspended, normal operation of the application's program is halted, and the TC1796 begins (or resumes) executing a special debug monitor program. When the application is suspended, a suspend request signal is generated by the TC1796 and sent to all modules. If bit SPEN is set to 1, the operation

of the peripheral module is stopped when the suspend signal is asserted. If SPEN is set to 0, the module does not react to the suspend request signal but continues its normal operation. This feature allows each peripheral module to be adapted to the unique requirements of the application being debugged. Setting SPEN bits is usually performed by a debugger.

This feature is necessary because application requirements typically determine whether on-chip modules should be stopped or left running when an application is suspended for debugging. For example, a peripheral unit that is controlling the motion of an external device through motors in most cases must not be stopped so as to prevent damage of the external device due to the loss of control through the peripheral. On the other hand, it makes sense to stop the system timer while the debugger is actively controlling the chip because it should only count the time when the user's application is running.

Note that it is never appropriate for application software to set the SPEN bit. The debug suspend mode should only be set by a debug software. To guard against application software accidentally setting SPEN, bit SPEN is specially protected by the mask bit SBWE. The SPEN bit can only be written if, during the same write operation, SBWE is set, too. Application software should never set SBWE to 1. In this way, user software can not accidentally alter the value of the SPEN bit that has been set by a debugger.

Note: The operation of the Watchdog Timer is always automatically stopped in debug suspend mode.

Entering Disabled Mode

Software can request that a peripheral unit be put into Disabled Mode by setting DISR. A module will also be put into Disabled Mode if the sleep mode is requested and the module is configured to allow Sleep Mode.

In Secure Shut-off Mode, a module first finishes any operation in progress, then proceeds with an orderly shut down. When all sub-components of the module are ready to be shut down, the module signals its clock control unit, which turns off the clock to this peripheral unit, that it is now ready for shut down. The status bit DISS is updated by the peripheral unit accordingly.

The kernel logic of the peripheral unit and its FPI Bus interface must both perform shut-down operations before the clock can be shut off in Secure Shut-off Mode. This is performed as follows. The peripheral module's FPI Bus interface provides an internal acknowledge signal as soon as any current bus interface operation is finished. For example, if there is a DMA write access to a peripheral in progress when a disable request is detected, the access will be terminated correctly. Similarly, the peripheral's kernel provides an internal acknowledge signal when it has entered a stable state. The clock control unit for that peripheral module shuts off the module's clock when it receives both acknowledge signals.

During emulation and debugging, it may be necessary to monitor the instantaneous state of the machine – including all or most of its modules – at the moment a software

Clock System and Control

breakpoint is reached. In such cases, it may not be desired that the kernel of a module finish whatever transaction is in progress before stopping, because that might cause important states in this module to be lost. Fast Shut-off Mode, controlled by bit FSOE, is available for this situation.

If FSOE = 0, modules are stopped as described above. This is called Secure Shut-off Mode. The module kernel is allowed to finish whatever operation is in progress. The clock to the unit is then shut off if both the bus interface and the module kernel have finished their current activity. If Fast Shut-off Mode is selected (FSOE = 1), clock generation to the unit is stopped as soon as any outstanding bus interface operation is finished. The clock control unit does not wait until the kernel has finished its transaction. This option stops the unit's clock as fast as possible, and the state of the unit will be the closest possible to the time of the occurrence of the software breakpoint.

Note: In all TC1796 modules except MultiCAN and DMA, the only shut down operating mode that is available is the Fast Shut-off Mode TC1796, regardless of the state of the FSOE bit.

Whether Secure Shut-off Mode or Fast Shut-off Mode is required depends on the application, the needs of the debugger, and the type of unit. For example, the analog-to-digital converter might allow the converter to finish a running analog conversion before it can be suspended. Otherwise the conversion might be corrupted and a wrong value could be produced when Debug Suspend Mode is exited and the unit is enabled again. This would affect further emulation and debugging of the application's program.

On the other hand, if a problem is observed to relate to the operation of the external analog-to-digital converter itself, it might be necessary to stop the unit as fast as possible in order to monitor its current instantaneous state. To do this, the Fast Shut-off Mode option would be selected. Although proper continuation of the application's program might not be possible after such a step, this would most likely not matter in such a case.

Note that it is never appropriate for application software to set the FSOE bit. Fast Shut-off Mode should only be set by debug software. To guard against application software accidentally setting FSOE, bit FSOE is specially protected by the mask bit SBWE. The SPEN bit can only be written if, during the same write operation, SBWE is set, too. Application software should never set SBWE to 1. In this way, user software can not accidentally alter the value of the FSOE bit. Note that this is the same guard mechanism used for the SPEN bit.

Module Clock Divider Control

Two peripheral modules of the TC1796, ASC0_CLC and STM_CLC, have a RMC control bit field in their CLC registers. This Run Mode Clock control bit field makes it possible to slow down the CLC clock via a programmable clock divider circuit.

Clock System and Control

A value of 00_H in RMC disables the clock signals to these modules (CLC clock is switched off). If RMC is not equal to 00_H, the CLC clock for a unit is generated as

$$f_{\text{CLC}} = f_{\text{SYS}} / \text{RMC}_{\text{MOD}} \quad (3.4)$$

where RMC is the content of its CLC register RMC field with a range of 1 to 255. If RMC is not available in a CLC register, the CLC clock frequency f_{CLC} is always equal to the frequency of f_{SYS} .

Note: The number of module clock cycles (wait states) that are required for a “destructive read” access (means: flags/bits are set/cleared by a read access) to a module register of a peripheral unit depends on the selected CLC clock frequency.

Therefore, a slower CLC clock (selected via bit field RMC in the CLC register) may result in a longer read cycle access time on the FPI Buses for peripheral units with “destructive read” access (e.g. the ASC).

3.3.3 Fractional Divider Operation

This section describes the module clock generation using the Fractional Divider.

3.3.3.1 Overview

The fractional divider makes it possible to generate a module clock from an input clock using a programmable divider. The fractional divider divides the input clock f_{IN} either by the factor $1/n$ or by a fraction of $n/1024$ for any value of n from 0 to 1023, and outputs the clock signal, f_{OUT} . The fractional divider is controlled by the FDR register. **Figure 3-6** shows the fractional divider block diagram.

The adder logic of the fractional divider can be configured for two operating modes:

- Reload counter (addition of +1), generating an output clock pulse on counter overflow
- Adder that adds a STEP value to the RESULT value and generates an output clock pulse on counter overflow

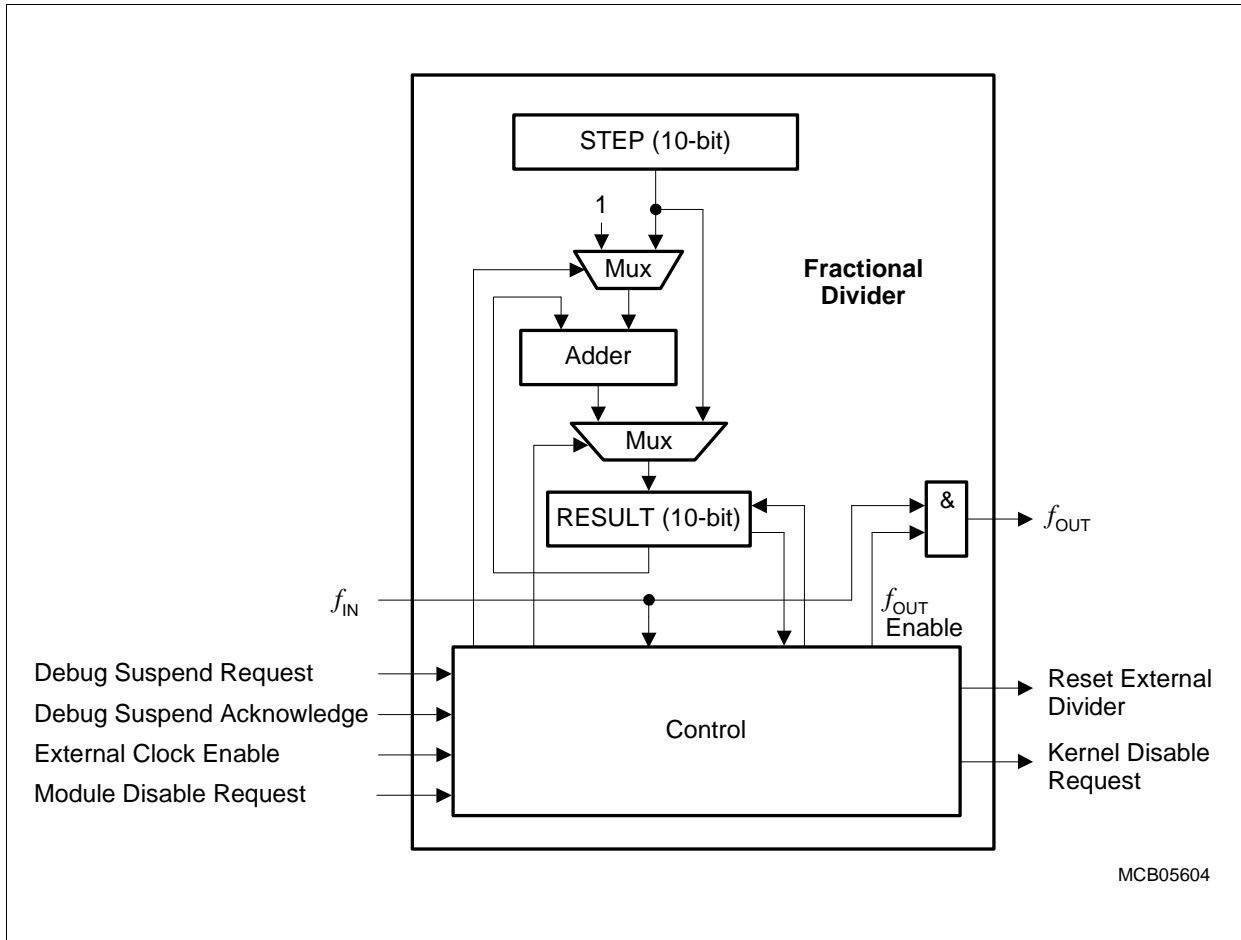


Figure 3-6 Fractional Divider Block Diagram

The adder logic of the fractional divider can be configured for two operating modes:

- **Normal Mode:** Reload counter (RESULT = RESULT + 1), generating an output clock pulse on counter overflow.
- **Fractional Divider Mode:** Adder that adds a STEP value to the RESULT value and generates an output clock pulse on counter overflow.

The fractional divider is further controlled by several input and output signals. The purpose of these signals is described in [Table 3-7](#).

Table 3-7 Fractional Divider Control I/O Lines

Signal	I/O	Description
Debug Suspend Request	Input	This input becomes active when a general suspend request is issued from the debug system to the on-chip modules.
Debug Suspend Acknowledge		This input is driven with the disable acknowledge signal from the module kernel. This disable acknowledge signal is activated by the module kernel as a response to a suspend request that has been generated by the fractional divider via the Kernel Disable Request signal.
External Clock Enable		This input can be used to synchronize the fractional divider clock generation to external events.
Module Disable Request		This input is connected to the disable request output from the CLC logic (see Figure 3-5). An active signal at this input activates the Kernel Disable Request signal.
Kernel Disable Request	Output	This output signal becomes active when either the Module Disable Request input or the Debug Suspend Request input become active.
Reset External Divider		This output signal makes it possible to control (stop/reset) external divider stages which have f_{OUT} as input.

Note: In the TC1796, the fractional divider input clock f_{IN} is also referred to as f_{CLC} and the fractional divider input clock f_{OUT} is also referred to as f_{MOD} (see [Figure 3-5](#)).

3.3.3.2 Fractional Divider Operating Modes

The fractional divider has two operating modes:

- Normal divider mode
- Fractional divider mode

Normal Divider Mode

In normal divider mode (FDR.DM = 01_B), the fractional divider behaves as a reload counter (addition of +1) that generates an output clock pulse at f_{OUT} on the transition from 3FF_H to 000_H. FDR.RESULT represents the counter value and FDR.STEP determines the reload value.

The output frequencies in normal divider mode are defined according to the following formulas:

$$f_{OUT} = f_{IN} \times \frac{1}{n} \quad , \text{ with } n = 1024 - \text{STEP} \quad (3.5)$$

In order to get $f_{OUT} = f_{IN}$ STEP must be programmed with 3FF_H. **Figure 3-7** shows the operation of the normal divider mode with a reload value of FDR.STEP = 3FD_H. The clock signal f_{OUT} is the AND combination of the f_{OUT} Enable signal with f_{IN} .

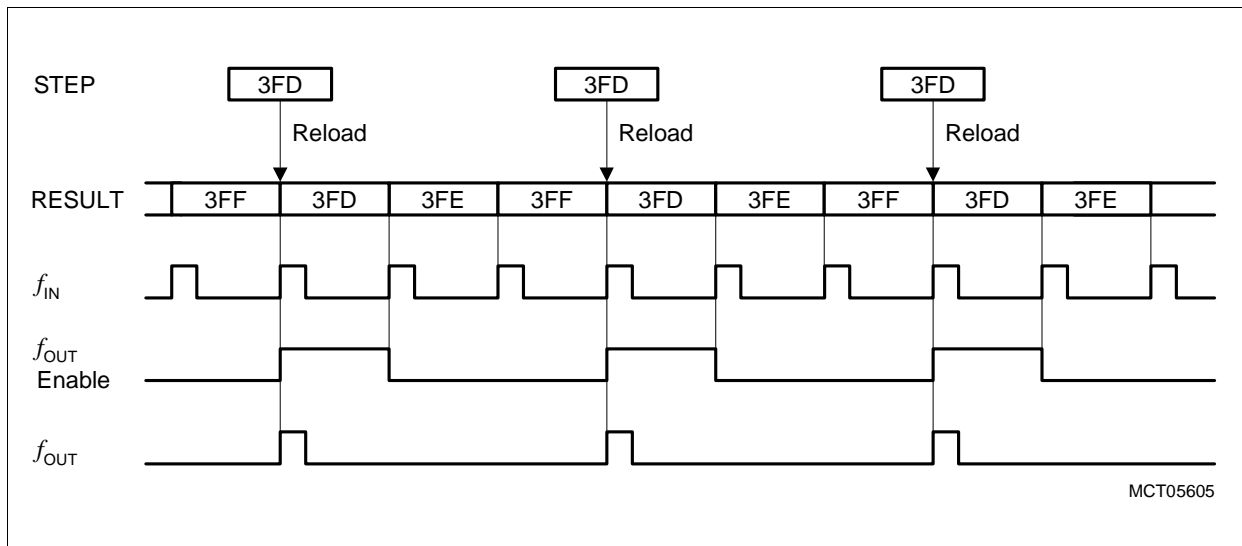


Figure 3-7 Normal Mode Timing

Fractional Divider Mode

When the fractional divider mode is selected (FDR.DM = 10_B), the output clock f_{OUT} is derived from the input clock f_{IN} by division of a fraction of $n/1024$ for any value of n from 0 to 1023. In general, the fractional divider mode makes it possible to program the average output clock frequency with a higher accuracy than in normal divider mode.

In fractional divider mode, an output clock pulse at f_{OUT} is generated depending on the result of the addition FDR.RESULT + FDR.STEP. If the addition leads to an overflow over 3FF_H, a pulse is generated at f_{OUT} . Note that in fractional divider mode the clock f_{OUT} can have a maximum period jitter of one f_{IN} clock period.

The output frequencies in fractional divider mode are defined according to the following formulas:

$$f_{OUT} = f_{IN} \times \frac{n}{1024} \quad , \text{ with } n = 0-1023 \quad (3.6)$$

Figure 3-8 shows the operation of the fractional divider mode with a reload value of FDR.STEP = 234_H (= factor 564/1024 = 0.55). The clock signal f_{OUT} is the AND combination of the f_{OUT} Enable signal with f_{IN} .

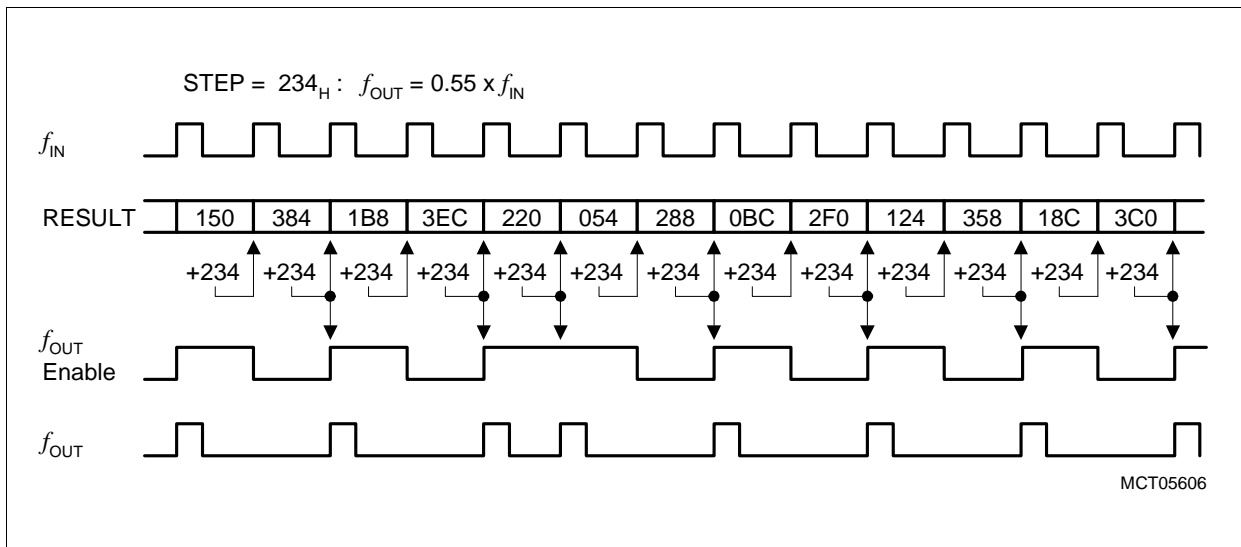


Figure 3-8 Fractional Divider Mode Timing

Suspend Mode Control

The operation of the fractional divider can be controlled by the Debug Suspend Request input. This input is activated in suspend mode by the on-chip debug control logic. In suspend mode, module registers are accessible for read and write actions, but other module internal functions are frozen. Suspend mode is entered one f_{IN} clock cycle after the Debug Suspend Request has been acknowledged by the Debug Suspend Acknowledge signal (granted suspend mode) **and** FDR.SC is not equal 00_B (clock output signal disabled). Suspend mode is immediately entered when bit SM is set to 1 **and** FDR.SC is not equal 00_B (immediate suspend mode).

The state of the Debug Suspend Request and Debug Suspend Acknowledge signal is latched in two status flags of register FDR, SUSREQ and SUSACK. Debug Suspend Request and (Debug Suspend Acknowledge or bit SM) must remain set both to maintain the suspend mode.

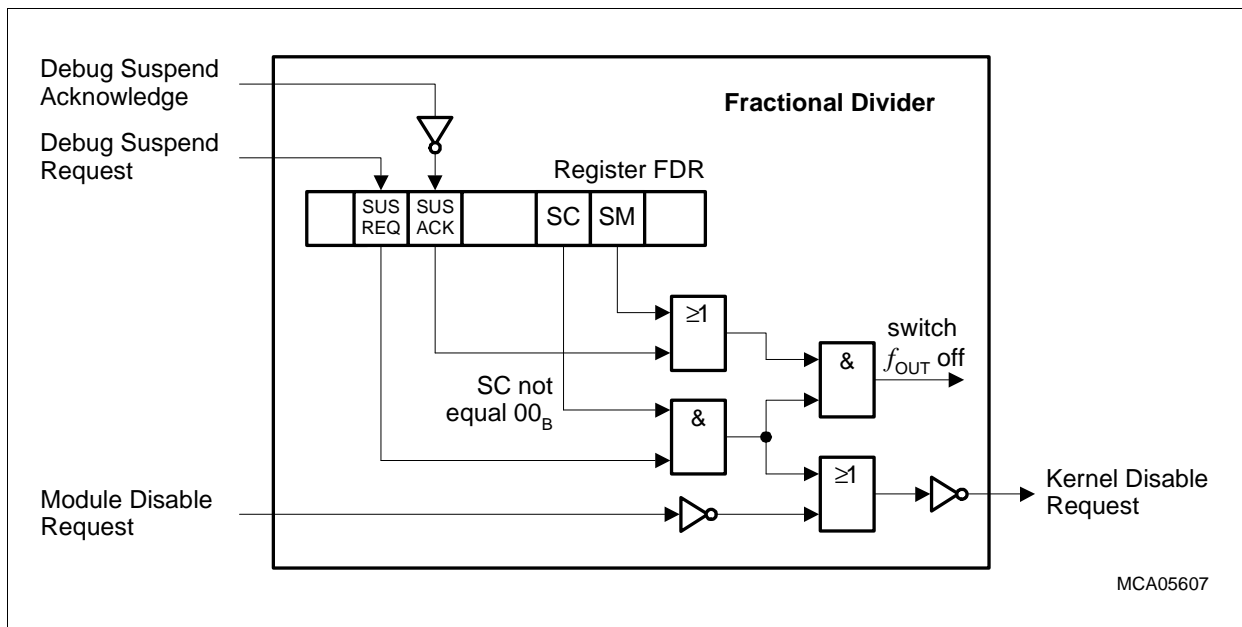


Figure 3-9 Suspend Mode Configuration

The Kernel Disable Request signal becomes always active when the Module Disable Request signal is activated, independently of the suspend mode settings in the fractional divider logic.

External Clock Enable

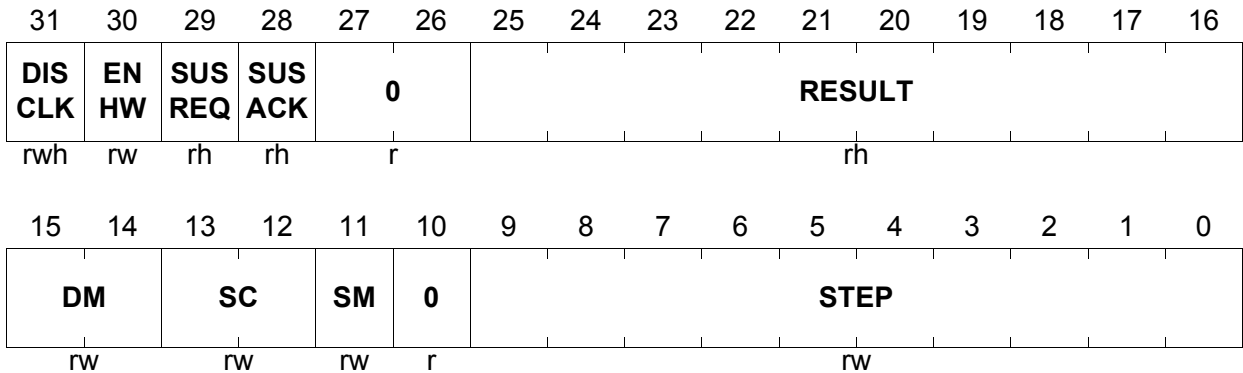
When the module clock generation has been disabled by software (setting FDR.DISCLK = 1), the disable state can be exited (hardware controlled) when the External Clock Enable input = 1. This feature is enabled when FDR.ENHW = 1.

3.3.3.3 Fractional Divider Register

FDR

Fractional Divider Register

Reset Value: 0000 0000_H



Field	Bits	Type	Description
STEP	[9:0]	rw	<p>Step Value</p> <p>In normal divider mode, STEP contains the reload value for RESULT.</p> <p>In fractional divider mode, this bit field determines the 10-bit value that is added to RESULT with each input clock cycle.</p>
SM	11	rw	<p>Suspend Mode</p> <p>SM selects between granted or immediate suspend mode.</p> <p>0_B Granted suspend mode selected</p> <p>1_B Immediate suspend mode selected</p>

Clock System and Control

Field	Bits	Type	Description
SC	[13:12]	rw	<p>Suspend Control This bit field determines the behavior of the fractional divider in suspend mode (bit SUSREQ and SUSACK set).</p> <p>00_B Clock generation continues. 01_B Clock generation is stopped and the clock output signal is not generated. RESULT is not changed except when writing bit field DM with 01_B or 10_B. 10_B Clock generation is stopped and the clock output signal is not generated. RESULT is loaded with 3FF_H. 11_B Same as SC = 10_B but signal Reset External Divider is 1 (independently of bit field DM).</p>
DM	[15:14]	rw	<p>Divider Mode This bit fields determines the functionality of the fractional divider block.</p> <p>00_B Fractional divider is switched off; no output clock is generated. The Reset External Divider signal is 1. RESULT is not updated (default after reset). 01_B Normal divider mode selected. 10_B Fractional divider mode selected. 11_B Fractional divider is switched off; no output clock is generated. RESULT is not updated.</p>
RESULT	[25:16]	rh	<p>Result Value In normal divider mode, RESULT acts as reload counter (addition +1). In fractional divider mode, this bit field contains the result of the addition RESULT + STEP. If DM is written with 01_B or 10_B, RESULT is loaded with 3FF_H.</p>
SUSACK	28	rh	<p>Suspend Mode Acknowledge 0_B Suspend mode is not acknowledged. 1_B Suspend mode is acknowledged. Suspend mode is entered when SUSACK and SUSREQ are set.</p>

Clock System and Control

Field	Bits	Type	Description
SUSREQ	29	rh	<p>Suspend Mode Request</p> <p>0_B Suspend mode is not requested. 1_B Suspend mode is requested. Suspend mode is entered when SUSREQ and SUSACK are set.</p>
ENHW	30	rw	<p>Enable Hardware Clock Control</p> <p>0_B Bit DISCLK cannot be cleared by hardware by a high level of the External Clock Enable input signal. 1_B Bit DISCLK is cleared by hardware while the External Clock Enable input signal is at high level.</p>
DISCLK	31	rwh	<p>Disable Clock</p> <p>0_B Clock generation of f_{OUT} is enabled according to the setting of bit field DM. 1_B Fractional divider is stopped. Signal f_{OUT} becomes inactive. No change except when writing bit field DM.</p> <p>In case of a conflict between hardware clearing and software setting of DISCLK, the software setting wins. Any write or read-modify-write action leads to the described behavior. As a result, read-modify-write operations should be avoided.</p>
0	10, [27:26]	r	<p>Reserved Read as 0; should be written with 0.</p>

Note: See also [Table 3-8](#) for further functional behavior of FDR bit fields and module operation.

Note: The Fractional Divider Registers are Endinit-protected.

Table 3-8 Fractional Divider Function Table

Mode	SC	DM	Reset Ext. Divider Signal	Result	f_{OUT}	Operation of Fractional Divider
Normal Mode	–	00 _B	1	unchanged	inactive	switched off
		01 _B	0	continuously updated ¹⁾	active	normal divider mode
		10 _B				fractional divider mode
		11 _B		unchanged	inactive	switched off
Suspend Mode	00 _B	00 _B	1	unchanged	inactive	switched off
		01 _B	0	continuously updated ¹⁾	active	normal divider mode
		10 _B				fractional divider mode
		11 _B		unchanged	inactive	switched off
	01 _B	00 _B	1	unchanged		switched off
		01 _B	0	loaded with 3FF _H		halted
		10 _B				
		11 _B		unchanged	inactive	switched off
	10 _B	00 _B	1	loaded with 3FF _H	inactive	switched off
		01 _B	0			halted
		10 _B				
		11 _B				switched off
	11 _B	–	1	loaded with 3FF _H	inactive	switched off

1) Each write operation to FDR with DM = 01_B or 10_B sets RESULT to 3FF_H.

Implementation

FDR registers are implemented for several modules of the TC1796. The name of these FDR registers is always preceded by the module name (e.g. SSC0_FDR is the FDR register for the SSC0 module). **Table 3-9** defines which module is equipped with a FDR register.

In the implementation parts of the modules using a fractional divider (see **Table 3-9**), the signal f_{OUT} is described as a clock signal and not as clock enable signal.

3.3.4 Module Clock Register Implementations

Table 3-9 shows which of the CLC register bits/bit fields are implemented for each peripheral module in the TC1796 and which modules are equipped with a fractional divider.

Table 3-9 Clock Generation Implementation of the TC1796 Peripheral Modules

Module		DISR	DISS	SPEN	EDIS	SBWE	FSOE	RMC	Fract.	
Name	State after Reset	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5		Divider ¹⁾	
ADC0 ADC1	off	✓	✓	✓	✓	✓	✓	–	✓	
FADC	off	✓	✓	✓	✓	✓	✓	–	✓	
ASC0 and ASC1	off	✓	✓	✓	✓	✓	✓	8-bit	–	
SSC0	off	✓	✓	✓	✓	✓	✓	–	✓	
SSC1	off	✓	✓	✓	✓	✓	✓	–	✓	
MultiCAN	off	✓	✓	✓	✓	✓	✓	–	✓	
DMA	on	✓	✓	✓	✓	✓	✓	–	–	
EBU	on	✓	✓	–	–	–	–	–	–	
GPTA0 GPTA1 LTCA2	off	✓	✓	✓	✓	✓	✓	–	✓	
MLI0	off	not implemented, MLI is connected to DMA_CLC								✓
MLI1	off	not implemented, MLI is connected to DMA_CLC								✓
MSC0	off	✓	✓	✓	✓	✓	✓	–	✓	
MSC1	off	✓	✓	✓	✓	✓	✓	–	✓	
PCP2	off	different bit definitions ²⁾								
PLL	on	different bit definitions							–	–
STM	on	✓	✓	✓	✓	✓	✓	3-bit	–	

1) Further info on FDR implementations see [Table 3-10](#).

2) Automatic clock switch-off capability if PCP is idle.

Note: The ports of the TC1796, SCU, and WDT do not provide CLC registers.

3.3.5 Fractional Divider Register Implementations

Table 3-10 shows the implementations specific differences of the fractional divider functionalities.

Table 3-10 FDR Register Implementations

FDR Register	Suspend Mode Acknowledge Operation¹⁾	ENHW²⁾	Reset Ext. Divider³⁾
CAN_FDR	Acknowledge depends on module state	–	–
ADC0_FDR	Acknowledge depends on module states	–	resets analog parts of ADCs
FADC_FDR	Acknowledge depends on module state	–	–
GPTA0_FDR	Always immediately acknowledged; independently from module states	from MultiCAN	–
MLI0_FDR	Acknowledge depends on module state	–	–
MLI1_FDR	Acknowledge depends on module state	–	–
MSC0_FDR	Acknowledge depends on module state	from MultiCAN	–
MSC1_FDR	Acknowledge depends on module state	from MultiCAN	–
SSC0_FDR	Always immediately acknowledged; independently from module state	from MultiCAN	–
SSC1_FDR	Always immediately acknowledged; independently from module state	from MultiCAN	–
SCU_SCLKFDR	Always immediately acknowledged; independently from module state	–	external divider-by-2

- 1) This column shows whether a suspend acknowledge from a FDR controlled module depends on the module's state or not. If a suspend acknowledge depends from the module state, typically module operations such as serial transmissions are terminated before a suspend request is acknowledged back to the fractional divider. Note that bit FDR.SM must be cleared (granted suspend mode selected) when using the suspend mode acknowledge/grant functionality. If immediate suspend mode is selected (FDR.SM = 1), suspend mode is entered at once (if FDR.SC not equal 00_B) independently from the suspend acknowledge answer from the module.
- 2) This column shows whether the External Clock Enable input of a fractional divider is controlled by on-chip hardware (source module see comment) or not ("–").
- 3) This column shows whether the Reset External Divider output of the fractional divider is used (purpose see comment) or not ("–").

3.4 System Clock Output Control

The System Clock SYSCLK (alternate function of GPIO port line P1.12) is generated by a fractional divider block with a subsequent divide-by-2 stage (see [Figure 3-10](#)). This allows the SYSCLK frequency to be highly independent from the system frequency. Furthermore, the SYSCLK signal has duty cycle of 50% (with a small period jitter). After reset, SYSCLK is at low level when the Reset External Divider signal is at 1 (fractional divider stopped).

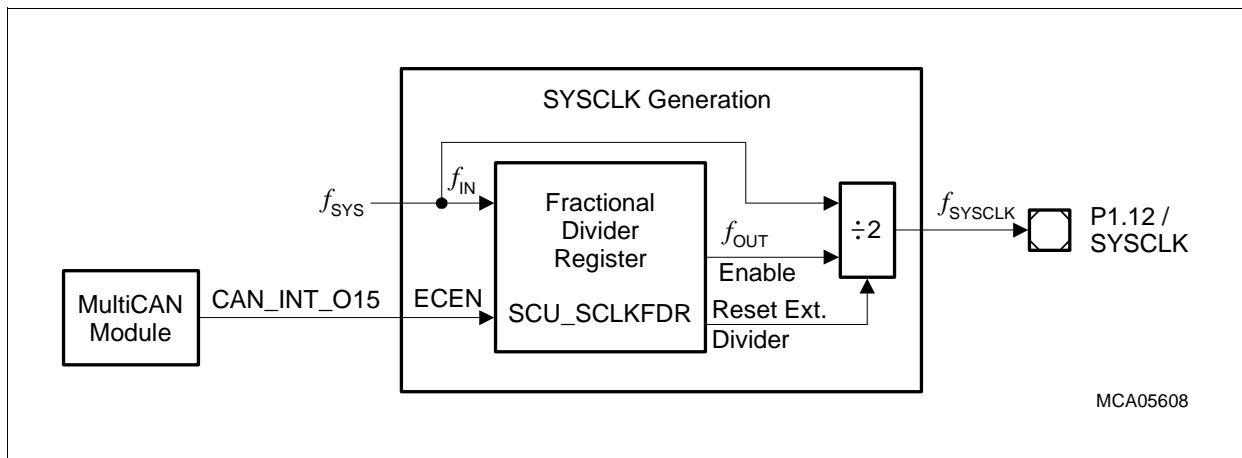


Figure 3-10 SYSCLK Generation

In normal divider mode, f_{SYSCLK} has a duty cycle of approximately 50%. The output frequency of f_{SYSCLK} in normal divider mode is defined according to the following formula:

$$f_{\text{SYSCLK}} = f_{\text{OUT}}/2 = \frac{f_{\text{SYS}}}{2 \times n} \quad , \text{ with } n = 1024 - \text{STEP} \quad (3.7)$$

In fractional divider mode, f_{OUT} is derived from the input clock f_{IN} by division of a fraction of $n/1024$ for any value of n from 0 to 1023. In general, the fractional divider mode makes it possible to program the average output clock frequency with a higher accuracy than in normal divider mode. Note that in fractional divider mode, the clock f_{OUT} can have a maximum period jitter of one f_{IN} clock period (f_{SYSCLK} has a maximum period jitter of one half f_{IN} clock period).

The output frequency of f_{SYSCLK} in fractional divider mode is defined according to the following formula:

$$f_{\text{SYSCLK}} = f_{\text{OUT}}/2 = f_{\text{SYS}} \times \frac{n}{2 \times 1024} \quad , \text{ with } n = 0-1023 \quad (3.8)$$

3.4.1 System Clock Fractional Divider Register

SCU_SCLKFDR

SCU System Clock Fractional Divider Register

(F000000C_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIS CLK	EN HW	SUS REQ	SUS ACK	0		RESULT									
rwh	rw	rh	rh	r		rh									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DM		SC		SM	0	STEP									
rw		rw		rw	r	rw									

Field	Bits	Type	Description
STEP	[9:0]	rw	Step Value Reload or addition value for RESULT.
SM	11	rw	Suspend Mode SM selects granted or immediate suspend mode.
DM	[15:14]	rw	Divider Mode DM selects normal or fractional divider mode.
SC	[13:12]	rw	Suspend Control SC determines the behavior of the fractional divider in suspend mode.
RESULT	[25:16]	rh	Result Value Bit field for the addition result.
SUSACK	28	rh	Suspend Mode Acknowledge Indicates state of SPNDACK signal.
SUSREQ	29	rh	Suspend Mode Request Indicates state of SPND signal.
ENHW	30	rw	Enable Hardware Clock Control No function for SYSCLK generation. Should be written with 0, will read back the last value written
DISCLK	31	rwh	Disable Clock Makes it possible to disable the clock generation independently of the setting of the other functions

Field	Bits	Type	Description
0	10, [27:26]	r	Reserved Read as 0; should be written with 0.

Note: This is only a short summary of the fractional divider behavior. The details on the fractional divider register functionality are described on [Page 3-29](#).

4 Reset and Boot Operation

This chapter describes the conditions under which the TC1796 will be reset, the reset and boot operations, and the available boot options.

4.1 Reset and Boot Overview

When the TC1796 device is first powered up, several boot parameters such as the start location of the code have to be defined to enable proper start operation of the device. To accommodate this, the device has a separate Power-On Reset ($\overline{\text{PORST}}$) pin and a number of configuration pins which are sampled during the power-on reset sequence or the hardware reset. At the end of this sequence, the sampled values are latched, and can not be modified until the next power-on reset. This guarantees stable conditions during the normal operation of the device.

To reset the device while it is operating, two options exist. For reset causes coming from the external world, a reset input pin, $\overline{\text{HDRST}}$, is provided. If software detects conditions that require to reset the device, it can perform a software reset through writing to a special register, the Reset Request (RST_REQ) register.

The Watchdog Timer (WDT) module is also capable of resetting the device if it detects a malfunction in the system. If the WDT is not serviced correctly and/or not serviced in time, it first generates an NMI request to the CPU (this allows the CPU to gather debug information), and then resets the device after a predefined time-out period.

After a reset has been executed, the Reset Status (RST_SR) register provides information on the type of the last reset and the selected boot configuration.

The external reset pin, $\overline{\text{HDRST}}$, has a double-function. It serves as a reset input from the external world to reset the device, and it serves as a reset output to the external world to indicate that the device has executed a reset. For this purpose, pin $\overline{\text{HDRST}}$ is implemented as a bi-directional open-drain pin with an internal weak pull-up device.

Please note, that any reset pulse generated on $\overline{\text{HDRST}}$ will be prolonged to reach a minimum pulse width. This ensures proper system reset, even it is caused by a very short pulse on $\overline{\text{HDRST}}$ coming from the system.

The boot configuration information required by the device to perform the desired start operation after a power-up reset includes the start location for the code execution, and the activation of special modes. This information is supplied to the chip via a number of dedicated input pins that are sampled and latched with the hardware reset $\overline{\text{HDRST}}$ or the power-on reset $\overline{\text{PORST}}$. However, the software reset provides the special option to alter these parameters to allow a different start configuration after the software reset has finished.

Attention: To ensure safe operation, the power-on reset operation must be completed prior to utilisation of the chip. With a completed power-on reset operation, a sequence of initialization of the chip is executed

Reset and Boot Operation

which includes the initialization of the ICACHE and TagRAM memories in the PMI. In the emulation device, the EEC is initialized upon the completion of a power-on reset. Therefore, activation of the HDRST signal earlier than the maximum Power-on Reset Boot Time must be prevented. Please refer to the maximum Power-on Reset Boot Time defined in the Power, Pad and Reset Timing section of the TC1796 Data Sheet.

4.1.1 Reset Status and Control Registers

The Reset Status Register RST_SR indicates the cause of a reset and the selected boot configuration. The Reset Request Register RST_REQ is used to cause a software reset.

4.1.1.1 Reset Status Register

After a reset, the reset status register RST_SR indicates the type of reset which has occurred and which parts of the TC1796 were affected by the last reset operation. RST_SR also holds the state of the boot configuration pins HWCFG[3:0] (Port 10) that have been sampled with the $\overline{\text{HDRST}}$ inactive (low-to-high) transition. Register RST_SR is a read-only register.

RST_SR

Reset Status Register

(F000014_H)

Reset Values: see [Table 4-1](#)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	WDT RST	SFT RST	HD RST	PWO RST	0			TMP LS	HW BRK IN	0	HWCFG				
r	rh	rh	rh	rh	r			rh	rh	r	rh				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												RS EXT	0	RS STM	
r												rh	r	rh	

Field	Bits	Type	Description
RSSTM	0	rh	System Timer Reset Status 0 _B The system timer was not reset. 1 _B The system timer was reset.
RSEXT	2	rh	HDRST Input State during Last Reset 0 _B $\overline{\text{HDRST}}$ was not activated. 1 _B $\overline{\text{HDRST}}$ was activated.
HWCFG	[19:16]	rh	Boot Configuration Selection Status This bit field indicates the status of the configuration pins HWCFG[3:0] at Port 10 that has been latched at the rising edge of $\overline{\text{HDRST}}$. HWCFG[3:0] is assigned to P10.[3:0].

Reset and Boot Operation

Field	Bits	Type	Description
HWBRKIN	21	rh	Latched State of BRKIN Input This bit indicates the logical state of the $\overline{\text{BRKIN}}$ input that has been latched at the end of a power-on reset.
TMPLS	22	rh	Latched State of TESTMODE Input This bit indicates the logical state of the $\overline{\text{TESTMODE}}$ input pin that has been latched at the end of a power-on reset.
PWORST	27	rh	Power-On Reset Status Flag 0 _B The last reset was not a power-on reset. 1 _B The last reset was a power-on reset.
HDRST	28	rh	Hardware Reset Status Flag 0 _B The last reset was not a hardware reset. 1 _B The last reset was a hardware reset.
SFTRST	29	rh	Software Reset Status Flag 0 _B The last reset was not a software reset. 1 _B The last reset was a software reset.
WDTRST	30	rh	Watchdog Reset Status Flag 0 _B The last reset was not a watchdog reset. 1 _B The last reset was a watchdog reset.
0	1, [15:3], 20, [26:23], 31	r	Reserved Read as 0.

Table 4-1 defines the reset values of the RST_SR register depending on the reset source.

Table 4-1 Reset Values of Register RST_SR

Reset Source	Reset Values
Power-on Reset	0000 1000 0XX0 XXXX 0000 0000 0000 0101 _B
Hardware Reset	0001 0000 0XX0 XXXX 0000 0000 0000 0000 _B
Software Reset	0010 0000 0XX0 XXXX 0000 0000 0000 0X0X _B
Watchdog Timer Reset	0100 0000 0XX0 XXXX 0000 0000 0000 0101 _B

4.1.1.2 Reset Request Register

The reset request register RST_REQ is used to generate a software reset. Unlike the other reset causes, the software reset can exclude functions (System Timer reset and hardware reset (HDRST) generation) from the reset. Additionally, the boot configuration information can be set by software.

RST_REQ is Endinit-protected, meaning that bit WDT_CON0.ENDINIT must be set to 0 first through the password-protected access scheme provided for the watchdog timer register WDT_CON0. Once access is gained through the Endinit-protection scheme, RST_REQ can be written, causing the requested software reset.

RST_REQ

Reset Request Register

(F000010_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0							SW BO OT	0	SW BRK IN	0	SWCFG				
r							rw	r	rw	r	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												RR EXT	0	RR STM	
r												rw	r	rw	

Field	Bits	Type	Description
RRSTM	0	rw	Reset Request for the System Timer 0 _B Do not reset the System Timer 1 _B Reset the System Timer
RREXT	2	rw	Reset Request for External Devices 0 _B Do not activate hardware reset output $\overline{\text{HDRST}}$ 1 _B Activate hardware reset output $\overline{\text{HDRST}}$
SWCFG	[19:16]	rw	Software Boot Configuration A software boot configuration different from the external applied hardware configuration can be specified with these bits. The encoding of this bit field is equal to the RST_SR.HWCFG line encoding.
SWBRKIN	21	rw	Software Break Signal Boot Value This bit determines the desired value for the $\overline{\text{BRKIN}}$ input signal to be used for software boot.

Reset and Boot Operation

Field	Bits	Type	Description
SWBOOT	24	rw	Software Boot Configuration Selection 0 _B The previously latched hardware configuration stored in RST_SR.HWCFG is used as boot selection. 1 _B The software configuration as programmed in bit field SWCFG is used as boot selection.
0	1, 20, [15:3], [23:22], [31:25]	r	Reserved Read as 0; should be written with 0.

Note: Please refer to the [Table 4-3](#) in this chapter for detailed value configuration for the SWCFG bit field as well as for SWBRKIN and SWBOOT bits.

4.2 Reset Operations

4.2.1 Power-On Reset

The $\overline{\text{PORST}}$ pin performs a power-on reset, also called cold reset. Driving the $\overline{\text{PORST}}$ pin low causes an asynchronous reset of the entire device. The device then enters its power-on reset sequence.

The PLL has its own power-on reset circuitry and is not affected by any other reset condition besides a low signal transition on the $\overline{\text{PORST}}$ pin. With an active power-on reset, the PLL is disconnected from the oscillator and will start running at its base frequency.

Simultaneously with $\overline{\text{PORST}}$ low, the reset circuitry drives the $\overline{\text{HDRST}}$ pin low and waits for the following two conditions to occur:

1. The system clock is active
2. Pin $\overline{\text{PORST}}$ is put to inactive level (driven high)

When both of these conditions are met and $\overline{\text{HDRST}}$ is not further pulled low externally, the power-on reset sequence is terminated synchronously with the next system clock transition.

The rising edge of $\overline{\text{PORST}}$ causes the state of some of the configuration pins for the PLL and the boot options to be latched into the appropriate registers. Others are latched with the rising edge of $\overline{\text{HDRST}}$. Fields in the Reset Status Register (RST_SR) are set to inform the user about this complete reset of the device. The power-on reset indication flag is set, while all other reset cause indication flags are cleared. Fields in this register which are set include the power-on reset indication flag (PWORST), as well as the reset status flags for the System Timer (RSSTM) and the reset output pin state (RSEXT).

The time $\overline{\text{PORST}}$ has to be active after the supply voltage is stable depends mainly on the settling time of the power supply lines. It does not directly depend on the oscillator start up time because a system clock signal will be provided already by the PLL in PLL Base Mode as soon as the power supply lines are stable. This system clock signal is a slow clock signal based on the VCO base frequency. More details on the PLL Base Mode are discussed on [Page 3-11](#).

$\overline{\text{PORST}}$ is equipped with a noise-suppression filter that suppresses glitches below 10 ns pulse width. $\overline{\text{PORST}}$ pulses with a width above 100 ns are safely recognized as a valid signal. The noise-suppression filter is switched-off when pin $\text{BYPASS} = 1$.

4.2.2 External Hardware Reset

The external hardware reset pin $\overline{\text{HDRST}}$ serves as an external reset input as well as a reset output. It is an active-low, bi-directional open-drain pin with an internal weak pull-up. An active-low signal at this pin causes the chip to enter its hardware reset sequence.

Reset and Boot Operation

The $\overline{\text{HDRST}}$ (output) pin is held low for $1024 f_{\text{SYS}}$ clock cycles by the reset circuitry until its internal reset sequence is terminated.

When the sequence is terminated, the reset circuitry then releases $\overline{\text{HDRST}}$ (that is, it does not actively drive $\overline{\text{HDRST}}$ anymore, so that the weak pull-up can try to drive the pin high). It then begins monitoring the level of the pin. If $\overline{\text{HDRST}}$ is still low (indicating that it is still being driven low externally), the reset circuitry holds the chip in hardware reset until a high level is detected. The hardware reset sequence is then terminated and flag RST_SR.HDRST is set.

The PLL is not affected by an external hardware reset and continues to operate.

In order to safely recognize a valid hardware reset, $\overline{\text{HDRST}}$ must be active for at least four f_{CPU} clock cycles.

$\overline{\text{HDRST}}$ is equipped with a noise-suppression filter which suppresses glitches below 10 ns pulse width. $\overline{\text{HDRST}}$ pulses with a width above 100 ns are safely recognized as a valid signal. The noise-suppression filter is switched-off when pin BYPASS = 1.

4.2.3 Software Reset

A software reset is invoked by writing the appropriate bits in the reset request register RST_REQ. Unlike the other reset types, the software reset can include/exclude optionally the system timer reset and the external reset output $\overline{\text{HDRST}}$ generation from becoming active. To exclude one of these two system functions from software reset, the corresponding bits in RST_REQ (RRSTM or RREXT) must be set to 0. Additionally, a software reset can be executed with a programmable software boot configuration value (bit field RST_REQ.SWCFG) instead of the last hardware boot configuration value (as latched in bit field RST_SR.HWCFG).

To perform a software reset, the reset request register RST_REQ must be written. However, RST_REQ is Endinit-protected to avoid triggering of an unintentional software reset. Bit WDT_CON0.ENDINIT must be cleared via the password-protected access scheme. When this is done, a write access to RST_REQ can then be performed.

After the write access to RST_REQ, the entering into the software reset takes some time in which instructions can be still executed. Therefore, it is recommended not to execute important instructions anymore after the instruction that writes to RST_REQ (e.g. entering a endless loop).

4.2.4 Watchdog Timer Reset

A Watchdog Timer overflow or access error occurs only in response to severe and/or unknown malfunctions of the TC1796, either caused by software or hardware errors. Therefore, a Watchdog Timer reset occurs when an overflow of the Watchdog Timer takes place.

Before the Watchdog Timer generates its reset, it first indicates a Non-Maskable Interrupt (NMI) at the beginning of a prewarning phase and enters a time-out mode. The

Reset and Boot Operation

NMI invokes the trap service routine and can save critical states of the microcontroller for later examination to determine the cause of the Watchdog Timer failure. However, it is not possible to stop or terminate the Watchdog Timer's time-out mode or prevent the pending watchdog reset.

However, software can prevent the Watchdog Timer by issuing a software reset on its own. Since the cause of the system failure is presumably unknown at that time, and it is presumably uncertain which functions of the TC1796 are operating properly, it is recommended that the software reset is configured to reset all system functions including the System Timer and external reset output $\overline{\text{HDRST}}$, and to use the hardware boot configuration bit field as boot configuration source indicator.

If the NMI trap handler does not perform a software reset, or if the system is so compromised that the trap handler cannot be executed, the Watchdog Timer will cause a Watchdog Timer reset to occur at the end of its time-out mode period. The actions performed on a Watchdog Timer reset sequence are the same as are performed for an external hardware reset. At the end of the Watchdog Timer reset sequence, bits WDTRST , RSSTM , and RSEXT are set in register RST_SR . All other reset flags are cleared.

Watchdog Timer Reset Lock

When the system emerges from any reset condition, the Watchdog Timer becomes active, and, unless prevented by initialization software, will eventually time out. Ordinarily, initialization software will configure the Watchdog Timer and commence servicing it on a regular basis to indicate that it is functioning properly. Should the system be malfunctioning such that initialization and service are not performed in a timely fashion, the Watchdog Timer will time out, causing a Watchdog Timer reset.

If the TC1796 system is so corrupted that it is chronically unable to service the Watchdog Timer, the danger could arise that the system would be continuously reset every time the Watchdog Timer times out. This could lead to serious system instability, and to the loss of information about the original cause of the failure.

However, the reset circuitry of the TC1796 is designed to detect this condition. If a Watchdog Timer error occurs while one or both of the Watchdog Timer error flags (WDT_SR.WDTAE and WDT_SR.WDTOE) are already set to 1, the reset circuitry locks the TC1796 permanently in reset (Reset Lock, $\overline{\text{HDRST}}$ permanently active) until the next power-on reset occurs by activation of the $\overline{\text{PORST}}$ pin.

This situation could arise, for example, if the connection to external code memory is lost or memory becomes corrupt, such that no valid code can be executed, including the initialization code. In this case, the initial time-out period of the Watchdog Timer cannot be properly terminated by software. The Watchdog Timer error flag WDTOE will be set when the Watchdog Timer overflows, and a Watchdog Timer reset will be triggered (after the watchdog reset pre-warning phase). The error flag WDTOE is not cleared by the Watchdog Timer reset that subsequently occurs. After finishing the Watchdog Timer

Reset and Boot Operation

reset sequence, the TC1796 will again attempt to execute the initialization code. If still the code cannot be executed because of connection problems, the WDTOE bit will not have been cleared by software. Again, the Watchdog Timer will time out and generate a Watchdog Timer reset. However, this time the reset circuitry detects that WDTOE is still set while a Watchdog Timer error has occurred, indicating danger of cyclic resets. The reset circuitry then puts the TC1796 in Reset Lock. This state can only be deactivated again through a power-on reset.

4.2.5 Debug System Reset

The debug system is not automatically reset by the regular resets except for the power-on reset. It is not effected by a software resets and by a Watchdog Timer reset. A hardware reset becomes only effective if at the same time the OCDS reset is active as well.

Note: TriCore, PCP, DMA, RBCU and SBCU have integrated debug modules that are part of the debug system.

4.2.6 Module Reset Behavior

Table 4-2 lists how the various functions of the TC1796 are affected through a reset depending on the reset type. A black square means that this function/unit is reset to its default state.

Reset and Boot Operation

Table 4-2 Effect of Reset on Device Functions

Module / Function		Watchdog Reset	Software Reset	Hardware Reset	Power-On Reset
Boot Configuration taken from		Bit field HWCFG	Bit fields HWCFG or SWCFG	Bit field HWCFG	Bit field HWCFG
CPU		■	■	■	■
SCU		■ ¹⁾	■ ¹⁾	■ ¹⁾	■ ¹⁾
BCUs, Bus System		■	■	■	■
Peripherals, PCP (except System Timer)		■	■	■	■
System Timer		■	Optional ²⁾	Not affected	■
On-chip Static RAMs	DMI, PMI	Not affected, reliable	Not affected, reliable	Not affected, reliable	Affected, unreliable
	DMU, PMU	Not affected, unreliable	Not affected, unreliable	Not affected, unreliable	Affected, unreliable
	PCP Memory	Not affected ³⁾	Not affected ³⁾	Not affected ³⁾	Affected, unreliable
On-Chip Caches ⁴⁾		■	■	■	■
Flash		■	■	■	■
Oscillator, PLL		Not affected	Not affected	Not affected	■
EBU		■	■	■	■
EBU Pins		Depending on Reset Configuration			
Port Pins		Tri-stated, weak pull-up active ⁵⁾			
NMI Pin		Not affected	Disabled	Disabled	Disabled
Reset Out Pin HDRST		■	Optional	■	■
OCDS L1 Debug System		Only if JTAG reset is also active ⁶⁾			■

- 1) For two of the SCU registers (PLL_CLC, RST_SR), the reset value depends on the reset source.
- 2) "Affected" or "not affected" depends on bit RST_REQ.RRSTM.
- 3) If only the PCP accesses its memory, the contents are reliable. If an access from the FPI Bus is performed while the reset is activated, the content is unreliable.
- 4) The actual data contents of the cache are not affected through a reset; however, the cache tag information is cleared, resulting in an "empty" cache.
- 5) While $\overline{\text{PORST}}$ is active it is guaranteed that the pins are in tri-state mode even if the core supply is not applied. Therefore the external power supply should activate $\overline{\text{PORST}}$ if any power failure is detected.
- 6) Default JTAG reset state (open JTAG pins) is active. A connected debugger tool controls the JTAG reset.

4.2.7 Booting Scheme

When the TC1796 is reset, it needs an indication how it has to start after the reset sequence is finished. The TC1796 internal state is usually cleared through a reset, especially in the case of a power-up reset. Thus, boot configuration information needs to be applied by the external hardware through input pins. The boot configuration information is required for:

- The start location of the code execution, and
- Activation of special modes and conditions

For the start of code execution and activation of special mode, the TC1796 implements two basic booting schemes: A hardware booting scheme which is invoked through external pins; and a software booting scheme, where software can determine the boot options, overriding the externally-applied options.

Boot Options

Inputs HWCFG[3:0] (Port P10.[3:0]) and input $\overline{\text{BRKIN}}$ determine the boot mode and boot location. [Table 4-3](#) shows the boot options/selections which are available in the TC1796. The source for $\overline{\text{BRKIN}}$ and HWCFG[3:0] can be either the corresponding bits HWBRKIN and HWCFG[3:0] from register RST_SR (sampled from configuration pins) or the software configuration bits SWBRKIN and SWCFG[3:0] from register RST_REQ.

The target boot address (program counter start address DFFF FFFC_H) is located at the end of the Boot ROM space. The Boot ROM code then decides how to proceed for the selected boot modes. When Boot ROM code is left, jump instructions to different dedicated addresses are executed. Note that pin $\overline{\text{TESTMODE}}$ must be at high level for all normal boot selection shown in [Table 4-3](#).

Table 4-3 TC1796 Boot Selections

BRKIN	HWCFG [3:0]	Type of Boot	Boot ROM Exit Jump Address
Normal Boot Options			
1	0000 _B	Enter bootstrap loader mode 1: Serial ASC0 boot via ASC0 pins	D400 0000 _H
	0001 _B	Enter bootstrap loader mode 2: Serial CAN boot via CAN pins	
	0010 _B	Start from internal PFLASH	A000 0000 _H
	0011 _B	Alternate Boot Mode (ABM): start from internal PFLASH after CRC check is correctly executed; enter a serial bootstrap loader mode ¹⁾ if CRC check fails;	as defined in ABM header or D400 0000 _H

Table 4-3 TC1796 Boot Selections (cont'd)

BRKIN	HWCFG [3:0]	Type of Boot	Boot ROM Exit Jump Address
1	0100 _B	Start from external memory with EBU as master, using $\overline{CS0}$; automatic EBU configuration ²⁾ ;	A100 0000 _H
	0101 _B	Alternate Boot Mode (ABM): start from external memory with CRC check and EBU as master, using $\overline{CS0}$; enter a serial bootstrap loader mode ¹⁾ if CRC checks fails; automatic EBU configuration ²⁾ ;	as defined in ABM header or D400 0000 _H
	0110 _B	Start from external memory with EBU as participant, using $\overline{CS0}$; automatic EBU configuration ²⁾ ;	A100 0000 _H
	0111 _B	Alternate Boot Mode (ABM): start from external memory with CRC check and EBU as participant, using $\overline{CS0}$; enter a serial bootstrap loader mode ¹⁾ if CRC checks fails; automatic EBU configuration ²⁾ ;	as defined in ABM header or D400 0000 _H
	1000 _B	Start from emulation memory if emulation device TC1796ED is available; in case of TC1796 execute stop loop;	if TC1796ED: AFF0 0000 _H
	1111 _B	Enter bootstrap loader mode 3: Serial ASC0 boot via CAN pins	D400 0000 _H
	others	Reserved; execute stop loop;	–

Debug Boot Options

0	0000 _B	Tri-state chip	–
	1000 _B	Go to external emulator space with EBU as master, using $\overline{CSEMU/CSCOMB}$	DE00 0000 _{H)}
	others	Reserved; execute stop loop;	–

1) The type of bootstrap loader mode is selected by SCU_SCLIR.SWOPT[2:0] bit field.

2) The EBU fetches the boot configuration from address offset 4 using $\overline{CS0}$.

4.2.7.1 Normal Boot Options

The normal boot options are invoked when $\overline{\text{BRKIN}}$ is inactive (at high level). The TC1796 has three options for booting into normal operation:

- Starting code execution from external memory
- Starting code execution from internal PFLASH
- Starting code execution from internal SPRAM after a bootstrap loader program has downloaded a program via a serial interface (ASC0 or CAN).

If a reserved boot selection is detected for HWCFG[3:0] with $\overline{\text{BRKIN}} = 1$, the TC1796 stops further Boot ROM control operations and enters into an endless loop by executing a jump instruction to itself (can be aborted by a WDT reset).

Serial Boot via Bootstrap Loader

The bootstrap loader (BSL) is a software part of the Boot ROM which provides a mechanism to load a program code into the scratchpad RAM (SPRAM) of the PMI. The program code to be loaded is received serially either via the ASC0 or CAN interfaces. After loading of the code, the bootstrap loader program jumps directly to address D400 0000_H (start address of the PMI scratchpad RAM) and begins executing the program code that has been loaded. Further details about the BSL are described in [Section 4.4](#).

External Boot

In external boot modes, code execution is started in external memory via the EBU at a fixed address (A100 0000_H). In Alternate Boot Modes, the code start address is defined by one of the two ABM headers (see [Page 4-30](#)).

In order to access external memory after reset, the EBU must have information about the type and access mechanism of the external memory. For this purpose, a special external bus access is executed by the EBU in order to retrieve information about the external code memory. This access is performed to address offset 0000 0004_H of the memory connected to $\overline{\text{CS0}}$, using access parameters such that regardless of the type and characteristics of the external memory, configuration information can be read from the memory into the EBU. By examining this information, the EBU determines the exact requirements for accesses to the external memory. It then configures the control registers accordingly, and performs the first instruction fetch from address A100 0000_H. All further accesses to external memory during external boot will activate $\overline{\text{CS0}}$ (until the EBU becomes reprogrammed for example by a user program).

Internal PFLASH Boot

After execution of the start-up code, two PFLASH boot options exist:

- PFLASH boot option 1 directly jumps to the internal flash.
- PFLASH boot option 2 (Alternate Boot Mode) first checks the flash via checksum calculation, whether one of two pre-defined entry locations contain valid code and jumps to a start address as defined by the ABM header.

4.2.7.2 Debug Boot Options

The debug boot options are invoked when $\overline{\text{BRKIN}}$ is active (at low level). The TC1796 has three options for booting into debug operation:

- Tri-state chip
- Starting code execution from external emulator memory
- Execution of an endless (stop) loop

Tri-state Chip

If $\text{HWCFG}[3:0] = 0000_{\text{B}}$, all pins of the TC1796 are put into tri-state mode. In this mode, all pins are deactivated, including the oscillator, and internal circuitry is held in a low-power mode. This mode allows e.g. board level test equipment or emulator probes to actively drive lines which are connected to TC1796 pins.

Emulator Memory Start

If $\text{HWCFG}[3:0] = 1000_{\text{B}}$, the TC1796 starts execution out of the external memory space using the EBU and the chip select line CSCOMB .

Execute Endless Loop

If a reserved boot selection is detected for $\text{HWCFG}[3:0]$ with $\overline{\text{BRKIN}} = 0$, the TC1796 stops further Boot ROM control operations and enters into an endless loop by executing a jump instruction to itself (can be aborted by a WDT reset).

4.3 Boot ROM

The internal 16 Kbyte Boot ROM (BROM) has two parts:

- 8 Kbyte reserved for boot code (Boot ROM)
- 8 Kbyte reserved for factory test routines (Test ROM).

Note: The expression “Boot ROM” in the text always refers to the 8 Kbyte boot code part of the BROM.

4.3.1 Addressing

As the original architectural position of the BROM is on the FPI Bus and the boot vector to the Boot ROM is hardwired in the CPU to this location. The internal BROM in the TC1796 is visible from the PMI side at three different locations, as can be seen in the memory map:

- In segment 8 (cached) starting at location 8FFF C000_H
- In segment 10 (non-cached) starting at location AFFF C000_H
- In segment 13 (FPI space) starting at location DFFF C000_H.

From DMI side, the internal ROM is not visible in segment 13. Accesses to these addresses will be routed to the FPI Bus and will cause a bus error.

The hardware-controlled start address after reset is address DFFF FFFC_H. At this location (within the BROM), a jump to start address AFFF F180_H is programmed to guarantee continuation of start program execution after reset.

Because the reset start address is fixed, the Boot ROM is mapped to the upper part of the internal BROM, at locations ..E000 - ..FFFF_H, and the TestROM is mapped to the lower part of internal BROM (..C000 - ..DFFF_H).

4.3.2 Program Structure

The different sections of the 8 Kbyte Boot ROM provide the following functionalities.

Startup Procedure

The startup procedure is the main control program in the Boot ROM which is always started after every reset operation. It initializes the chip, checks the hardware or software configuration selections, and branches to the corresponding boot routines. From a user point of view, the startup procedure lengthens the reset time of the TC1796.

The startup procedure is responsible for the correct initialization of the on-chip resources. For example, the startup procedure monitors the Flash ramp-up phase and waits until the Flash memories are ready to be used. The startup procedure further controls the Flash read protection, initializes SRAM redundancy settings, and copies the unique chip identifier from Flash into the LDRAM memory.

Reset and Boot Operation

During the startup procedure, the NMI interrupt, the watchdog timer, and the debug interfaces are disabled. The NMI interrupt has to be enabled by the user software (SCU_CON.NMIEN bit set) after proper setup of the interrupt vector table.

The reset configuration indicated in register RST_SR is evaluated by the startup procedure for selection of the configured operation and of program start memory.

Bootstrap Loader

The Bootstrap loader (BSL) is a software part of the Boot ROM which provides a mechanism to load a program code into the Scratchpad RAM (SPRAM) of the PMI. The program code to be loaded is received serially either via the ASC0 or CAN interfaces. After loading of the code, the bootstrap loader jumps directly to address D400 0000_H (start address of SPRAM) and begins executing the program code that has been loaded. Further details about the BSL are described in [Section 4.4](#).

Alternate Boot Modes

The alternate boot modes will only branch to an user program in PFLASH or external code memory if a CRC checksum test shows no error. If the CRC check fails, a bootstrap loader mode is entered instead of user program execution.

Tuning Protection

Tuning protection is supported to protect user software in external Flash memory from being manipulated and to safely detect changes of this user software.

Emulation Support

If the device currently used is an emulation device (TC1796ED), a boot capability from emulation memory is supported. For the standard TC1796 this boot option cannot be activated.

Test and Stress Routines

The 8 Kbyte TestROM is reserved for special routines, which are used for testing, stressing and qualification of the component. This boot option should never be activated by a user program.

Boot ROM Program Flow

[Figure 4-1](#) shows the basic Boot ROM program flow. This flow diagram has one input (reset address DFFF FFFC_H) and several outputs that depend on the selected boot option (as described in [Table 4-3](#)).

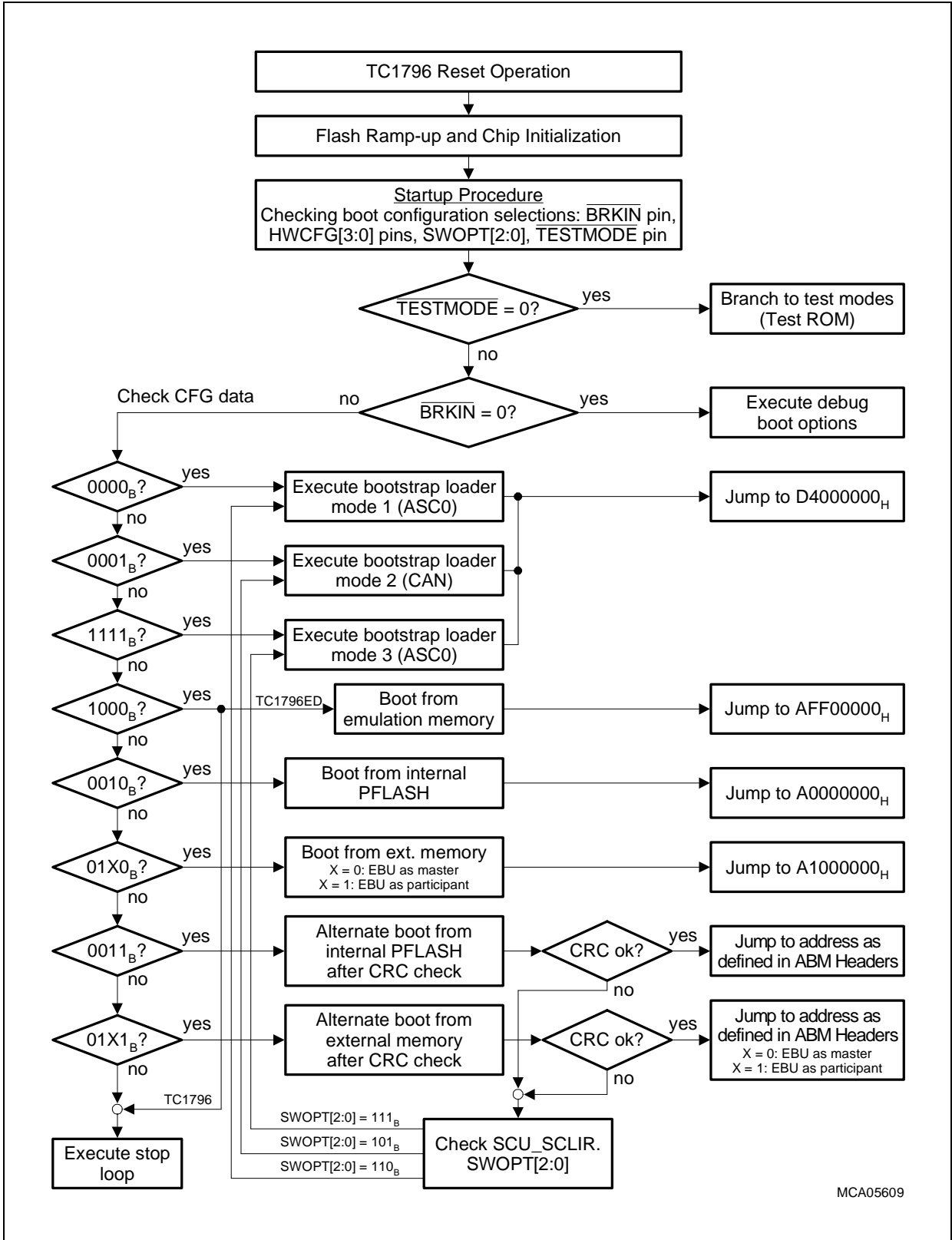


Figure 4-1 Boot ROM Flow Diagram

4.3.3 Initial State after Boot ROM Exit

The Boot ROM code is always executed after every reset operation. Depending on the program flow through the Boot ROM code and the Boot ROM exit path, several resources of the TC1796 on-chip hardware have been used and have been programmed. This means that the state of on-chip hardware resources that have been used by the Boot ROM code may differ from the device reset state as described by the register reset values. This section describes which initial state of the on-chip hardware resources is left after a specific Boot ROM exit.

Table 4-4 Hardware Status after Boot ROM Exit

Boot ROM Exit Path	State of On-Chip Hardware Resources
For all boot modes	CPU registers D[15:0] and A[15:0] are changed (note that its reset values are XXXX XXXX _H); Memory locations in LDRAM are used: D000 0000 _H - D000 000F _H : updated with unique Chip ID D000 0010 _H - D000 0107 _H : updated with Boot ROM data SCU_STAT[15:13] are set to 001 _B . FLASH_MARP.TRAPDIS and FLASH_MARD.TRAPDIS are cleared; FLASH_FCON.WSWLHIT is set to 110 _B
Branch to test modes	Only applicable for test purposes
Execute debug boot options	The following registers have been changed: CPR0_0L, CPM0, and TR0EVT
Exit of Bootstrap loader mode 1 (ASC0)	P5_IOCR0 has been changed (P5.1/TXD0A); ASC0 module is initialized and enabled; PLL_CLC has been changed
Exit of Bootstrap loader mode 2 (CAN)	P6_IOCR8 has been changed (P6.9/TXDCAN0); MultiCAN module is initialized and enabled; registers of CAN node 0, message object 0, and message object 1 have been used; PLL_CLC has been changed
Exit of Bootstrap loader mode 3 (ASC0)	P6_IOCR8 has been changed (P6.9/TXD0B); ASC0 module is initialized and enabled; PLL_CLC has been changed
Boot from emulation memory	EBU_CON has been changed
Boot from PFLASH	No further changes
Alternate boot from PFLASH	Memory checker module has been used
Boot from EBU	No further changes
Alternate boot from EBU	Memory checker module has been used
Execute Stop Loop	BTV has been changed

4.4 Bootstrap Loader (BSL)

The bootstrap loader (BSL) is a software part which is integrated in the TC1796 Boot ROM. The BSL provides a mechanism to load a program code via a serial interface (ASC or CAN) into the scratchpad RAM (SPRAM) of the PMI. After loading of the code, the BSL jumps directly to address D400 0000_H (start address of the PMI scratchpad RAM) and begins executing the program code that has been loaded. The BSL automatically calculates the baud rate of the serial data streams.

Table 4-5 shows the three BSL modes with its parameters.

Table 4-5 Bootstrap Loader Selections

Bootstrap Loader Mode	Selection ¹⁾		Associated I/O Lines	
	BRKIN	HWCFG [3:0]	Receive	Transmit
Bootstrap Loader Mode 1 (BSL1): ASC Boot via ASC0 Pins	1	0000 _B	P5.0 / RXD0A	P5.1 / TXD0A
Bootstrap Loader Mode 2 (BSL2): CAN Boot		0001 _B	P6.8 / RXDCAN0	P6.9 / TXDCAN0
Bootstrap Loader Mode 3 (BSL3): ASC Boot via CAN Pins		1111 _B	P6.8 / RXD0B	P6.9 / TXD0B

1) The bootstrap loader mode selections in alternate boot modes see **Table 4-3**.

With the low-to-high signal transition of the hardware reset signal $\overline{\text{HDRST}}$ or the power-on reset signal PORST , the input pins $\overline{\text{BRKIN}}$ and $\text{HWCFG}[3:0]$ of the TC1796 are latched. If one of the latched $\overline{\text{BRKIN}}/\text{HWCFG}[3:0]$ signal combination of **Table 4-5** is detected, the bootstrap loader is started and the selected bootstrap loader mode is entered.

The bootstrap loader can also be started by a software reset. For this purpose, bit RST_REQ.SWBRKIN and bit field RST_REQ.SWCFG must be loaded with the corresponding $\overline{\text{BRKIN}}/\text{HWCFG}[3:0]$ code, and bit RST_REQ.SWBOOT must be set (see also RST_REQ register description at **Page 4-5**).

When a boot option for a bootstrap loader mode is detected, the TC1796 jumps to address DFFF FFFC_H which is the last word address of the Boot ROM. During execution of the bootstrap loader, the Watchdog Timer interrupts and NMI interrupts are disabled.

4.4.1 Bootstrap Loader Mode 1 - ASC Boot via ASC0 Pins

The ASC boot mode of the bootstrap loader moves 128 bytes of program code/data into the SPRAM and starts executing the loaded code from address D400 0000_H (address of the first transmitted byte). Most probably the initially-loaded program code will load additional code or data. For that purpose, it can directly use the ASC0 interface that has already been initialized during the bootstrap loader execution. **Figure 4-2** shows the actions that take place in bootstrap loader mode 1.

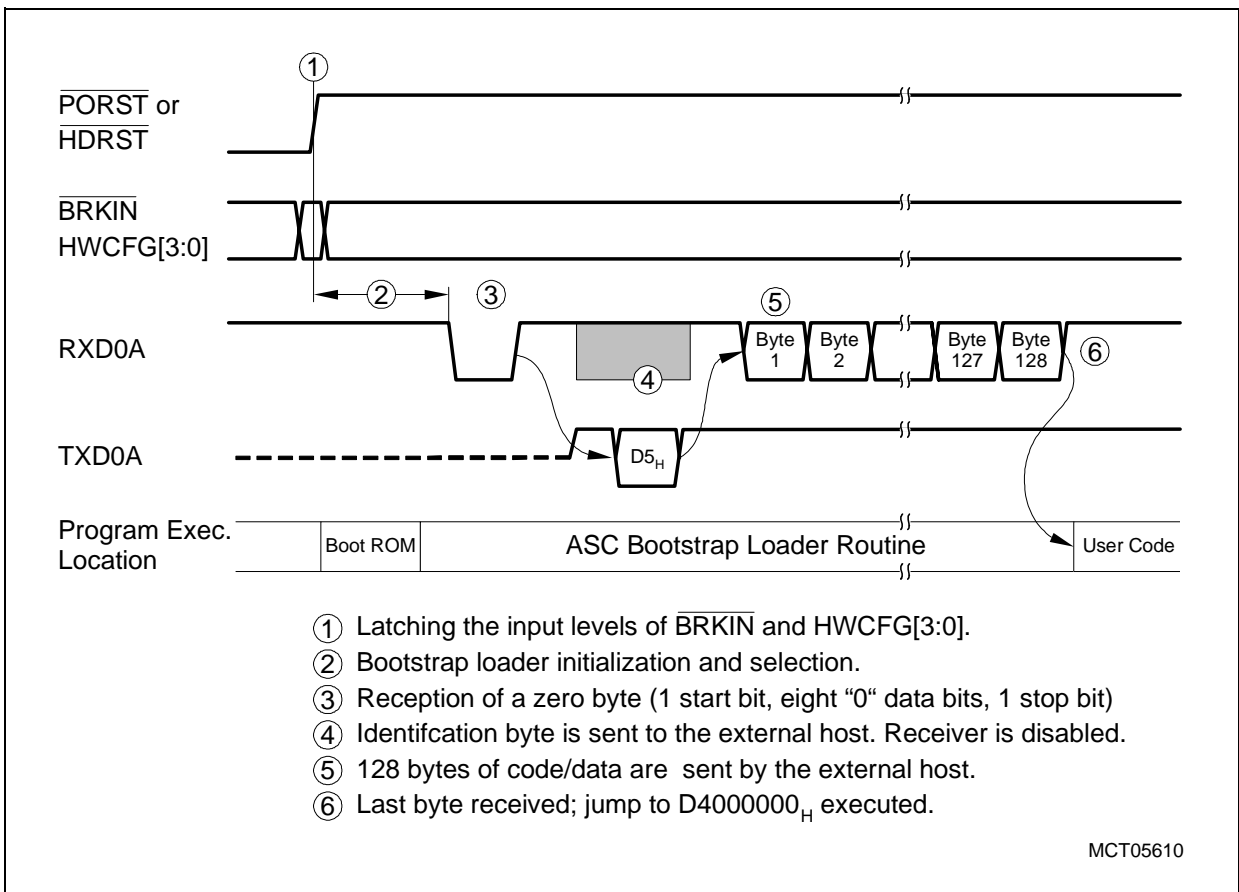


Figure 4-2 ASC0 Bootstrap Loader Sequence (Mode 1)

As the first task, the ASC0 determines the baud rate at which the external host is communicating. This task requires the external host to transmit an initialization byte to the TC1796 at the RXD0A input line of the ASC0. The initialization byte is built up by one low level start bit, eight low-level data bits, and 1 stop bit (a low pulse with a width of nine serial bit cells). The bootstrap loader software measures the width of the initialization byte, calculates the baud rate, and writes the corresponding ASC0 registers with the values that select the detected baud rate.

After the baud rate calculation and initialization (receive pin remains disabled during this time), an identification byte (D5_H) is sent to the external host indicating that the TC1796 is ready to accept a data transfer from the host of exactly 128 bytes (32 words). After the

Reset and Boot Operation

identification byte has been transmitted, the ASC0 receive pin is enabled again. The bootstrap loader software now enters two receive loops. The inner loop waits until it has received four bytes. The outer loop writes one word (four bytes) to the scratchpad RAM (SPRAM) program memory of the PMI. These loops are running until 128 bytes have been received.

After the reception of the 128 bytes, the bootstrap loader software is finished and a jump to address D400 0000_H is executed. This address is the 32-bit word location in the SPRAM at which the first four bytes of the transmitted 128 bytes have been written.

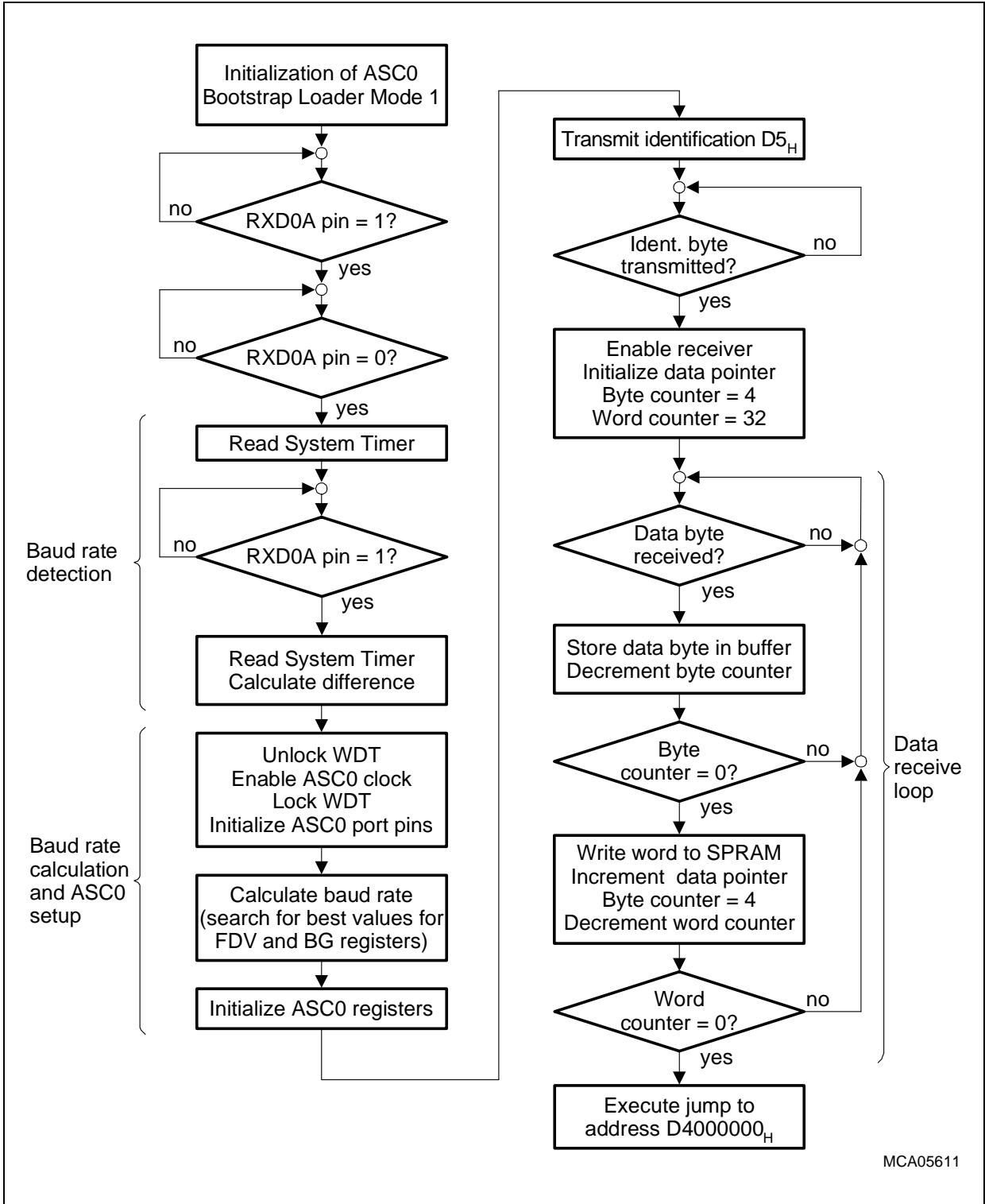


Figure 4-3 Flow Diagram of Bootstrap Loader ASC0 Boot Modes

4.4.2 Bootstrap Loader Mode 2 - CAN Boot via CAN Pins

The CAN boot mode of the bootstrap loader provides a mechanism to load program code/data via the MultiCAN module into the SPRAM, and start executing the loaded code from address D400 0000_H (address of the first transmitted byte). A single data frame always transmits eight code/data bytes from the external host to the TC1796. The number of data frames to be received in CAN boot mode is programmable and depends on the required amount of code/data to be transmitted. The 16-bit data message count value DMSGC determines the number of code/data bytes to be transmitted as $8 \times \text{MSGC}$ with $\text{MSGC} = 1$ to 2^{16} . Therefore, the 48 Kbyte SPRAM of the TC1796 can be loaded completely during CAN boot mode.

The communication between TC1796 and external host is based on the following three CAN standard frames:

- Initialization frame - sent by the external host to the TC1796
- Acknowledge frame - sent by the TC1796 to the external host
- Data frame(s) - sent by the external host to the TC1796

The initialization frame is used in the TC1796 for baud rate detection. After a successful baud rate detection is reported to the external host by the acknowledge frame, data is transmitted by data frames. [Table 4-6](#) shows the parameters and settings of the three CAN standard frames used for CAN bootstrap loader.

Table 4-6 CAN Bootstrap Loader Frames

Frame Type	Parameter	Description
Initialization Frame	Identifier	11-bit, don't care
	DLC = 8	Data length code, 8 bytes transmitted within CAN frame
	Data byte 0	55 _H
	Data byte 1	55 _H
	Data byte 2	Acknowledge message identifier ACKID, low byte
	Data byte 3	Acknowledge message identifier ACKID, high byte
	Data byte 4	Data message count DMSGC, low byte
	Data byte 5	Data message count DMSGC, high byte
	Data byte 6	Data message identifier DMSGID, low byte
	Data byte 7	Data message identifier DMSGID, high byte

Table 4-6 CAN Bootstrap Loader Frames (cont'd)

Frame Type	Parameter	Description
Acknowledge Frame	Identifier	Acknowledge message identifier ACKID as received by data bytes [3:2] of the initialization frame
	DLC = 4	Data length code, 4 bytes transmitted within CAN frame
	Data bytes 0/1	Contents of bit-timing register
	Data bytes 2/3	Copy of acknowledge identifier from initialization frame
Data frame	Identifier	Data message identifier DMSGID as sent by data bytes [7:6] of the initialization frame
	DLC = 8	Data length code, 8 bytes transmitted within CAN frame
	Data bytes 0 to 7	Data bytes, assigned to increasing destination (SPRAM) addresses

Initialization Phase

As in the ASC boot mode, as first task the CAN bootstrap loader has to determine the CAN baud rate at which the external host is communicating. This task requires the external host to send initialization frames continuously to the TC1796. The first two data bytes of the initialization frame include a 2-byte 5555_H baud rate detection pattern, an 11-bit (2-byte) identifier ACKID¹⁾ for the acknowledge frame, a 16-bit data message count value DMSGC, and an 11-bit (2-byte) identifier DMSGID¹⁾ to be used by the data frame.

When CAN boot mode is entered, the CAN bootstrap loader program starts measuring signal pulses at the RXDCAN0 input. After reception and pulse measurements of a 5555_H bit pattern (data bytes 0 and 1 of the initialization frame), the CAN bootstrap loader program calculates the CAN baud rate and programs the baud rate registers of the MultiCAN module. The TC1796 is now ready to receive CAN frames with the baud rate of the external host.

Acknowledge Phase

In the acknowledge phase, the bootstrap loader waits until it receives the next correctly recognized initialization frame from the external host, and acknowledges this frame by generating a dominant bit in its ACK slot. Afterwards, the bootstrap loader transmits an acknowledge frame back to the external host indicating that it is now ready to receive

1) The CAN bootstrap loader copies the two identifier bytes received in the initialization frame directly to register MOAR. Therefore, the respective fields in the initialization frame must contain the intended identifier padded with two dummy bits at the lower end and extended with bitfields IDE (=0_b) and PRI (=01_b) at the upper end.

Reset and Boot Operation

data frames. The acknowledge frame uses the message identifier ACKID that has been received with the initialization frame. The eight data bytes of the acknowledge frame are a copy of the data bytes of the recognized initialization frame.

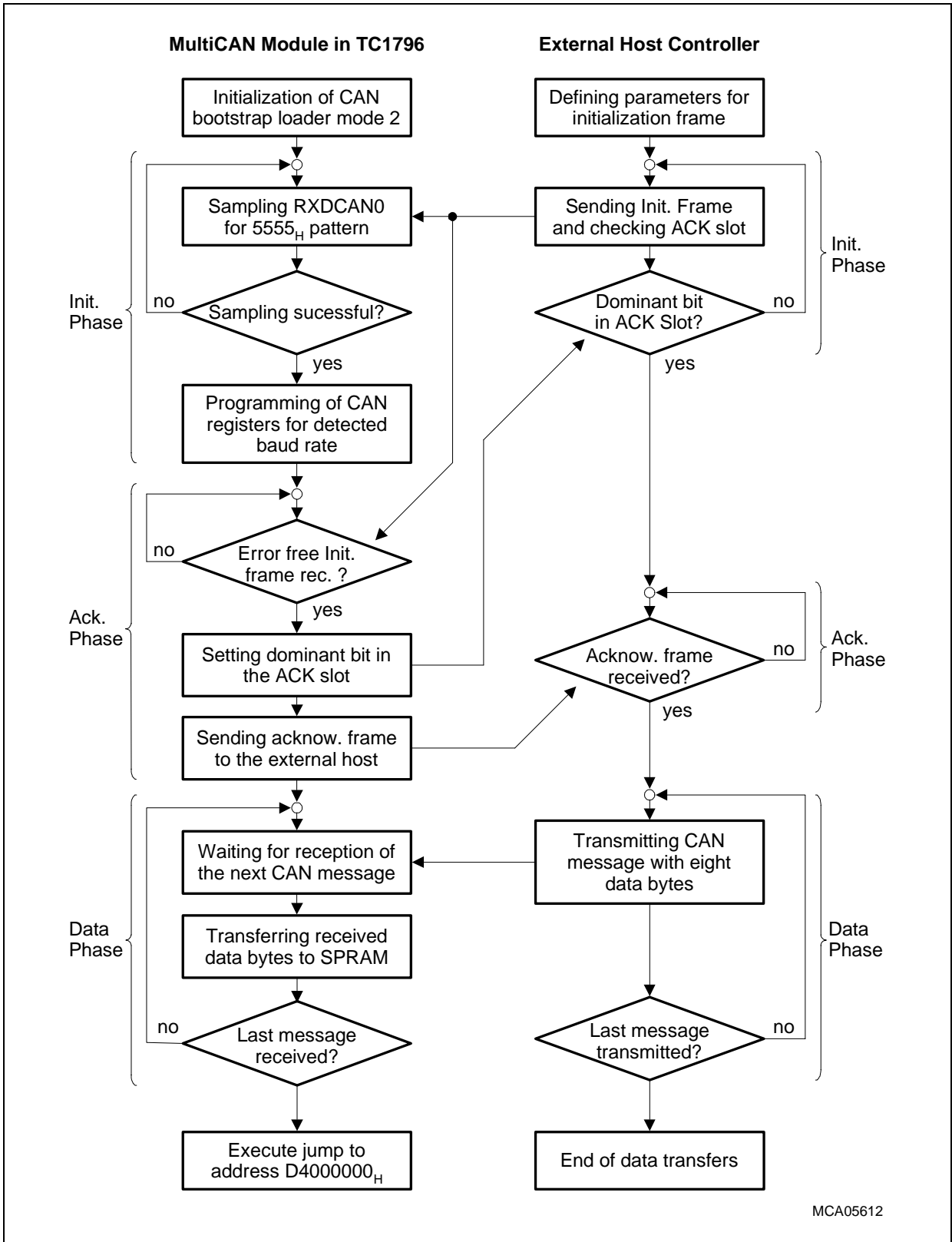
Data Transmission Phase

In the data transmission phase, data frames are sent by the external host and received by the TC1796. The data frame uses the 11-bit (2-byte) data message identifier DMSGID that has been sent with the initialization frame. Eight data bytes are transmitted with each data frame. The first data byte is stored in SPRAM at D400 0000_H. Consecutive data bytes are stored at incrementing addresses.

Both communication partners evaluate the data message count DMSGC until the requested number of CAN data frames has been transmitted. After the reception of the last CAN data frame, the bootstrap loader software is finished and executes a jump to address D400 0000_H. This address is the first 32-bit word location in the SPRAM.

Timing Parameters

There are no general restrictions for CAN timings of the external host. After a reset of external host and TC1796, the external host can start transmitting initialization frames. If no acknowledge frame is sent back within a certain time as defined in the external host (e.g. after a dedicated number of initialization frame transmissions), the external host can decide that the TC1796 is not able to establish the CAN boot communication link.



MCA05612

Figure 4-4 Flow Diagram of CAN Bootstrap Loader Mode

4.4.3 Bootstrap Loader Mode 3 - ASC Boot via CAN Pins

Except for different connections to serial port lines of ASC0, the bootstrap loader mode 3 is identical with bootstrap loader mode 1. The serial data input of the ASC0 is connected to RXD0B which is an alternate function of P6.8/RXDCAN0 and the serial data output of the ASC0 is connected to TXD0B which is an alternate function of P6.9/TXDCAN0.

4.4.4 Alternate Boot Modes

The alternate boot modes (ABM) make it possible to execute a CRC check procedure before jumping to the program start addresses in internal PFLASH or external memory (with EBU as master or as participant). In case of a CRC error in the checked code, the BSL is entered, and loading of serial (ASC or CAN) code/data can be initiated.

To use ABMs, the user has to define the parameters for the CRC checks in the two ABM headers. These parameters are:

- The program start address in case of a positive CRC check
- The start address of the memory range which has to be checked with CRC checksum
- The end address of the memory range which has to be checked with CRC checksum
- The checksums for the checked memory range and for the ABM header itself.

Basically the CRC check procedure of the Boot ROM program consists of three steps:

Step 1: The first CRC check is executed with the parameters of the primary ABM header. If the first CRC check passes, the user program is started at the address as defined in the first word of the primary ABM header.

Step 2: If the first CRC check of the primary ABM header fails, a second CRC check with the parameters of the secondary ABM header is performed. If this second CRC check passes, the user program is started at an alternate start address as defined in the first word of the secondary ABM header.

Step 3: If the second CRC check also fails, one of the three BSL modes is entered. The BSL mode to be used is defined by the content of bit field SCU_SCLIR.SWOPT[2:0]. This bit field contains the state of the Port 0 pins P0.[2:0] that was latched at the last rising edge of HDRST (further details on SCU_SCLIR see [Page 10-24](#)).

Table 4-7 ABM Bootstrap Loader Selections

SWOPT[2:0]	Selected ABM Bootstrap Loader Mode
110 _B	Bootstrap Loader Mode 1: ASC Boot via ASC0 Pins
101 _B	Bootstrap Loader Mode 2: CAN Boot
111 _B ¹⁾	Bootstrap Loader Mode 3: ASC Boot via CAN Pins
others	Reserved; do not use these combinations

Reset and Boot Operation

- 1) This value is default after reset without connecting P0.[2:0] to a low or high level. Port 0 pins are inputs after reset with a pull-up device connected.

Note: For CRC generation and error checking, the Boot ROM software uses the TC1796 on-chip memory checker module with an initial value of FFFF FFFF_H for the memory checker result register before the checksum is generated. Further details about this module are described in [Section 12.4 on Page 12-110](#).

Definition of the ABM Headers

There are two ABM headers defined, a primary and a secondary (alternate). The two ABM headers have a size of 32 bytes (8 words) and are located at fix addresses in the internal or external Flash, as shown in [Table 4-8](#). Alternatively, cached addressing (segment 8) or non-cached addressing (segment 10) can be used.

Table 4-8 ABM Header Locations

Internal Flash Memory (PFLASH)		External Flash Memory	
Base Address	End Address	Base Address	End Address
Primary ABM Header			
8001 FFE0 _H	8001 FFFF _H	8100 FFE0 _H	8100 FFFF _H
A001 FFE0 _H	A001 FFFF _H	A100 FFE0 _H	A100 FFFF _H
Secondary ABM Header			
8003 FFE0 _H	8003 FFFF _H	8108 FFE0 _H	8108 FFFF _H
A003 FFE0 _H	A003 FFFF _H	A108 FFE0 _H	A108 FFFF _H

Table 4-9 shows the structure of the ABM headers.

Table 4-9 Structure of ABM Headers

Address ¹⁾	Value	Function
XXXX XxE0 _H	32-bit start address	Program/code start address
XXXX XxE4 _H	DEAD BEEF _H	Identifier string
XXXX XxE8 _H	32-bit address (checksum start)	32-bit aligned start address of memory range to be checked
XXXX XxEC _H	32-bit address (checksum end)	32-bit aligned end address (last word address) of memory range to be checked
XXXX XXF0 _H	32-bit CRC value CRC _{RANGE}	Expected 32-bit CRC result for memory range to be checked
XXXX XXF4 _H	CRC _{RANGE} inverted	Inverted expected 32-bit CRC result for memory range to be checked
XXXX XXF8 _H	32-bit CRC value CRC _{HEAD}	CRC result of current ABM header from offset (byte) address E0 _H to F7 _H
XXXX XXFC _H	CRC _{HEAD} inverted	Inverted CRC result of current ABM header from offset (byte) address E0 _H to F7 _H

1) XXXX XX is 8001 FF_H/A001 FF_H for primary ABM header and 8003 FF_H/A003 FF_H for secondary ABM header (see also [Table 4-8](#)).

The time needed for CRC generation depends on the length of the memory range to be checked. In case of internal PFLASH check, the duration can be roughly calculated as follows:

- CRC of header: To be defined
- CRC of a Flash space of 400 bytes: To be defined

5 System Control Unit

The System Control Unit (SCU) of the TC1796 handles system control tasks. All the system functions described in this chapter are tightly coupled; thus, they are conveniently handled by one unit, the SCU.

The system tasks of the SCU that are covered in this chapter are:

- Power Management (see [Page 5-2](#))
- Configuration Input Sampling (see [Page 5-8](#))
- External Request Unit (see [Page 5-9](#))
- Special System Interrupts (see [Page 5-35](#))
- On-chip SRAM Parity Control (see [Page 5-37](#))
- Pad Driver Temperature Compensation Control (see [Page 5-41](#))
- Die Temperature Sensor (see [Page 5-48](#))
- GPTA1 Input IN1 Control (see [Page 5-49](#))
- Pad Test Mode Control (see [Page 5-50](#))
- Emergency Stop Output Control for GPTA and MSC (see [Page 5-57](#))
- Analog Input 7 Testmode (see [Page 5-60](#))
- Miscellaneous SCU Registers (see [Page 5-64](#))
- SCU Registers and Address Map (see [Page 5-61](#))

5.1 Power Management

This section describes the power management system of the TC1796. Topics covered here include the internal system interfaces, external interfaces, state diagrams, and the operations of the CPU and peripherals. The power management state machine is also described.

5.1.1 Power Management Overview

The TC1796 power-management system allows software to configure the various processing units so that they automatically adjust to draw the minimum necessary power for the application.

As shown in [Table 5-1](#), there are three power management modes available:

- Run Mode
- Idle Mode
- Sleep Mode

Table 5-1 Power Management Mode Summary

Mode	Description
Run	The system is fully operational. All clocks and peripherals are enabled, as determined by software.
Idle	The CPU clock is disabled, waiting for a condition to return it to Run Mode. Idle Mode can be entered by software when the processor has no active tasks to perform. All peripherals remain powered and clocked. Processor memory is accessible to peripherals. A reset, Watchdog Timer event, a falling edge on the NMI pin, or any enabled interrupt event will return the system to Run Mode.
Sleep	The system clock signal is distributed only to those peripherals programmed to operate in Sleep Mode. The other peripheral module will be shut down by the suspend signal. Interrupts from operating peripherals, the Watchdog Timer, a falling edge on the NMI pin, or a reset event will return the system to Run Mode. Entering this state requires an orderly shut-down controlled by the Power Management State Machine.

The operation of each system component in each of these states can be configured by software. The power-management modes provide flexible reduction of power consumption through a combination of techniques, including:

- Stopping the CPU clock
- Stopping the clocks of other system components individually
- Clock-speed reduction of some peripheral components individually

System Control Unit

The Power Management State Machine (PMSM) controls the power management mode of all system components during Run Mode, Idle Mode, and Sleep Mode. The PMSM continues to operate in Idle Mode and Sleep Mode, even if all other system components have been disabled, so that it can reawaken the system as needed. This flexibility in power management provides minimum power consumption for any application.

Besides these explicit software-controlled power-saving modes, in the TC1796 special attention has been paid to automatic power-saving in those operating units which are currently not required or idle. In that case they are shut off automatically until their operation is required again.

In typical operation, Idle Mode and Sleep Mode may be entered and exited frequently during the run time of an application. For example, system software will typically cause the CPU to enter Idle Mode each time it has to wait for an interrupt before continuing its tasks. In Sleep Mode and Idle Mode, wake-up is performed automatically when any enabled interrupt signal is detected, or if the Watchdog Timer signals the CPU an NMI trap.

5.1.2 Power Management Control and Status Register, PMG_CSR

The set of registers used for power management is divided between central TC1796 components and peripheral components. The PMG_CSR register provides software control and status information for the PMSM. There are individual clock control registers for peripheral components because the Sleep Mode behavior of each peripheral component is programmable. When entering Idle Mode and Sleep Mode, the PMSM directly controls TC1796 components such as the CPU, but indirectly controls peripheral components through their clock control registers.

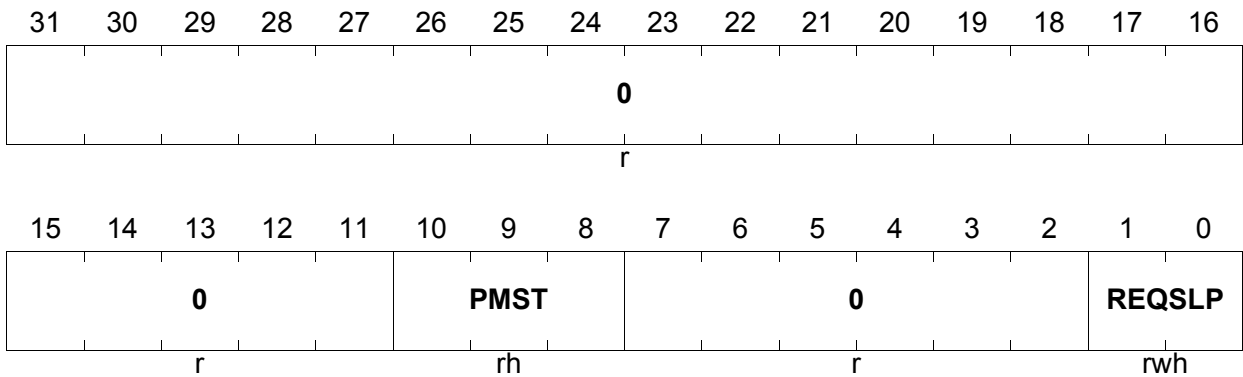
System Control Unit

PMG_CSR

Power Management Control and Status Register

(F0000034_H)

Reset Value: 0000 0100_H



Field	Bits	Type	Function
REQSLP	[1:0]	rwh	Idle Mode and Sleep Mode Request 00 _B Normal Run Mode 01 _B Request Idle Mode 10 _B Request Sleep Mode 11 _B Reserved; do not use this combination In Idle Mode or Sleep Mode, these bits are cleared in response to an enabled interrupt, or when bit 15 of the Watchdog Timer count register (the WDT_SR.TIM[15] bit) changes from 0 to 1.
PMST	[10:8]	rh	Power Management State Machine Status 000 _B Waiting for PLL lock condition 001 _B Normal Run Mode 010 _B Idle Mode requested 011 _B Idle Mode acknowledged 100 _B Sleep Mode 101 _B Undefined, reserved 110 _B Undefined, reserved 111 _B Undefined, reserved
0	[7:2], [31:11]	r	Reserved Read as 0; should be written with 0.

5.1.3 Power Management Modes

This section describes in more detail the power management modes, their operations, and how power management modes are entered and exited. It also describes the behavior of TC1796 system components in all power management modes.

5.1.3.1 Idle Mode

The Idle Mode is requested by software when writing to register PMG_CSR with bit field REQSLP = 01_B. After requesting the Idle Mode, the power management state machine posts an idle request signal to the CPU. The CPU finishes its current operation, sends an acknowledge signal back to the PMSM, and then enters an inactive state in which the CPU clocks and the DMI and PMI memory units are shut off.

Other system components that are able to write to register PMG_CSR can also request the Idle Mode. For example, the PCP or DMA controller can request Idle Mode by writing to the PMG_CSR register.

During Idle Mode, memory accesses to the DMI and PMI via the local memory buses (DLMB and PLMB) cause these units to awaken automatically to handle the transactions. When memory transactions are complete, the DMI and PMI return to Idle Mode again.

The system will return to Run Mode through the occurrence of any of the following conditions:

- An interrupt signal is received from an enabled interrupt source.
- A NMI request is received either from an external source via the NMI pin, or from the Watchdog Timer. The Watchdog Timer triggers an NMI trap request in Idle Mode when its count value (WDT_SR.WD TTIM) changes from 7FFF_H to 8000_H.
- An external power-on reset signal ($\overline{\text{PORST}}$), or hardware reset signal (HDRST) is received.
- A software reset is requested by a FPI Bus agent by writing to the reset request register RST_REQ.

If any of these conditions arise, the TC1796 immediately awakens and returns to Run Mode. If it is awakened by a hardware or software reset signal, the TC1796 system begins its reset sequence. If it is awakened by a Watchdog Timer overflow event, it executes the instruction following the one that was last executed before Idle Mode was entered. If it is awakened by an NMI signal or interrupt signal, the CPU will immediately vector to the appropriate handler.

5.1.3.2 Sleep Mode

The Idle Mode is requested by software when writing to register PMG_CSR with bit field REQSLP = 10_B.

Entering Sleep Mode

Sleep Mode is entered in two steps:

1. The CPU is put into Idle Mode as described in the previous section. When the PMSM receives the Idle acknowledge signal back from the CPU, it proceeds with the second step.
2. A sleep signal is broadcasted on the FPI Bus. Each FPI Bus interface unit receives this signal. The response of each FPI Bus unit to the sleep signal is determined by its clock control register. These clock control registers must have been previously configured by software.

TC1796 State During Sleep Mode

Sleep Mode is disabled for a unit if its clock control register bit EDIS is set to 1. The sleep signal is ignored in this case and the corresponding unit continues normal operation. If EDIS = 0, Sleep Mode is enabled for this unit and the sleep signal will cause this unit to enter Sleep Mode. Two actions then occur:

- The unit's bus interface finishes whatever transaction was in progress when the signal was received.
- The unit's functions are suspended.

The TriCore architecture qualifies the actions in step 2 as follows. Depending on the module's clock control register Fast Shut-Off Enable bit, FSOE, the module's clocks are either immediately stopped (FSOE = 1), or the unit is allowed to finish ongoing operations (FSOE = 0) before the clocks are stopped. For example, setting FSOE to 1 for a serial port will stop all actions in the serial port immediately when the sleep signal is received. Ongoing transmissions or receptions will be aborted. If FSOE is 0, ongoing transmissions or receptions will be completed, and then the clock will be shut off. The purpose of setting FSOE = 1 is to allow a debugger to observe the internal state of a peripheral unit immediately.

Exiting Sleep Mode

The system will be returned to Run Mode by the same events that exit Idle Mode. The response of the CPU to being awakened is also the same as for Idle Mode. Peripheral units that have entered Sleep Mode will switch back to their selected Run Mode operation.

5.1.3.3 States of TC1796 Units in Power Management Modes

Table 5-2 summarizes the state of the various units of the TC1796 during Run Mode, Idle Mode, and Sleep Mode.

Table 5-2 States of TC1796 Units in Power Management Modes

Unit	Run Mode	Idle Mode	Sleep Mode
Main Oscillator & PLL	On	On	On
CPU	Executing	Idle	Idle
DMI & PMI	Active	Idle, but accessible	Idle, but accessible
DMU & PMU	Active	Accessible	Accessible
Flash Module	Active	Active	Powered down
Watchdog Timer	Functioning as programmed	Functioning as programmed	Functioning as programmed
FPI Bus Peripherals	Functioning as programmed	Functioning as programmed	Peripherals with suspend enabled are shut down
Debug Units	Functioning	Functioning	Functioning
External Bus Controller (EBU)	Functioning as programmed	Functioning as programmed	Functioning as programmed
Ports	Functioning as programmed	Functioning as programmed	Functioning as programmed

5.2 Configuration Input Sampling

Several device pins of the TC1796 are latched either by a hardware reset or power-on reset operation. The latched states of such pins may directly determine the start-up configuration and conditions of the TC1796. The latched pins are divided into two types:

- Software configuration inputs, which are latched into a register and have no direct influence on the device start-up configuration
- Hardware configuration inputs, which directly influence the start-up behavior of the TC1796

Table 5-3 determines which pins are latched either by a hardware or power-on reset operation. The state of the corresponding signals is always latched at the rising edges of $\overline{\text{HDRST}}$ or $\overline{\text{PORST}}$.

Table 5-3 Configuration Input Sampling

Configuration Type	Reset Signal	Pins	Remark
Software Configuration	$\overline{\text{HDRST}}$	P0.[15:0]	The states of these pins are latched in register SCU_SCLIR (see also Section 10.3.3.2 on Page 10-24).
Hardware Configuration	$\overline{\text{HDRST}}$	P10.[3:0] (HWFCG[3:0])	The states of these pins are used for selection of the Boot option (see Table 4-3 on Page 4-12). Its state is latched in bit field RST_SR.HWCFG.
		$\overline{\text{BRKIN}}$	The state of this pin is used for selection of the Boot option and for debug purposes.
		P5.3/TXD1A	The state of this pin determines whether oscillator bypass mode is selected or not (see also Table 3-1 on Page 3-9)
		BYPASS	The state of this pin determines whether P5.3/TXD1A is evaluated or not. Additionally, the latched state of BYPASS controls the noise <u>suppression filters</u> of pins $\overline{\text{PORST}}$, $\overline{\text{HDRST}}$, and $\overline{\text{NMI}}$.
	$\overline{\text{PORST}}$	$\overline{\text{TESTMODE}}$	The state of this pin is used for device test mode selection. For normal device operation, this pin should be always be set to 1 level.

5.3 External Request Unit (ERU)

In many cases, external events are used to trigger actions inside a microcontroller. This can be done by triggering external interrupts, which are then serviced by the CPU. Another possibility is to activate module functions that are related to external signals, such as the start of analog-to-digital conversions, counting control for a timer unit or to start a DMA transfer.

Due to the large variety of possible conditions of external signals, the simple generation of interrupt events after the detection of edges of the external signals might not be enough. Therefore, it can become necessary to check for patterns (gating of functions) or to reroute trigger events from one block to another.

In the TC1796, a flexible External Request Unit (ERU) makes it possible to generate trigger events that are able to generate interrupts, trigger a DMA transfers, or start analog-to-digital conversions.

Features

- Edge-detection of an input signal (rising, falling, or both edges)
- Event generation with combined conditions of input signals
- Pattern detection of input pins (e.g gating functionality)
- Flexible input signal selection
- Generation of multiple output trigger events possible

The block diagram of the ERU is shown in [Figure 5-1](#).

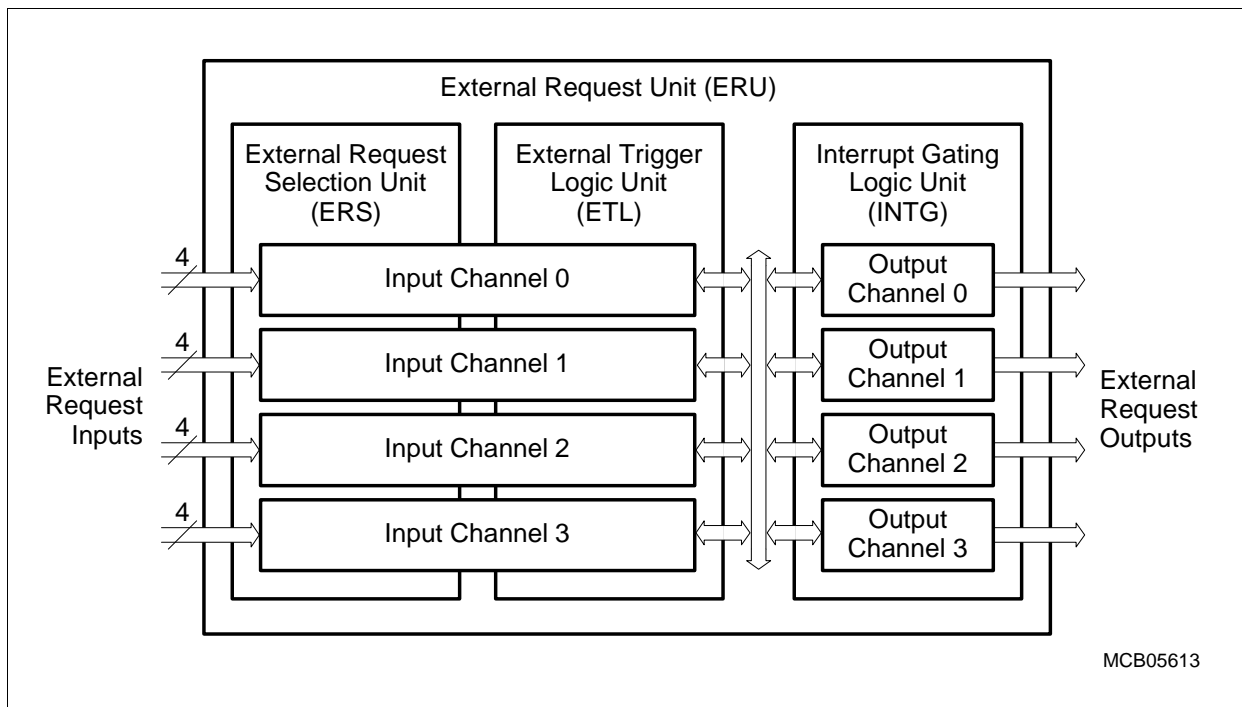


Figure 5-1 External Request Unit Block Diagram

The External Request Select unit (ERS) has three parts:

- The External Request Selection unit (ERS) selects one input signal for each input channel.
- The External Trigger Logic unit (ETL) selects the input channel's input signal trigger condition, and provides output signals to the output channel.
- The Interrupt Gating unit (INTG) combines the detected events of the ETL and provides pattern recognition for each output channel, as well as triggering and gating functions for A/D converters or DMA controller.

The functional blocks of the ERU are described in detail in the following sections.

5.3.1 Input Channel

Figure 5-2 shows the basic structure of an input channel. The input channel has two parts, one part located in the ERS and one part located in the ETL. Two of the four input channels are always controlled by one External Input Channel Register EICRn (n = 0, 1; n = 0 applies to input channel 0 and 1, n = 1 applies to input channel 2 and 3). In the following description, the index number "x" indicates the input channel number.

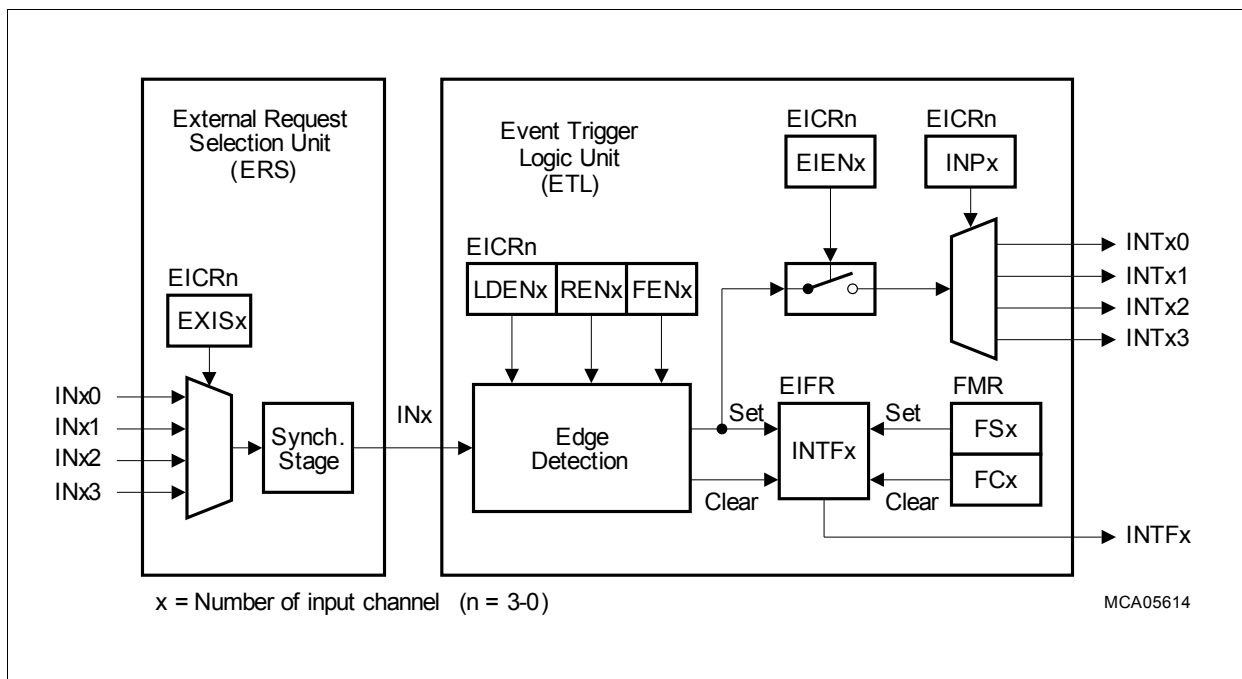


Figure 5-2 External Request Unit Input Channel

An input channel of the ERS contains an 4-to-1 input multiplexer (controlled by bit field EICRn.EXISx) that selects one of four possible request input lines INx[3:0]. The selected signal then becomes synchronized. This is required for further control purposes that are done in the Event Trigger Logic Unit (ETL) of the input channel.

System Control Unit

The ETL of the input channel analyzes the synchronized output signal IN_x of the ERS by an edge-detection logic. This edge-detection block generates a pulse (one clock cycle long) when a signal transition is detected. The selection which signal transition (edge) should generate a pulse (event) is done by two control bits, one for the rising edge ($EICR_n.REN_x$) and one for the falling edge ($EICR_n.FEN_x$). Therefore, only a rising or falling edge, or both edges of the input signal can be detected. When a selected signal transition is detected, the external interrupt flag $EIFR_n.INTF_x$ becomes always set by hardware, even it has been set previously. The interrupt flag $EIFR_n.INTF_x$ is available as output signal $INTF_x$ of the ETL.

The hardware reset condition for $EIFR_n.INTF_x$ can be controlled in two different ways. These two modes are selected by bit $EIFR_n.LDEN_x$.

- $EIFR_n.LDEN_x = 0$:
Flag $EIFR_n.INTF_x$ is used as a sticky bit, indicating that the selected signal transition has been detected by the edge-detection logic at least once. In this mode, bit $EIFR_n.INTF_x$ can be cleared only by software.
- $EIFR_n.LDEN_x = 1$:
In this mode, flag $EIFR_n.INTF_x$ becomes cleared by hardware if an edge is detected at the IN_x signal, that has not been selected. This means, $EIFR_n.INTF_x$ is cleared by hardware at a IN_x rising edge when $EICR_n.FEN_x = 1$ and $EIFR_n.INTF_x$ is cleared by hardware at a IN_x falling edge when $EICR_n.REN_x = 1$. If both bits, $EICR_n.FEN_x$ and $EICR_n.REN_x$, are set, flag $EIFR_n.INTF_x$ becomes never cleared by hardware. In this case, flag $EIFR_n.INTF_x$ can be only cleared by software.

Flag $EIFR_n.INTF_x$ can be set or cleared by software using the bits FS_x or FC_x in the flag modification register FMR . Writing to FMR with bit $FMR.FS_x$ set sets flag $EIFR_n.INTF_x$. Writing to FMR with bit $FMR.FC_x$ set clears flag $EIFR_n.INTF_x$. If both bits are set for one input channel x , a set operation of the flag will be executed.

Each output pulse of the edge-detection block that sets the $EIFR_n.INTF_x$ flag by hardware can be routed also to one of the $INT_x[3:0]$ outputs of the ETL unit. The external interrupt enable bit $EICR_n.EIEN_x = 1$ enables the interrupt output pulse at the $INT_x[3:0]$ outputs. Bit field $EICR_n.INP_x$ controls which of the $INT_x[3:0]$ outputs is selected for the output pulse.

5.3.2 Output Channel

The output channel logic shown in **Figure 5-3** is built in for each of the four output channels.

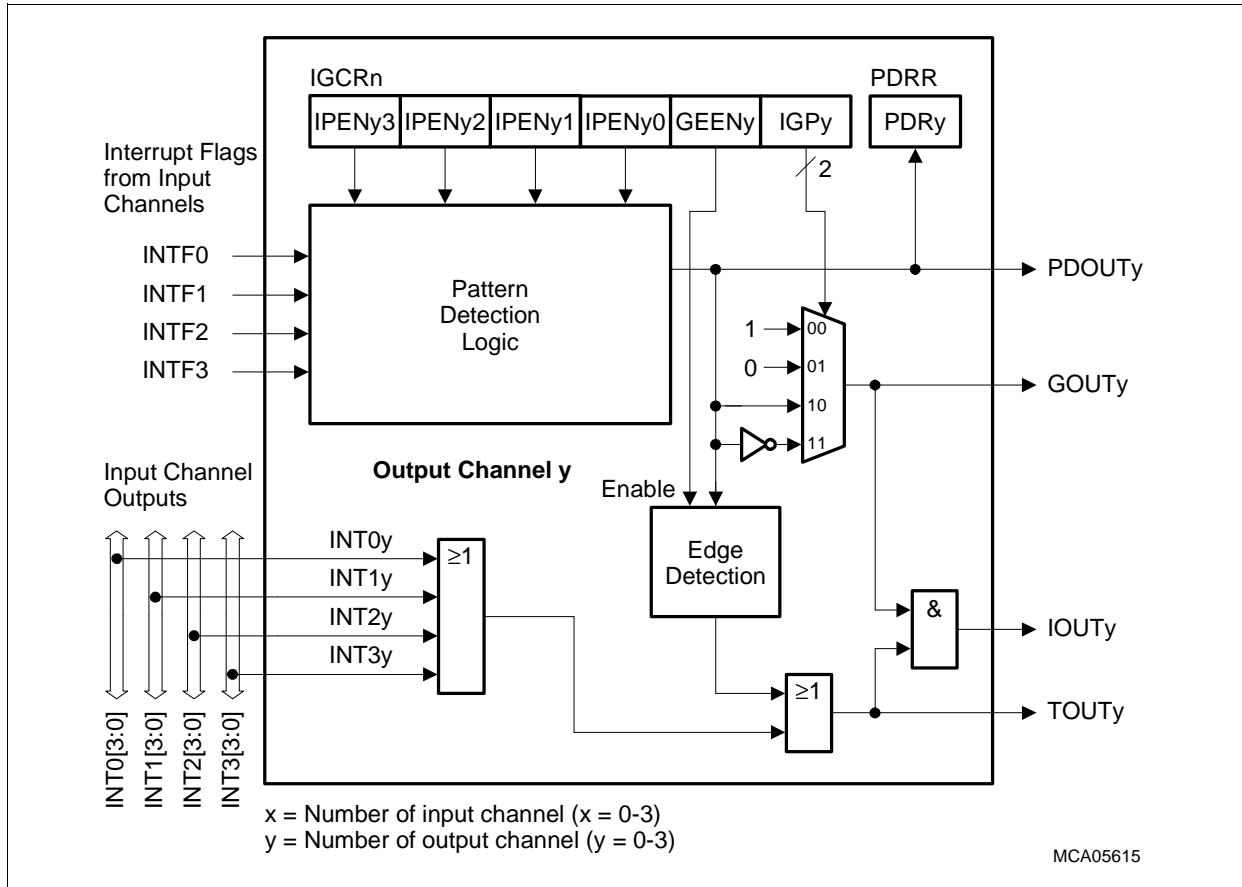


Figure 5-3 Interrupt Gating Logic (Output Channel y)

The output signals of the output channel y can be used to trigger peripheral actions, DMA requests, or interrupts.

PDOUY Output Signal

The pattern detection output PDOUY can be used to enable/disable peripheral functions while a certain condition (pattern) is detected (level controlled output). PDOUY can be typically used to enable/disable analog-to-digital conversions. The output PDOUY is active as long as the programmed condition of input signals (INTFx) is met.

Each INTFx output signal from input channel x is connected to each output channel y. Bit IPENy[3:0] determine whether flag INTFx of input channel x takes part in the pattern detection of output channel y. The output signal PDOUY is an AND combination of all INTFx inputs that are enabled by IPENyx = 1. In other words, the PDOUY signal

becomes active (high) when the selected input channel interrupt flags are at active (high) level. The state of flag PDRR.PDRy indicates the actual state of the PDOUTy signal.

GOUTy Output Signal

The gating output GOUTy represents the programmable level of PDOUTy. This output signal is not connected/used in TC1796 but referred in the register descriptions.

TOUTy Output Signal

The trigger output TOUTy combines all related event trigger sources. This output can be used to request ADC conversions or to start other peripheral actions. TOUTy outputs pulses.

The incoming high-level active interrupt request pulses INT0y to INT3y from the event trigger logic are combined to one common interrupt request for the corresponding output channel. The incoming signals remain inactive when an event has been detected but another output channel has been selected by INPx in the channel event trigger logic. As a result, only those interrupt requests are taken into account, that are targeting this output channel (y).

IOUT Output Signal

The interrupt output IOUTy can be connected to an interrupt node, and combines the gating functionality GOUTy (programmed level of PDOUTy) and the trigger functionality of TOUTy. It can also be used to trigger module actions, such as DMA requests or GPTA actions. IOUTy outputs pulses.

Output Channel Control Logic

A very useful additional feature is the possibility to gate the incoming interrupt requests with the indication flags INTFx (x = 0-3) coming from the Event Trigger Logic of the Input Channel x. This allows more gating capability for the interrupt generation. The output channel can be activated when a trigger event occurs while a certain pattern is detected (IGP = 10_B), or while this pattern is not detected (IGP = 11_B). If an indication flag should not be taken into account for the gating of the interrupt requests, the related IPENyx bit has to be 0.

Bit GEENy = 1 enables the generation of a trigger event for output channel y when the result of the pattern detection changes. When using this feature, a trigger (e.g. for an interrupt) is generated during the first clock cycle when a pattern is detected or when it is no longer detected.

The output of the gating AND of all interrupts request are also delayed by one clock cycle to detect a change of the gating status (XOR), corresponding to the pattern detection. The bit GEENy (gating edge enable) makes it possible to generate a pulse whenever the gating status changes. Combined with the bit field IGPy, an event can be generated

when a pattern is detected, when a pattern is no longer detected, or if both conditions are true.

5.3.3 External Request Unit Implementation

This section describes how the ERU is interconnected within other on-chip modules.

The connections of the ERU input lines and the connections between input and output channels are shown in **Figure 5-4**. **Figure 5-5** shows how the outputs of the output channels are connected to the A/D converters ADC0 and ADC1, to the DMA controller, and to the GPTA modules.

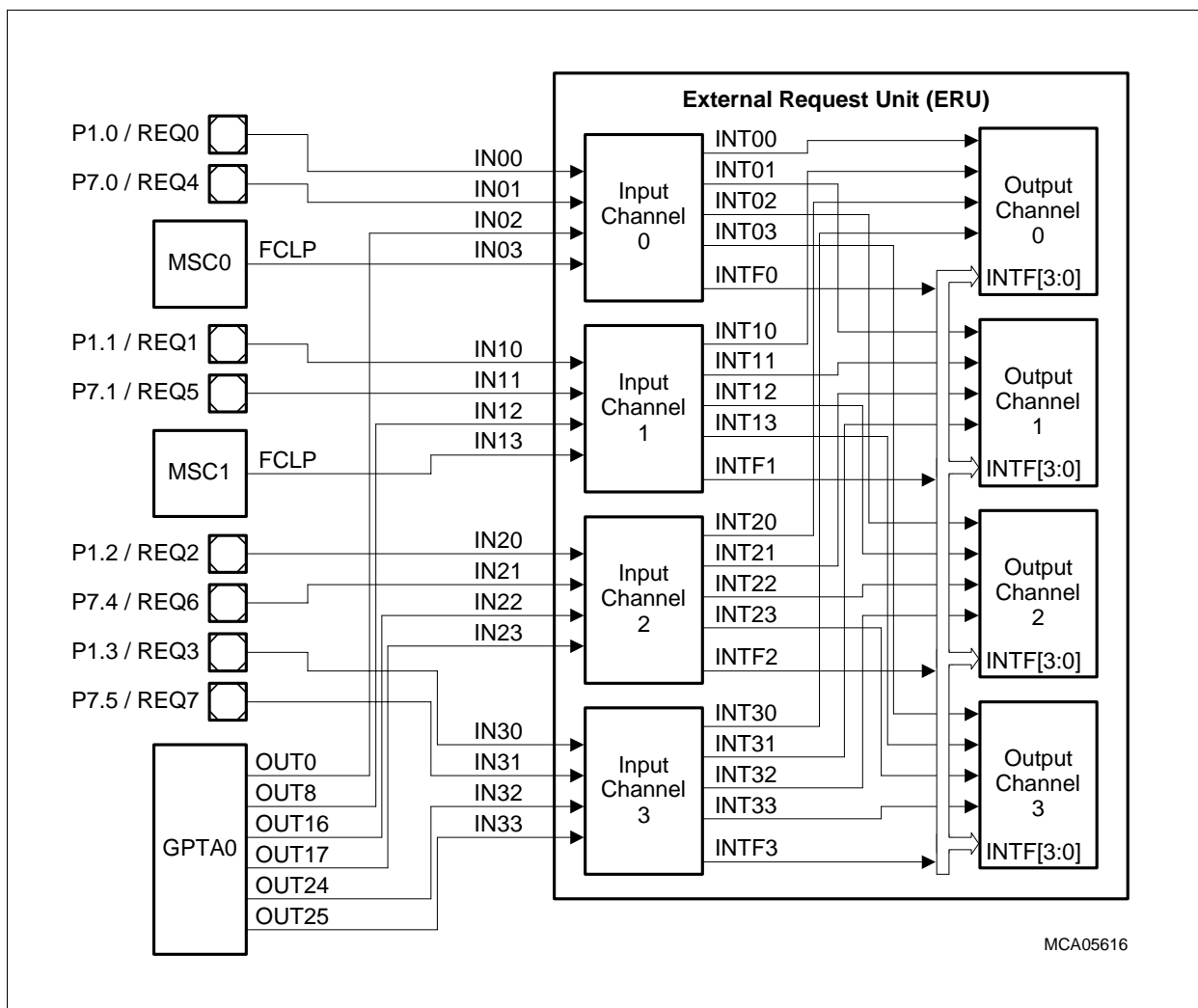


Figure 5-4 Input Connections of External Request Unit

As shown in **Figure 5-5**, additional trigger lines ROUT are generated by the GPTA0 module. These lines detect rising edges at three dedicated GPTA0 outputs and can be used to trigger ADC actions, for example. The GPTA0 outputs are connected to an edge-detection logic to obtain a trigger pulse each time a rising edge is detected.

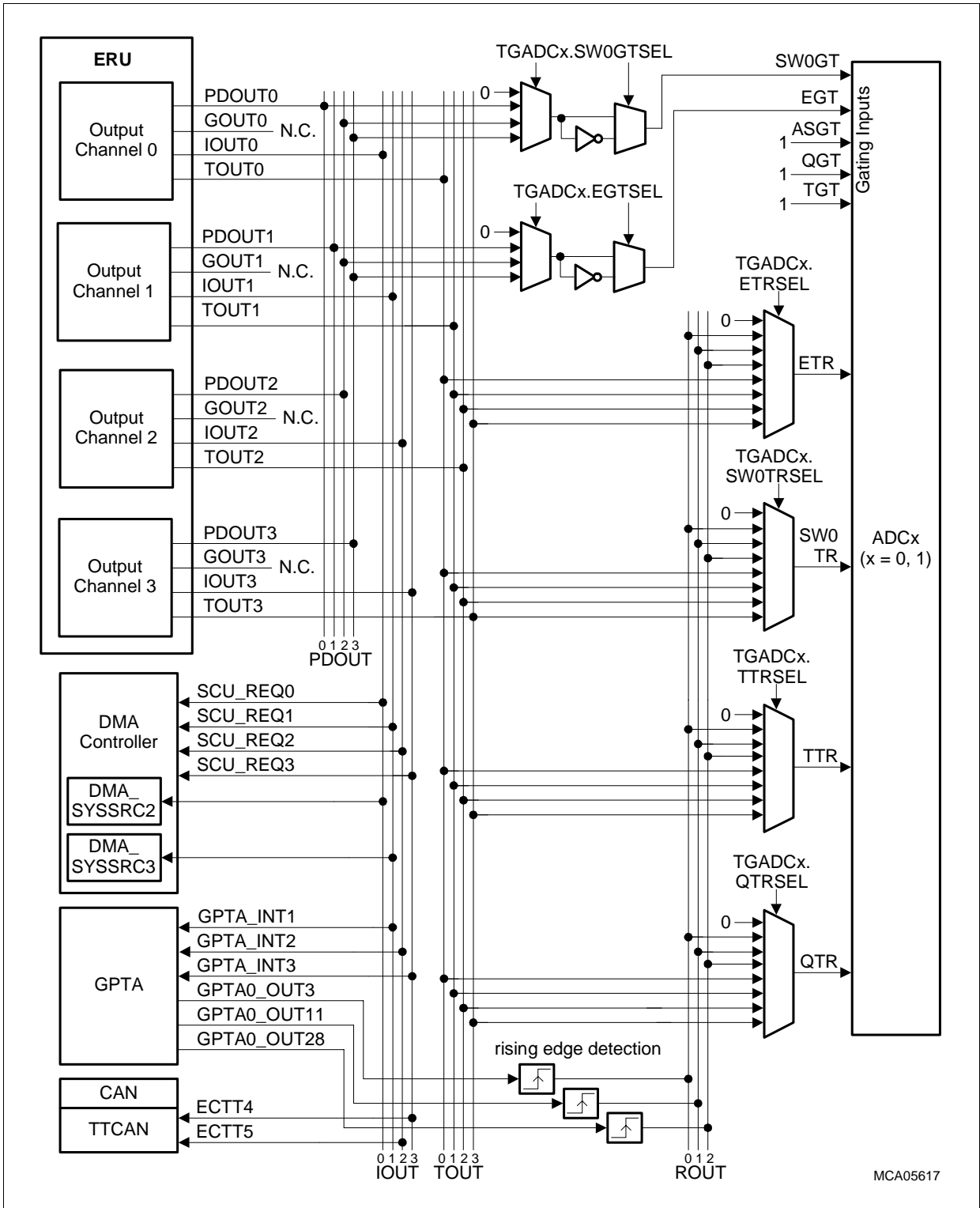


Figure 5-5 Output Connections of External Request Unit

Note: The trigger and gating selection is independent for ADC0 and ADC1 (index “x”).

5.3.4 External Request Unit Registers

Table 5-4 refer to the registers associated with the ERU Kernel as well as the two TC1796 implementation-specific ERU registers, which control the ERU connections with the A/D converters ADC0 and ADC1.

Table 5-4 External Trigger Request Unit Kernel Registers

Register Short Name	Register Long Name	Description see
ERU Kernel Registers		
EICR0	External Input Channel Register 0	Page 5-17
EICR1	External Input Channel Register 1	Page 5-20
EIFR	External Input Flag Register	Page 5-23
FMR	Flag Modification Register	Page 5-24
PDRR	Pattern Detection Result Register	Page 5-25
IGCR0	Interrupt Gating Register 0	Page 5-26
IGCR1	Interrupt Gating Register 1	Page 5-29
Implementation-specific ERU Registers		
TGADC0	Trigger Gating ADC0 Register	Page 5-32
TGADC1	Trigger Gating ADC1 Register	Page 5-32

System Control Unit

The External Input Channel Register EICR0 and EICR1 for the external input channels 0 to 3 contain bits to configure the input gating logic IGL and the event trigger logic ETL. A maximum of 12 input channels are supported by one input unit (defined by the maximum number of IGCRy.IPENx bits).

EICR0

External Input Channel Register 0

(F000080_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	INP1			EI EN1	LD EN1	R EN1	F EN1	0	EXIS1			0			
r	rw			rw	rw	rw	rw	r	rw			r			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	INP0			EI EN0	LD EN0	R EN0	F EN0	0	EXIS0			0			
r	rw			rw	rw	rw	rw	r	rw			r			

Field	Bits	Type	Description
EXIS0	[5:4]	rw	External Input Selection 0 This bit field determines which input line is selected for signal IN0. 00 _B Input IN00 is selected. 01 _B Input IN01 is selected. 10 _B Input IN02 is selected. 11 _B Input IN03 is selected.
FEN0	8	rw	Falling Edge Enable 0 This bit determines if the falling edge of signal IN0 is used to set bit INTF0. 0 _B The falling edge is not used. 1 _B The detection of a falling edge of IN0 generates a trigger event (INTF0 becomes set).
REN0	9	rw	Rising Edge Enable 0 This bit determines if the rising edge of signal IN0 is used to set bit INTF0. 0 _B The rising edge is not used. 1 _B The detection of a rising edge of IN0 generates a trigger event (INTF0 becomes set).

System Control Unit

Field	Bits	Type	Description
LDEN0	10	rw	<p>Level Detection Enable 0</p> <p>This bit determines if bit INTF0 is cleared automatically if an edge of the input signal IN0 is detected, which has not been selected (rising edge with REN0 = 0 or falling edge with FEN0 = 0).</p> <p>0_B Bit INTF0 will not be cleared. 1_B Bit INTF0 will be cleared.</p>
EIEN0	11	rw	<p>External Interrupt Enable 0</p> <p>This bit enables the generation of a trigger event for request channel 0 (e.g. for interrupt generation) when a selected edge is detected.</p> <p>0_B The trigger event is disabled. 1_B The trigger event is enabled.</p>
INP0	[14:12]	rw	<p>Interrupt Node Pointer</p> <p>This bit field determines the destination (output channel) for trigger event 0 (if enabled by EIEN0).</p> <p>X00_B The event of input channel 0 triggers output channel 0 (signal INT00). X01_B The event of input channel 0 triggers output channel 1 (signal INT01). X10_B The event of input channel 0 triggers output channel 2 (signal INT02). X11_B The event of input channel 0 triggers output channel 3 (signal INT03).</p>
EXIS1	[21:20]	rw	<p>External Input Selection 1</p> <p>This bit field determines which input line is selected for signal IN1.</p> <p>00_B Input IN10 is selected. 01_B Input IN11 is selected. 10_B Input IN12 is selected. 11_B Input IN13 is selected.</p>
FEN1	24	rw	<p>Falling Edge Enable 1</p> <p>This bit determines if the falling edge of signal IN1 is used to set bit INTF1.</p> <p>0_B The falling edge is not used. 1_B The detection of a falling edge of IN1 generates a trigger event (INTF1 becomes set).</p>

System Control Unit

Field	Bits	Type	Description
REN1	25	rw	<p>Rising Edge Enable 1 This bit determines if the rising edge of signal IN1 is used to set bit INTF1.</p> <p>0_B The rising edge is not used. 1_B The detection of a rising edge of IN1 generates a trigger event (INTF1 becomes set).</p>
LDEN1	26	rw	<p>Level Detection Enable 1 This bit determines if bit INTF1 is cleared automatically if an edge of the input signal IN1 is detected, which has not been selected (rising edge with REN1 = 0 or falling edge with FEN1 = 0).</p> <p>0_B Bit INTF1 will not be cleared. 1_B Bit INTF1 will be cleared.</p>
EIEN1	27	rw	<p>External Interrupt Enable 1 This bit enables the generation of a trigger event for request channel 1 (e.g. for interrupt generation) when a selected edge is detected.</p> <p>0_B The trigger event is disabled. 1_B The trigger event is enabled.</p>
INP1	[30:28]	rw	<p>Interrupt Node Pointer This bit field determines the destination (output channel) for trigger event 1 (if enabled by EIEN1).</p> <p>X00_B The event of input channel 1 triggers output channel 0 (signal INT10). X01_B The event of input channel 1 triggers output channel 1 (signal INT11). X10_B The event of input channel 1 triggers output channel 2 (signal INT12). X11_B The event of input channel 1 triggers output channel 3 (signal INT13).</p>
0	[3:0], [7:6], [19:15], [23:22], 31	r	<p>Reserved Read as 0; should be written with 0.</p>

EICR1

External Input Channel Register 1

(F000084_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		INP3		EI EN3	LD EN3	R EN3	F EN3	0		EXIS3		0			
r		rw		rw	rw	rw	rw	r		rw		r			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		INP2		EI EN2	LD EN2	R EN2	F EN2	0		EXIS2		0			
r		rw		rw	rw	rw	rw	r		rw		r			

Field	Bits	Type	Description
EXIS2	[5:4]	rw	External Input Selection 2 This bit field determines which input line is selected for signal IN2. 00 _B Input IN20 is selected. 01 _B Input IN21 is selected. 10 _B Input IN22 is selected. 11 _B Input IN23 is selected.
FEN2	8	rw	Falling Edge Enable 2 This bit determines if the falling edge of signal IN2 is used to set bit INTF2. 0 _B The falling edge is not used. 1 _B The detection of a falling edge of IN2 generates a trigger event (INTF3 becomes set).
REN2	9	rw	Rising Edge Enable 2 This bit determines if the rising edge of signal IN2 is used to set bit INTF2. 0 _B The rising edge is not used. 1 _B The detection of a rising edge of IN2 generates a trigger event (INTF2 becomes set).

System Control Unit

Field	Bits	Type	Description
LDEN2	10	rw	<p>Level Detection Enable 2</p> <p>This bit determines if bit INTF2 is cleared automatically if an edge of the input signal IN2 is detected, which has not been selected (rising edge with REN2 = 0 or falling edge with FEN2 = 0).</p> <p>0_B Bit INTF2 will not be cleared. 1_B Bit INTF2 will be cleared.</p>
EIEN2	11	rw	<p>External Interrupt Enable 2</p> <p>This bit enables the generation of a trigger event for request channel 2 (e.g. for interrupt generation) when a selected edge is detected.</p> <p>0_B The trigger event is disabled. 1_B The trigger event is enabled.</p>
INP2	[14:12]	rw	<p>Interrupt Node Pointer</p> <p>This bit field determines the destination (output channel) for trigger event 2 (if enabled by EIEN2).</p> <p>X00_B The event of input channel 2 triggers output channel 0 (signal INT20). X01_B The event of input channel 2 triggers output channel 1 (signal INT21). X10_B The event of input channel 2 triggers output channel 2 (signal INT22). X11_B The event of input channel 2 triggers output channel 3 (signal INT23).</p>
EXIS3	[21:20]	rw	<p>External Input Selection 3</p> <p>This bit field determines which input line is selected for signal IN3.</p> <p>00_B Input IN30 is selected. 01_B Input IN31 is selected. 10_B Input IN32 is selected. 11_B Input IN33 is selected.</p>
FEN3	24	rw	<p>Falling Edge Enable 3</p> <p>This bit determines if the falling edge of signal IN3 is used to set bit INTF3.</p> <p>0_B The falling edge is not used. 1_B The detection of a falling edge of IN3 generates a trigger event (INTF3 becomes set).</p>

System Control Unit

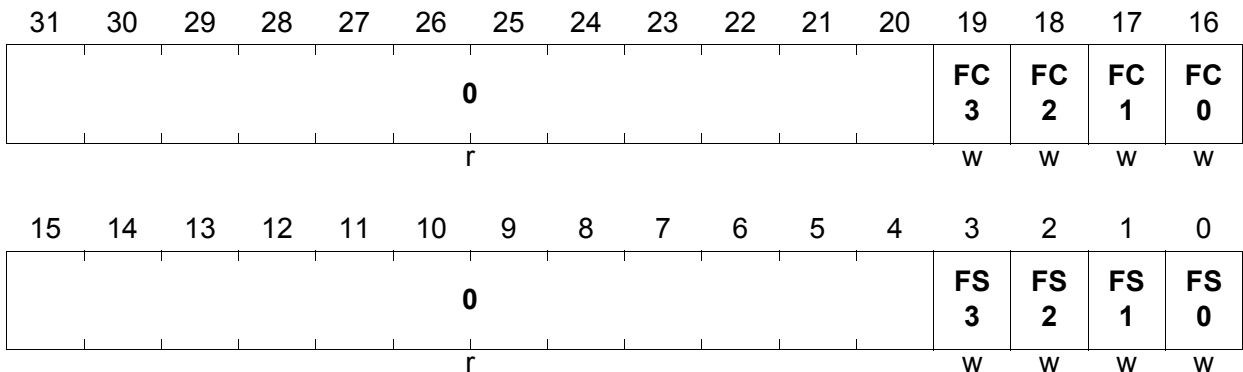
Field	Bits	Type	Description
REN3	25	rw	<p>Rising Edge Enable 3 This bit determines if the rising edge of signal IN3 is used to set bit INTF3.</p> <p>0_B The rising edge is not used. 1_B The detection of a rising edge of IN3 generates a trigger event (INTF3 becomes set).</p>
LDEN3	26	rw	<p>Level Detection Enable 3 This bit determines if bit INTF3 is cleared automatically if an edge of the input signal IN3 is detected, which has not been selected (rising edge with REN3 = 0 or falling edge with FEN3 = 0).</p> <p>0_B Bit INTF3 will not be cleared. 1_B Bit INTF3 will be cleared.</p>
EIEN3	27	rw	<p>External Interrupt Enable 3 This bit enables the generation of a trigger event for request channel 3 (e.g. for interrupt generation) when a selected edge is detected.</p> <p>0_B The trigger event is disabled. 1_B The trigger event is enabled.</p>
INP3	[30:28]	rw	<p>Interrupt Node Pointer This bit field determines the destination (output channel) for trigger event 3 (if enabled by EIEN3).</p> <p>X00_B The event of input channel 3 triggers output channel 0 (signal INT30). X01_B The event of input channel 3 triggers output channel 1 (signal INT31). X10_B The event of input channel 3 triggers output channel 2 (signal INT32). X11_B The event of input channel 3 triggers output channel 3 (signal INT33).</p>
0	[3:0], [7:6], [19:15], [23:22], 31	r	<p>Reserved Read as 0; should be written with 0.</p>

System Control Unit

The Flag Modification Register is a write-only register that is used to set and to clear the bits INTFx in register EIFR. If a set event and a clear event (hardware or software) for bit INTFx occur at the same time, the set event is taken into account.

FMR

Flag Modification Register (F00008C_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
FSx (x = 0-3)	x	w	Set Flag INTFx for Channel x Setting this bit will set the corresponding bit INTFx in register EIFR. Reading this bit always delivers a 0. 0 _B The bit x in register EIFR is not modified. 1 _B The bit x in register EIFR is set.
FCx (x = 0-3)	16 + x	w	Reset Flag INTFx for Channel x Setting this bit will clear the corresponding bit INTFx in register EIFR. Reading this bit always delivers a 0. 0 _B The bit x in register EIFR is not modified. 1 _B The bit x in register EIFR is reset.
0	[15:4], [31:20]	r	Reserved Read as 0; should be written with 0.

Note: This register is virtual and does not contain any flip-flop.

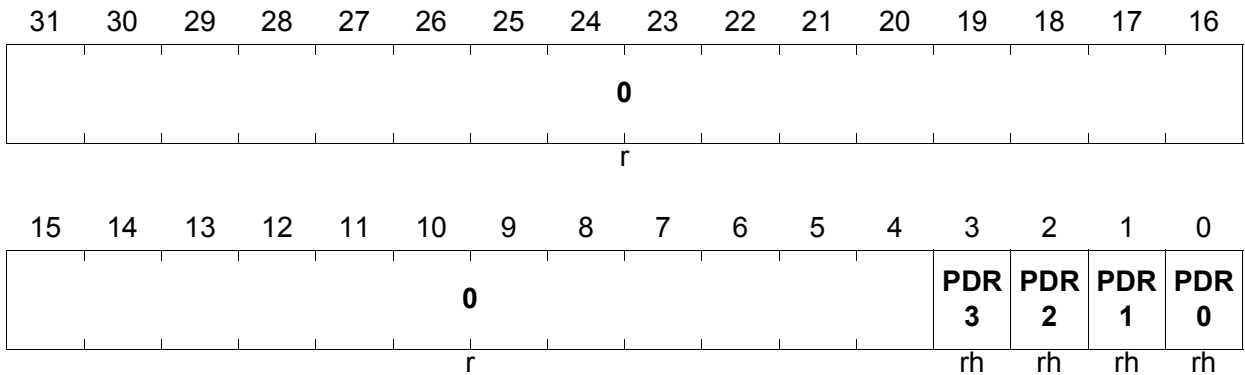
System Control Unit

The Pattern Detection Result Register monitors the combinatorial output status of the pattern detection units.

PDRR

Pattern Detection Result Register (F0000090_H)

Reset Value: 0000 000F_H



Field	Bits	Type	Description
PDR_y (y = 0-3)	y	rh	Pattern Detection Result of Channel y This bit monitors the output status of the pattern detection for the output channel y.
0	[31:4]	r	Reserved Read as 0; should be written with 0.

System Control Unit

The Interrupt Gating Control Registers IGCR0 and IGCR1 contain bits to enable the pattern detection and to control the gating for output channel 0 to 3 (e.g. for interrupt nodes or peripherals).

IGCR0

Interrupt Gating Register 0

(F0000094_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
IGP1		GE EN1		0								IPEN 13		IPEN 12		IPEN 11		IPEN 10	
rw		rw		r								rw		rw		rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
IGP0		GE EN0		0								IPEN 03		IPEN 02		IPEN 01		IPEN 00	
rw		rw		r								rw		rw		rw		rw	

Field	Bits	Type	Description
IPEN0x (x = 0-3)	x	rw	<p>Interrupt Pattern Enable for Channel 0</p> <p>Bit IPEN0x determines the flag INTFx of channel x takes part in the pattern detection for the gating of the requests for the output signals GOUTy and IOUTy.</p> <p>0_B The bit INTFx does not take part in the pattern detection.</p> <p>1_B The bit INTFx is taken into consideration for the pattern detection.</p>
GEEN0	13	rw	<p>Generate Event Enable 0</p> <p>Bit GEEN0 enables the generation of a trigger event for output channel 0 when the result of the pattern detection changes. When using this feature, a trigger (e.g. for an interrupt) is generated during the first clock cycle when a pattern is detected or when it is no longer detected.</p> <p>0_B The trigger generation at a change of the pattern detection result is disabled.</p> <p>1_B The trigger generation at a change of the pattern detection result is enabled.</p>

Field	Bits	Type	Description
IGP0	[15:14]	rw	<p>Interrupt Gating Pattern 0</p> <p>Bit field IGP0 determines how the pattern detection influences the output lines GOUT0 and IOUT0.</p> <p>00_B The detected pattern is not taken into account. An activation of IOUT0 is always possible due to a trigger event.</p> <p>01_B The detected pattern is not taken into account. An activation of IOUT0 is not possible.</p> <p>10_B The detected pattern is taken into account. An activation of IOUT0 is only possible due to a trigger event while the pattern is detected.</p> <p>11_B The detected pattern is taken into account. An activation of IOUT0 is only possible due to a trigger event while the pattern is not detected.</p>
IPEN1x (x = 0-3)	16+x	rw	<p>Interrupt Pattern Enable for Channel 1</p> <p>Bit IPEN1x determines if the flag INTFx of channel x takes part in the pattern detection for the gating of the requests for the output signals GOUTy and IOUTy.</p> <p>0_B The bit INTFx does not take part in the pattern detection.</p> <p>1_B The bit INTFx is taken into consideration for the pattern detection.</p>
GEEN1	29	rw	<p>Generate Event Enable 1</p> <p>Bit GEEN1 enables the generation of a trigger event for output channel 1 when the result of the pattern detection changes. When using this feature, a trigger (e.g. for an interrupt) is generated during the first clock cycle when a pattern is detected, or when it is no longer detected.</p> <p>0_B The trigger generation at a change of the pattern detection result is disabled.</p> <p>1_B The trigger generation at a change of the pattern detection result is enabled.</p>

Field	Bits	Type	Description
IGP1	[31:30]	rw	<p>Interrupt Gating Pattern 1</p> <p>Bit field IGP1 determines how the pattern detection influences the output lines GOUT1 and IOUT1.</p> <p>00_B The detected pattern is not taken into account. An activation of IOUT1 is always possible due to a trigger event.</p> <p>01_B The detected pattern is not taken into account. An activation of IOUT1 is not possible.</p> <p>10_B The detected pattern is taken into account. An activation of IOUT1 is only possible due to a trigger event while the pattern is detected.</p> <p>11_B The detected pattern is taken into account. An activation of IOUT1 is only possible due to a trigger event while the pattern is not detected.</p>
0	[12:4], [28:20]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

System Control Unit

IGCR1

Interrupt Gating Register 1

(F0000098_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
IGP3		GE EN3		0								IPEN 33		IPEN 32	IPEN 31	IPEN 30
rw		rw		r								rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
IGP2		GE EN2		0								IPEN 23		IPEN 22	IPEN 21	IPEN 20
rw		rw		r								rw		rw	rw	rw

Field	Bits	Type	Description
IPEN2x (x = 0-3)	x	rw	<p>Interrupt Pattern Enable for Channel 2</p> <p>Bit IPEN2x determines if the flag INTFx of channel x takes part in the pattern detection for the gating of the requests for the output signals GOUTy and IOUTy.</p> <p>0_B The bit INTFx does not take part in the pattern detection.</p> <p>1_B The bit INTFx is taken into consideration for the pattern detection.</p>
GEEN2	13	rw	<p>Generate Event Enable 2</p> <p>Bit GEEN2 enables the generation of a trigger event for output channel 2 when the result of the pattern detection changes. When using this feature, a trigger (e.g. for an interrupt) is generated during the first clock cycle when a pattern is detected, or when it is no longer detected.</p> <p>0_B The trigger generation at a change of the pattern detection result is disabled.</p> <p>1_B The trigger generation at a change of the pattern detection result is enabled.</p>

Field	Bits	Type	Description
IGP2	[15:14]	rw	<p>Interrupt Gating Pattern 2</p> <p>Bit field IGP2 determines how the pattern detection influences the output lines GOUT2 and IOUT2.</p> <p>00_B The detected pattern is not taken into account. An activation of IOUT2 is always possible due to a trigger event.</p> <p>01_B The detected pattern is not taken into account. An activation of IOUT2 is not possible.</p> <p>10_B The detected pattern is taken into account. An activation of IOUT2 is only possible due to a trigger event while the pattern is detected.</p> <p>11_B The detected pattern is taken into account. An activation of IOUT2 is only possible due to a trigger event while the pattern is not detected.</p>
IPEN3x (x = 0-3)	16+x	rw	<p>Interrupt Pattern Enable for Channel 3</p> <p>Bit IPEN3x determines if the flag INTFx of channel x takes part in the pattern detection for the gating of the requests for the output signals GOUTy and IOUTy.</p> <p>0_B The bit INTFx does not take part in the pattern detection.</p> <p>1_B The bit INTFx is taken into consideration for the pattern detection.</p>
GEEN3	29	rw	<p>Generate Event Enable 3</p> <p>Bit GEEN3 enables the generation of a trigger event for output channel 3 when the result of the pattern detection changes. When using this feature, a trigger (e.g. for an interrupt) is generated during the first clock cycle when a pattern is detected, or when it is no longer detected.</p> <p>0_B The trigger generation at a change of the pattern detection result is disabled.</p> <p>1_B The trigger generation at a change of the pattern detection result is enabled.</p>

Field	Bits	Type	Description
IGP3	[31:30]	rw	<p>Interrupt Gating Pattern 3</p> <p>Bit field IGP3 determines how the pattern detection influences the output lines GOUT3 and IOUT3.</p> <p>00_B The detected pattern is not taken into account. An activation of IOUT3 is always possible due to a trigger event.</p> <p>01_B The detected pattern is not taken into account. An activation of IOUT3 is not possible.</p> <p>10_B The detected pattern is taken into account. An activation of IOUT3 is only possible due to a trigger event while the pattern is detected.</p> <p>11_B The detected pattern is taken into account. An activation of IOUT3 is only possible due to a trigger event while the pattern is not detected.</p>
0	[12:4], [28:20]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

System Control Unit

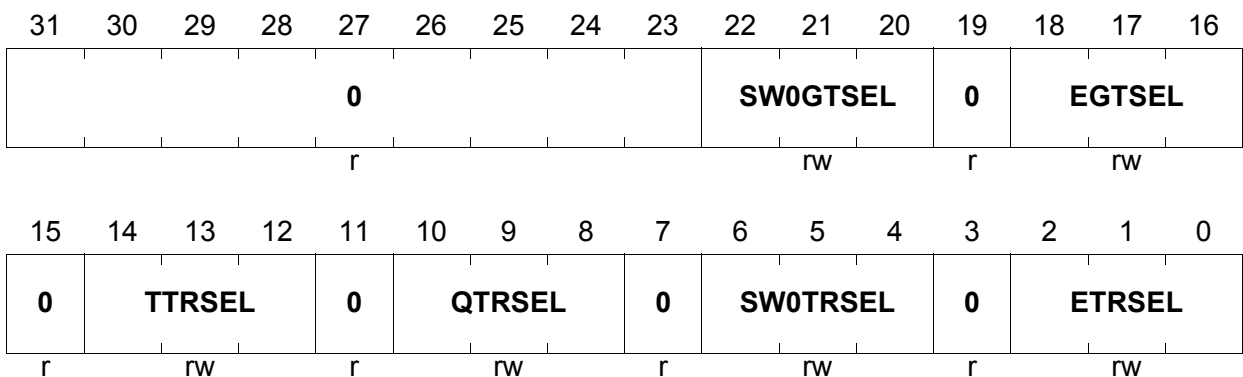
The Trigger Gating ADC0 Register TGADC0 contains bit fields that determine the assignment of the output signals of the external request unit to the trigger and gating inputs of ADC0. The Trigger Gating ADC1 Register TGADC1 contains bit fields that determine the assignment of the output signals of the external request unit to the trigger and gating inputs of ADC1. Suffix “x” (x = 0,1) in the bit descriptions refers to the corresponding A/D converter, ADC0 or ADC1.

TGADC0

Trigger Gating ADC0 Register (F000009C_H) Reset Value: 0000 0000_H

TGADC1

Trigger Gating ADC1 Register (F00000A0_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
ETRSEL	[2:0]	rw	<p>External Trigger Request Selection</p> <p>This bit determines which trigger source will be used for the external trigger request input ETR of ADCx.</p> <p>000_B ETR is constant at 0 level (trigger function switched off).</p> <p>001_B ROUT0 is connected to ETR.</p> <p>010_B ROUT1 is connected to ETR.</p> <p>011_B ROUT2 is connected to ETR.</p> <p>100_B TOUT0 is connected to ETR.</p> <p>101_B TOUT1 is connected to ETR.</p> <p>110_B TOUT2 is connected to ETR.</p> <p>111_B TOUT3 is connected to ETR.</p>

Field	Bits	Type	Description
SW0TRSEL	[6:4]	rw	<p>SW0 Trigger Request Selection</p> <p>This bit determines which trigger source will be used for the SW0 trigger request input SW0TR of ADCx.</p> <p>000_B SW0TR is constant at 0 level (trigger function switched off).</p> <p>001_B ROUT0 is connected to SW0TR.</p> <p>010_B ROUT1 is connected to SW0TR.</p> <p>011_B ROUT2 is connected to SW0TR.</p> <p>100_B TOUT0 is connected to SW0TR.</p> <p>101_B TOUT1 is connected to SW0TR.</p> <p>110_B TOUT2 is connected to SW0TR.</p> <p>111_B TOUT3 is connected to SW0TR.</p>
QTRSEL	[10:8]	rw	<p>Queue Trigger Request Selection</p> <p>This bit determines which trigger source will be used for the queue trigger request input QTR of ADCx.</p> <p>000_B QTR is constant at 0 level (trigger function switched off).</p> <p>001_B ROUT0 is connected to QTR.</p> <p>010_B ROUT1 is connected to QTR.</p> <p>011_B ROUT2 is connected to QTR.</p> <p>100_B TOUT0 is connected to QTR.</p> <p>101_B TOUT1 is connected to QTR.</p> <p>110_B TOUT2 is connected to QTR.</p> <p>111_B TOUT3 is connected to QTR.</p>
TTRSEL	[14:12]	rw	<p>Timer Trigger Request Selection</p> <p>This bit determines which trigger source will be used for the timer trigger request input TTR of ADCx.</p> <p>000_B TTR is constant at 0 level (trigger function switched off).</p> <p>001_B ROUT0 is connected to TTR.</p> <p>010_B ROUT1 is connected to TTR.</p> <p>011_B ROUT2 is connected to TTR.</p> <p>100_B TOUT0 is connected to TTR.</p> <p>101_B TOUT1 is connected to TTR.</p> <p>110_B TOUT2 is connected to TTR.</p> <p>111_B TOUT3 is connected to TTR.</p>

Field	Bits	Type	Description
EGTSEL	[18:16]	rw	<p>External Gating Selection</p> <p>This bit determines which trigger source will be used for the external gating input EGT of ADCx.</p> <p>000_B EGT is constant at 0 level (externally triggered conversions permanently disabled).</p> <p>001_B PDOUT1 is connected to EGT.</p> <p>010_B PDOUT2 is connected to EGT.</p> <p>011_B PDOUT3 is connected to EGT.</p> <p>100_B EGT is constant at 1 level (externally triggered conversions permanently enabled).</p> <p>101_B <u>PDOUT1</u> is connected to EGT.</p> <p>110_B <u>PDOUT2</u> is connected to EGT.</p> <p>111_B <u>PDOUT3</u> is connected to EGT.</p>
SW0GTSEL	[22:20]	rw	<p>SW0 Gating Selection</p> <p>This bit determines which trigger source will be used for the SW0 gating input SW0GT of ADCx.</p> <p>000_B SW0GT is constant at 0 level (SW0 triggered conversions permanently disabled).</p> <p>001_B PDOUT0 is connected to SW0GT.</p> <p>010_B PDOUT2 is connected to SW0GT.</p> <p>011_B PDOUT3 is connected to SW0GT.</p> <p>100_B SW0GT is constant at 1 level (SW0 triggered conversions permanently enabled).</p> <p>101_B <u>PDOUT0</u> is connected to SW0GT.</p> <p>110_B <u>PDOUT2</u> is connected to SW0GT.</p> <p>111_B <u>PDOUT3</u> is connected to SW0GT.</p>
0	3, 7, 11, 15, 19, [31:23]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

5.4 Special System Interrupts

For some of the possible interrupts in the system, the interrupt control logic is not directly controlled in the module, but via the SCU, e.g. for the FPU interrupts, which are generated in the CPU, but have to be processed outside the CPU.

5.4.1 FPU Interrupts

The FPU provides two interrupts outputs that are activated in case of error conditions:

- Signal FPU_Exception1, which is activated if any of the error flags **including** the inexact flag FX is set
- Signal FPU_Exception2, which is activated if any of the error flags **excluding** the inexact flag FX is set

Both interrupts are combined to one interrupt request output that is controlled by register DMA_SYSSCR0 in the DMA controller. Bit SCU_CON.FIEN determines whether an inexact condition will lead to an interrupt or not.

When an FPU interrupt is generated, the related FPU status flags are latched into the corresponding bits of the SCU Status Register SCU_STAT (see [Page 5-67](#)). Thus, the status of the last floating point instruction that caused an interrupt can be read from SCU_STAT.

The exception status information at the FPU output bus FPU_Exception_PSW is connected to the bits in the SCU_STAT register as shown in [Figure 5-6](#).

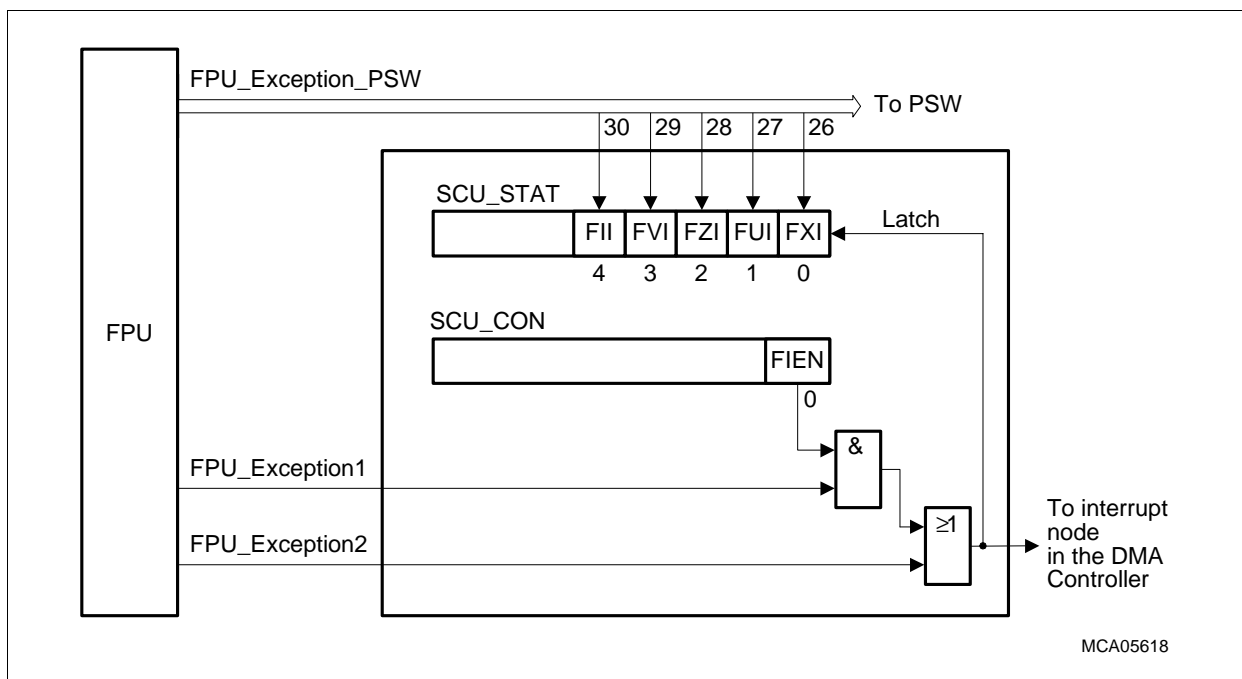


Figure 5-6 FPU Interrupt Control

The FPU interrupt uses the system interrupt node DMA_SYSSRC0 located in the DMA controller. A description of this register can be found on [Page 12-108](#). Note that DMA_SYSSRC0.TOS should be written with 00_B because FPU interrupt should only be serviced by the CPU (and not by the PCP).

5.4.2 Flash Interrupt

The flash module can generate an interrupt when the following conditions occur:

- End-of-busy state
- Protection error
- Sequence error
- Single-bit ECC error

Each source can be individually enabled and disabled. The detailed description of the flash interrupt generation can be found at [“Flash Interrupt Generation and Control” on Page 7-36](#). The Flash interrupt uses the system interrupt node DMA_SYSSRC1 located in the DMA module.

5.4.3 External Interrupts

As shown in [Figure 5-5 on Page 5-15](#), the External Request Unit provides two interrupt sources for events on external pins. The interrupt events can be defined by programming the corresponding registers in the ERU.

The two ERU interrupts are controlled by the system interrupt nodes DMA_SYSSRC2 and DMA_SYSSRC3 located in the DMA module. A description of this register can be found on [Page 12-108](#).

5.5 SRAM Parity Control

In TC1796, several on-chip memory blocks are equipped with a parity error detection logic. Each memory block as defined in [Table 5-5](#) provides a parity error detection logic. This logic asserts a parity error signal when a parity error is detected in the related memory block. An active parity error signal sets a parity error flag, PFLx (x = 0-6). If enabled by the specific parity enable control bit PENx, a non-maskable (NMI) is generated.

The bits PENx (parity error x enable) and PFLx (parity error flag) are located in the registers SCU_PETCR and SCU_PETSR respectively.

Table 5-5 On-chip SRAM with Parity Error Detection

Module	Memory Block	Parity Error Flag	Parity Enable Bit
DMI	Data Memory (LDRAM, DPRAM)	PFL0	PEN0
PMI	Code Scratchpad RAM & Instruction Cache (SPRAM, ICACHE)	PFL1	PEN1
	Program Cache Tag RAM	PFL2	PEN2
DMU	Data Memory (SBRAM, SRAM)	PFL3	PEN3
PCP	Parameter RAM (PRAM)	PFL4	PEN4
	Code Memory (CMEM)	PFL5	PEN5
CAN	Module memories	PFL6	PEN6

After a Boot ROM exit, the parity logic is generally enabled (initial value of SCU_STAT.PARAV = 1), but all specific parity enable control bits PENx are cleared (parity disabled). During normal operation of the TC1796, the SRAM parity error logic can be generally disabled only once (PARAV clear) when the SCU_CON.RPARAV bit is written with 1. Note that if PARAV = 0, the SRAM parity error logic cannot be enabled anymore except by a power-on reset operation.

Before the parity logic of an SRAM memory block can be used the first time after a power-on reset operation (before setting its SCU_PETCR.PENx enable bit), the corresponding memories must be completely initialized by writing every memory location of it once (exceptions: MultiCAN module memories, Program Cache Tag RAM, and ICACHE). Otherwise, unpredictable parity errors may occur after setting its SCU_PETCR.PENx enable bit.

Furthermore, before setting any SCU_PETCR.PENx enable bit after a power-on reset, register SCU_PETSR should be read once to clear parity error flags, that could have been set by parity logic tests during the Boot ROM code execution.

Figure 5-7 shows the functionality of the SRAM parity error control in the SCU.

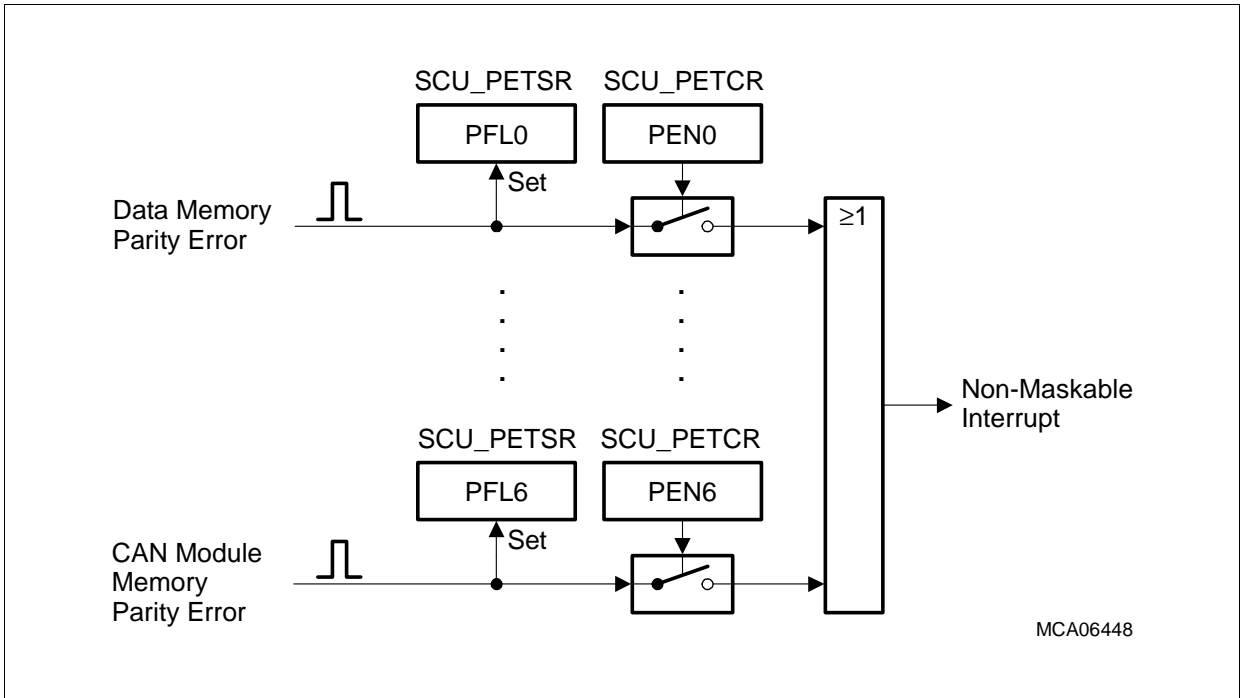


Figure 5-7 Control of SRAM Parity Error Detection in SCU

5.5.1 Parity Error Trap Registers

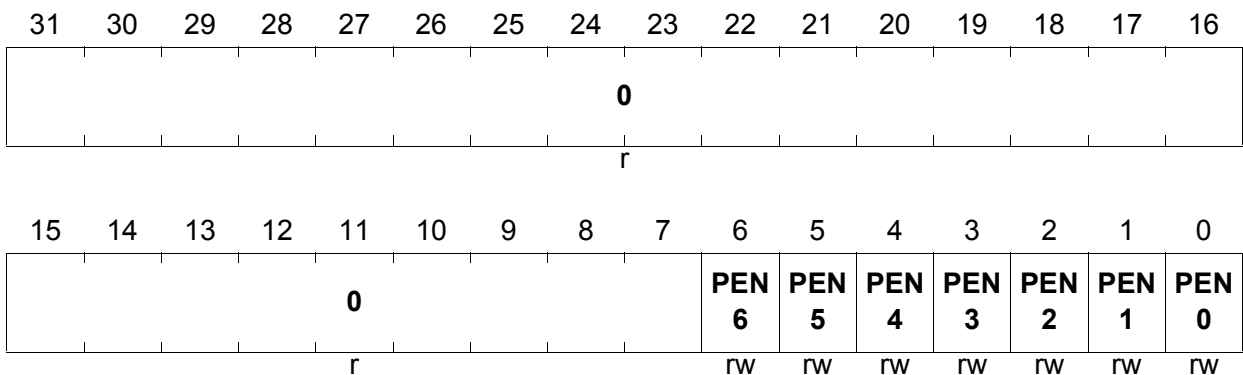
Additional details about the NMI handling of SRAM parity errors are described in section “SRAM Parity Error NMI” on Page 14-26.

SCU_PETCR

SCU Parity Error Trap Control Register

(F0000D0_H)

Reset Value: 0000 0000_H



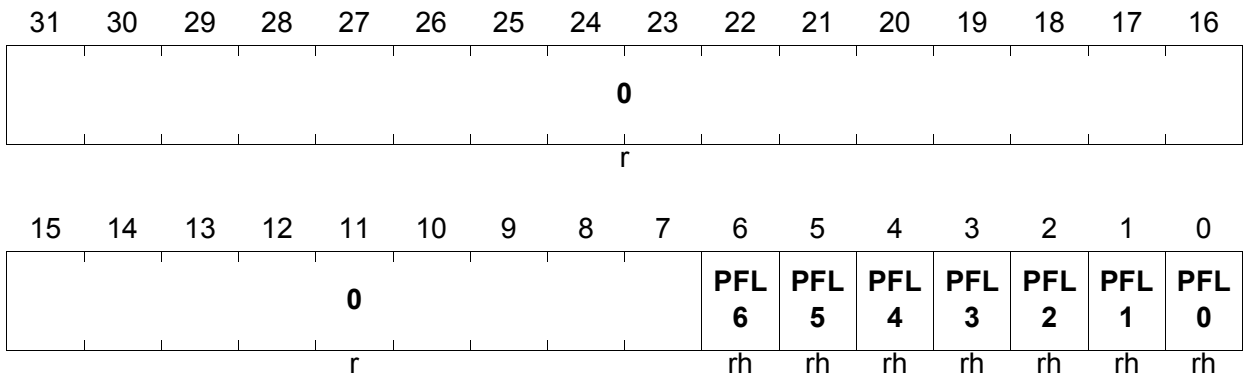
Field	Bits	Type	Description
PENx (x = 0-6)	x	rw	Parity Error Trap Enable for SRAM Module x These bits determine whether an NMI trap is generated if a parity error is detected in the associated SRAM memory module. The assignment of the enable bits to the SRAM modules is defined in Table 5-5 . 0 _B NMI parity error trap is disabled. 1 _B NMI parity error trap is enabled.
0	[31:7]	r	Reserved Read as 0; should be written with 0.

Note: SFR SCU_PETCR is Endinit-protected for write operations.

SCU_PETSR
SCU Parity Error Trap Status Register

(F0000D4_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PFL_x (x = 0-6)	x	rh	Parity Error Flag for SRAM Module x These bits indicate whether a parity error has been detected in the associated SRAM memory module. The assignment of the error flags to the SRAM modules is defined in Table 5-5 . 0 _B No parity error detected. 1 _B Parity error is detected. The PFL _x bits are cleared by hardware after a read access.
0	[31:7]	r	Reserved Read as 0.

Note: SCU_PETSR is a read-only register. Writing to SCU_PETSR results in a bus error.

5.6 Pad Driver Temperature Compensation Control

In the TC1796, two groups of pads can be controlled separately by a temperature compensation control logic:

- Class B1/B2 pads of the EBU interface when used as outputs
- Class A2 pads in strong-driver mode (see [Table 5-6](#) on [Page 5-43](#))

5.6.1 Functional Description

The temperature compensation for the pad output drivers makes it possible to get stable driver output characteristics within dedicated portions of the specified temperature range. As shown in [Figure 5-8](#), the temperature compensation requires a 100 kHz reference input clock signal that is derived from the bus clock f_{SYS} by a programmable divider (TCDIV). This reference input clock f_{REF} is fed into a 12-bit free-running counter (SCOUNT). After each overflow of SCOUNT, the pad oscillator circuit (f_{POSC}), which is located in the pad area, and an 8-bit counter (THCOUNT) are enabled for one clock period of the reference clock. The oscillator circuit is temperature-sensitive and typically has a much higher frequency (8 - 16 MHz) than f_{REF} . The 8-bit counter counts the f_{POSC} pulses and its count value THCOUNT when stopped again is compared against three threshold values stored in THMAX, THMED, and THMIN. These thresholds must be set accordingly to the application needs for proper operation of the temperature compensation.

The temperature compensation can be controlled for two separate groups: All outputs at GPIO ports (Port 0 to 10), and all EBU output lines.

The clock divider TCDIV is programmed via bit field SCU_TCCON.TCDIV. TCDIV can be calculated using the following formula:

$$TCDIV = \text{RoundDown} (2.5 \times f_{SYS}) - 1 \quad (5.1)$$

The resulting divide factor is TCDIV + 1 and leads to an update of the temperature compensation approximately every 41 ms.

Example: for $f_{SYS} = 75$ MHz, $TCDIV = \text{RoundDown} (2.5 \times 75) - 1 = 186_D (= BA_H)$.

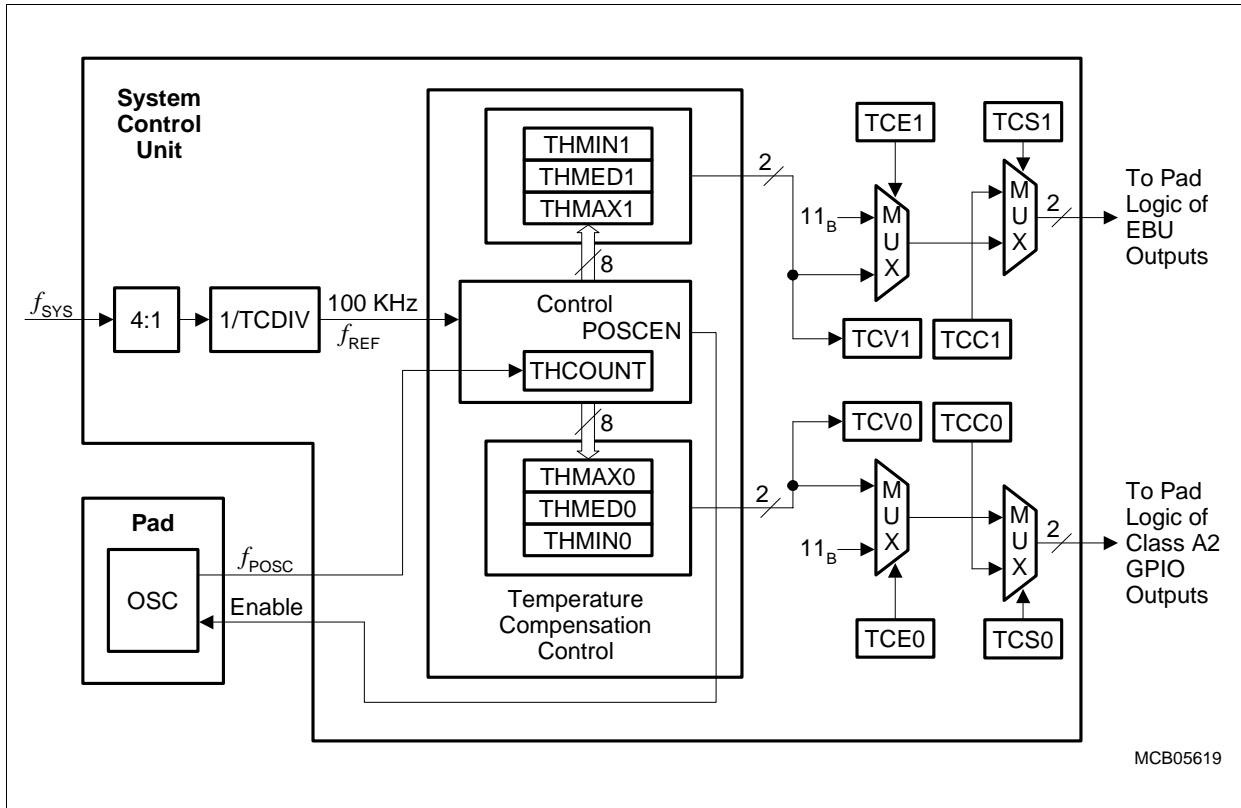


Figure 5-8 Pad Driver Temperature Compensation Block Diagram

Generally, temperature compensation is a transparent feature. Bit field SCU_TCLR_x.THCOUNT ($x = 0, 1$) of both temperature compensation x level registers provides direct access to the actual compensation (counter) value and allows software adjustment control of the port temperature compensation logic. This is useful for two situations:

1. **Device testing:** The function of the compensation mechanism can be verified during production testing or characterization.
2. **User control:** During operation the device can be controlled via externally provided compensation values rather than via the internal mechanism.

Temperature compensation is initialized by programming register SCU_TCCON (enable and prescaler for 100 kHz reference clock) and register SCU_TCLR0 and SCU_TCLR1 (levels values).

The temperature-dependent counter value THCOUNT is compared against three thresholds values (correspond to a certain temperature level) at which the 2-bit output signals to the pad logic are switched (see [Figure 5-9](#)). The three threshold values are defined by three bit fields in the port temperature compensation 0/1 level registers SCU_TCLR0 and SCU_TCLR1. The pads support different driving levels and are controlled via the 2-bit output signals.

The switching thresholds are evaluated hierarchically (see [Table 5-6](#)). For proper operation the relationship $THMIN > THMED > THMAX$ must be true. With increasing temperature, the compensation value `SCU_TCLRx.THCOUNT` decreases.

Table 5-6 Switching Threshold Hierarchy

Output Driver Control	Relationship between THCOUNT and THMAX, THMED, THMIN
Maximum Level	$THCOUNT < THMAX$
High Level	$THMED \geq THCOUNT > THMAX$
Low Level	$THMIN \geq THCOUNT > THMED$
Minimum Level	$THCOUNT > THMIN$

Note: The reset value FF_H for the thresholds ensures that the drivers operate on maximum level if the threshold values have not been initialized.

Note: Reprogramming of the threshold levels in registers `SCU_TCLR0` and `SCU_TCLR1` should only be executed while the temperature compensation is disabled.

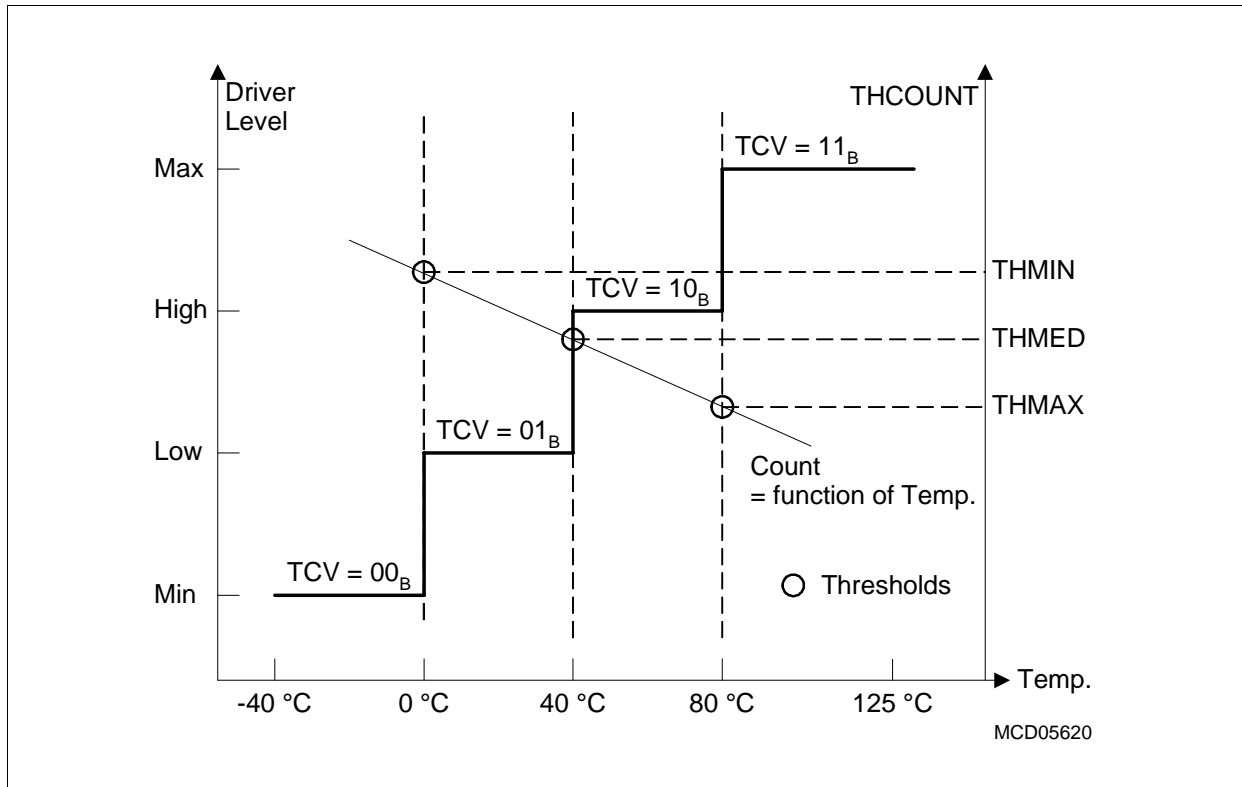


Figure 5-9 Temperature Compensation Switching Thresholds

5.6.2 Temperature Compensation Registers

Note: Control/status bits with index 0 are assigned for output control of class A2 GPIO output lines. Control/status bits with index 1 are assigned for EBU output control.

SCU_TCCON

SCU Temperature Compensation Control Register

(F000048_H)

Reset Value: 0003 0003_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								TCS 1	0	TCC 1	TCE 1	0	TCV 1		
r								rw	r	rw	rw	r	rh		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCDIV								TCS 0	0	TCC 0	TCE 0	0	TCV 0		
rw								rw	r	rw	rw	r	rh		

Field	Bits	Type	Description
TCV0	[1:0]	rh	<p>Temperature Compensation 0 Value</p> <p>This bit field indicates the compensation value that is generated for the temperature compensation logic 0. TCV0 is fed to the temperature compensation logic 0 outputs while bit TCS0 = 0.</p> <p>00_B Minimum driving strength required, i.e. very low temperature.</p> <p>..._B ...</p> <p>11_B Maximum driving strength required, i.e. very high temperature (default after reset).</p>
TCE0	3	rw	<p>Temperature Compensation 0 Enable</p> <p>0_B Temperature compensation logic 0 is deactivated (default after reset). The port drivers are at maximum driver level.</p> <p>1_B Temperature compensation logic 0 is active. THCOUNT is periodically updated.</p>
TCC0	[5:4]	rw	<p>Temperature Compensation 0 Control</p> <p>This 2-bit value is fed to the temperature compensation logic 0 outputs while bit TCS0 = 1. Encoding of TCC0 is equal to TCV0.</p>

System Control Unit

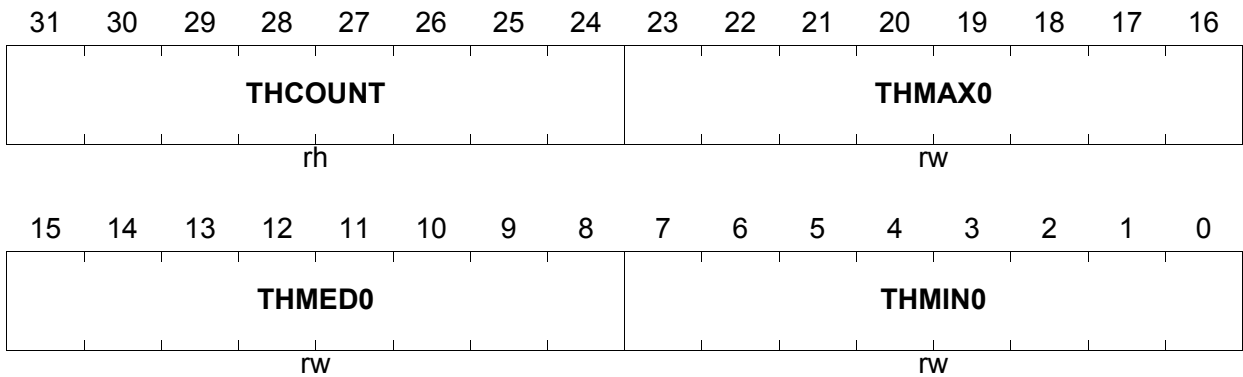
Field	Bits	Type	Description
TCS0	7	rw	Temperature Compensation 0 Source 0_B Temperature compensation logic 0 is controlled by the temperature compensation control hardware. 1_B Temperature compensation logic 0 is controlled by software via bit field TCC0.
TCDIV	[15:8]	rw	Temperature Compensation Clock Divider This value controls the input clock divider for the temperature compensation logic.
TCV1	[17:16]	rh	Temperature Compensation 1 Value This bit field indicates the compensation value which is generated for the temperature compensation logic 1. TCV0 is fed to the temperature compensation logic 0 outputs while bit TCS1 = 0. 00_B Minimum driving strength required, i.e. very low temperature. \dots_B ... 11_B Maximum driving strength required, i.e. very high temperature (default after reset).
TCE1	19	rw	Temperature Compensation 1 Enable 0_B Temperature compensation logic 1 is deactivated (default after reset). The port drivers are at maximum driver level. 1_B Temperature compensation logic 1 is active. THCOUNT is periodically updated.
TCC1	[21:20]	rw	Temperature Compensation 1 Control This 2-bit value is fed to the temperature compensation logic 1 outputs while bit TCS1 = 1. Encoding of TCC1 is equal to TCV1.
TCS1	23	rw	Temperature Compensation 1 Source 0_B Temperature compensation logic 1 is controlled by the temperature compensation control hardware. 1_B Temperature compensation logic 1 is controlled by software via bit field TCC1.
0	2, 6, 18, 22, [31:24]	r	Reserved Read as 0; should be written with 0.

System Control Unit

SCU_TCLR0

**SCU Temperature Compensation 0 Level Register
(F000058_H)**

Reset Value: 02FF FFFF_H



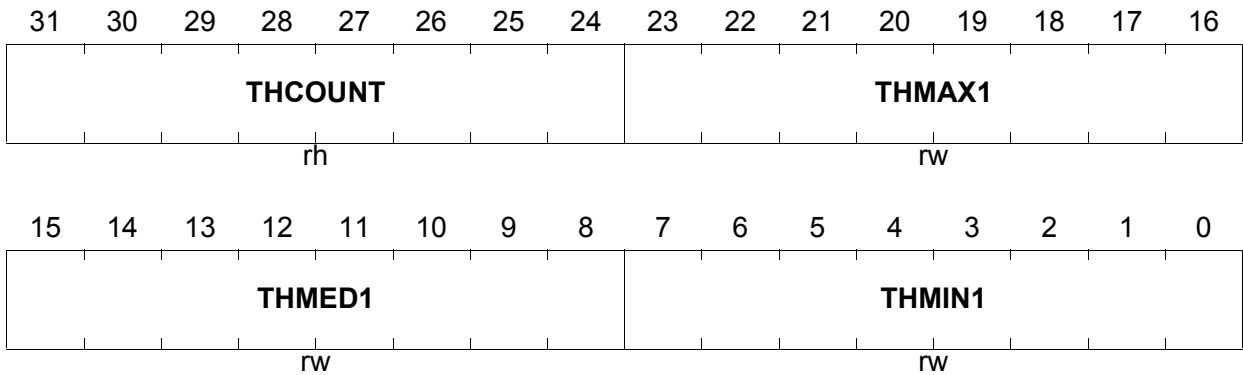
Field	Bits	Type	Description
THMIN0	[7:0]	rw	Minimum Threshold for Temp. Compensation 0 Driver level = Low or Min
THMED0	[15:8]	rw	Medium Threshold for Temp. Compensation 0 Driver level = High or Low
THMAX0	[23:16]	rw	Maximum Threshold for Temp. Compensation 0 Driver level = Max or High
THCOUNT	[31:24]	rh	Threshold Counter Returns the actual count value of the counter that counts the f_{POSC} clock pulses. THCOUNT is only updated when temperature compensation is enabled (TCE0 or TCE1 or both set).

SCU_TCLR1

SCU Temperature Compensation 1 Level Register

(F000005C_H)

Reset Value: 02FF FFFF_H



Field	Bits	Type	Description
THMIN1	[7:0]	rw	Minimum Threshold for Temp. Compensation 1 Driver level = Low or Min
THMED1	[15:8]	rw	Medium Threshold for Temp. Compensation 1 Driver level = High or Low
THMAX1	[23:16]	rw	Maximum Threshold for Temp. Compensation 1 Driver level = Max or High
THCOUNT	[31:24]	rh	Threshold Counter Returns the actual count value of the counter that counts the f_{POSC} clock pulses. THCOUNT is only updated when temperature compensation is enabled (TCE0 or TCE1 or both set).

5.7 Die Temperature Sensor

The TC1796 provides an on-chip Die Temperature Sensor (DTS). The output voltage of the DTS can be measured using analog input AIN31 of the ADC1 module. For measuring the DTS output voltage, the following conditions must be met:

- The DTS circuitry has to be enabled by setting `SCU_CON.DTSON = 1`. After enabling the DTS, a settling time of about 10 μ s has to be respected before starting a DTS measurement.
- Bit `ADC1_CHCON15.EMUX[0]` has to be set to 1 for channel 15 of ADC1.
- The analog input AIN31 of ADC1 has to be requested for conversion (corresponding `GRPS = 1`). With this request, the reference voltages of the DTS (V_{AREF_DTS} , V_{AGND_DTS}) and the sensor output signal (V_{DTS}) are connected to ADC1 for AD conversion.

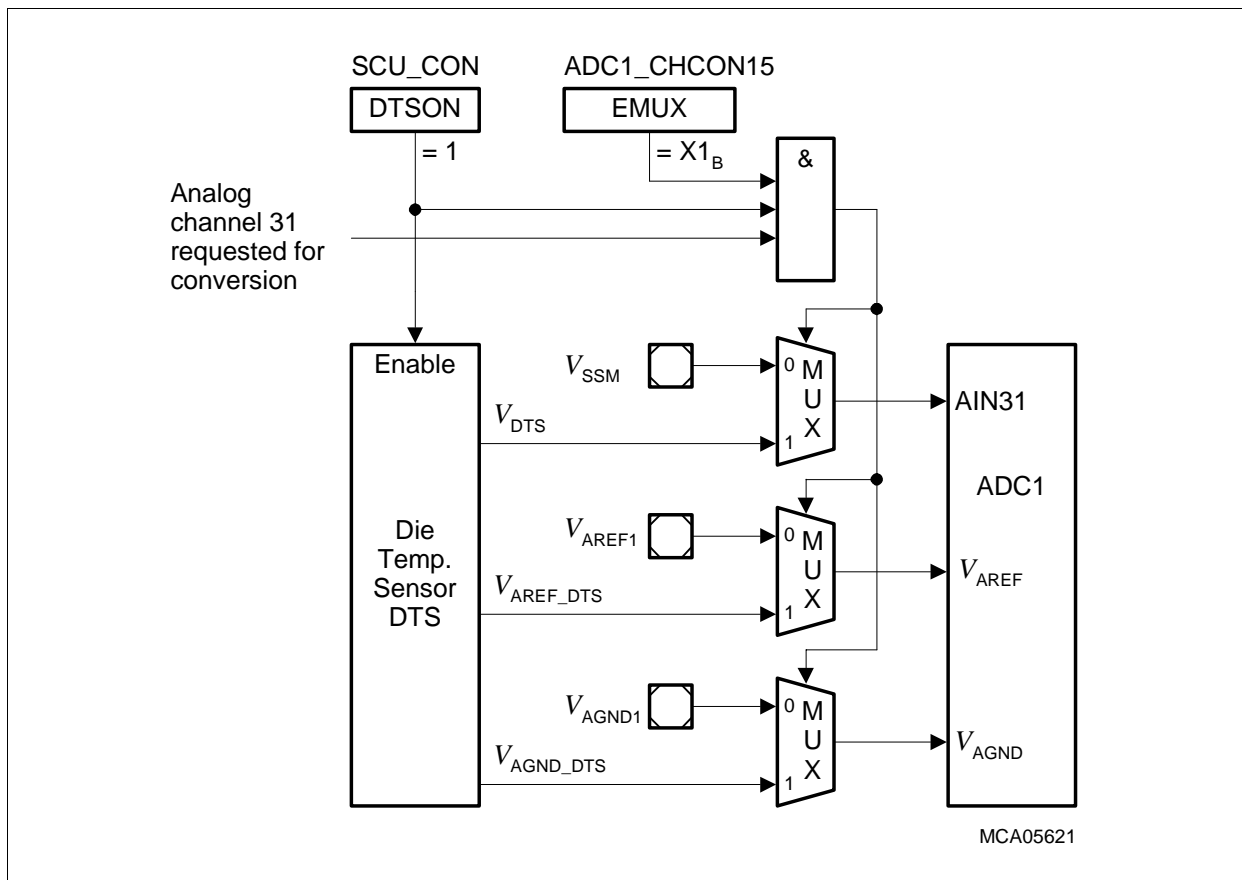


Figure 5-10 Die Temperature Sensor Selection

The die temperature can be determined according to the equation:

$$\text{temp_ao[V]} = 1.0 \text{ V} + (\text{die temperature} + 40 \text{ }^\circ\text{C}) / 190 \quad (5.2)$$

The DTS operates with an accuracy of $\pm 10 \text{ }^\circ\text{C}$.

5.8 GPTA1 Input IN1 Control

In the TC1796, the input line IN1 of the GPTA0, GPTA1, and LTCA2 module can be used to measure the baud rate of an ASC0 or ASC1 receiver input signal with GPTA1. This feature is controlled by a bit of the SCU, SCU_CON.GIN1S. Bit GIN1S controls a 4-to-1 multiplexer that makes it possible to switch several port lines to the IN1 input of the GPTA1 module. After reset, the nominal GPTA input IN1 (P2.9) is connected to IN1 of GPTA.

Table 5-7 GPTA1 Input Line IN1 Connections

SCU_CON.GIN1S	GPTA1 Input IN1 Connected to
00 _B	P2.9 / IN1 (default after reset)
01 _B	P5.0 / RXD0A
10 _B	P6.8 / RXD0B
11 _B	P6.10 / RXD1B

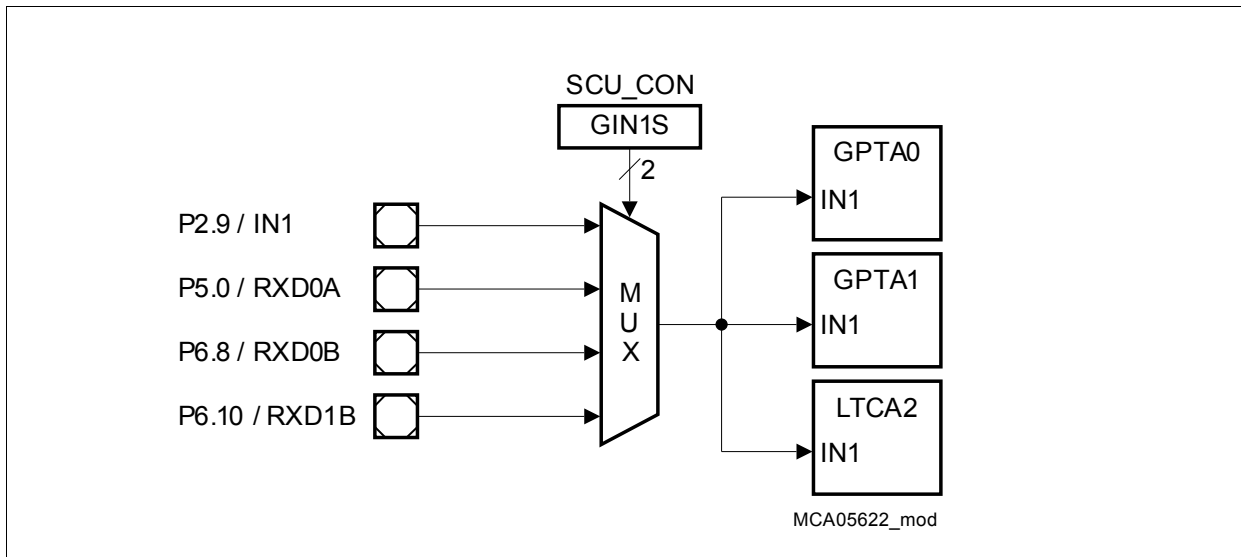


Figure 5-11 GPTA1 Input IN1 Control

5.9 Pad Test Mode Control

The pad test mode control logic in the SCU can be used for in-system tests of board connections for dedicated pins (pins without GPIO functionality). The pad test mode can be enabled in the normal operating mode of the TC1796. A special enable procedure (two-word write sequence) avoids unintentionally enabling the pad test mode.

The pad test mode control logic especially makes it possible to:

- Output a value (low- or high-level) to dedicated pins
- Read the logic level on dedicated pins

Figure 5-12 shows the pad test mode control logic for one pad/pin. Test data is read or written via four pad test data registers. These registers with numbering index n ($n = 0-3$) make it possible to access at maximum sixteen of the dedicated pins, independently from each other.

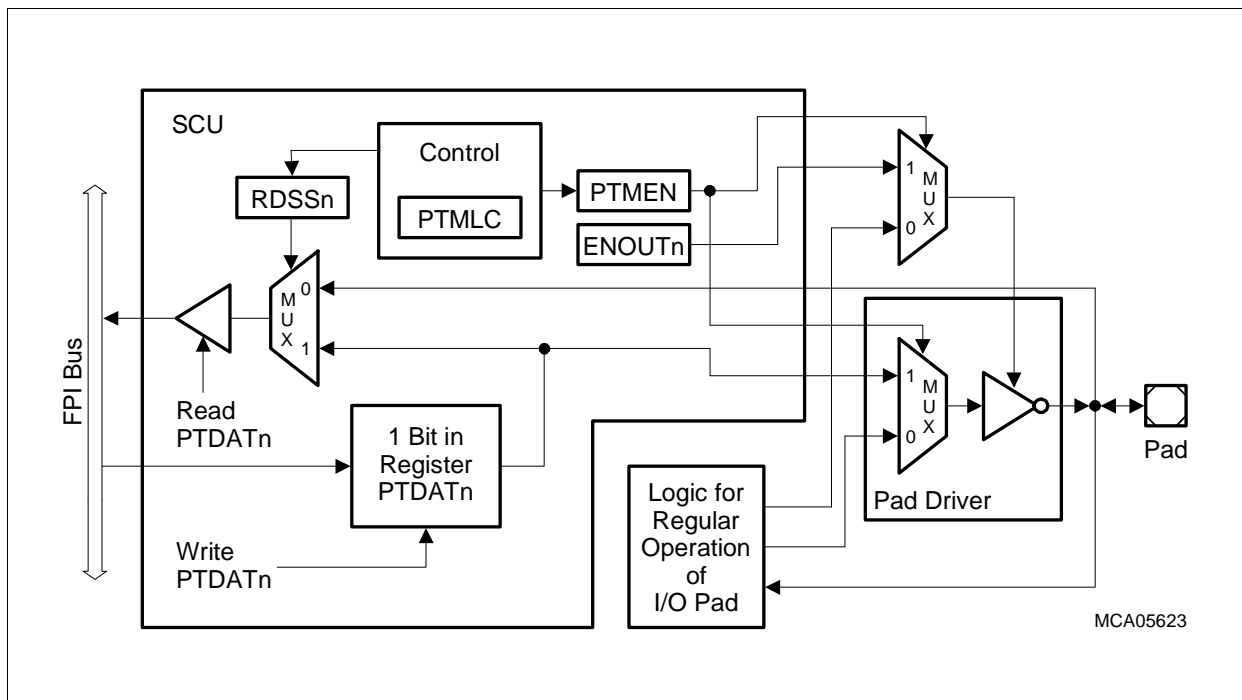


Figure 5-12 Pad Test Mode Control in the SCU

When the pad test mode is disabled ($PTMEN = 0$), the dedicated pads are in their normal operating mode. The pad output drivers are controlled by the hardware part that determines whether the dedicated pad/pin is used for input, output, or for I/O purposes.

When the pad test mode is enabled, $PTMEM$ is set. A value written into register $PTDATn$ is always output at the corresponding pad as inverted state. This means when writing a 1 (0) to a bit of register $PTDATn$, a low (high) level will be available at the corresponding dedicated pin. When reading register $PTDATn$, either the (non-inverted) logic level at the dedicated pad ($RDSSn = 0$) or the value of the $PTDATn$ register bit ($RDSSn = 1$) can be

read back. The ENOUTn bits determine whether or not the logic level state as defined by the bits in the SCU_PTDATn register is output to the pad/pin.

5.9.1 Pad Test Mode Enabling

To enable the pad test mode, the following two-word write sequence must be executed:

1. Writing SCU_PTCON with lock code PTMLC = 5A_H
2. Writing SCU_PTCON with lock code PTMLC = A5_H. After this write operation pad test mode is enabled, as indicated by PTMEN = 1. Bits ENOUTn and RDSSn determine the requested pad test mode configuration (enable, input selection).

When pad test mode is enabled, the test mode configuration (as defined through ENOUTn and RDSSn) can be changed without leaving the pad test mode by writing SCU_PTCON with new values for bits ENOUTn and RDSSn and PTMLC = A5_H. In pad test mode, any other write operation to SCU_PTCON with lock code PTMLC not equal to A5_H terminates the pad test mode.

After pad test mode has been enabled via the two-word write operation to SCU_PTCON, it can be disabled again by any reset operation or a write operation to SCU_PTCON.

5.9.2 Pad Test Mode Registers

The pad test mode control logic contains five registers.

Table 5-8 Pad Test Mode Registers

Register Short Name	Register Long Name	Description see
SCU_PTCON	SCU Pad Test Control Register	Page 5-52
SCU_PTDAT0	SCU Pad Test Data Register 0	Page 5-53
SCU_PTDAT1	SCU Pad Test Data Register 1	Page 5-54
SCU_PTDAT2	SCU Pad Test Data Register 2	Page 5-55
SCU_PTDAT3	SCU Pad Test Data Register 3	Page 5-56

System Control Unit

SCU_PTCON

SCU Pad Test Control Register (F0000B0_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PTM EN		0						0			0		0		
rh		r						rw			rw		rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RD SS3	RD SS2	RD SS1	RD SS0	EN OUT 3	EN OUT 2	EN OUT 1	EN OUT 0	PTMLC							
rw	rw	rw	rw	rw	rw	rw	rw	w							

Field	Bits	Type	Description
PTMLC	[7:0]	w	Pad Test Mode Lock Code This bit field must be written with a special two-word write sequence to enable the pad test mode. PTMLC is always read as 00 _H .
ENOUT_n (n = 0-3)	n + 8	rw	Enable Data Output for Pad Test Data Register n In pad test mode (PTMEN = 1), these bits determine whether or not the logic level state as defined by the bits in the SCU_PTDAT _n register is switched as inverted state to the pad/pin. 0 _B Pad drivers of the pad test data register n related pads are disabled. 1 _B Pad drivers of the pad test data register n related pads are enabled and the bits in the pad test data register n are output as inverted state at the related pins.
RDSS_n (n = 0-3)	n + 12	rw	Read Source Selection for Pad Test Data Register n In pad test mode (PTMEN = 1), these bits determine the read source for the SCU_PTDAT _n register. 0 _B Logic levels of the pad test data register n related pads/pins are read. 1 _B Pad test data register n bits are read.
0	16	rw	Reserved Read as 0 after reset; returns the value that is written.

System Control Unit

Field	Bits	Type	Description
0	17, 18	rw	Reserved Read as 0; bits must be written with 0.
PTMEN	31	rh	Pad Test Mode Enable Flag 0 _B Pad test mode disabled (default after reset) 1 _B Pad test mode enabled
0	[30:19]	r	Reserved Read as 0; should be written with 0.

SCU_PTDAT0

SCU Pad Test Data Register 0 (F0000B4_H) **Reset Value: XXXX XXXX_H**

31		30		29		28		27		26		25		24		23		22		21		20		19		18		17		16	
MR	W	ADV	RD	WR	RD	BC3	BC2	BC1	BC0	A23	A22	A21	A20	A19	A18	A17	A16														
rwh		rwh		rwh		rwh		rwh		rwh		rwh		rwh		rwh		rwh		rwh		rwh		rwh		rwh		rwh			
15		14		13		12		11		10		9		8		7		6		5		4		3		2		1		0	
A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0																
rwh		rwh		rwh		rwh		rwh		rwh		rwh		rwh		rwh		rwh		rwh		rwh		rwh		rwh		rwh			

Field	Bits	Type	Description
An (n = 0-23)	n	rwh	Pad Test Value for/of EBU Address Bus Line An
BC0	24	rwh	Pad Test Value for/of BC0
BC1	25	rwh	Pad Test Value for/of BC1
BC2	26	rwh	Pad Test Value for/of BC2
BC3	27	rwh	Pad Test Value for/of BC3
RD	28	rwh	Pad Test Value for/of RD
RDWR	29	rwh	Pad Test Value for/of RD/WR
ADV	30	rwh	Pad Test Value for/of ADV
MRW	31	rwh	Pad Test Value for/of MR/W

Note: In pad test mode, the bits in SCU_PTDAT0 are output to the pad/pin in inverted state: a 0 generates a high level and a 1 generates a low level at the pad/pin.

System Control Unit

SCU_PT DAT1

SCU Pad Test Data Register 1 (F0000B8_H)

Reset Value: XXXX XXXX_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	BAA	WAIT	CSCOMB	CS3	CS2	CS1	CS0	BREQ	HLDA	HOLD	NMI	HDRST	TRCLK	BRKOUT	BRKIN
r	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rh	rwh	rwh	rwh	rwh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
TRn (n = 0-15)	n	rwh	Pad Test Value for/of TRn
BRKIN	16	rwh	Pad Test Value for/of BRKIN
BRKOUT	17	rwh	Pad Test Value for/of BRKOUT
TRCLK	18	rwh	Pad Test Value for/of TRCLK
HDRST	19	rwh	Pad Test Value for/of HDRST
NMI	20	rh	Pad Test Value of NMI
HOLD	21	rwh	Pad Test Value for/of HOLD
HLDA	22	rwh	Pad Test Value for/of HLDA
BREQ	23	rwh	Pad Test Value for/of BREQ
CS0	24	rwh	Pad Test Value for/of CS0
CS1	25	rwh	Pad Test Value for/of CS1
CS2	26	rwh	Pad Test Value for/of CS2
CS3	27	rwh	Pad Test Value for/of CS3
CSCOMB	28	rwh	Pad Test Value for/of CSCOMB
WAIT	29	rwh	Pad Test Value for/of WAIT
BAA	30	rwh	Pad Test Value for/of BAA
0	31	r	Reserved Reading this bit always returns 0.

System Control Unit

Note: In pad test mode, the bits in SCU_PTDAT1 are output to the pad/pin in inverted state: a 0 generates a high level and a 1 generates a low level at the pad/pin.

SCU_PTDAT2

SCU Pad Test Data Register 2 (F0000BC_H) **Reset Value: XXXX 0XXX_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SLS I0	SCL K0	MR ST0	MT SR0	SLS O1	SLS O0	0									
rwh	rwh	rwh	rwh	rwh	rwh	rw									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						TST RES	TES TMO DE	TMS	TDO	TDI	TCK	T RST	BF CLK O	BF CLK I	BYP ASS
r						rh	rh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
BYPASS	0	rwh	Pad Test Value for/of BYPASS
BFCLKI	1	rwh	Pad Test Value for/of BFCLKI
BFCLKO	2	rwh	Pad Test Value for/of BFCLKO
TRST	3	rwh	Pad Test Value for/of TRST
TCK	4	rwh	Pad Test Value for/of TCK
TDI	5	rwh	Pad Test Value for/of TDI
TDO	6	rwh	Pad Test Value for/of TDO
TMS	7	rwh	Pad Test Value for/of TMS
TESTMODE	8	rh	Pad Test Value of TESTMODE
TSTRES	9	rh	Pad Test Value of TSTRES
0	[15:10]	r	Reserved Reading this bit always returns 0.
0	[25:16]	rw	Reserved Read as 0 after reset; returns the value that is written.
SLSO0	26	rwh	Pad Test Value for/of SLSO0
SLSO1	27	rwh	Pad Test Value for/of SLSO1
MTSR0	28	rwh	Pad Test Value for/of MTSR0
MRST0	29	rwh	Pad Test Value for/of MRST0

System Control Unit

Field	Bits	Type	Description
SCLK0	30	rwh	Pad Test Value for/of SCLK0
SLSI0	31	rwh	Pad Test Value for/of SLSI0

Note: In pad test mode, the bits in SCU_PTDAT2 are output to the pad/pin in inverted state: a 0 generates a high level and a 1 generates a low level at the pad/pin.

SCU_PTDAT3

SCU Pad Test Data Register 3 (F0000C0_H) **Reset Value: XXXX XXXX_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
D31	D30	D29	D28	D27	D26	D25	D24	D23	D22	D21	D20	D19	D18	D17	D16
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
Dn (n = 0-31)	n	rwh	Pad Test Value for/of EBU Data Bus Line Dn

Note: In pad test mode, the bits in SCU_PDAT3 are output to the pad/pin in inverted state: a 0 generates a high level and a 1 generates a low level at the pad/pin.

In synchronous mode (selected by $\text{MODE} = 0$), the HWCFG1 signal is sampled for a inactive-to-active level transition, and an emergency stop flag EMSF is set if the transition is detected. The setting of EMSF activates the EMGSTOP signal. An emergency case can only be terminated by clearing EMSF via software. The synchronous control logic is clocked by the system clock f_{SYS} . This results in a small delay between the HWCFG1 signal and EMGSTOP signal generation. If the system clock is switched off (not applicable in TC1796), the synchronous mode control logic is frozen and not able to react to transitions at input HWCFG1. In this case, the asynchronous mode can be used to put GPTA outputs to dedicated logic levels.

In asynchronous mode (selected by $\text{MODE} = 1$), the occurrence of an active level at input HWCFG1 immediately activates the EMGSTOP signal to the GPTA outputs and the MSC modules even if the synchronous control part is inactive and not clocked by the system clock f_{SYS} . Of course, a valid-to-invalid transition of HWCFG1 (emergency case is released) also immediately deactivates the EMGSTOP signal.

The POL bit determines the active level of the emergency stop input signal HWCFG1. The MODE bit selects synchronous or asynchronous mode for emergency stop signal generation. The EMSF flag can be enabled/disabled for setting (control bit ENON) and it can be set or cleared by software, too (bit field EMSFM).

5.10.1 GPTA Output Emergency Control in the GPIO Ports

The selection of which port line of a GPIO port is affected by an active EMGSTOP signal is done in the Emergency Stop Registers (Pn_ESR), which are located in the port logics. Each of the GPIO lines that can be assigned as GPTA output has its own emergency stop enable bit Pn_ESR.ENx . If the emergency stop signal EMGSTOP becomes active, one of two states can be selected:

- Emergency stop function disabled ($\text{Pn_ESR.ENx} = 0$):
A GPTA output line remains connected to the GPTA module (alternate function).
- Emergency stop function enabled ($\text{Pn_ESR.ENx} = 1$):
A GPTA output line is disconnected immediately from the GPTA module (alternate function) and connected to the corresponding bit of the Pn_OUT output register Pn_OUT.Px .

5.10.2 MSC Emergency Control Selection

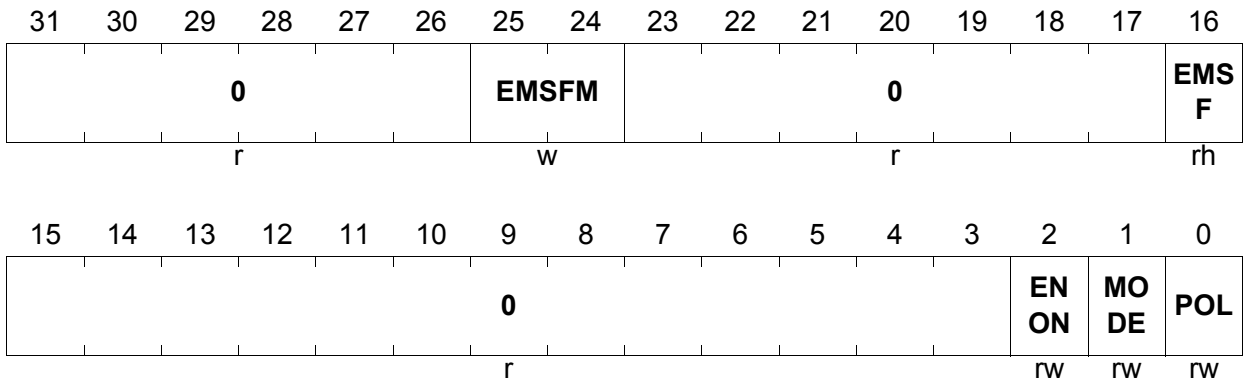
With an active emergency stop signal ($\text{EMGSTOP} = 1$), the two MSC modules are selectively put bits of the next following serial downstream frame(s) to a programmable logic value instead of an actual data value. In an emergency case, the bits in the downstream shift registers SRL/SRH that are enabled for the emergency stop feature (corresponding bits in the emergency stop register ESR set) are loaded with the corresponding bit of the downstream data register DD instead with the bits from the ALTINL/ALTINH data source. Therefore, it takes a calculable time until the transmitted downstream frame with the emergency stop information reaches the receiver device.

5.10.3 Emergency Stop Register

The Emergency Stop Register SCU_EMSR contains control and status bits/flags of the emergency stop input logic.

SCU_EMSR

SCU Emergency Stop Register (F0000044_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
POL	0	rw	Input Polarity This bit determines the polarity of the emergency input line HWCFG1. 0 _B HWCFG1 is high active. 1 _B HWCFG1 is low active.
MODE	1	rw	Mode Selection This bit determines the operating mode of the EMGSTOP signal. 0 _B Synchronous mode selected; EMGSTOP is derived from the state of flag EMSF. 1 _B Asynchronous mode selected; EMGSTOP is directly derived from the state of the input signal HWCFG1.
ENON	2	rw	Enable ON This bit enables the (hardware) setting of flag EMSF by an inactive-to-active level transition of input signal HWCFG1. 0 _B Setting of EMSF by hardware is disabled. 1 _B Setting of EMSF by hardware is enabled.

Field	Bits	Type	Description
EMSF	16	rh	Emergency Stop Flag This bit indicates if an emergency stop condition has occurred. 0 _B An emergency stop has not occurred. 1 _B An emergency stop has occurred and signal EMGSTOP becomes active (if MODE = 0).
EMSFM	[25:24]	w	Emergency Stop Flag Modification This bit field makes it possible to set or clear flag EMSF by software. In case of a simultaneous hardware and software modification request, the hardware operation will be executed. 00 _B EMSF remains unchanged. 01 _B EMSF is set. 10 _B EMSF is cleared. 11 _B EMSF remains unchanged. EMSFM is always read as 00 _B .
0	[15:3], [23:17], [31:26]	r	Reserved Read as 0; should be written with 0.

5.11 Analog Input 7 Testmode

When setting bit SCU_CON.AN7TM, the analog input AN7 is pulled low via a resistor of max. 900 Ω. This feature makes it possible to check the correct operation of the ADC during operation. The conversion result should be near the minimum (depending on the source resistance of the connected sensor).

5.12 SCU Registers and Address Map

This section refers to all registers which are located in the SCU address range. Some of these registers are described in other chapters of this User's Manual. The entries in column entitled "Description see" of [Table 5-9](#) point to the pages on which such registers are described in detail.

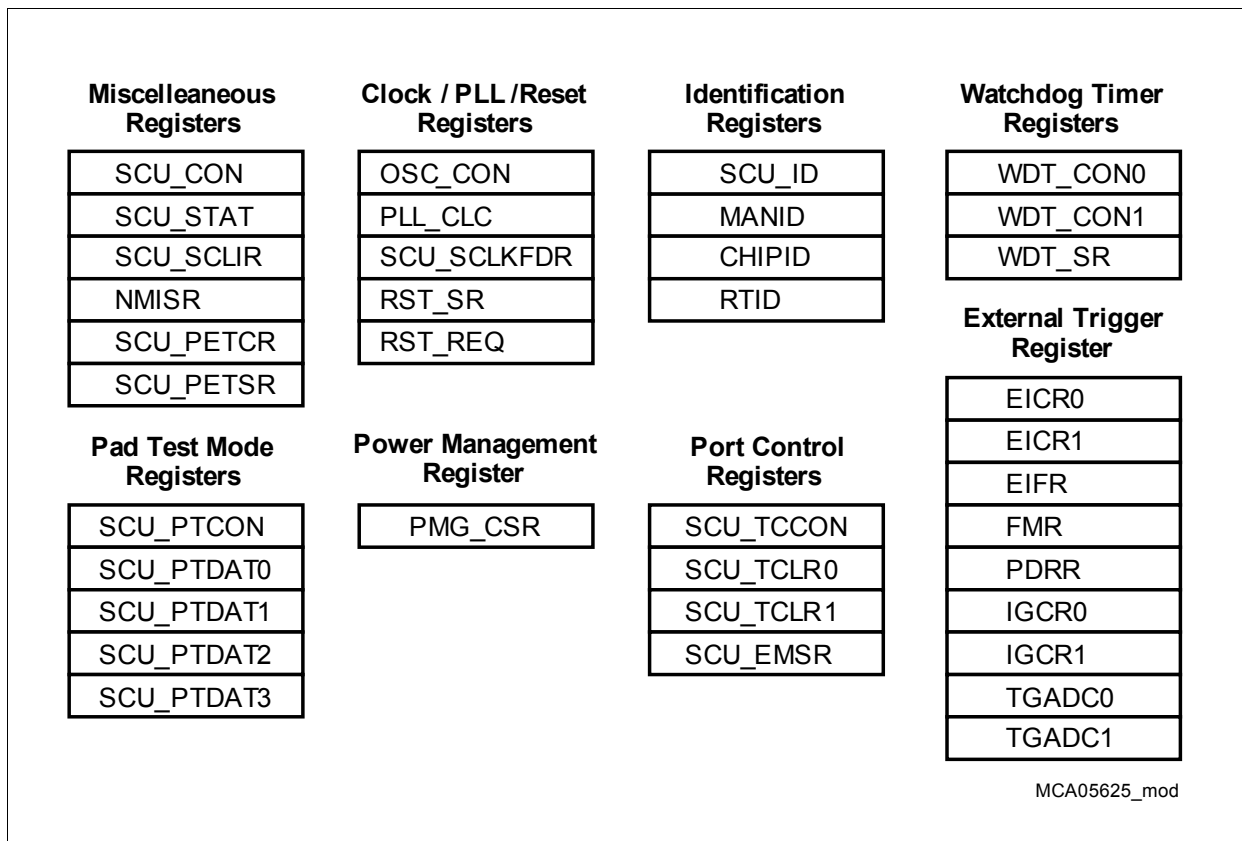


Figure 5-14 SCU Registers

The complete and detailed address map of all SCU registers is described in [Table 18-3](#) on [Page 18-7](#) of the TC1796 User's Manual System Units part (Volume 1).

Table 5-9 SCU Registers

Register Short Name	Register Long Name	Offset Address	Description see
SCU_ID	SCU Module IDentification Register	08 _H	Page 5-69
SCU_SCLKFDR	SCU System Clock Fractional Divider Register	0C _H	Page 3-42
RST_REQ	Reset Request Register	10 _H	Page 4-5
RST_SR	Reset Status Register	14 _H	Page 4-3

System Control Unit

Table 5-9 SCU Registers (cont'd)

Register Short Name	Register Long Name	Offset Address	Description see
OSC_CON	Oscillator Control Register	18 _H	Page 3-8
WDT_CON0	Watchdog Timer Control Register 0	20 _H	Page 16-29
WDT_CON1	Watchdog Timer Control Register 1	24 _H	Page 16-31
WDT_SR	Watchdog Timer Status Register	28 _H	Page 16-32
NMISR	NMI Status Register	2C _H	Page 14-27
PMG_CSR	Power Management Control and Status Register	34 _H	Page 5-4
SCU_SCLR	SCU Software Configuration Latched Inputs Register	38 _H	Page 10-24
PLL_CLC	PLL Clock Control Register	40 _H	Page 3-16
SCU_EMSR	SCU Emergency Stop Register	44 _H	Page 5-59
SCU_TCCON	SCU Temperature Compensation Control Register	48 _H	Page 5-44
SCU_CON	SCU Control Register	50 _H	Page 5-64
SCU_STAT	SCU Status Register	54 _H	Page 5-67
SCU_TCLR0	SCU Temperature Compensation 0 Level Register	58 _H	Page 5-46
SCU_TCLR1	SCU Temperature Compensation 1 Level Register	5C _H	Page 5-47
MANID	Manufacturer Identification Register	70 _H	Page 5-69
CHIPID	Chip Identification Register	74 _H	Page 5-70
RTID	Redesign Tracing Identification Register	78 _H	Page 5-71
EICR0	External Input Channel Register 0	80 _H	Page 5-17
EICR1	External Input Channel Register 1	84 _H	Page 5-20
EIFR	External Input Flag Register	88 _H	Page 5-23
FMR	Flag Modification Register	8C _H	Page 5-24
PDRR	Pattern Detection Result Register	90 _H	Page 5-25
IGCR0	Interrupt Gating Register 0	94 _H	Page 5-26
IGCR1	Interrupt Gating Register 1	98 _H	Page 5-29
TGADC0	Trigger Gating ADC0 Register	9C _H	Page 5-32
TGADC1	Trigger Gating ADC1 Register	A0 _H	Page 5-32

Table 5-9 SCU Registers (cont'd)

Register Short Name	Register Long Name	Offset Address	Description see
SCU_PTCON	SCU Pad Test Control Register	B0 _H	Page 5-52
SCU_PTDAT0	SCU Pad Test Data Register 0	B4 _H	Page 5-53
SCU_PTDAT1	SCU Pad Test Data Register 1	B8 _H	Page 5-54
SCU_PTDAT2	SCU Pad Test Data Register 2	BC _H	Page 5-55
SCU_PTDAT3	SCU Pad Test Data Register 3	C0 _H	Page 5-56
SCU_PETCR	SCU Parity Error Trap Control Register	D0 _H	Page 5-39
SCU_PETSR	SCU Parity Error Trap Status Register	D4 _H	Page 5-40

5.13 Miscellaneous SCU Registers

This section includes descriptions of the following registers:

- SCU Control Register SCU_CON
- SCU Status Register SCU_STAT
- SCU Module Identification Register SCU_ID
- Manufacturer Identification Register MANID
- Chip Identification Register CHIPID
- Redesign Tracing Identification Register RTID

5.13.1 SCU Control Register

SCU_CON

SCU Control Register

(F000050_H)

Reset Value: FF00 0002_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ONE								ZERO				GIN1S		SSC 0 PDR	SLS PDR
rw								rw				rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		RPA RAV		LD EN	DTS ON	0		AN7 TM	NMI EN	EPU D	CS GEN	CS OEN	CS EEN	FIEN	
r		rw		rw	rw	r		rw	rws	rw	rw	rw	rw	rw	

Field	Bits	Type	Description
FIEN	0	rw	FPU Inexact Interrupt Enable 0 _B Inexact error condition (setting FX flag) after a FPU calculation will not generate an interrupt. 1 _B Inexact error condition (setting FX flag) after a FPU calculation will generate an interrupt.
CSEEN	1	rw	CSEMU Enable 0 _B <u>CSEMU</u> will not activate <u>CSCOMB</u> 1 _B CSEMU will activate CSCOMB (default after reset) See also Figure 13-10 on Page 13-29 .

System Control Unit

Field	Bits	Type	Description
CSOEN	2	rw	<p>CSOVL Enable</p> <p>0_B CSOVL will not activate $\overline{\text{CSCOMB}}$ (default after reset)</p> <p>1_B CSOVL will activate $\overline{\text{CSCOMB}}$</p> <p>See also Figure 13-10 on Page 13-29.</p>
CSGEN	3	rw	<p>CSGLB Enable</p> <p>0_B CSGLB will not activate $\overline{\text{CSCOMB}}$ (default after reset).</p> <p>1_B CSGLB will activate $\overline{\text{CSCOMB}}$.</p> <p>See also Figure 13-10 on Page 13-29.</p>
EPUD	4	rw	<p>EBU Pull-up Disable</p> <p>0_B Pull-up resistors are enabled (default after reset).</p> <p>1_B Pull-up resistors are disabled.</p>
NMIEN	5	rws	<p>NMI Enable</p> <p>0_B NMI is disabled (default).</p> <p>1_B NMI is enabled.</p> <p>This bit is cleared with any reset. It can be only set by software and will remain in this state until the next reset. Writing a zero to this bit has no effect. The NMI is described in detail at Section 14.10 on Page 14-25.</p>
AN7TM	6	rw	<p>Analog Input 7 Test Mode</p> <p>0_B Pull down of analog input 7 is disabled (default after reset).</p> <p>1_B Pull down of analog input 7 is enabled.</p>
DTSON	9	rw	<p>Die Temperature Sensor On</p> <p>0_B Die temperature sensor is switched off.</p> <p>1_B Die temperature sensor is switched on.</p>
LDEN	10	rw	<p>LVDS Driver Enable</p> <p>0_B The LVDS drivers are disabled and in power-down mode.</p> <p>1_B The LVDS drivers are enabled.</p> <p>See also Page 10-85.</p>
RPARAV	11	rw	<p>Reset SRAM Parity Available Bit</p> <p>0_B No action</p> <p>1_B Clear bit SCU_STAT.PARAV</p> <p>This bit is always read as 0.</p> <p>See also “SRAM Parity Control” on Page 5-37.</p>

System Control Unit

Field	Bits	Type	Description
SLSPDR	16	rw	SLSO0/SLSO1 Pad Driver Strength Selection 0 _B Strong driver, sharp edge selected. 1 _B Strong driver, soft edge selected. See also Page 10-83 .
SSC0PDR	17	rw	SSC0 Pad Driver Strength Selection 0 _B Strong driver, sharp edge selected 1 _B Strong driver, soft edge selected See also Page 10-83 .
GIN1S	[19:18]	rw	GPTA Input1 Source Select 00 _B IN1 of GPTA input array 01 _B RxD0A 10 _B RxD0B 11 _B RxD1B Details of GIN1S control are described in Section 5.8 on Page 5-49 .
ZERO	[23:20]	rw	Spare 0 Control Bits This bit field contains bits that are reserved for future SCU control tasks. Field ZERO is set to 00 _H after reset. ZERO bits should be written with 00 _H . Reading ZERO bits will return the value last written.
ONE	[31:24]	rw	Spare 1 Control Bits This bit field contains bits that are reserved for future SCU control tasks. Field ONE is set to FF _H after reset. ONE bits should be written with FF _H . Reading ONE bits will return the value last written.
0	[8:7], [15:12]	r	Reserved Read as 0; should be written with 0.

5.13.2 SCU Status Register

The SCU Status Register SCU_STAT holds the status bits for the FPU interrupt (see [Section 5.4.1 on Page 5-35](#)).

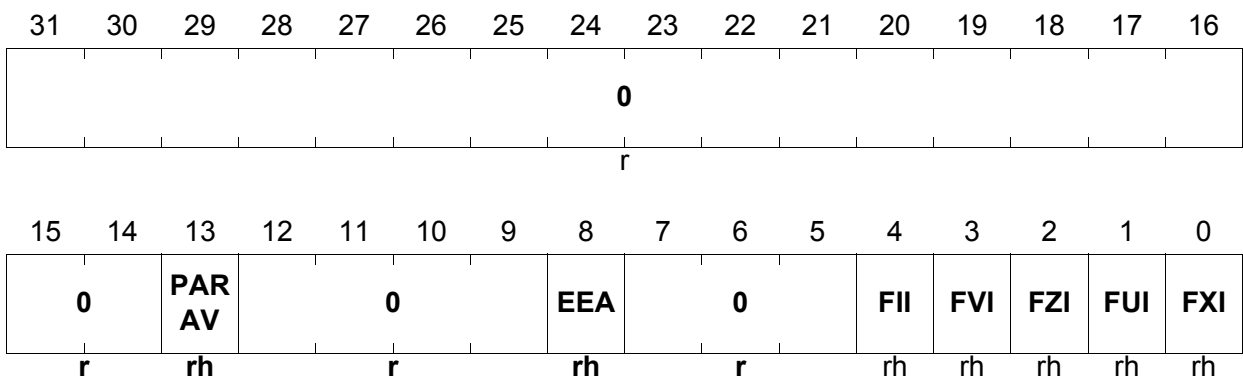
The Boot ROM code is always executed after every reset operation. Depending on the program flow through the Boot ROM code and the Boot ROM exit path, several resources of the TC1796 on-chip hardware have been used and are programmed. This means that the state of on-chip hardware resources that have been used by the Boot ROM code, may differ from the device reset state as described by the register reset values. This section describes which initial state of the on-chip hardware resources is left after a specific Boot ROM exit.

SCU_STAT

SCU Status Register

(F000054_H)

Reset Value: 0000 E000_H¹⁾



1) The initial value of SCU_STAT after Boot ROM exit is 0000 2000_B.

Field	Bits	Type	Description
FXI	0	rh	FPU Inexact Result Indication Flag Indicates the state of the FPU's FX status flag latched during the last FPU interrupt.
FUI	1	rh	FPU Underflow Error Indication Flag Indicates the state of the FPU's FU status flag latched during the last FPU interrupt.
FZI	2	rh	FPU Divide by Zero Error Indication Flag Indicates the state of the FPU's FZ status flag latched during the last FPU interrupt.
FVI	3	rh	FPU Overflow Error Indication Flag Indicates the state of the FPU's FV status flag latched during the last FPU interrupt.

System Control Unit

Field	Bits	Type	Description
FII	4	rh	FPU Invalid Operation Error Indication Flag Indicates the state of the FPU's FI status flag latched during the last FPU interrupt.
EEA	8	rh	Emulation Extension Available Indicates if the emulation extension (= 1796ED) is available or not (= production device) 0 _B EEC is not available. 1 _B EEC is available.
PARAV	13	rh	Parity Available 0 _B SRAM parity logic is disabled. 1 _B SRAM parity logic is enabled. This bit is set with any power-on reset. It can be cleared by software by writing bit SCU_CON.RPARAV. It cannot be set by software. More details about PARAV are described below Table 5-5 on Page 5-37 .
0	[7:5], [12:9], [31:14]	r	Reserved Read as 0.

5.13.3 Device Identification Registers

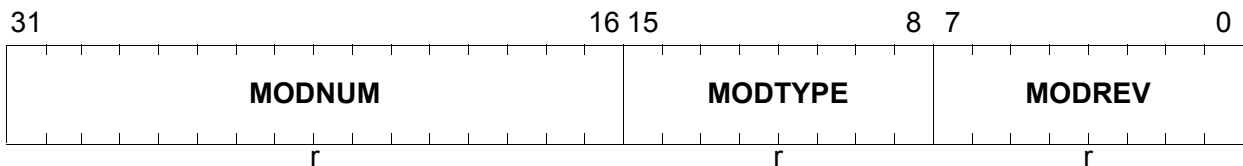
The SCU Module Identification Register ID contains read-only information about the SCU module version.

SCU_ID

SCU Module Identification Register

(F0000008_H)

Reset Value: 002C C0XX_H



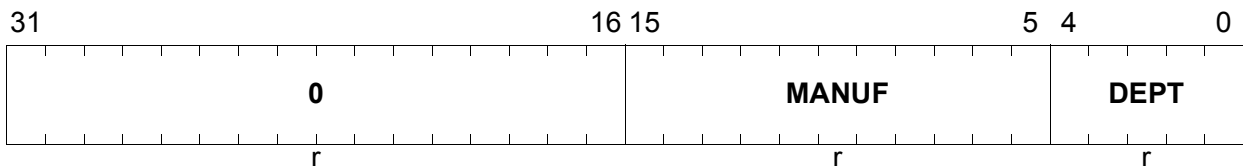
Field	Bits	Type	Description
MODREV	[7:0]	r	Module Revision Number MODREV defines the module revision number. The value of a module revision starts with 01 _H (first revision).
MODTYPE	[15:8]	r	Module Type This bit field defines the module as a 32-bit module: C0 _H
MODNUM	[31:16]	r	Module Number Value This bit field defines the module identification number for the SCU: 002C _H

MANID

Manufacturer Identification Register

(F0000070_H)

Reset Value: 0000 1820_H



Field	Bits	Type	Description
DEPT	[4:0]	r	Department Identification Number = 00 _H : indicates the Automotive & Industrial microcontroller department within Infineon Technologies.

System Control Unit

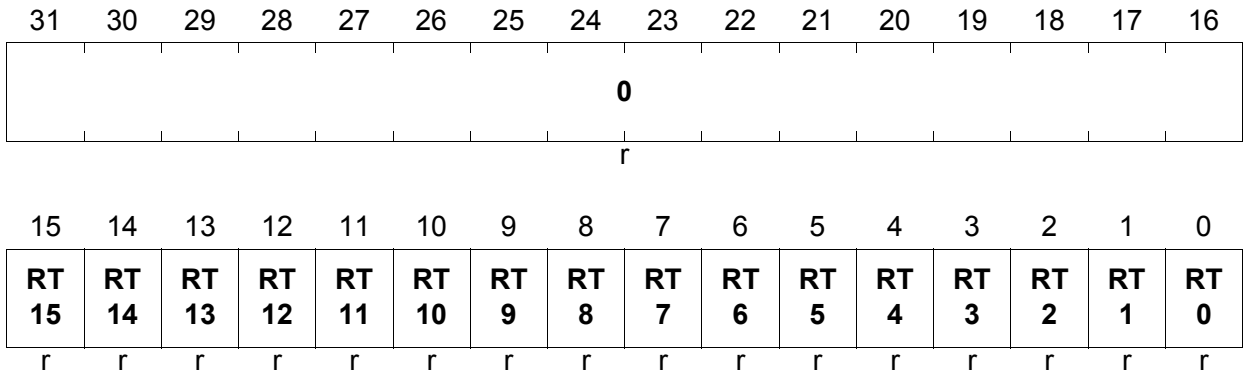
The Redesign Tracing Register RTID provides a means of signalling minor redesigns that are not reflected in the CHIPID.CHREV bit field.

RTID

Redesign Tracing Identification Register

(F0000078_H)

Reset Value: 0000 XXXX_H



Field	Bits	Type	Description
RTn (n = 0-15)	n	r	Redesign Trace Bit n 0 _B No change indicated 1 _B A change has been made (without changing bit field CHIPID.CHREV). RTn can be used, e.g., for minor redesign stepping identification purposes.
0	[31:16]	r	Reserved Read as 0.

Note: The RTID reset value for a major design step (without modifications) is 0000_H.

6 On-Chip System Buses and Bus Bridges

The TC1796 has four independent on-chip buses

- Program Local Memory Bus (PLMB)
- Data Local Memory Bus (DLMB)
- System Peripheral Bus (SPB)
- Remote Peripheral Bus (RPB)

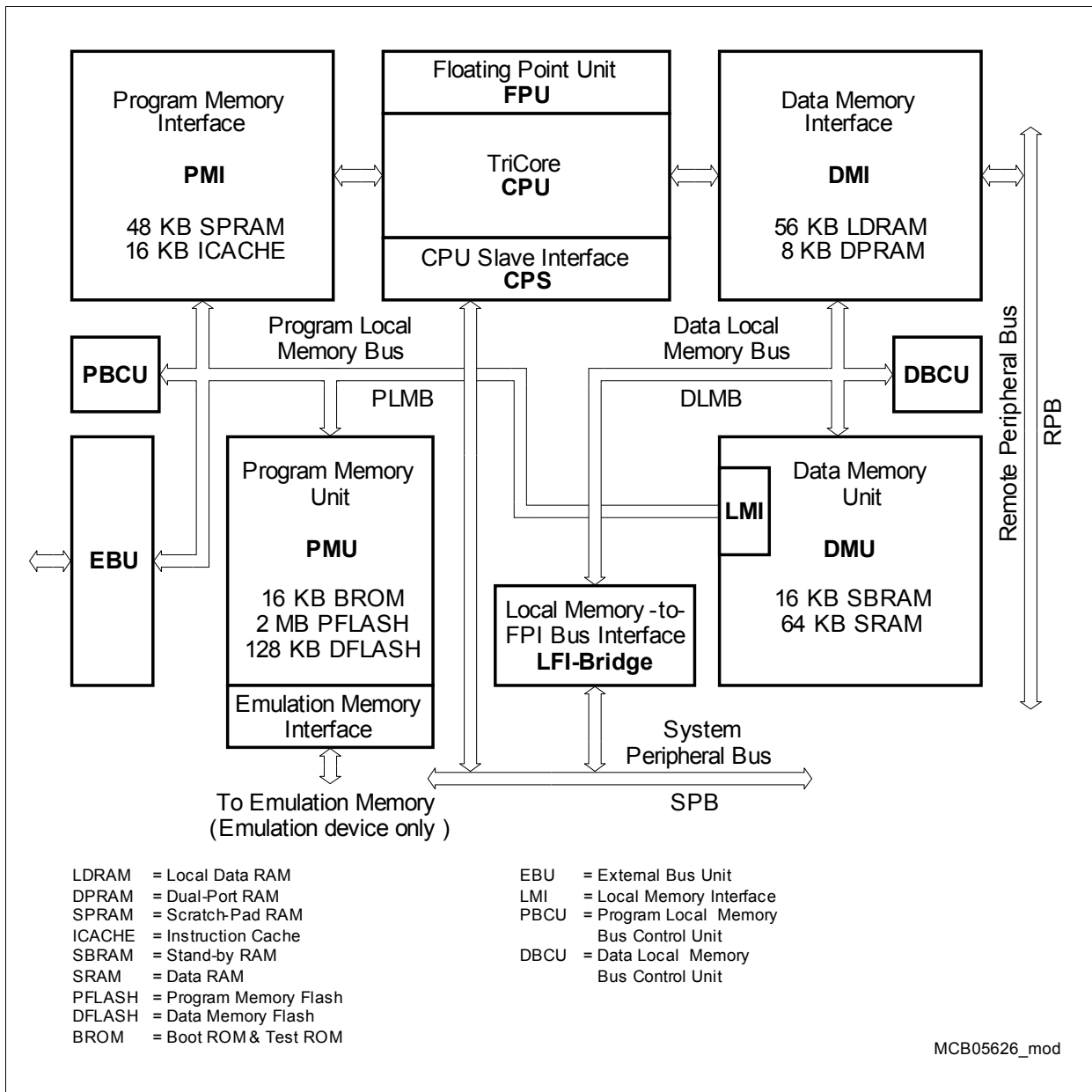


Figure 6-1 Buses in TC1796 Processor Subsystem

The two LMBs (PLMB and DLMB) connect the TriCore CPU to its local resources for instruction fetches and data accesses.

On-Chip System Buses and Bus Bridges

The SPB is accessible by the CPU via the LFI Bridge.

The RPB connects the peripherals with high data rates (SSC, ADC, FADC) to the Dual-port memory (DPRAM) in the DMI, relieving the SPB and the two LMBs from these data transfers. The RPB is controlled by a bus switch which is located in the DMA controller.

The two LMBs (PLMB and DLMB) run at CPU clock speed f_{CPU} , whereas SPB and RPB run at system clock speed f_{SYS} . Note that f_{SYS} can be equal to f_{CPU} or half the f_{CPU} frequency, but f_{SYS} is limited to max. 75 MHz.

6.1 Program and Data Local Memory Buses

The PLMB and DLMB are identical LMBs especially designed for the 32-bit TriCore system technology. Both buses operate in the same manner. The following terminology is used for these buses:

Table 6-1 LMB Terms

Term	Description
Agent	An LMB agent is any master or slave device which is connected to the LMB.
Master	An LMB master device is an LMB agent which is able to initiate transactions on the LMB. It is also able to react as a LMB slave.
Slave	An LMB slave device is an LMB agent which is not able to initiate transactions on the LMB. It is only able to handle operations that are dedicated to it by a LMB master device.

6.1.1 Overview

The LMB is a synchronous and pipelined bus with variable block size transfer support. The protocol supports 8-, 16-, 32-, and 64-bit single transactions and 2/4 wide 64-bit block transfers.

The LMB has the following features:

- Optimized for high speed and high performance data transfers
- 32-bit address bus, 64-bit data bus
- Simple central arbitration per cycle
- Slave-controlled wait state insertion
- Address pipelining (max depth - 2)
- Support of atomic operations LDMST, ST.T and SWAP.W
- Block transfers with variable block length (two or four 64-bit data transactions)

6.1.2 Transaction Types

6.1.2.1 Single Transfers

Single transfers are all transactions that are initiated by any instruction (code or data) of the TriCore 1 CPU and that require a system resource which not part of the TriCore 1 PMI or DMI. The only exceptions are the following instructions:

- LDMST, ST.T and SWAP.W generate atomic transfers
- Cache miss instructions generate block transfers

6.1.2.2 Block Transfers

Block transfers are only issued in two ways:

1. By the PMI in case of a cache miss.
2. By the PCP if it uses a BCOPY instruction.

Block transfers work in the same way as single transfers, except that only one address phase with two or four data phases is generated.

6.1.2.3 Atomic Transfers

Atomic transfers are initiated by instructions that require two single transfers (e.g. read-modify-write instructions such as LDMST, ST.T and SWAP.W). During an atomic transfer, any other LMB master is blocked for gaining bus ownership.

6.1.3 Address Alignment Rules

Depending on the data size, there are rules that determine the address alignment of an LMB transfer.

1. Byte accesses must be always located on byte address boundaries.
2. Half-word accesses must be aligned to addresses with address line $A_0 = 0$.
3. Word accesses must be aligned to addresses with address lines $A[1:0] = 00_B$.
4. Double-word accesses must be aligned to addresses with address lines $A[2:0] = 000_B$.
5. Block transfers must be aligned identical as double-word addresses.

6.1.4 Reaction of a Busy Slave

If an LMB slave is busy at an incoming LMB transaction request, it can delay the execution of the LMB transaction. The requesting LMB master releases the LMB for one cycle after the LMB transaction request in order to allow the LMB slave to indicate if it is ready to handle the requested LMB transaction.

On-Chip System Buses and Bus Bridges

Note: For the LMB default master, the one cycle gap does not result in a performance loss because it is granted the LMB in this cycle as default master if no other master requests the LMB for some other reasons.

6.1.5 LMB Basic Operation

Figure 6-2 describes some basic bus operations of the LMB.

Bus Cycle	1	2	3	4	5
Transfer 1	Request / Grant	Address Cycle	Data Cycle		
Transfer 2		Request / Grant	Address Cycle	Data Cycle	
Transfer 3		Request / Grant		Address Cycle	Data Cycle

MCA05628_mod

Figure 6-2 Basic LMB Transactions

Transfer 1 displays the three cycles of any LMBs transaction:

1. **Request/Grant Cycle:** The LMB master attempts to perform a read or write transfer and requests for the LMB. If the LMB is available, it is granted in the same cycle by the LMB bus controller.
2. **Address Cycle:** After the request/grant cycle, the master puts the address on the LMB and all LMB slave devices check whether they are addressed for the following data cycle.
3. **Data Cycle:** In the data cycle, either the LMB master puts write data on the LMB which is read by the LMB slave (write cycle), or vice versa (read cycle).

Transfers 2 and 3 show the conflict when two masters try to use the LMB, and how the conflict is resolved. In the example, the LMB master of transfer 2 has a higher priority than the LMB master of transfer 3.

During a block transfer, the address cycle of a second transfer is extended until the data cycles of the block transfer are finished. In the example shown in **Figure 6-3**, transfer 1 is a block transfer while transfer 2 is a single transfer.

On-Chip System Buses and Bus Bridges

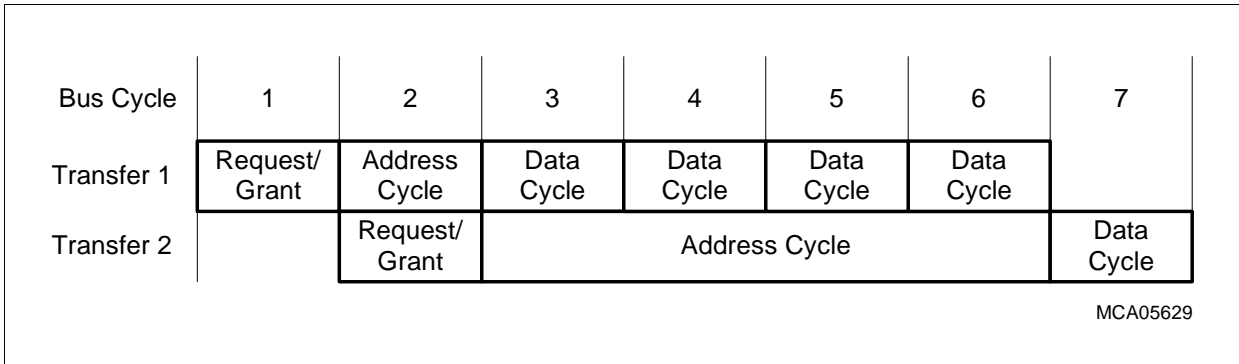


Figure 6-3 LMB Block Transactions

6.2 Local Memory Bus Controller Units

Each of the two LMBs in the TC1796 have a LMB Bus Control Unit (LBCU), one for the DLMB (DBCUC) and one for the PLMB (PBCUC). Where the description in this section refers to LBCUC, the related topic is also valid for DBCUC and PBCUC.

6.2.1 Basic Operation

The LBCUC handles the cycle sequences of the transfers that have been requested by the LMB master devices. The LBCUC is also able to detect bus protocol violations and addressing of un-implemented addresses. In case of a bus error, the LBCUC captures all relevant data such as bus address, bus data and bus status information in register where the information can be analyzed by software.

6.2.2 LMB Bus Arbitration

All LMB master devices requesting the LMB will participate in an arbitration round. Arbitration rounds are performed in each cycle that precedes a possible address cycle. Each LMB master device has a fixed priority as shown in [Table 6-2](#).

Table 6-2 Priority of Master PLMB Agents

LMB Bus	Priority	LMB Bus Master
PLMB	Low priority	DMU
	High priority	PMI (Default Master)
DLMB	Low priority	LFI
	High priority	DMI (Default Master)

For all the masters requesting an LMB (PLMB or DLMB) during any one cycle, the master that is granted the LMB is the one with the highest priority.

6.2.2.1 LMB Bus Default Master

When no LMB master is requesting the LMB, it is granted to the LMB default master. This means if the default master needs the LMB in the next cycle, it can enter the address cycle without running through a request/grant cycle.

6.2.3 LMB Bus Error Handling

When an error occurs on LMB, the LMB bus control unit captures and stores data about the erroneous condition and generates a service request if enabled to do so. The conditions that force an error-capture event are:

- Un-implemented address: No LMB slave responds to an address target
- Error acknowledge: An LMB slave responds with an error to a transaction

When a transaction causes an error, the address and data phase signals of the transaction causing the error are captured and stored in the following registers:

- The LMB Error Address Register (LEADDR) stores the LMB address that has been captured during the last erroneous LMB transaction.
- The LMB Error Data Low/High Registers (LEDATL/LEDTAH) store the 64-bit LMB data bus information that has been captured during the last erroneous LMB transaction.
- The LMB Error Attribute Register (LEATT) stores status information about the bus error event.

If more than one LMB transaction generates a bus error, only the first bus error is captured. After a bus error has been captured, the capture mechanism must be released again by software.

If a transaction from the PCP, the DMA or the Cerberus causes a bus error on the DLMB, the originating masters are not informed about this bus error because they are not a DLMB master agent. With each bus error-capture event, a service request is generated and an interrupt can be generated if enabled and configured in the corresponding service request register.

6.2.4 DLMB and PLMB Bus Registers

This section describes the registers of the DBCU/PBCU modules. The complete and detailed address maps of DBCU/PBCU are described in [Table 18-38/](#)[Table 18-41](#) on [Page 18-122/](#)[Page 18-125](#) of this TC1796 System Units (Vol. 1 of 2) User's Manual.

LMB Bus Register Overview

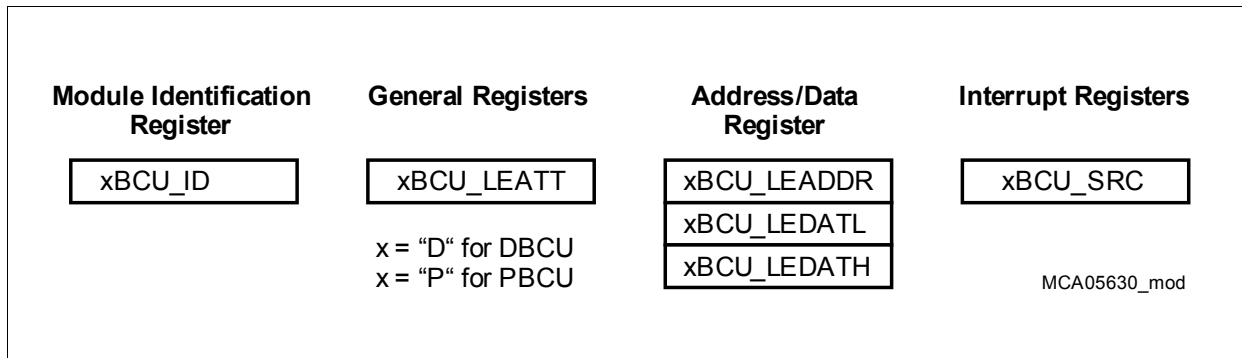


Figure 6-4 DLMB/PLMB Bus Control Unit Registers

Table 6-3 Registers Address Space

Module	Base Address	End Address	Note
DBCUC	F87F FA00 _H	F87F FAFF _H	–
PBCUC	F87F FE00 _H	F87F FEFF _H	–

Table 6-4 Registers Overview - LMB Bus Control Unit Registers

Register Short Name ¹⁾	Register Long Name	Offset Address	Description see
xBCU_ID	xBCU Module Identification Register	08 _H	Page 6-8
xBCU_LEATT	xBCU LMB Error Attribute Register	20 _H	Page 6-9
xBCU_LEADDR	xBCU LMB Error Address Register	24 _H	Page 6-12
xBCU_LEDATL	xBCU LMB Error Data Low Register	28 _H	Page 6-12
xBCU_LEDATH	xBCU LMB Error Data High Register	2C _H	Page 6-13
xBCU_SRC	xBCU Service Request Control Register	FC _H	Page 6-13

1) Prefix x = "D" stands for "DBCUC" and x = "P" for "PBCUC".

On-Chip System Buses and Bus Bridges

DBCUCLEATT

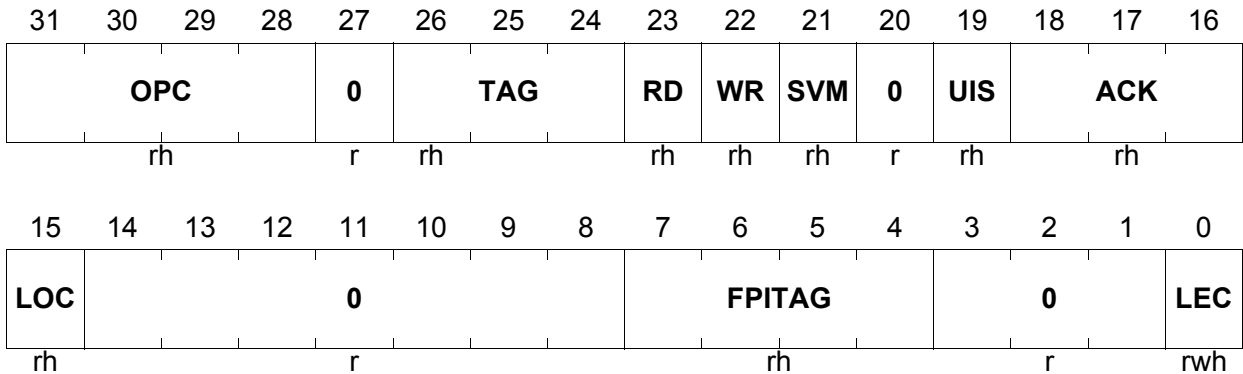
DBCUCLEATT LMB Error Attribute Register (20_H)

Reset Value: XXXX XXX0_H

PBCUCLEATT

PBCUCLEATT LMB Error Attribute Register (20_H)

Reset Value: XXXX XXX0_H



Field	Bits	Type	Description
LEC	0	rwh	<p>Lock Error Capture</p> <p>This bit indicates and controls whether the error-capture mechanism is unlocked or locked.</p> <p>0_B The error-capture mechanism is unlocked. The next LMB bus error will be captured.</p> <p>1_B The error-capture mechanism is locked. The registers LEADDR, LEDATL, LEDATH, and bits [31:4] of LEATT contain valid data.</p> <p>LEC is automatically set when an LMB bus error has been captured. Any further LMB bus error is not captured if LEC = 1. When writing a 1 to LEC, the error-capture mechanism becomes unlocked and is ready for the next LMB bus error-capture event.</p>
FPITAG	[7:4]	rh	<p>FPI Bus Master TAG</p> <p>This bit field indicates the FPI Bus master tag in case of an LMB bus error.</p> <p>Note that the FPI Bus master tag is only of interest if the erroneous LMB transfer was initiated either by the PCP, DMA, or by Cerberus.</p>
LOC	15	rh	<p>LMB Bus Lock State</p> <p>This bit indicates the bus lock state in case of an LMB bus error.</p> <p>0_B LMB bus error occurred at an atomic transfer.</p> <p>1_B LMB bus error occurred at a single or block transfer.</p>

On-Chip System Buses and Bus Bridges

Field	Bits	Type	Description
ACK	[18:16]	rh	<p>LMB Bus Slave Response State This bit indicates status information of the LMB slave device in case of an LMB bus error.</p> <p>000_B Slave is in normal operation. 010_B Slave is busy. 011_B Slave has an error encountered. other Reserved</p>
UIS	19	rh	<p>Un-implemented Address This bit indicates whether the LMB bus error occurred by an un-implemented address.</p> <p>0_B LMB slave address is valid. 1_B Invalid LMB slave address occurred.</p>
SVM	21	rh	<p>LMB Bus Supervisor Mode This bit indicates whether the LMB bus error occurred in supervisor mode or in user mode.</p> <p>0_B Transfer was initiated in supervisor mode. 1_B Transfer was initiated in user modes.</p>
WR	22	rh	<p>LMB Bus Write Error Indication This bit indicates whether the LMB bus error occurred at a write cycle (see Table 6-5).</p>
RD	23	rh	<p>LMB Bus Read Error Indication This bit indicates whether the LMB bus error occurred at a read cycle (see Table 6-5).</p>
TAG	[26:24]	rh	<p>LMB Master TAG This bit field indicates the LMB master device in case of a LMB bus error.</p> <p>DBC_U_LEATT register (DLMB): 000_B LFI 001_B DMI other Reserved</p> <p>PBC_U_LEATT register (PLMB): 000_B LMI 001_B PMI other Reserved</p>

On-Chip System Buses and Bus Bridges

Field	Bits	Type	Description
OPC	[31:28]	rh	LMB Bus Error Transaction Type This bit field indicates the type of transfer at which the LMB bus error occurred. 0000 _B 8-bit data single transfer 0001 _B 16-bit data single transfer 0010 _B 32-bit data single transfer 0011 _B 64-bit data single transfer 1000 _B 2 × 64-bit data block transfer 1001 _B 4 × 64-bit data block transfer other Reserved
0	[3:1], [14:8], 20, 27	r	Reserved Read as 0; should be written with 0.

LEATT[31:4] contains valid read data only if its bit LEC bit is set.

Table 6-5 LMB Read/Write Error Indication

RD	WR	LMB Bus Cycle
0	0	LMB bus error occurred at the read cycle of an atomic transfer.
0	1	LMB bus error occurred at a read cycle of a single transfer.
1	0	LMB bus error occurred at a write cycle of a single transfer or at the write cycle of an atomic transfer.
1	1	Does not occur

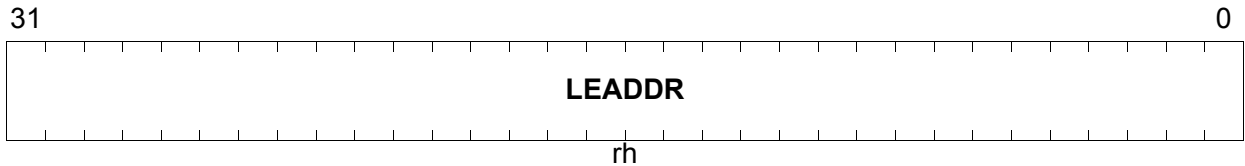
On-Chip System Buses and Bus Bridges

DBCU_LEADDR

DBCU LMB Error Address Register (24_H) **Reset Value: XXXX XXXX_H**

PBCU_LEADDR

PBCU LMB Error Address Register (24_H) **Reset Value: XXXX XXXX_H**



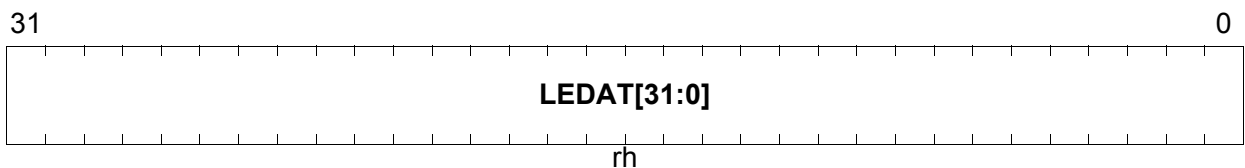
Field	Bits	Type	Description
LEADDR	[31:0]	rh	LMB Address This bit field holds the LMB address that has been captured at an LMB bus error. LEADDR only contains valid read data when bit LEC in the corresponding register LEATT is set.

DBCU_LEDATL

DBCU LMB Error Data Low Register (28_H) **Reset Value: XXXX XXXX_H**

PBCU_LEDATL

PLMB LMB Error Data Low Register (28_H) **Reset Value: XXXX XXXX_H**



Field	Bits	Type	Description
LEDAT[31:0]	[31:0]	rh	LMB Bus Address Bits [31:0] This bit field holds the lower 32-bit part of the 64-bit LMB data that has been captured at an LMB bus error. LEDAT[31:0] only contains valid read data when bit LEC in the corresponding register LEATT is set.

On-Chip System Buses and Bus Bridges

DBCUC LEDATH

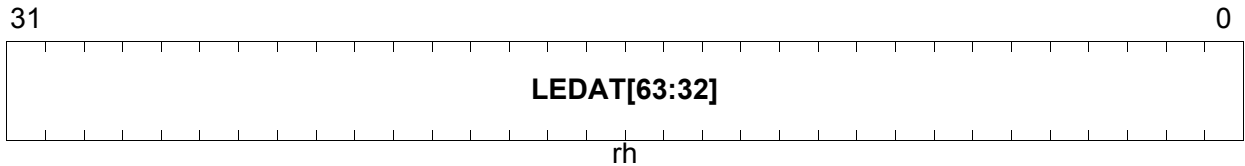
DBCUC LMB Error Data High Register (2C_H)

Reset Value: XXXX XXXX_H

PBCUC LEDATH

PBCUC LMB Error Data High Register (2C_H)

Reset Value: XXXX XXXX_H



Field	Bits	Type	Description
LEDAT[63:32]	[31:0]	rh	<p>LMB Bus Address Bits [31:0]</p> <p>This bit field holds the upper 32-bit part of the 64-bit LMB data that has been captured at an LMB bus error.</p> <p>LEDAT[63:32] only contains valid read data when bit LEC in the corresponding register LEATT is set.</p>

DBCUC SRC

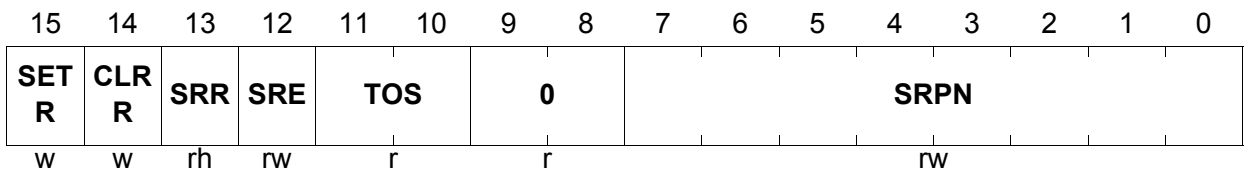
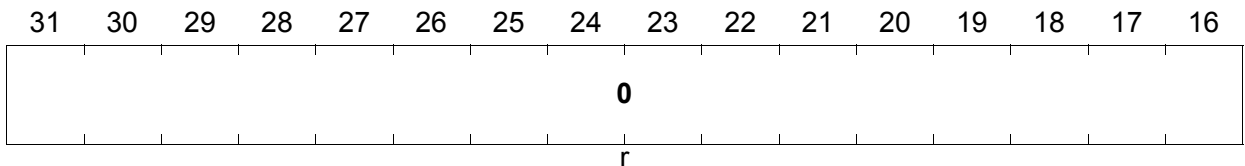
DLMB Service Request Control Register(FC_H)

Reset Value: 0000 0000_H

PBCUC SRC

PBCUC Service Request Control Register(FC_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SRPN	[7:0]	rw	Service Request Priority Number
TOS	[11:10]	r	<p>Type-of-Service State</p> <p>Always read as 00_B. This means type-of-service is associated with interrupt bus 0 (CPU interrupt arbitration bus).</p>

On-Chip System Buses and Bus Bridges

Field	Bits	Type	Description
SRE	12	rw	Service Request Enable
SRR	13	rh	Service Request Flag
CLRR	14	w	Request Flag Clear Bit
SETR	15	w	Request Flag Set Bit
0	[9:8], [31:16]	r	Reserved Read as 0; should be written with 0.

Note: Further details on interrupt handling and processing are described in [Chapter 14](#) of this TC1796 System Units (Vol. 1 of 2) User's Manual.

6.3 Local Memory to FPI Bus Interface (LFI Bridge)

6.3.1 Functional Overview

The LFI Bridge is a bi-directional bus bridge between the DLMB and the System Peripheral FPI Bus (SPB). The bridge supports all transactions types of both the LMB Bus and FPI Bus.

The bridge is not direction-transparent; this means that the master TAG of a bus master is not forwarded to the other side of the bridge and is replaced instead by the master TAG of the LFI Bridge itself.

In order to avoid deadlocks, priority is given to transactions initiated either by PCP, DMA, or by Cerberus.

The bridge supports the pipelining of both connected buses. Therefore, no additional delay is created except for bus protocol conversions.

Address Translation

Addresses of SPB transfers (initiated either by the PCP, DMA controller, or Cerberus) via the LFI Bridge that address a DLMB or PLMB slave device are translated into a DLMB address according [Table 6-6](#).

Table 6-6 SPB to DLMB Address Translation

Transaction Destination	SPB Access Range	Translated DLMB Address
SBRAM/SRAM	E800 0000 _H - E800 FFFF _H	C000 0000 _H - C000 FFFF _H
Reserved	E801 0000 _H - E83F FFFF _H	C001 0000 _H - C03F FFFF _H
DMI LDRAM and DPRAM	E840 0000 _H - E840 FFFF _H	D000 0000 _H - D000 FFFF _H
Reserved	E841 0000 _H - E84F FFFF _H	D0010000 _H - D00F FFFF _H
PMI SPRAM	E850 0000 _H - E850 BFFF _H	D400 0000 _H - D400 BFFF _H
Reserved	E850 C000 _H - E85F FFFF _H	D400 C000 _H - D40F FFFF _H

Bus Errors at Writes via the LFI Bridge

When a write operation has been initiated and directed to the LFI Bridge by a SPB bus master, the LFI Bridge handles the write transaction at the LMBs (DLMB and PLMB) autonomously. If the write operation at the LMB results in a bus error, the BCU of the corresponding LMB detects the bus error and generates a LMB bus error interrupt. There is no bus error generated at the SPB side in this case because of the nature of a posted write operation.

On-Chip System Buses and Bus Bridges

The equivalent behavior occurs when an LMB master initiates a write to a SPB slave device. In this case, SPB bus errors are detected by the SBCU but not at the LMB side. Note that this behavior occurs only at write operations via the LFI Bridge. It also can be triggered by an erroneous write cycle of a read-modify-write bus transaction.

6.3.2 LFI Register

This section describes the kernel register of the LFI Bridge.

LFI Register

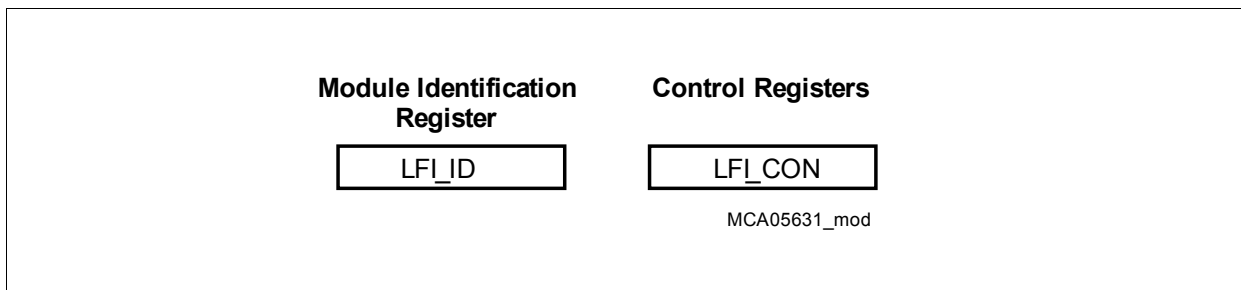


Figure 6-5 LFI Register

The complete and detailed address map of LFI is described in [Table 18-42](#) on [Page 18-126](#) of this TC1796 System Units (Vol. 1 of 2) User's Manual.

Table 6-7 Registers Address Space - LFI Bridge

Module	Base Address	End Address	Note
LFI	F87F FF00 _H	F87F FFFF _H	–

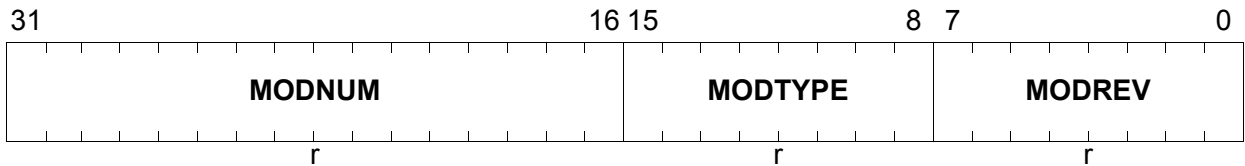
Table 6-8 Registers Overview - LFI Register

Register Short Name	Register Long Name	Offset Address	Description see
LFI_ID	LFI Module Identification Register	08 _H	Page 6-17
LFI_CON	LFI Configuration Register	10 _H	Page 6-17

On-Chip System Buses and Bus Bridges

LFI_ID

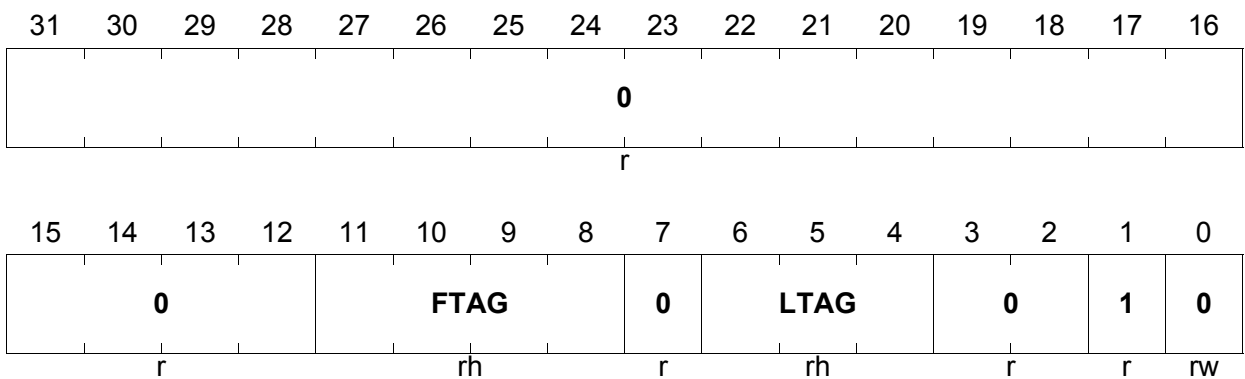
LFI Module Identification Register (08_H) **Reset Value: 000C C0XX_H**



Field	Bits	Type	Description
MODREV	[7:0]	r	Module Revision Number MODREV defines the module revision number. The value of a module revision starts with 01 _H (first revision).
MODTYPE	[15:8]	r	Module Type This bit field defines the module as a 32-bit module: C0 _H
MODNUM	[31:16]	r	Module Number Value This bit field defines the module identification number for the LFI: 000C _H

LFI_CON

LFI Configuration Register (10_H) **Reset Value: 0000 0B02_H**



Field	Bits	Type	Description
0	0	rw	Reserved Returns 0 if read; must be written with 0.
1	1	r	Reserved Returns 1 if read; should be written with 1.

On-Chip System Buses and Bus Bridges

Field	Bits	Type	Description
LTAG	[6:4]	rh	LMB Bus (DLMB) Tag ID In the TC1796, the bit field LTAG = 000 _B .
FTAG	[11:8]	rh	FPI Bus (SPB) Tag ID In the TC1796, the bit field FTAG = 1011 _B , which reflects the tag number of the LFI Bridge on the SPB.
0	[3:2], 7, [31:12]	r	Reserved Returns 0 if read.

6.4 System and Remote Peripheral Bus

The TC1796 has two on-chip FPI buses:

- System Peripheral Bus (SPB)
 - System bus for on-chip peripherals (except for SSCs and ADCs)
- Remote Peripheral Bus (RPB)
 - Mainly dedicated to DMA transactions of SSCs and ADCs with DMI memories

This section gives an overview of the two on-chip FPI buses. It describes its bus control units, the bus characteristics, bus arbitration, scheduling, prioritizing, error conditions, and debugging support. Both FPI buses have the same FPI-Bus functionality. If required, implementation-specific differences are described.

6.4.1 Overview

The FPI buses interconnect the on-chip peripheral units with the TC1796 processor subsystem. **Figure 6-6** gives an overview of the FPI Buses and the modules connected to them.

The FPI Bus is designed to be quick to acquire by on-chip functional units, and quick to transfer data. The low setup overhead of the FPI Bus access protocol guarantees fast FPI Bus acquisition, which is required for time-critical applications.

The FPI Bus is designed to sustain high transfer rates. For example, a peak transfer rate of up to 160 Mbyte/s can be achieved with the 32-bit data bus at 40 MHz bus clock. Multiple data transfers per bus arbitration cycle allow the FPI Bus to operate at close to its peak bandwidth.

Additional features of the FPI Bus include:

- Optimized for high speed and high performance
- Support of multiple bus masters
- 32-bit wide address and data buses
- 8-, 16-, and 32-bit data transfers
- 64-, 128-, and 256-bit block transfers
- Central simple per-cycle arbitration
- Slave-controlled wait state insertion
- Support of atomic operations LDMST, ST.T and SWAP.W
- Designed to minimize EMI and power consumption

The functional units of the TC1796 are connected to the FPI bus via FPI bus interfaces. An FPI Bus interface act as bus agent, requesting bus transactions on behalf of their functional unit, or responding to bus transaction requests.

There are two types of bus agents:

- FPI Bus master agents can initiate FPI Bus transactions and can also act as slaves.
- Slave agents can only react and respond to FPI Bus transaction requests in order to read or write internal registers of slave modules as for example memories.

On-Chip System Buses and Bus Bridges

When an FPI Bus master attempts to initiate a transfer on the FPI Bus, it first signals a request for bus ownership to the bus control unit. When bus ownership is granted by the bus control unit, an FPI Bus read or write transaction is initiated. The unit targeted by the transaction becomes the FPI Bus slave, and responds with the requested action.

Some functional units operate only as slaves, while others can operate as either masters or slaves. In the TC1796, DMI and PMI (via the LFI Bridge), PCP, DMA, and Cerberus operate as FPI Bus masters. On-chip peripheral units are typically FPI Bus slaves. **Figure 6-6** shows the two FPI Buses and the interface types of the various modules.

FPI Bus arbitration is performed by the Bus Control Unit (BCU) of the FPI Bus. In case of bus errors, the BCU generates an interrupt request to the CPU and provides debugging information about the actual bus error to the CPU.

In the TC1796, device external memory/peripheral accesses are handled via the EBU as part of the PMU. Therefore, the FPI Buses are not required for such type of device external transactions.

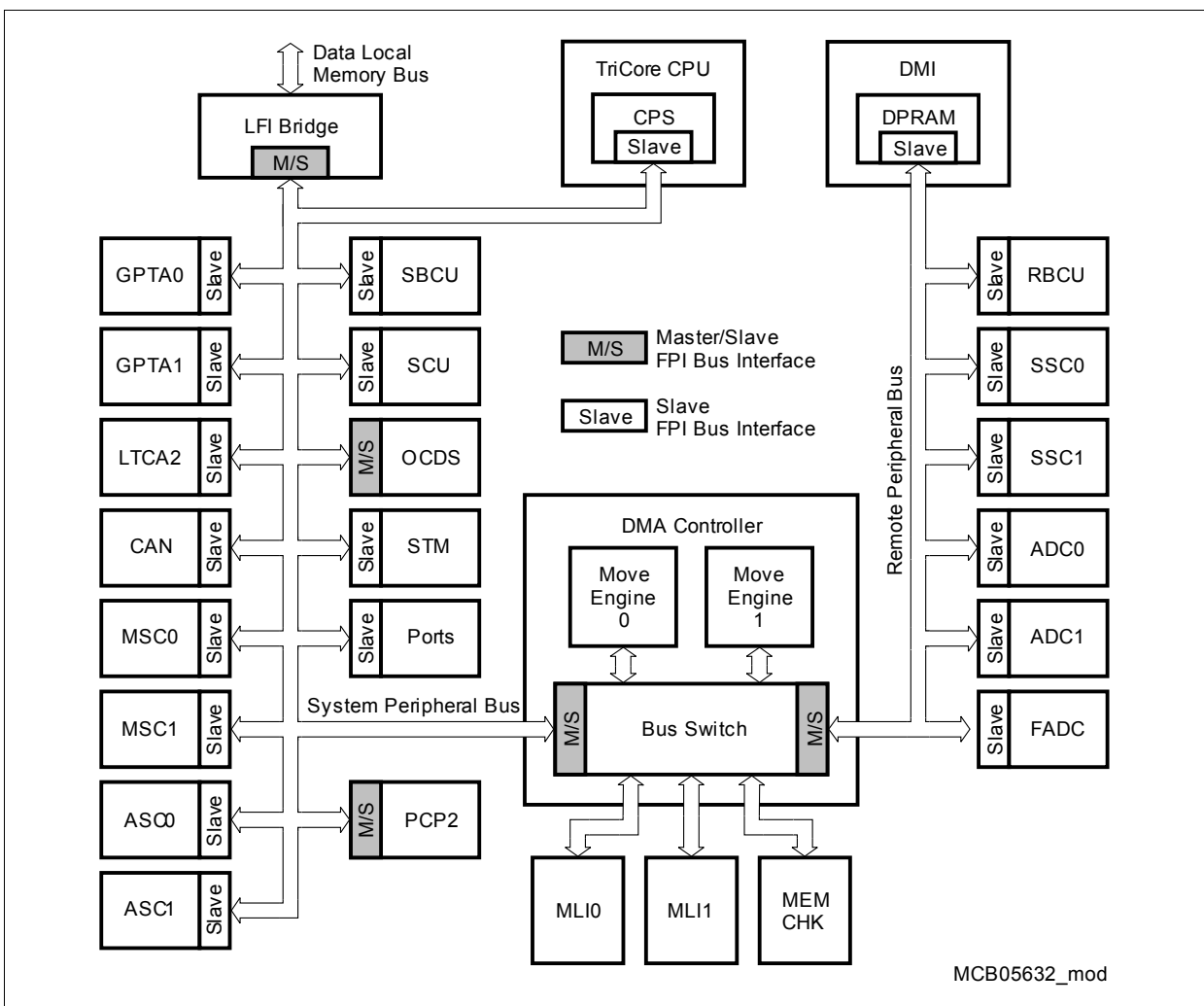


Figure 6-6 TC1796 FPI Buses Block Diagram

The Bus Switch in the DMA controller operates as a bus bridge between the SPB and RPB.

6.4.2 Bus Transaction Types

This section describes the SPB transaction types.

Single Transfers

Single transfers are byte, half-word, and word transactions that target any slave connected either to SPB or RPB. Note that the LFI Bridge operates as an SPB master.

Block Transfers

Block transfers operate in principle in the same way as single transfers do, but one address phase is followed by multiple data phases. Block transfers can be composed of 2 word, 4 word, or 8 word transfers.

Note: In general, block transfers (2 word, 4 word, or 8 word) cannot be executed in the TC1796 with peripheral units that operate as FPI Bus slaves during an FPI Bus transaction.

Block transfers are initiated by the following CPU instructions: LD.D, LD.DA, MOV.D, ST.D and ST.DA. The BCOPY instruction of the PCP also initiates a block transfer transaction on the FPI Bus.

Atomic Transfers

Atomic transfers are generated by LDMST, ST.T and SWAP.W instructions that require two single transfers. The read and write transfer of an atomic transfer are always locked and cannot be interrupted by another bus masters. Atomic transfers are also referenced as read-modify-write transfers.

Note: See also [Table 6-12](#) for available FPI Bus transfer types.

6.4.3 Reaction of a Busy Slave

If an FPI Bus slave is busy at an incoming FPI Bus transaction request, it can delay the execution of the FPI Bus transaction. The requesting FPI Bus master releases the FPI Bus for one cycle after the FPI Bus transaction request, in order to allow the FPI slave to indicate if it is ready to handle the requested FPI Bus transaction. This sequence is repeated as long as the slave indicates that it is busy.

Note: For the FPI Bus default master, the one cycle gap does not result in a performance loss because it is granted the FPI Bus in this cycle as default master if no other master requests the FPI Bus for some other reasons.

6.4.4 Address Alignment Rules

FPI Bus address generation is compliant with the following rules:

- Half-word transactions must have a half-word aligned address ($A_0 = 0$). Half-word accesses on byte lanes 1 and 2 addresses are illegal.
- Word transactions must always have word-aligned addresses ($A[1:0] = 00_B$).
- Block transactions must always have block-type aligned addresses.

6.4.5 FPI Bus Basic Operations

This section describes some basic transactions on the FPI Bus.

The example in [Figure 6-7](#) shows the three cycles of an FPI Bus operation:

1. **Request/Grant Cycle:** The FPI Bus master attempts to perform a read or write transfer and requests for the FPI Bus. If the FPI Bus is available, it is granted in the same cycle by the FPI Bus controller.
2. **Address Cycle:** After the request/grant cycle, the master puts the address on the FPI Bus, and all FPI Bus slave devices check whether they are addressed for the following data cycle.
3. **Data Cycle:** In the data cycle, either the master puts write data on the FPI Bus which is read by the FPI Bus slave (write cycle) or vice versa (read cycle).

Transfers 2 and 3 show the conflict when two masters try to use the FPI Bus and how the conflict is solved. In the example, the FPI Bus master of transfer 2 has a higher priority than the FPI Bus master of transfer 3.

Bus Cycle	1	2	3	4	5
Transfer 1	Request / Grant	Address Cycle	Data Cycle		
Transfer 2		Request / Grant	Address Cycle	Data Cycle	
Transfer 3		Request/Grant		Address Cycle	Data Cycle

MCA05634_mod

Figure 6-7 Basic FPI Bus Transactions

At a block transfer, the address cycle of a second transfer is extended until the data cycles of the block transfer are finished. In the example of [Figure 6-8](#), transfer 1 is a block transfer, while transfer 2 is a single transfer.

On-Chip System Buses and Bus Bridges

Bus Cycle	1	2	3	4	5	6	7
Transfer 1	Request/ Grant	Address Cycle	Data Cycle	Data Cycle	Data Cycle	Data Cycle	
Transfer 2		Request/ Grant	Address Cycle				Data Cycle

MCA05635

Figure 6-8 FPI Bus Block Transactions

6.5 FPI Bus Control Units (SBCU and RBCU)

The TC1796 incorporates two BCUs: one for the SPB, called SBCU, and the other for the RPB, called RBCU. The register name module prefix “x” of xBCU refers to the SBCU with x = S and to the RBCU with x = R.

6.5.1 FPI Bus Arbitration

The arbitration unit of the BCU determines whether it is necessary to arbitrate for FPI Bus ownership, and, if so, which available bus requestor gets the FPI Bus for the next data transfer. During arbitration, the bus is granted to the requesting agent with the highest priority. If no request is pending, the bus is granted to a default master. If no bus master takes the bus, the BCU itself will drive the FPI Bus to prevent it from floating electrically.

6.5.1.1 Arbitration on the System Peripheral Bus

The TC1796 SPB has five bus agents that can become a SPB bus master. Each agent is supplied a arbitration priority as shown in [Table 6-9](#). DMA controller and OCDS agents can be assigned to low or high priorities by software.

Table 6-9 Priority of TC1796 SPB Bus Agents

Priority	Agent	Comment
highest	Any bus requestor meeting the starvation protection criteria is assigned this priority	Highest priority, used only for starvation protection
	On-Chip Debug System (Cerberus), high priority	Priority selection by software
	Peripheral Control Processor	Default master 0
	DMA Controller, high priority channels	Priority selection by software
	LFI Bridge	Default master 1
	DMA Controller, low priority channels	Priority selection by software
lowest	On-Chip Debug System (Cerberus), low priority	Priority selection by Software

If there is no request from a SPB bus master, the SPB is granted to a default master (PCP or LFI Bridge) which has been at last the default master.

6.5.1.2 Arbitration on the Remote Peripheral Bus

The TC1796 RPB has only one bus master agent, the DMA Controller. Therefore, the RPB does not need the RBCU to arbitrate the bus, and the DMA controller is always its default master.

6.5.1.3 Starvation Prevention

Starvation prevention is a feature that is especially important for the SPB. Because the priority assignment of the SPB agents is fixed, it is possible that a lower-priority bus requestor may never be granted the bus if a higher-priority bus requestor continuously asks for, and receives, bus ownership. To protect against bus starvation of lower-priority masters, an optional feature of the TC1796 will detect such cases and momentarily raise the priority of the lower-priority requestor to the highest priority (above all other priorities), thereby guaranteeing it access.

Starvation protection employs a counter that is incremented each time an arbitration is performed in the BCU. When this counter reaches a user-programmable threshold value, for each active bus request a request flag is stored in the BCU. This flag is cleared automatically when a master is granted the bus.

When the counter reaches the threshold value, it is automatically reset to zero and starts counting up again. When the next period is finished, an active request of a master from which the request flag was set, a starvation event happened. This master will now be set to the highest priority and will be granted service. If there are several masters to which this starvation condition applies, they are served in the order of their hard-wired priority ranking.

Starvation protection can be enabled and disabled through bit `SBCU_CON.SPE`. The sample period of the counter is programmed through bit field `SBCU_CON.SPC`. `SPC` should be set to a value at least greater than or equal to the number of masters. Its reset value is 40_H .

Note: The RPB also provides the SPE starvation protection enable bit and the SPC starvation protection counter bit field. Because of the RPB single master agent configuration, `RBCU_CON.SPE` can be always disabled.

6.5.2 FPI Bus Error Handling

When an error occurs on a FPI Bus, its BCU captures and stores data about the erroneous condition and generates a service request if enabled to do so. The error conditions that force an error-capture are:

- Error acknowledge: An FPI Bus slave responds with an error to a transaction.
- Un-implemented address: No FPI Bus slave responds to a transaction request.
- Time-out: A slave does not respond to a transaction request within a certain time window. The number of bus clock cycles that can elapse until a bus time-out is generated is defined by bit field `xBCU_CON.TOUT`.

When a transaction causes an error, the address and data phase signals of the transaction causing the error are captured and stored in registers.

- The Error Address Capture Register (`xBCU_EADD`) stores the 32-bit FPI Bus address that has been captured during the erroneous FPI Bus transaction.

On-Chip System Buses and Bus Bridges

- The Error Data Capture Registers (xBCU_EDAT) stores the 32-bit FPI Bus data bus information that has been captured during the erroneous FPI Bus transaction.
- The Error Control Capture Register (xBCU_ECON) stores status information of the bus error event.

If more than one FPI Bus transaction generates a bus error, only the first bus error is captured. After a bus error has been captured, the capture mechanism must be released again by software.

If a write transaction from TriCore causes an error on the SPB, the originating master is not informed about this error as it is not an SPB master agent. With each bus error-capture event, a service request is generated, and an interrupt can be generated if enabled and configured in the corresponding service request register.

Interpreting the BCU Control Register Error Information

Although the address and data values captured in registers xBCU_EADD and xBCU_EDAT, respectively, are self-explanatory, the captured FPI Bus control information needs some more explanation.

Register xBCU_ECON captures the state of the read (RDN), write (WRN), Supervisor Mode (SVM), acknowledge (ACK), ready (RDY), abort (ABT), time-out (TOUT), bus master identification lines (TAG) and transaction operation code (OPC) lines of the FPI Bus.

The SVM signal is set to 1 for an access in Supervisor Mode and set to 0 for an access in User Mode. The time-out signal indicates if there was no response on the bus to an access, and the programmed time (via xBCU_TOUT) has elapsed. TOUT is set to 1 in this case. An acknowledge code has to be driven by the selected slave during each data cycle of an access. These codes are listed in [Table 6-10](#).

Table 6-10 FPI Bus Acknowledge Codes

Code (ACK)	Description
00 _B	NSC: No Special Condition.
01 _B	SPT: Split Transaction (not used in the TC1796).
10 _B	RTY: Retry. Slave can currently not respond to the access. Master needs to repeat the access later.
11 _B	ERR: Bus Error, last data cycle is aborted.

Each master on the FPI Bus is assigned a 4-bit identification number, the master TAG number (see [Table 6-11](#)). This makes it possible to distinguish which master has performed the current transaction.

Table 6-11 FPI Bus Master TAG Assignments

TAG-Number	FPI Bus	Description
1001 _B	SPB	Peripheral Control Processor (PCP2)
1010 _B		DMA Controller
1011 _B		LFI Bridge
1100 _B		OCDS (Cerberus)
1111 _B	RPB	DMA Controller
Others	–	Reserved

Transactions on the FPI Bus are classified via a 4-bit operation code (see [Table 6-12](#)). Note that split transactions (OPC = 1000_B to 1110_B) are not used in the TC1796.

Table 6-12 FPI Bus Operation Codes (OPC)

OPC	Description
0000 _B	Single Byte Transfer (8-bit)
0001 _B	Single Half-Word Transfer (16-bit)
0010 _B	Single Word Transfer (32-bit)
0100 _B	2-Word Block Transfer
0101 _B	4-Word Block Transfer
0110 _B	8-Word Block Transfer
1111	No operation
0011 _B , 0111 _B , 1000 _B - 1110 _B	Reserved

6.5.3 Clock Management

The BCU can be configured so that it shuts down automatically when not needed by disabling its internal clock. When it is needed again, for instance when a bus request signal is received from a master, the BCU will enable its clock and perform the arbitration. If no further bus activity is required after the transfer has completed, the BCU will automatically shut off its clock and return to idle mode.

Automatic power management is controlled through the xBCU_CON.PSE bit. When cleared to 0, power management is disabled, and the BCU clock is always active. This might be required, for instance, to debug both the active and idle FPI Bus states of an application via an external emulator or other debugging hardware.

6.5.4 BCU Debug Support

For debugging purposes, the BCU has the capability for breakpoint generation support. This OCDS debug capability is controlled by the Cerberus module and must be enabled by it (indicated by bit xBCU_DBCNTL.EO).

When BCU debug support has been enabled (EO = 1), any breakpoint request generated by the BCU to the Cerberus disarms the BCU breakpoint logic for further breakpoint requests. In order to rearm the BCU breakpoint logic again for the next breakpoint request generation, bit xBCU_DBCNTL.RA must be set. The status of the BCU breakpoint logic (armed or disarmed) is indicated by bit xBCU_DBCNTL.OA.

There are three types of trigger events:

- Address triggers
- Signal triggers
- Grant triggers

6.5.4.1 Address Triggers

The address debug trigger event conditions are defined by the contents of the xBCU_DBADR1, xBCU_DBADR2, and xBCU_DBCNTL registers. A wide range of possibilities arise for the creation of debug trigger events based on addresses. The following debug trigger events can be selected:

- Match on one signal address
- Match on one of two signal addresses
- Match on one address area
- Mismatch on one address area

Each pair of DBADR_x registers and DBCNTL.ONA_x bits determine one possible debug trigger event. The combination of these two possible debug trigger events defined by DBCNTL.CONCOM1 determine the address debug trigger event condition.

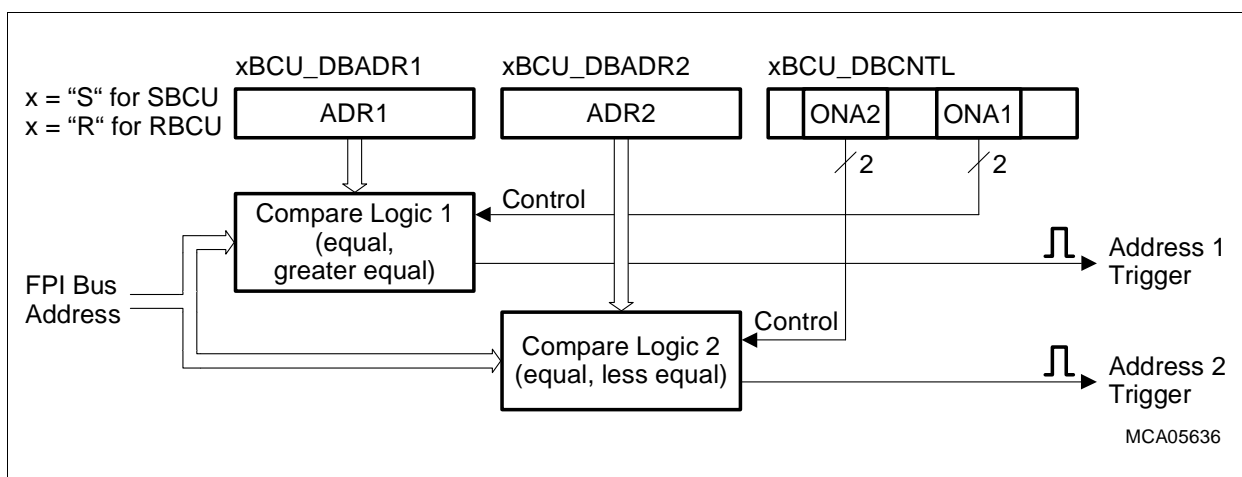


Figure 6-9 Address Trigger Generation

6.5.4.2 Signal Status Triggers

The signal status debug trigger event conditions are defined by the contents of the xBCU_DBBOS and xBCU_DBCNTL registers. Depending on the selected configuration a wide range of possibilities arise for the creation of a debug trigger event based on FPI Bus status signals. Possible combinations are:

- Match on a single signal status
- Match on a multiple signal status

With the multiple signal match conditions, all single signal match conditions are combined with a logical **AND** to the signal status debug trigger event signal. The selection whether or not a single match condition is selected can be enabled/disabled selectively for each condition via the xBCU_DBCNTL.ONBOSx bits.

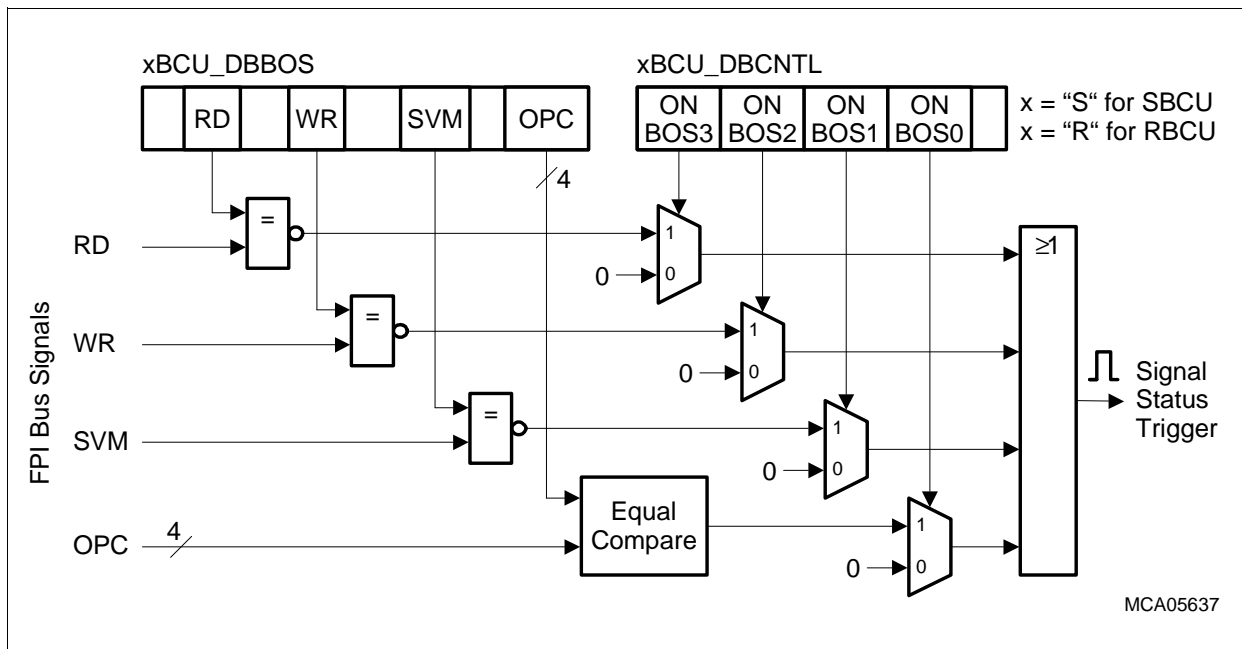


Figure 6-10 Signal Status Trigger Generation

6.5.4.3 Grant Triggers

The signal status debug trigger event conditions are defined via the registers xBCU_DBGRNT and xBCU_DBCNTL. Depending on the configuration of these registers, any combination of FPI Bus master trigger events can be configured. Only the enabled masters in the xBCU_DBGRNT register are of interest for the grant debug trigger event condition. The grant debug trigger event condition can be enabled/disabled via bit xBCU_DBCNTL.ONG (see [Figure 6-11](#)).

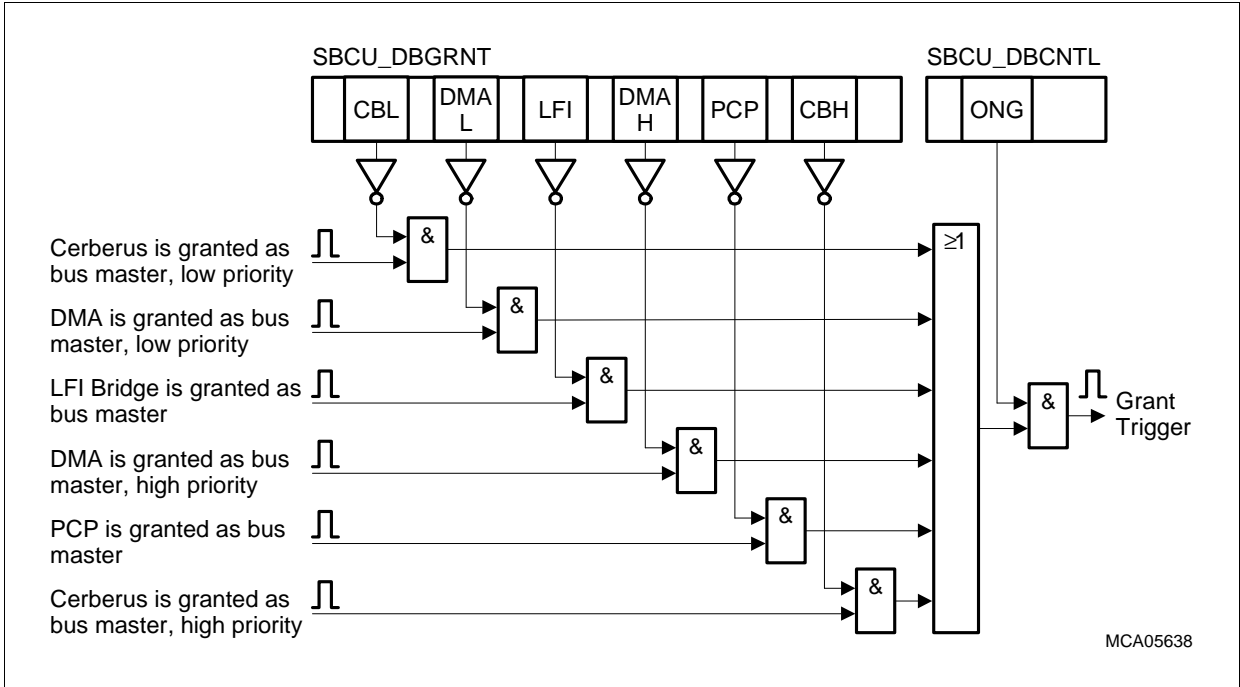


Figure 6-11 Grant Trigger Generation

6.5.4.4 Combination of Triggers

The combination of the four debug trigger signals to the single BCU breakpoint trigger event is defined via the bits CONCOM[2:0] of register xBCU_DBCNTL (see Figure 6-12). The two address trigger signals are combined to one address trigger that is further combined with signal status and grant trigger signals. A logical AND or OR combination can be selected for the BCU breakpoint trigger generation.

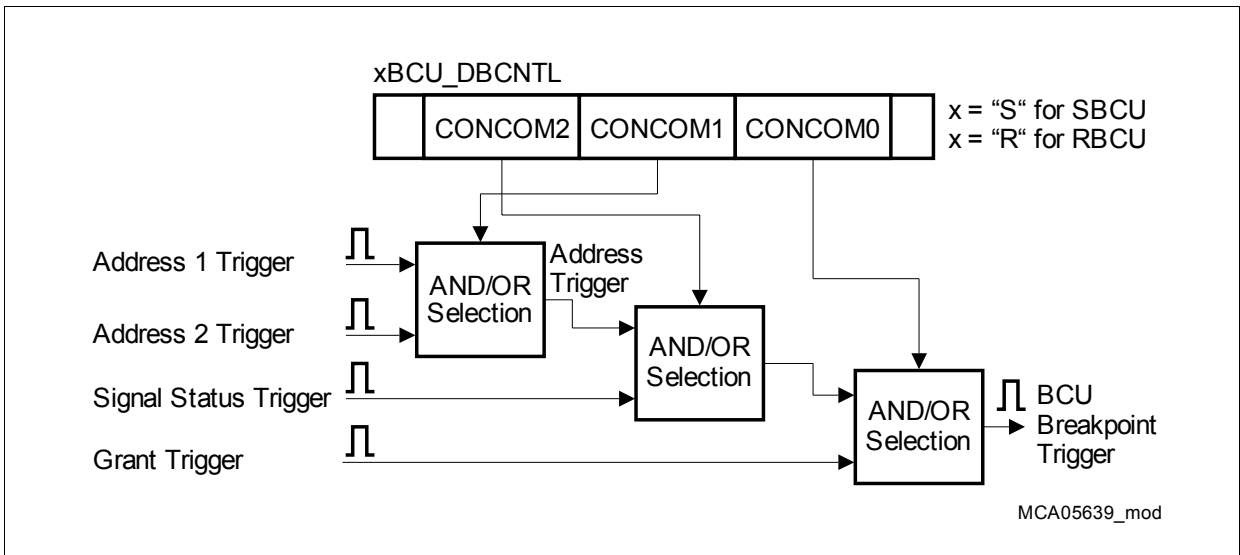


Figure 6-12 BCU Breakpoint Trigger Combination Logic

6.5.4.5 BCU Breakpoint Generation Examples

This section gives three examples of how BCU debug trigger events are programmed.

OCDS Debug Example 1

- Task: Generation of a BCU debug trigger event on any SPB write access to address 00002004_H or 000020A0_H by SPB masters PCP or LFI Bridge.

For this task, the following programming settings for the BCU breakpoint logic must be executed:

1. Writing SBCU_DBADR1 = 0000 2004_H
2. Writing SBCU_DBADR2 = 0000 20A0_H
3. Writing SBCU_DBCNTL = C1115010_H:
ONBOS[3:0] = 1100_B means that no signal status trigger is generated (disabled) for a read signal match AND write signal match condition according to the settings of bits RD and WR in register SBCU_DBBOS. Debug trigger event generation for supervisor mode signal match and opcode signal match condition is disabled.
ONA2 = 01_B means that the equal match condition for debug address 2 register is selected.
ONA1 = 01_B means that the equal match condition for debug address 1 register is selected.
ONG = 1 means that the grant debug trigger is enabled.
CONCOM[2:0] = 101_B means that the address trigger is created by address trigger 1 OR address trigger 2 (CONCOM1 = 0), and that the grant trigger is ANDed with the address trigger (CONCOM0 = 1), and that the signal status trigger is ANDed with the address trigger (CONCOM2 = 1).
RA = 1 means that the BCU breakpoint logic is rearmed.
4. Writing SBCU_DBGRNT = FFFFFFFD7_H means that the grant trigger for the SPB masters PCP and LFI Bridge is enabled.
5. Writing SBCU_DBBOS = 00001000_H means that the signal status trigger is generated on a write transfer and not on a read transfer.

OCDS Debug Example 2

- Task: generation of a BCU debug trigger event on any half-word access in user mode to address area 01FFFFFF_H to 02FFFFFF_H by any master.

For this task, the following programming settings for the BCU breakpoint logic must be executed:

1. Writing SBCU_DBADR1 = 01FFFFFF_H
2. Writing SBCU_DBADR2 = 02FFFFFF_H
3. Writing SBCU_DBCNTL = 32206010_H:
 ONBOS[3:0] = 0011_B means that the signal status trigger is disabled for a read or for write signal status match but enabled for supervisor mode match AND opcode match conditions according to the settings of bit SVM and bit field OPC in register SBCU_DBBOS.
 ONA2 = 10_B means that the address 2 trigger is generated if the FPI Bus address is greater or equal to SBCU_DBADR2.
 ONA1 = 10_B means that the address 1 trigger is generated if the FPI Bus address is greater or equal to SBCU_DBADR1.
 ONG = 0 means that the grant debug trigger is disabled.
 CONCOM[2:0] = 110_B means that the address trigger is created by address trigger 1 AND address trigger 2 (CONCOM1 = 1), and that the grant trigger is OR-ed with the address trigger (CONCOM0 = 0), and that the signal status trigger is ANDed with the address trigger (CONCOM2 = 1).
 RA = 1 means that the BCU breakpoint logic is rearmed.
4. Writing SBCU_DBGRNT = FFFFFFFF_H:
 means that no grant trigger for SPB masters is selected (“don’t care” because also disabled by ONG = 0).
5. Writing SBCU_DBBOS = 00000001_H:
 means that the signal status trigger is generated for read (RD = 0) and write (WR = 0) half-word transfers (OPC = 0001_B) in user mode (SVM = 0).

OCDS Debug Example 3

- Task: Generation of a BCU debug trigger event on any access into address area 01FFFFFF_H to FFFFFFFF_H by the PCP.

For this task the following programming settings for the BCU breakpoint logic must be executed:

1. Writing SBCU_DBADR1 = 01FFFFFF_H
2. Writing SBCU_DBADR2 = don’t care
3. Writing SBCU_DBCNTL = 00215010_H:
 ONBOS[3:0] = 0000_B means that a signal status trigger is generated for all FPI Bus opcodes not equal to a “no operation” opcode.
 ONA2 = 00_B means that no address 2 trigger is generated.
 ONA1 = 10_B means that the address 1 trigger is generated if the FPI Bus address is

On-Chip System Buses and Bus Bridges

greater or equal to SBCU_DBADR1.

ONG = 1 means that the grant debug trigger is enabled.

CONCOM[2:0] = 101_B means that the address trigger is created by address trigger 1 OR address trigger 2 (CONCOM1 = 0), and that the grant trigger is ANDed with the address trigger (CONCOM0 = 1), and that the signal status trigger is ANDed with the address trigger (CONCOM2 = 1).

RA = 1 means that the BCU breakpoint logic is rearmed.

4. Writing SBCU_DBGRNT = FFFFFFFF_H:
means that the grant trigger for the SPB bus master PCP is enabled.
5. Writing SBCU_DBBOS is "don't care". No signal trigger for SVM, WR, or RD is generated.

6.5.5 SBCU and RBCU Registers

The registers shown in [Figure 6-13](#) are available for the SBCU.

SBCU Registers Overview

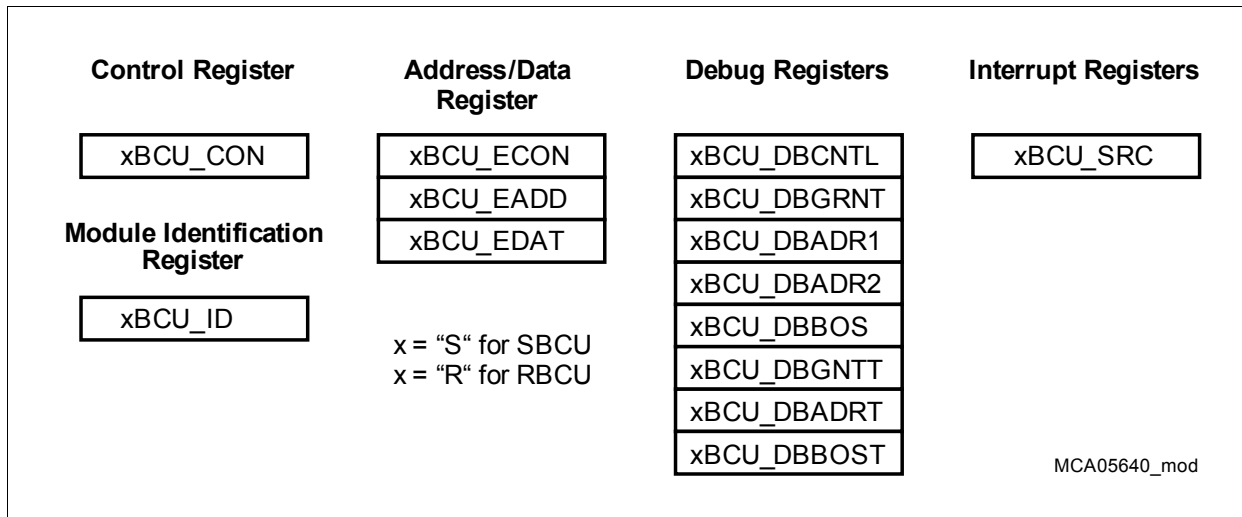


Figure 6-13 SBCU Registers

The complete and detailed address maps of SBCU/RBCU are described in [Table 18-4/Table 18-26](#) on [Page 18-11/Page 18-79](#) of this TC1796 System Units (Vol. 1 of 2) User's Manual.

Table 6-13 Registers Address Space

Module	Base Address	End Address	Note
SBCU	F000 0100 _H	F000 01FF _H	—
RBCU	F010 0000 _H	F010 00FF _H	—

Table 6-14 Registers Overview - SBCU Registers

Register Short Name ¹⁾	Register Long Name	Offset Address	Description see
xBCU_ID	xBCU Module Identification Register	08 _H	Page 6-36
xBCU_CON	xBCU Control Register	10 _H	Page 6-37
xBCU_ECON	xBCU Error Control Capture Register	20 _H	Page 6-39
xBCU_EADD	xBCU Error Address Capture Register	24 _H	Page 6-41
xBCU_EDAT	xBCU Error Data Capture Register	28 _H	Page 6-41
xBCU_DBCNTL	xBCU Debug Control Register	30 _H	Page 6-42

On-Chip System Buses and Bus Bridges

Table 6-14 Registers Overview - SBCU Registers (cont'd)

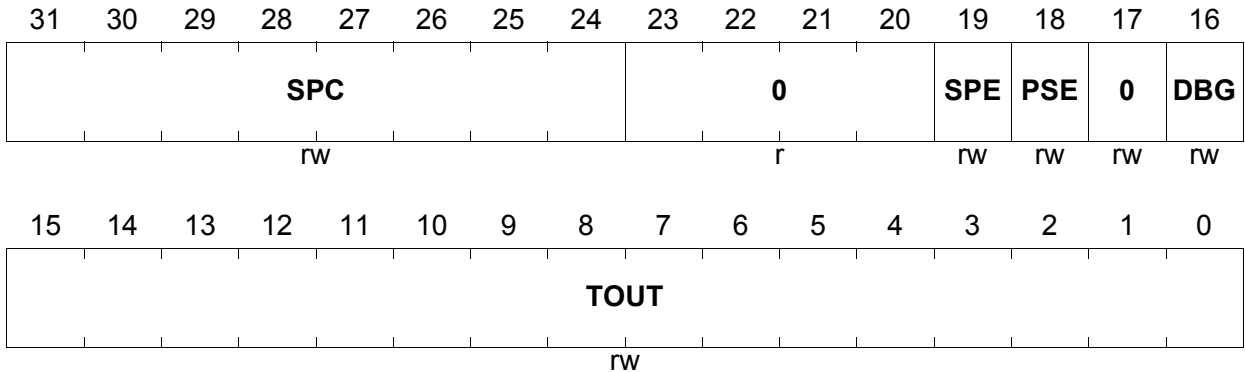
Register Short Name¹⁾	Register Long Name	Offset Address	Description see
SBCU_DBGRNT	SBCU Debug Grant Mask Register	0034 _H	Page 6-45
RBCU_DBGRNT	RBCU Debug Grant Mask Register		Page 6-47
xBCU_DBADR1	xBCU Debug Address 1 Register	0038 _H	Page 6-48
xBCU_DBADR2	xBCU Debug Address 2 Register	003C _H	Page 6-48
xBCU_DBBOS	xBCU Debug Bus Operation Signals Register	0040 _H	Page 6-49
SBCU_DBGNTT	SBCU Debug Trapped Master Register	0044 _H	Page 6-51
RBCU_DBGNTT	RBCU Debug Trapped Master Register		Page 6-53
xBCU_DBADRT	xBCU Debug Trapped Address Register	0048 _H	Page 6-54
xBCU_DBBOST	xBCU Debug Trapped Bus Operation Signals Register	004C _H	Page 6-55
xBCU_SRC	xBCU Service Request Control Register	00FC _H	Page 6-58

1) Prefix x = "S" stands for "SBCU" and x = "R" for "RBCU".

On-Chip System Buses and Bus Bridges

SBCU_CON
SBCU Control Register (10_H) Reset Value: 4009 FFFF_H

RBCU_CON
RBCU Control Register (10_H) Reset Value: 4009 FFFF_H



Field	Bits	Type	Description
TOUT	[15:0]	rw	BCU Bus Time-Out Value The bit field determines the number of System Peripheral Bus time-out cycles. Default after reset is FFFF _H (= 65536 bus cycles). ¹⁾
DBG	16	rw	BCU Debug Trace Enable 0 _B BCU debug trace disabled 1 _B BCU debug trace enabled (default after reset)
PSE	18	rw	BCU Power Saving (Automatic Clock Control) Enable 0 _B BCU power saving disabled (default after reset) 1 _B BCU power saving enabled
SPE	19	rw	BCU Starvation Protection Enable 0 _B BCU starvation protection disabled 1 _B BCU starvation protection enabled (default after reset)
SPC	[31:24]	rw	Starvation Period Control Determines the sample period for the starvation counter. Must be larger than the number of masters. The reset value is 40 _H .
0	17	rw	Reserved Read as 0 after reset; should be written with 0.

On-Chip System Buses and Bus Bridges

Field	Bits	Type	Description
0	[23:20]	r	Reserved Read as 0; should be written with 0.

- 1) When an access occurs from an SPB bus master to the RPB while the RPB is busy, the SPB bus master has to wait until the RPB is granted. Therefore, the SBCU time-out value must be larger than the RBCU time-out value: $SBCU_CON.TOUT_{min} = RBCU_CON.TOUT + 28$.

6.5.5.2 BCU Error Registers

The capture of bus error conditions is enabled by setting `BCU_CON.DBG` to 1. In case of a bus error, information about the condition will then be stored in the BCU error capture registers. The BCU error capture registers for SPB and RPB can then be examined by software to determine the cause of the FPI Bus error.

If enabled and a FPI Bus error occurs, the ECON registers holds the captured FPI Bus control information and an error count of the number of bus errors. The EADD registers store the captured FPI Bus address. The EDAT registers store the captured FPI Bus data.

If the capture of FPI Bus error conditions is disabled (`BCU_CON.DBG = 0`), the BCU error capture registers remain untouched.

Note: The BCU error capture registers store only the parameters of the first error. In case of multiple bus errors, an error counter `BCU_ECON.ERRCNT` shows the number of bus errors since the first error occurred. A hardware reset clears this bit field to zero, but the counter can be set to any value through software. This counter is prevented from overflowing, so a value of $2^{16} - 1$ indicates that at least this many errors have occurred, but there may have been more. After `BCU_ECON` has been read, the `BCU_ECON`, `BCU_EADD` and `BCU_EDAT` registers are re-enabled to trace FPI Bus error conditions.

On-Chip System Buses and Bus Bridges

SBCU_ECON

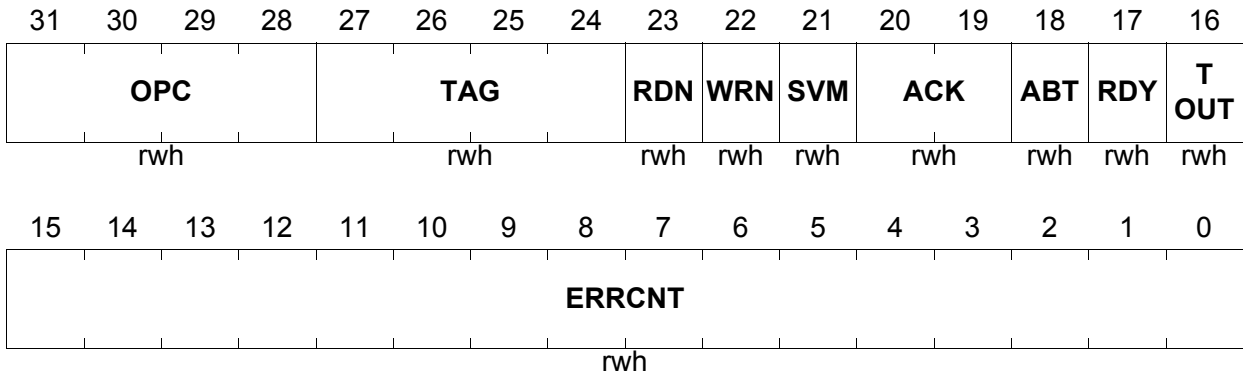
SBCU Error Control Capture Register (20_H)

Reset Value: 0000 0000_H

RBCU_ECON

RBCU Error Control Capture Register (20_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
ERRCNT	[15:0]	rwh	FPI Bus Error Counter ERRCNT is incremented on every occurrence of an FPI Bus error. ERRCNT is reset to 0000 _H after the ECON register is read. ¹⁾
TOUT	16	rwh	State of FPI Bus Time-Out Signal This bit indicates the state of the time-out signal at an FBI Bus error. 0 _B No time-out occurred. 1 _B Time-out has occurred.
RDY	17	rwh	State of FPI Bus Ready Signal This bit indicates the state of the ready signal at an FBI Bus error. 0 _B Wait state(s) have been inserted. Ready signal was active. 1 _B Ready signal was inactive.
ABT	18	rwh	State of FPI Bus Abort Signal This bit indicates the state of the abort signal at an FBI Bus error. 0 _B Master has aborted an FPI Bus transfer. Abort signal was active. 1 _B Abort signal was inactive.

On-Chip System Buses and Bus Bridges

Field	Bits	Type	Description
ACK	[20:19]	rwh	State of FPI Bus Acknowledge Signals This bit field indicates the acknowledge code that has been output by the selected slave at an FPI Bus error. Coding see Table 6-10 .
SVM	21	rwh	State of FPI Bus Supervisor Mode Signal This bit indicates whether the FPI Bus error occurred in supervisor mode or in user mode. 0 _B Transfer was initiated in supervisor mode. 1 _B Transfer was initiated in user modes.
WRN	22	rwh	State of FPI Bus Write Signal This bit indicates whether the FPI Bus error occurred at a write cycle (see Table 6-15).
RDN	23	rwh	State of FPI Bus Read Signal This bit indicates whether the FPI Bus error occurred at a read cycle (see Table 6-15).
TAG	[27:24]	rwh	FPI Bus Master Tag Number Signals This bit field indicates the FPI Bus master TAG number (definitions see Table 6-11).
OPC	[31:28]	rwh	FPI Bus Operation Code Signals The FPI Bus operation codes are defined in Table 6-12 .

1) For the SBCU_ECON.ERRCNT bit field, the following additional sentence must be added: in the TC1796, aborted accesses to a 0 wait state SPB slave may also increment ERRCNT when the slave generates an error acknowledge.

Table 6-15 FPI Bus Read/Write Error Indication

RD	WR	FPI Bus Cycle
0	0	FPI Bus error occurred at the read transfer of a read-modify-write transfer.
0	1	FPI Bus error occurred at a read cycle of a single transfer.
1	0	FPI Bus error occurred at a write cycle of a single transfer or at the write cycle of a read-modify-write transfer.
1	1	Does not occur

On-Chip System Buses and Bus Bridges

6.5.5.3 OCDS Registers

SBCU_DBCNTL

SBCU Debug Control Register (30_H) Reset Value: 0000 7003_H

RBCU_DBCNTL

RBCU Debug Control Register (30_H) Reset Value: 0000 7003_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ON BOS 3	ON BOS 2	ON BOS 1	ON BOS 0	0		ONA2	0		ONA1		0				ONG
r	r	r	r	r		r	r		r		r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CON COM 2	CON COM 1	CON COM 0				0				RA	0	OA	EO	
r	r	r	r				r				w	r	r	r	r

Field	Bits	Type	Description
EO	0	r	<p>Status of BCU Debug Support Enable</p> <p>This bit is controlled by the Cerberus and enables the BCU debug support.</p> <p>0_B BCU debug support is disabled.</p> <p>1_B BCU debug support is enabled. (default after reset)</p>
OA	1	r	<p>Status of BCU Breakpoint Logic</p> <p>0_B The BCU breakpoint logic is disarmed. Any further breakpoint activation is discarded.</p> <p>1_B The BCU breakpoint logic is armed.</p> <p>The OA bit is set by writing a 1 to bit RA. When OA is set, registers xBCU_DBGNTT, xBCU_DBADRT, and xBCU_DBBOST are reset.</p>
RA	4	w	<p>Rearm BCU Breakpoint Logic</p> <p>Writing a 1 to this bit rearms BCU breakpoint logic and sets bit OA = 1. RA is always reads as 0.</p>

On-Chip System Buses and Bus Bridges

Field	Bits	Type	Description
CONCOM0	12	rw	<p>Grant and Address Trigger Relation</p> <p>0_B Grant trigger condition and the address trigger condition are combined with a logical OR for further control.</p> <p>1_B The grant trigger condition and the address trigger condition are combined with a logical AND for further control (see Figure 6-12).</p>
CONCOM1	13	rw	<p>Address 1 and Address 2 Trigger Relation</p> <p>0_B Address 1 trigger condition and address 2 trigger condition are combined with a logical OR for further control.</p> <p>1_B Address 1 trigger condition and address 2 trigger condition are combined with a logical AND for further control (see Figure 6-12).</p>
CONCOM2	14	rw	<p>Address and Signal Trigger Relation</p> <p>0_B Address trigger condition and signal trigger condition are combined with a logical OR for further control.</p> <p>1_B Address trigger condition and the signal trigger condition are combined with a logical AND for further control (see Figure 6-12).</p>
ONG	16	rw	<p>Grant Trigger Enable</p> <p>0_B No grant debug event trigger is generated.</p> <p>1_B The grant debug event trigger is enabled and generated according the settings of register DBGRNT (see Figure 6-11).</p>
ONA1	[21:20]	rw	<p>Address 1 Trigger Control</p> <p>00_B No address 1 trigger is generated.</p> <p>01_B An address 1 trigger event is generated if the FPI Bus address is equal to DBADR1.</p> <p>10_B An address 1 trigger event is generated if FPI Bus address is greater or equal to DBADR1.</p> <p>11_B same as 00_B. See also Figure 6-9.</p>

On-Chip System Buses and Bus Bridges

Field	Bits	Type	Description
ONA2	[25:24]	rw	<p>Address 2 Trigger Control</p> <p>00_B No address 2 trigger is generated.</p> <p>01_B An address 2 trigger event is generated if the FPI Bus address is equal to DBADR2.</p> <p>10_B An address 2 trigger event is generated if FPI Bus address is greater or equal to DBADR2.</p> <p>11_B same as 00_B.</p> <p>See also Figure 6-9.</p>
ONBOS0	28	rw	<p>Opcode Signal Status Trigger Condition</p> <p>0_B A signal status trigger is generated for all FPI Bus opcodes except a “no operation” opcode.</p> <p>1_B A signal status trigger is generated if the FPI Bus opcode matches the opcode as defined in DBBOS.OPC (see Figure 6-10).</p>
ONBOS1	29	rw	<p>Supervisor Mode Signal Trigger Condition</p> <p>0_B The signal status trigger generation for the FPI Bus supervisor mode signal is disabled.</p> <p>1_B A signal status trigger is generated if the FPI Bus supervisor mode signal state is equal to the value of DBBOS.SVM (see Figure 6-10).</p>
ONBOS2	30	rw	<p>Write Signal Trigger Condition</p> <p>0_B The signal status trigger generation for the FPI Bus write signal is disabled.</p> <p>1_B A signal status trigger is generated if the FPI Bus write signal state is equal to the value of DBBOS.WR (see Figure 6-10).</p>
ONBOS3	31	rw	<p>Read Signal Trigger Condition</p> <p>0_B The signal status trigger generation for the FPI Bus read signal is disabled.</p> <p>1_B A signal status trigger is generated if the FPI Bus read signal state is equal to the value of DBBOS.RD (see Figure 6-10).</p>

On-Chip System Buses and Bus Bridges

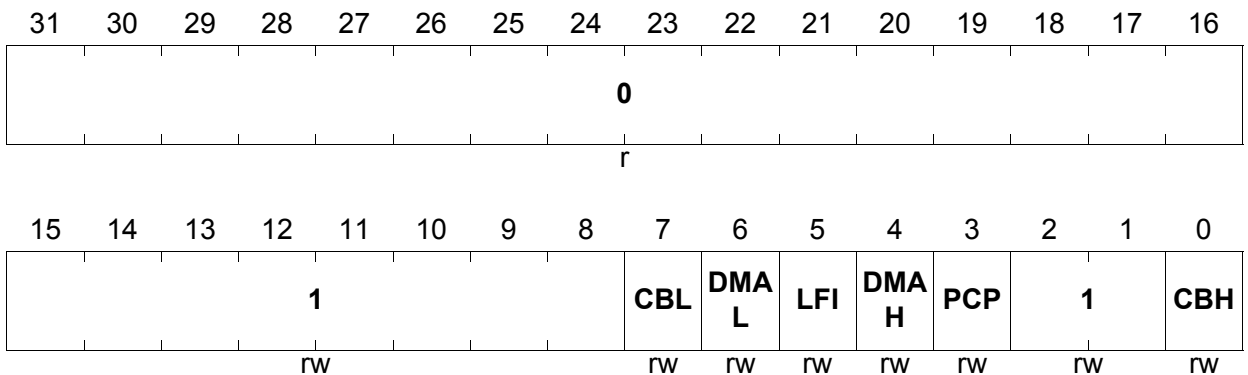
Field	Bits	Type	Description
0	[3:2], [11:5], 15, [19:17], [23:22], [27:26]	r	Reserved Read as 0; should be written with 0.

SBCU_DBGRT

SBCU Debug Grant Mask Register

(34_H)

Reset Value: 0000 FFFF_H



Field	Bits	Type	Description
CBH	0	rw	Cerberus Grant Trigger Enable, High Priority 0 _B FPI Bus transactions on SPB with high-priority Cerberus as bus master are enabled for grant trigger event generation. 1 _B FPI Bus transactions on SPB with high-priority Cerberus as bus master are disabled for grant trigger event generation.
1	[2:1], [15:8]	rw	Reserved Read as 1 after reset; reading these bits will return the value last written.
PCP	3	rw	PCP Grant Trigger Enable 0 _B FPI Bus transactions on SPB with PCP as bus master are enabled for grant trigger event generation. 1 _B FPI Bus transactions on SPB with PCP as bus master are disabled for grant trigger event generation.

On-Chip System Buses and Bus Bridges

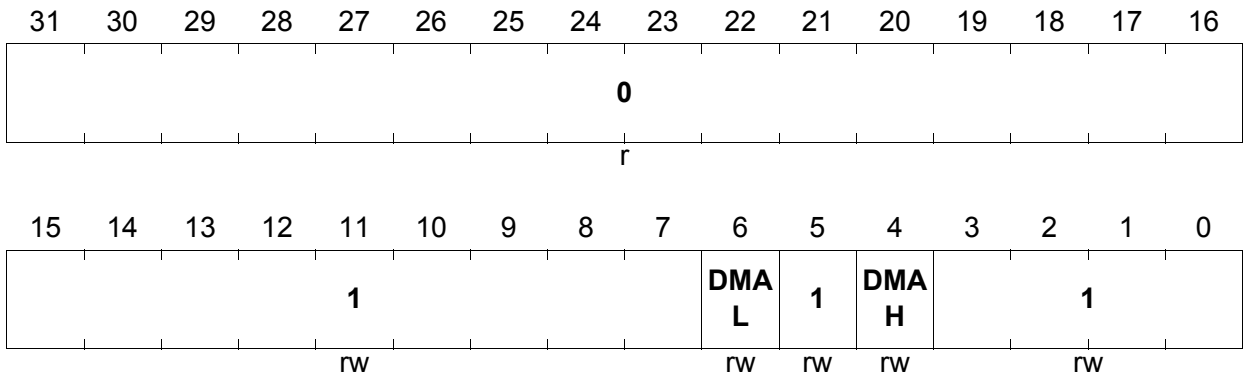
Field	Bits	Type	Description
DMAH	4	rw	<p>DMA Grant Trigger Enable, High Priority</p> <p>0_B FPI Bus transactions on SPB with low-priority DMA channels as bus master are enabled for grant trigger event generation.</p> <p>1_B FPI Bus transactions on SPB with low-priority DMA channels as bus master are disabled for grant trigger event generation.</p>
LFI	5	rw	<p>LFI Bridge Grant Trigger Enable</p> <p>0_B FPI Bus transactions on SPB with LFI Bridge as bus master are enabled for grant trigger event generation.</p> <p>1_B FPI Bus transactions on SPB with LFI Bridge as bus master are disabled for grant trigger event generation.</p>
DMAL	6	rw	<p>DMA Grant Trigger Enable, Low Priority</p> <p>0_B FPI Bus transactions on SPB with high-priority DMA channels as bus master are enabled for grant trigger event generation.</p> <p>1_B FPI Bus transactions on SPB with high-priority DMA channels as bus master are disabled for grant trigger event generation.</p>
CBL	7	rw	<p>Cerberus Grant Trigger Enable, Low Priority</p> <p>0_B FPI Bus transactions on SPB with low-priority Cerberus as bus master are enabled for grant trigger event generation.</p> <p>1_B FPI Bus transactions on SPB with low-priority Cerberus as bus master are disabled for grant trigger event generation.</p>
0	[31:16]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

On-Chip System Buses and Bus Bridges

RBCU_DBGRNT

RBCU Debug Grant Mask Register (34_H)

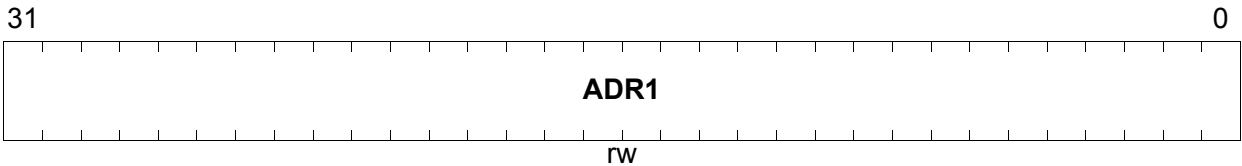
Reset Value: 0000 FFFF_H



Field	Bits	Type	Description
1	[3:0], 5, [15:7]	rw	Reserved Read as 1 after reset; reading these bits will return the value last written.
DMAH	4	rw	DMA Grant Trigger Enable, High Priority 0 _B FPI Bus transactions on RPB with low-priority DMA channels as bus master are enabled for grant trigger event generation. 1 _B FPI Bus transactions on RPB with low-priority DMA channels as bus master are disabled for grant trigger event generation.
DMAL	6	rw	DMA Grant Trigger Enable, Low Priority 0 _B FPI Bus transactions on RPB with high-priority DMA channels as bus master are enabled for grant trigger event generation. 1 _B FPI Bus transactions on RPB with high-priority DMA channels as bus master are disabled for grant trigger event generation.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

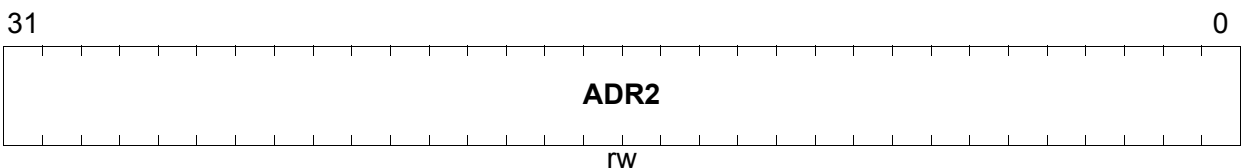
On-Chip System Buses and Bus Bridges

SBCU_DBADR1
SBCU Debug Address 1 Register (38_H) **Reset Value: 0000 0000_H**
RBCU_DBADR1
RBCU Debug Address 1 Register (38_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
ADR1	[31:0]	rw	Debug Trigger Address 1 This register contains the address for the address 1 trigger event generation.

SBCU_DBADR2
SBCU Debug Address 2 Register (3C_H) **Reset Value: 0000 0000_H**
RBCU_DBADR2
RBCU Debug Address 2 Register (3C_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
ADR2	[31:0]	rw	Debug Trigger Address 2 This register contains the address for the address 2 trigger event generation.

On-Chip System Buses and Bus Bridges

SBCU_DBBOS

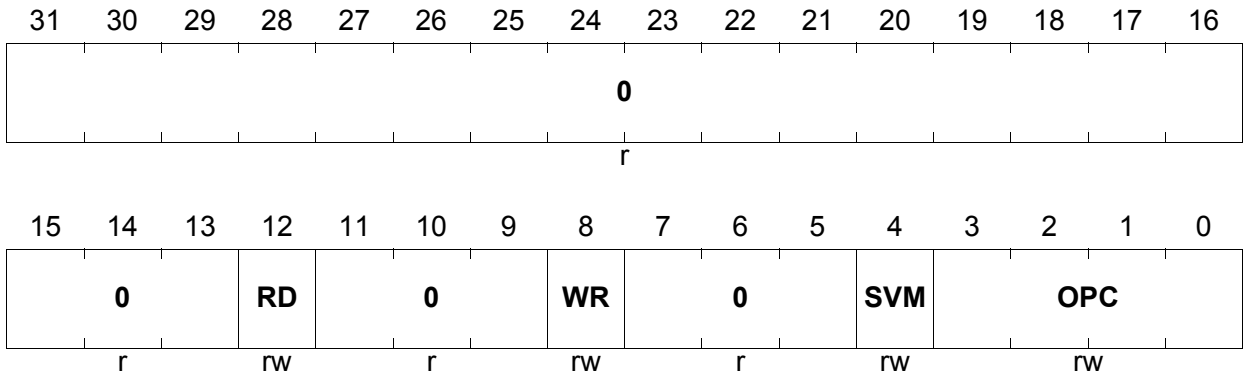
SBCU Debug Bus Operation Signals Register
(40_H)

Reset Value: 0000 0000_H

RBCU_DBBOS

RBCU Debug Bus Operation Signals Register
(40_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
OPC	[3:0]	rw	<p>Opcode for Signal Status Debug Trigger This bit field determines the type (opcode) of a FPI Bus transaction for which a signal status debug trigger event is generated (if enabled by DBCNTL.ONBOS0 = 1).</p> <p>0000_B Trigger on single byte transfer selected 0001_B Trigger on single half-word transfer selected 0010_B Trigger on single word transfer selected 0100_B Trigger on 2-word block transfer selected 0101_B Trigger on 4-word block transfer selected 0110_B Trigger on 8-word block transfer selected 1111_B Trigger on no operation selected others Reserved</p>
SVM	4	rw	<p>SVM Signal for Status Debug Trigger This bit determines the mode of a FPI Bus transaction for which a signal status debug trigger event is generated (if enabled by DBCNTL.ONBOS1 = 1).</p> <p>0_B Trigger on user mode selected 1_B Trigger on supervisor mode selected</p>

On-Chip System Buses and Bus Bridges

Field	Bits	Type	Description
WR	8	rw	<p>Write Signal for Status Debug Trigger This bit determines the state of the \overline{WR} signal of an FPI Bus transaction for which a signal status debug trigger event is generated (if enabled by DBCNTL.ONBOS2 = 1).</p> <p>0_B Trigger on a single write transfer or write cycle of an atomic transfer selected</p> <p>1_B No operation or read transaction selected</p>
RD	12	rw	<p>Write Signal for Status Debug Trigger This bit determines the state of the \overline{RD} signal of an FPI Bus transaction for which a signal status debug trigger event is generated (if enabled by DBCNTL.ONBOS3 = 1).</p> <p>0_B Trigger on a single read transfer or read cycle of an atomic transfer selected</p> <p>1_B No operation or write transfer selected</p>
0	[7:5], [11:9], [31:13]	r	<p>Reserved Read as 0; should be written with 0.</p>

On-Chip System Buses and Bus Bridges

Field	Bits	Type	Description
LFI	5	rh	<p>LFI Bridge FPI Bus Master Status This bit indicates whether the LFI Bridge was SPB bus master when the break trigger event occurred.</p> <p>0_B The LFI Bridge was not a SPB bus master. 1_B The LFI Bridge was SPB bus master.</p>
DMAL	6	rh	<p>Low-Priority DMA FPI Bus Master Status This bit indicates whether the low priority DMA channels were SPB bus master when the break trigger event occurred.</p> <p>0_B The low priority DMA channels were not a SPB bus master. 1_B The low priority DMA channels were SPB bus master. Bits CHNR_{xy} determine the DMA channel number.</p>
CBL	7	rh	<p>Low-Priority Cerberus FPI Bus Master Status This bit indicates whether the low-priority Cerberus was SPB bus master when the break trigger event occurred.</p> <p>0_B The low-priority Cerberus was not a SPB bus master. 1_B The low-priority Cerberus was SPB bus master.</p>
CHNR0_y (y = 0-7)	16 + y	rh	<p>DMA Channel Number Status These bits indicate which DMA channel with number 0_y was active when a DMA break trigger event occurred.</p> <p>0_B DMA channel 0_y was not active at a DMA break trigger event. 1_B DMA channel 0_y was active at a DMA break trigger event.</p>
CHNR1_y (y = 0-7)	24+ y	rh	<p>DMA Channel Number Status These bits indicate which DMA channel with number 1_y was active when a DMA break trigger event occurred.</p> <p>0_B DMA channel 1_y was not active at a DMA break trigger event. 1_B DMA channel 1_y was active at a DMA break trigger event.</p>
1	[2:1], [15:8]	r	<p>Reserved Read as 1.</p>

On-Chip System Buses and Bus Bridges

RBCU_DBGNTT

RBCU Debug Trapped Master Register

(44_H)

Reset Value: FFFF FFFF_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CH	CH	CH	CH	CH	CH	CH	CH	CH	CH	CH	CH	CH	CH	CH	CH
NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR
17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									DMA	1	DMA				
									L	r	H				
									r	rh	rh				

Field	Bits	Type	Description
1	[3:0], 5, [15:7]	rw	Reserved Read as 1.
DMAH	2	rh	High-Priority DMA FPI Bus Master Status This bit indicates whether the high-priority DMA channels were RPB bus master when the break trigger event occurred. 0_B The high-priority DMA channels were not a RPB bus master. 1_B The high-priority DMA channels were RPB bus master. Bits CHNRxy determine the DMA channel number.
DMAL	4	rh	Low-Priority DMA FPI Bus Master Status This bit indicates whether the low priority DMA channels were RPB bus master when the break trigger event occurred. 0_B The low priority DMA channels were not a RPB bus master. 1_B The low priority DMA channels were RPB bus master. Bits CHNRxy determine the DMA channel number.

On-Chip System Buses and Bus Bridges

Field	Bits	Type	Description
CHNR0y (y = 0-7)	16 + y	rh	DMA Channel Number Status These bits indicate which DMA channel with number 1y was active when a DMA break trigger event occurred at the RPB. 0 _B DMA channel 1y was not active at a DMA break trigger event at the RPB. 1 _B DMA channel 1y was active at a DMA break trigger event at the RPB.
CHNR1y (y = 0-7)	24 + y	rh	DMA Channel Number Status These bits indicate which DMA channel with number 1y was active when a DMA break trigger event occurred at the RPB. 0 _B DMA channel 1y was not active at a DMA break trigger event at the RPB. 1 _B DMA channel 1y was active at a DMA break trigger event at the RPB.

SBCU_DBADRT

SBCU Debug Trapped Address Register

(48_H)

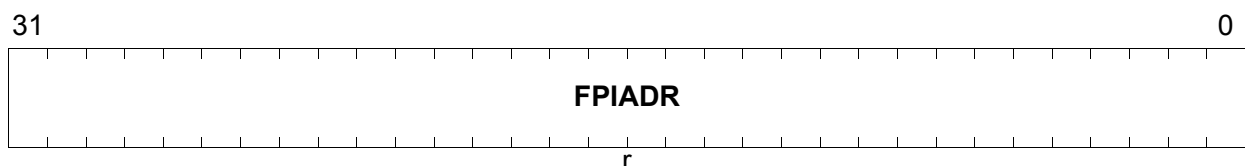
Reset Value: 0000 0000_H

RBCU_DBADRT

RBCU Debug Trapped Address Register

(48_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
FPIADR	[31:0]	r	FPI Bus Address Status This register contains the FPI Bus address that was captured when the OCDS break trigger event occurred.

On-Chip System Buses and Bus Bridges

SBCU_DBBOST

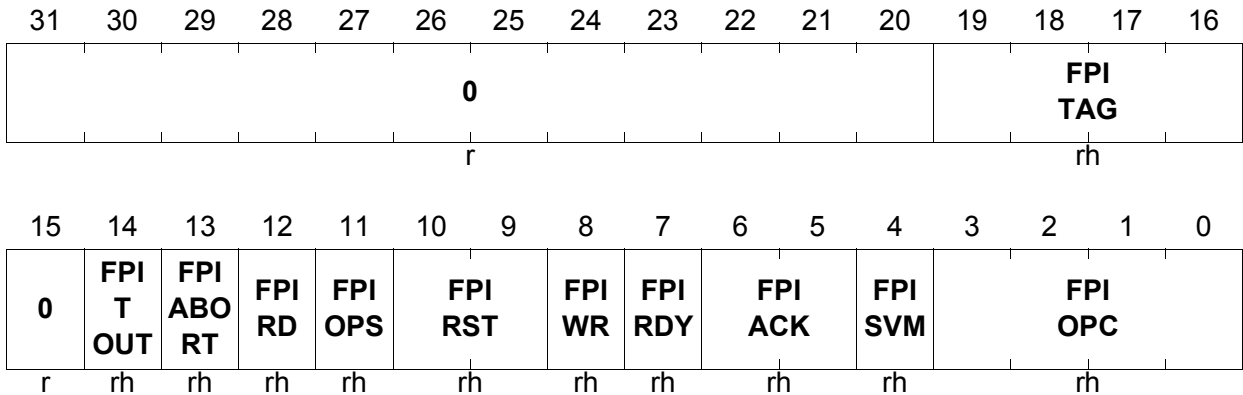
SBCU Debug Trapped Bus Operation Signals Register
(4C_H)

Reset Value: 0000 3180_H

RBCU_DBBOST

RBCU Debug Trapped Bus Operation Signals Register
(4C_H)

Reset Value: 0000 3180_H



Field	Bits	Type	Description
FPIOPC	[3:0]	rh	<p>FPI Bus Opcode Status</p> <p>This bit field indicates the type (opcode) of the FPI Bus transaction captured from the FPI Bus signal lines when the BCU break trigger event occurred.</p> <p>0000_B Single byte transfer 0001_B Single half-word transfer 0010_B Single word transfer 0100_B 2-word block transfer 0101_B 4-word block transfer 0110_B 8-word block transfer 1111_B No operation others Reserved</p>
FPISVM	4	rh	<p>FPI Bus Supervisor Mode Status</p> <p>This bit indicates the state of the supervisor mode signal captured from the FPI Bus signal lines when the BCU break trigger event occurred.</p> <p>0_B User mode 1_B Supervisor mode</p>

On-Chip System Buses and Bus Bridges

Field	Bits	Type	Description
FPIACK	[6:5]	rh	<p>FPI Bus Acknowledge Status This bit field indicates the acknowledge signal status captured from the FPI Bus signal lines when the BCU break trigger event occurred.</p> <p>00_B No special case 01_B Error 10_B Reserved 11_B Retry, slave did not respond</p>
FPIRDY	7	rh	<p>FPI Bus Ready Status This bit indicates the ready signal status captured from the FPI Bus signal lines when the BCU break trigger event occurred.</p> <p>0_B Last cycle of transfer 1_B Not last cycle of transfer</p>
FPIWR	8	rh	<p>FPI Bus Write Indication Status This bit indicates the write signal status captured from the FPI Bus signal lines when the BCU break trigger event occurred.</p> <p>0_B Single write transfer or write cycle of an atomic transfer 1_B No operation or read transfer</p>
FPIRST	[10:9]	rh	<p>FPI Bus Reset Status This bit field indicates the reset signal status captured from the FPI Bus signal lines when the BCU break trigger event occurred.</p> <p>00_B Reset of all FPI Bus components 11_B No reset others Reserved</p>
FPIOPS	11	rh	<p>FPI Bus OCDS Suspend Status This bit indicates the OCDS suspend signal status captured from the FPI Bus signal lines when the BCU break trigger event occurred.</p> <p>0_B No OCDS suspend request is pending 1_B An OCDS suspend request is pending</p>

On-Chip System Buses and Bus Bridges

Field	Bits	Type	Description
FPIRD	12	rh	<p>FPI Bus Read Indication Status This bit indicates the read signal status captured from the FPI Bus signal lines when the BCU break trigger event occurred.</p> <p>0_B Single read transfer or read cycle of an atomic transfer 1_B No operation or write transfer</p>
FPIABORT	13	rh	<p>FPI Bus Abort Status This bit indicates the abort signal status captured from the FPI Bus signal lines when the BCU break trigger event occurred.</p> <p>0_B A transfer that had already started was aborted 1_B Normal operation</p>
FPITOUT	14	rh	<p>FPI Bus Time-out Status This bit indicates the time-out signal status captured from the FPI Bus signal lines when the BCU break trigger event occurred.</p> <p>0_B Normal operation 1_B A time-out event was generated</p>
FPITAG	[19:16]	rh	<p>FPI Bus Master TAG Status This bit indicates the master TAG captured from the FPI Bus signal lines when the BCU break trigger event occurred. The master TAG identifies the master of the transfer which generated BCU break trigger event.</p> <p>0000_B Cerberus Interface (high-priority) 0001_B Peripheral Control Processor (PCP) 0010_B DMA Controller (high-priority channels) 0011_B LFI Bridge 0100_B DMA Controller (low-priority channels) 0101_B Cerberus Interface (low-priority) others Reserved</p>
0	15, [31:20]	rh	<p>Reserved Read as 0; should be written with 0.</p>

On-Chip System Buses and Bus Bridges

6.5.5.4 BCU Service Request Control Register

In case of a bus error, the BCU generates an interrupt request to the selected service provider (usually the CPU). This interrupt request is controlled through a standard service request control register.

SBCU_SRC

SBCU Service Request Control Register

(FC_H)

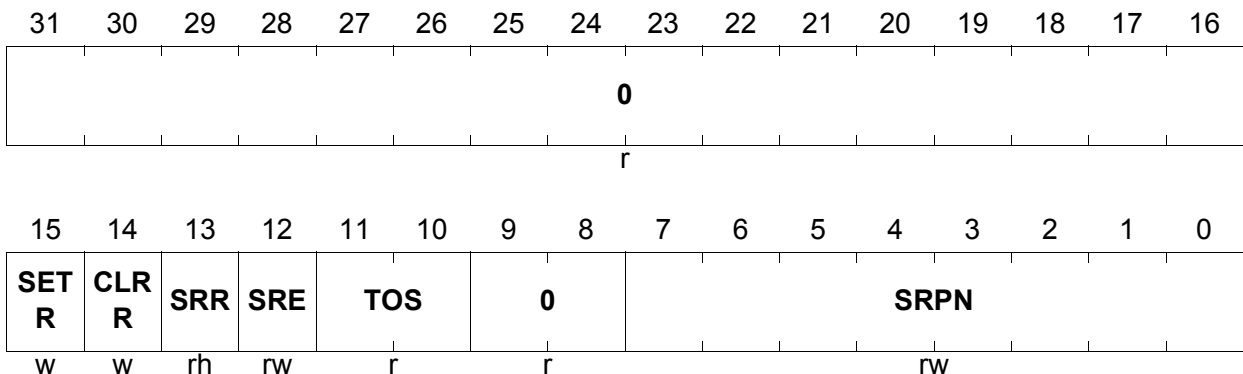
Reset Value: 0000 0000_H

RBCU_SRC

RBCU Service Request Control Register

(FC_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SRPN	[7:0]	rw	Service Request Priority Number
TOS	[11:10]	r	Type of Service Control The SBCU/RBCU can only be serviced by the CPU.
SRE	12	rw	Service Request Enable
SRR	13	rh	Service Request Flag
CLRR	14	w	Request Clear Bit
SETR	15	w	Request Set Bit
0	[9:8], [31:16]	r	Reserved Read as 0; should be written with 0.

Note: Further details on interrupt handling and processing are described in [Chapter 14](#) of this TC1796 System Units (Vol. 1 of 2) User's Manual.

7 Program Memory Unit

The Program Memory Unit (PMU) shown in [Figure 7-1](#) contains the following functional units:

- 2 Mbyte of Program Flash Memory (PFLASH)
- 128 Kbyte of Data Flash Memory (DFLASH)
- 16 Kbyte of Boot ROM (BROM)
 - 8 Kbyte Boot Code ROM
 - 8 Kbyte Factory Test ROM
- Program Local Memory Bus Interface
- Emulation Memory Interface

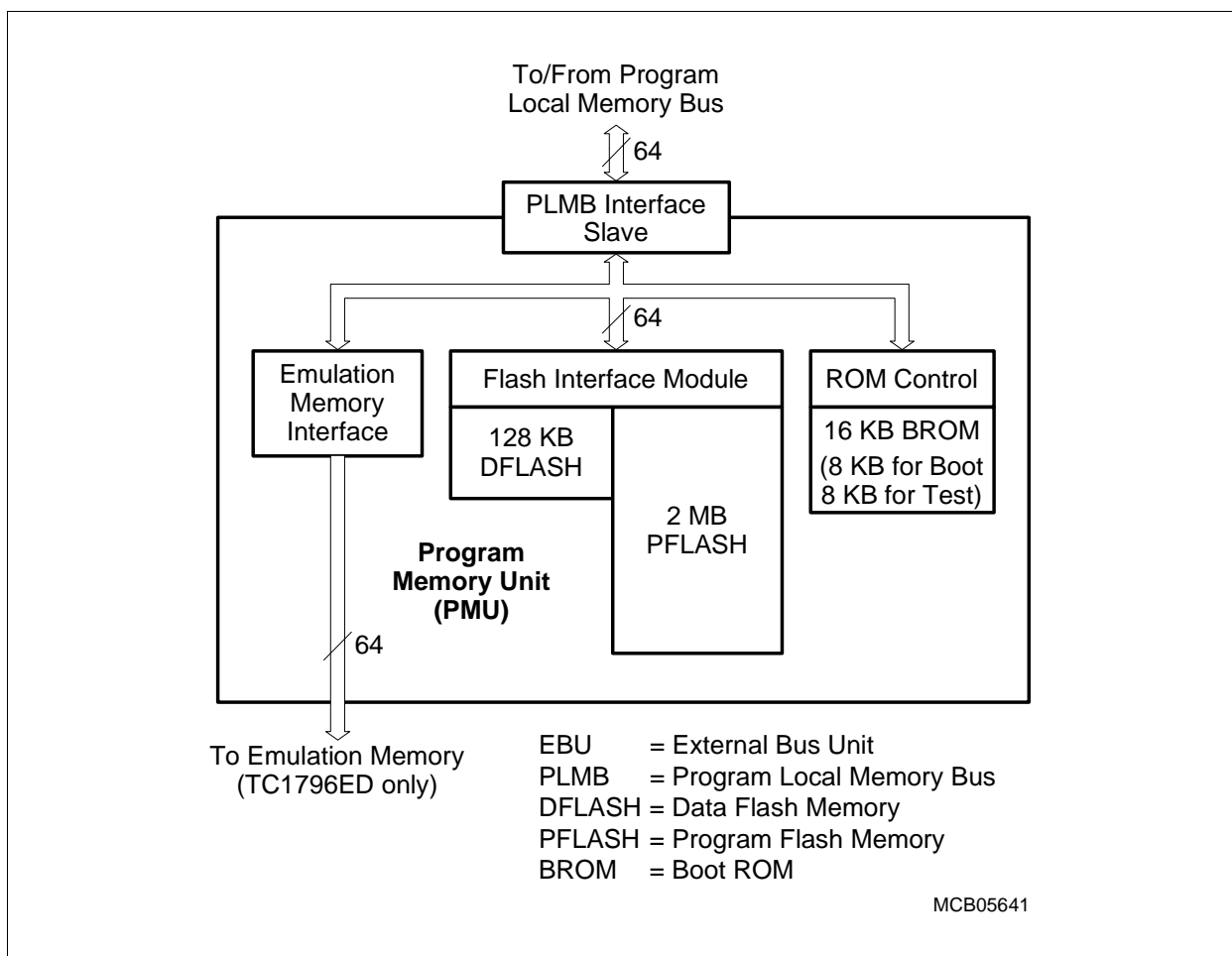


Figure 7-1 PMU Block Diagram

7.1 Boot ROM

The functionality of the 16 Kbyte Boot ROM (BROM) is described in [Chapter 4](#) (Reset and Boot Operation) at [Page 4-16](#).

7.2 Program & Data Flash Memory

The embedded Flash module of TC1796 incorporates a 2 Mbyte Flash memory for program code or constant data (called PFLASH or program Flash) and a 128 Kbyte Flash memory used for data storage (called DFLASH or data Flash).

Both, PFLASH and DFLASH, provide error correction of single-bit errors within a 64-bit read double-word, resulting in a very low failure rate. The programming quantity is one page, including 256 byte for the PFLASH and 128 byte for the DFLASH.

The PFLASH is implemented as one Flash bank. The DFLASH is built up by two Flash banks. This bank configuration allows combinations of concurrent Flash operations:

- Reading code or data from PFLASH while one bank of the DFLASH is busy with a program or erase operation.
- Reading data from one bank of the DFLASH while the other bank of the DFLASH is busy with a program or erase operation.
- Programming one bank of the DFLASH while the other bank of the DFLASH is busy with an erase operation and simultaneously reading from PFLASH.

Note: It is not possible to read data from DFLASH while the PFLASH is busy with a program or erase operation.

The embedded Flash module is divided into the following two sub-modules:

- The Flash Interface and Control Module (FIM)
 - Controls the execution of Flash commands (Flash command state machine FCS)
 - Handles error correction and ECC generation
 - Provides a PLMB bus interface to the PMI for instruction accesses and to the DMI module (via the LMI-Bridge) for data accesses.
- The Flash Array Module (FAM)
 - One PFLASH bank of 2 Mbyte
 - Two PFLASH banks of 64 Kbyte each
 - Control logic that includes assembly buffers and voltage generators, for example

The FIM and FAM are main parts of the Program Memory Unit (PMU). An overview of the PMU integration into the system architecture is shown in the TC1796 block and bus system diagrams (see [Page 1-9](#) and [Page 6-1](#)). A basic diagram of the Flash modules is shown in [Figure 7-2](#).

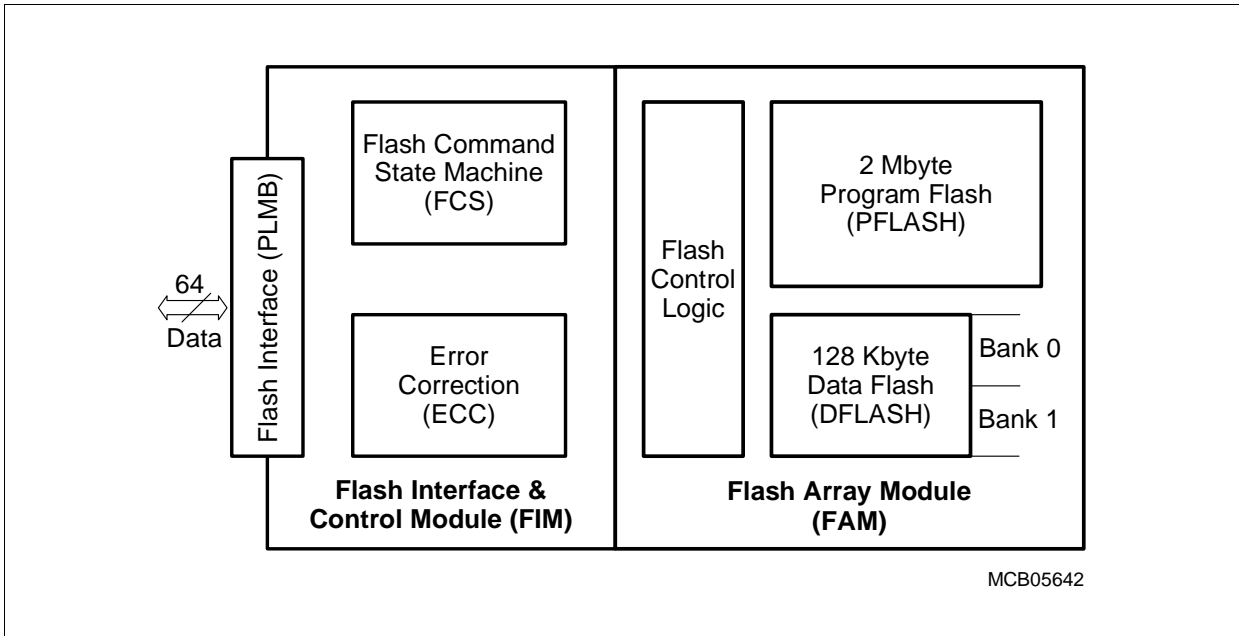


Figure 7-2 Basic Diagram of Flash Module

All Flash operations are initiated by transferring command sequences to the Flash that are based on the JEDEC standard. The command sequences to the Flash are high-level commands such as “Erase Sector”. State transitions during execution of the commands, such as termination of command execution or errors, are indicated by status flags and maskable interrupts. Command sequences are normally written to the PMU by the CPU but can also be issued by the PCP or the DMA controller.

7.2.1 Program Flash Overview

The PFLASH memory has a capacity of 2 Mbyte. The internal structure of the PFLASH is based on a sector architecture. For flexible erase, programming, and protection capability, the 2 Mbyte are divided into 13 sectors of eight 16 Kbyte sectors (combined into two 64 Kbyte physical sectors), one 128 Kbyte sector, one 256 Kbyte sector, and three 512 Kbyte sectors. PFLASH sectors are numbered as PSx with x = 0 to 12.

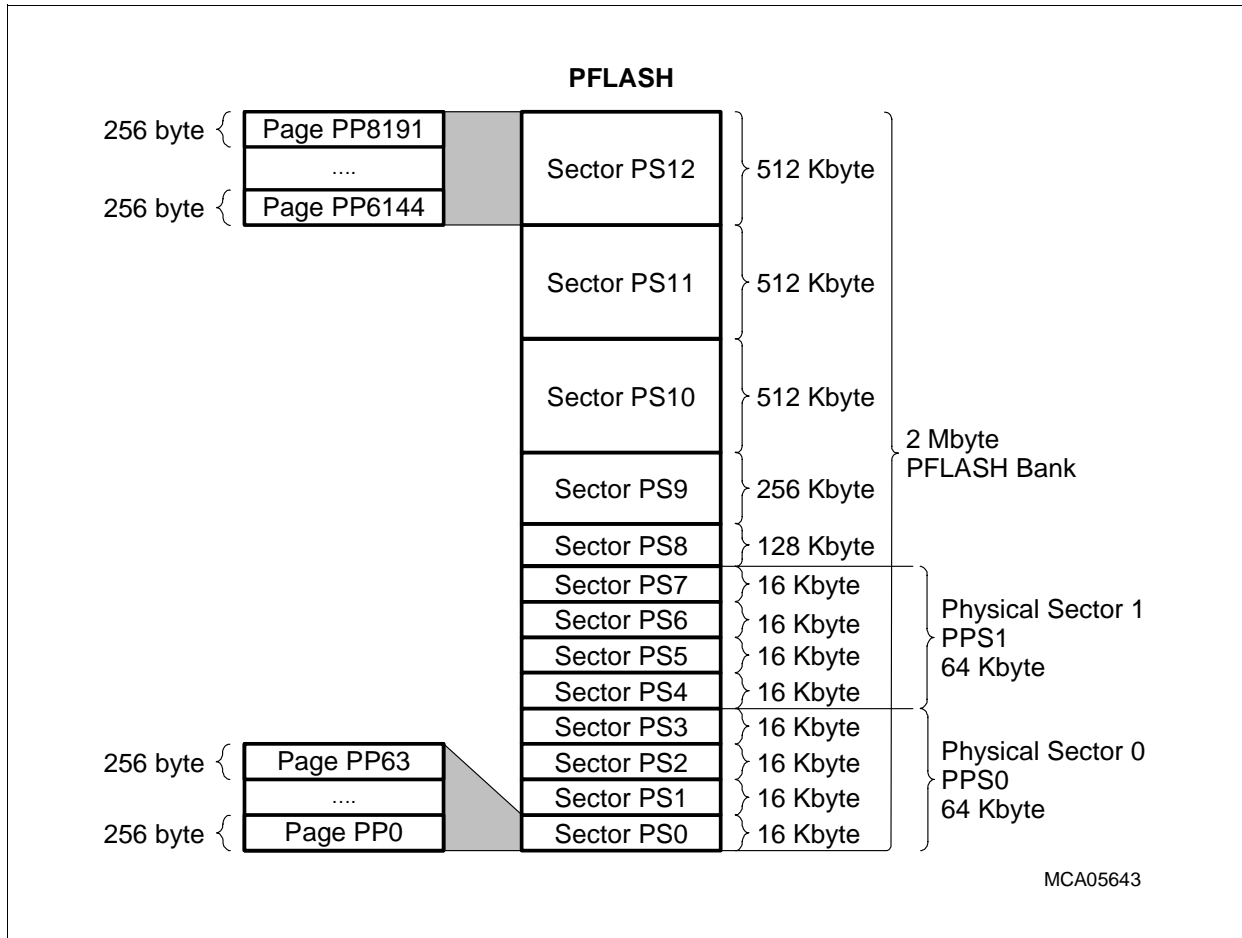


Figure 7-3 PFLASH Structure

The PFLASH operates in paging mode which makes it possible to load 64 words (= 256 byte) into a page assembly buffer with fast CPU accesses before this buffer is programmed into the Flash memory with one Write Page command. Thus, the programming width is always 256 byte. Programming of single words, bytes, or bits is not supported. The 256-byte wide PFLASH pages are numbered with PPy with y = 0 to 8191.

Only one complete sector can be erased. Program and erase operations are initiated and supervised by the Flash Command State Machine (FCS), but its execution is further controlled by control logic in the Flash Array Module (FAM).

Program Memory Unit

The PFLASH array delivers 256-bit wide read data with one read access. In the TC1796, these 256 bits are transferred to the CPU and PMI via the 64-bit wide Program Local Memory Bus (PLMB) using single-cycle burst transfers with four 64-bit transfers. During such a burst transfer, the next sequential 256-bit read data is prefetched from the PFLASH. Therefore, except a delay of the first initial read data access, sequential burst transfers are provided, supporting the highest possible instruction throughput from PFLASH to the CPU with execution of two 32-bit instructions in one cycle. Also non-cached instruction accesses are burst accesses with up to four double-word transfers.

Each PFLASH sector can be separately locked against erasing and reprogramming. Write operations to a locked and protected sector are only possible only if the sector is temporarily unlocked using a dedicated password check sequence. Additionally, also an OTP (One Time Programmable) function can be selected for each sector. An OTP sector is locked forever, and erasing and reprogramming is no longer possible.

Features of Program Flash

- 2 Mbyte on-chip program Flash memory
- Usable for instruction code execution or constant data storage
- 256-byte wide program interface
 - 256 bytes are programmed into PFLASH page in one step/command
- 256-bit read interface
 - Transfer from PFLASH to CPU/PMI by four 64-bit single-cycle burst transfers
- Dynamic correction of single-bit errors during read access
- Detection of double bit errors
- Fixed sector architecture
 - Eight 16 Kbyte, one 128 Kbyte, one 256 Kbyte, and three 512 Kbyte sectors
 - Each sector separately erasable
 - Each sector separately write-protectable
- Configurable read protection for complete PFLASH with sophisticated read access supervision, combined with write protection for complete PFLASH (protection against “Trojan horse” software)
- Configurable write protection for each sector
 - Each sector separately write-protectable
 - With capability to be re-programmed
 - With capability to be locked forever (OTP)
- Password mechanism for temporarily disable write or read protection
- On-chip programming voltage generation
- PFLASH is delivered in erased state (read all zeros)
- JEDEC standard based command sequences for PFLASH control
 - Write state machine controls programming and erase operations
 - Status and error reporting by status flags and interrupt
- Margin check for detection of problematic PFLASH bits

Program Memory Unit

Table 7-1 defines the sectors with their sector numbers, sector sizes, and address ranges. The PFLASH contains thirteen sectors PS[12:0] with different sizes. Two 64 Kbyte physical sectors PPS[1:0] are defined for the PFLASH.

Table 7-1 PFLASH Bank, Sector and Page Definitions

Numbering		Size	Cached Address Range	Non-Cached Address Range
PFLASH Bank				
PB		2 Mbyte	8000 0000 _H - 801F FFFF _H	A000 0000 _H - A01F FFFF _H
PFLASH Sectors				
PS0	PPS0 ¹⁾	16 Kbyte	8000 0000 _H - 8000 3FFF _H	A000 0000 _H - A000 3FFF _H
PS1		PPS0 ²⁾	8000 4000 _H - 8000 7FFF _H	A000 4000 _H - A000 7FFF _H
PS2		16 Kbyte	8000 8000 _H - 8000 BFFF _H	A000 8000 _H - A000 BFFF _H
PS3		16 Kbyte	8000 C000 _H - 8000 FFFF _H	A000 C000 _H - A000 FFFF _H
PS4	PPS1	16 Kbyte	8001 0000 _H - 8001 3FFF _H	A001 0000 _H - A001 3FFF _H
PS5		16 Kbyte	8001 4000 _H - 8001 7FFF _H	A001 4000 _H - A001 7FFF _H
PS6		16 Kbyte	8001 8000 _H - 8001 BFFF _H	A001 8000 _H - A001 BFFF _H
PS7		16 Kbyte	8001 C000 _H - 8001 FFFF _H	A001 C000 _H - A001 FFFF _H
PS8		128 Kbyte	8002 0000 _H - 8003 FFFF _H	A002 0000 _H - A003 FFFF _H
PS9		256 Kbyte	8004 0000 _H - 8007 FFFF _H	A004 0000 _H - A007 FFFF _H
PS10		512 Kbyte	8008 0000 _H - 800F FFFF _H	A008 0000 _H - A00F FFFF _H
PS11		512 Kbyte	8010 0000 _H - 8017 FFFF _H	A010 0000 _H - A017 FFFF _H
PS12		512 Kbyte	8018 0000 _H - 801F FFFF _H	A018 0000 _H - A01F FFFF _H
PFLASH Pages				
PP0		256 byte	8000 0000 _H - 8000 00FF _H	A000 0000 _H - A000 00FF _H
PPn ³⁾		256 byte	(8000 0000 _H + OFF) - (8000 0000 _H + OFF + FF _H) (with OFF = n × 100 _H)	(A000 0000 _H + OFF) - (A000 0000 _H + OFF + FF _H) (with OFF = n × 100 _H)
PP8191		256 byte	801F FF00 _H - 801F FFFF _H	A01F FF00 _H - A01F FFFF _H

1) PPSx (x = 0, 1) is a physical PFLASH sector.

2) PPSx (x = 0, 1) is a physical PFLASH sector.

3) n = 0-8191

Note: The sectors PS[7:0] are also called “logical sectors”. These logical sectors have a reduced endurance (50) compared to the physical sectors (1000). In order to

increase the endurance of a logical sector, its corresponding physical sector can be erased and reprogrammed after every 50 erase/program cycles of the logical sector.

7.2.2 Data Flash Overview

The on-chip DFLASH has a capacity of 128 Kbyte, organized in two independent banks/sectors of 64 Kbyte each, DB0/DS0 and DB1/DS1. The structure with two independent Flash banks makes it possible executing read accesses to one bank while erasing or programming the other array bank. Erase and programming operations can also be performed simultaneously in the two DFLASH banks, whereby an erase operation of one DFLASH bank is suspended by a programming operation of the other DFLASH bank, and automatically resumed afterwards. Each DFLASH bank can be erased only completely.

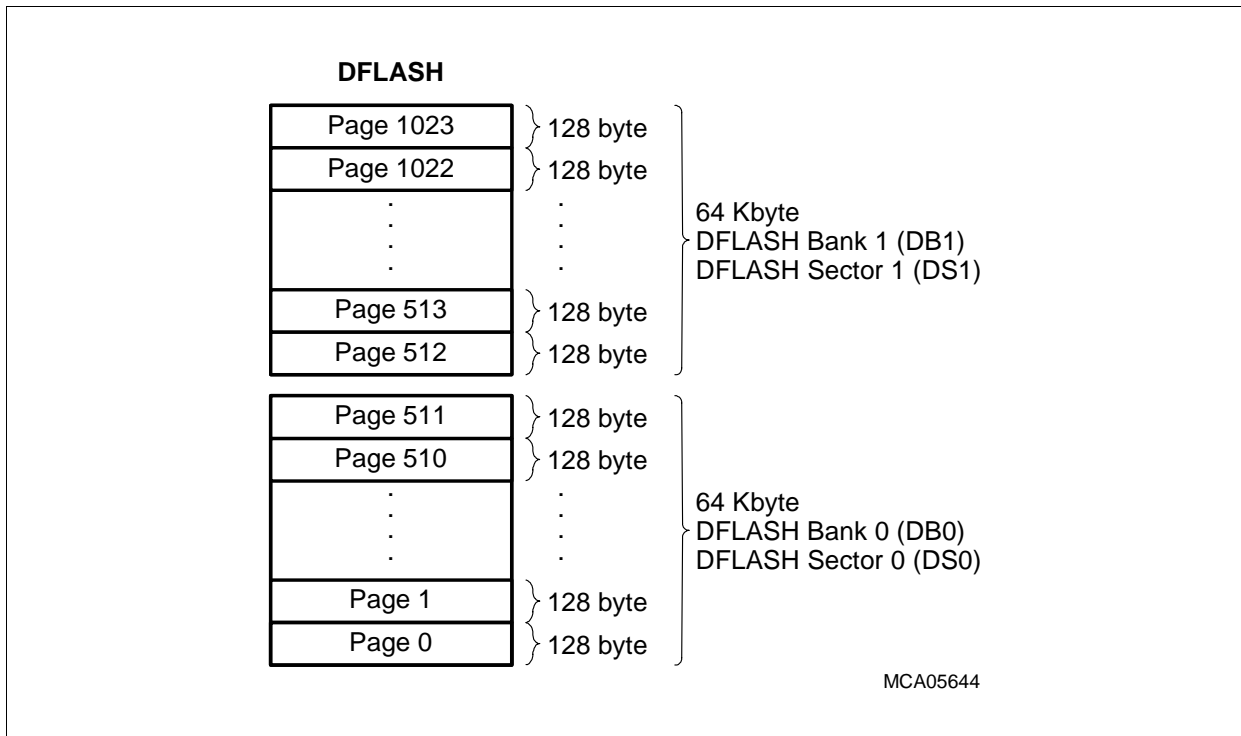


Figure 7-4 DFLASH Structure

For programming of the DFLASH, the data for one page (32 words = 128 bytes) must be loaded into an assembly buffer using fast CPU accesses before this assembly buffer is programmed into one page of the data Flash with one programming command operation. Thus, the programming width of the DFLASH is always 128 bytes. Programming of single words, bytes, or bits is not supported. The DFLASH uses the same JEDEC-standard based command sequences as the program Flash does.

Features of Data Flash

- 128 Kbyte on-chip data Flash memory, organized in two 64 Kbyte banks
- Usable for data storage with EEPROM functionality
- 128 byte program interface
 - 128 bytes are programmed into one DFLASH page by one step/command
- 64-bit read interface (no burst transfers)
- Dynamic correction of single-bit errors during read access
- Detection of double bit errors
- Fixed sector architecture
 - Two 64 Kbyte banks/sectors
 - Each sector separately erasable
- Configurable read protection (combined with write protection) for complete DFLASH together with PFLASH read protection
- Password mechanism to temporarily disable write and read protection
- Erasing/programming of one bank possible while reading data from the other bank
- Programming of one bank possible while erasing the other bank
- On-chip generation of programming voltage
- DFLASH is delivered in erased state (read all zeros)
- JEDEC-standard based command sequences for DFLASH control
 - Write state machine controls programming and erase operations
 - Status and error reporting by status flags and interrupt
- Margin check for detection of problematic DFLASH bits

The DFLASH contains two 64 Kbyte sectors, DS0 and DS1, which are identical with banks DB0 and DB1. DFLASH pages are always 128-byte wide.

Table 7-2 DFLASH Bank and Page Definitions

Numbering		Size	Cached Address Range	Non-Cached Address Range
DFLASH Sectors (and Banks)				
DS0 (= DB0)		64 Kbyte	8FE0 0000 _H - 8FE0 FFFF _H	AFE0 0000 _H - AFE0 FFFF _H
DS1 (= DB1)		64 Kbyte	8FE1 0000 _H - 8FE1 FFFF _H	AFE1 0000 _H - AFE1 FFFF _H
DFLASH Pages				
DS0	DP0	128 byte	8FE0 0000 _H - 8FE0 007F _H	AFE0 0000 _H - AFE0 007F _H
	DPn ¹⁾	128 byte	(8FE0 0000 _H + OFF) - (8FE0 0000 _H + OFF + 7F _H) (with OFF = n × 80 _H)	(AFE0 0000 _H + OFF) - (AFE0 0000 _H + OFF + 7F _H) (with OFF = n × 80 _H)
	DP511	128 byte	8FE0 FF80 _H - 8FE0 FFFF _H	AFE0 FF80 _H - AFE0 FFFF _H

Table 7-2 DFLASH Bank and Page Definitions (cont'd)

Numbering		Size	Cached Address Range	Non-Cached Address Range
DS1	DP512	128 byte	8FE1 0000 _H - 8FE1 007F _H	AFE1 0000 _H - AFE1 007F _H
	DPn ¹⁾	128 byte	8FE1 0000 _H + OFF - 8FE1 0000 _H + OFF + 7F _H (with OFF = n × 80 _H)	AFE1 0000 _H + OFF - AFE1 0000 _H + OFF + 7F _H (with OFF = n × 80 _H)
	DP1023	128 byte	8FE1 FF80 _H - 8FE1 FFFF _H	AFE1 FF80 _H - AFE1 FFFF _H

1) n = 0-1023

7.2.3 User Configuration Blocks Overview

The contents of the three User Configuration Blocks (UCB_m, m = 0-2) determine user-specific Flash configuration and protection functions such as keywords and sector-specific lock bits for the PFLASH. Each 1 Kbyte configuration block contains four UCB pages.

The UCBs are implemented as a Flash memory. The addressing of the UCBs overlays the cached addresses of the first 3 Kbyte of the PFLASH. The UCBs are not readable by the user. A user program can indirectly modify the contents of one UCB page only by two commands, the Write User Configuration Page and the Erase User Configuration Block command.

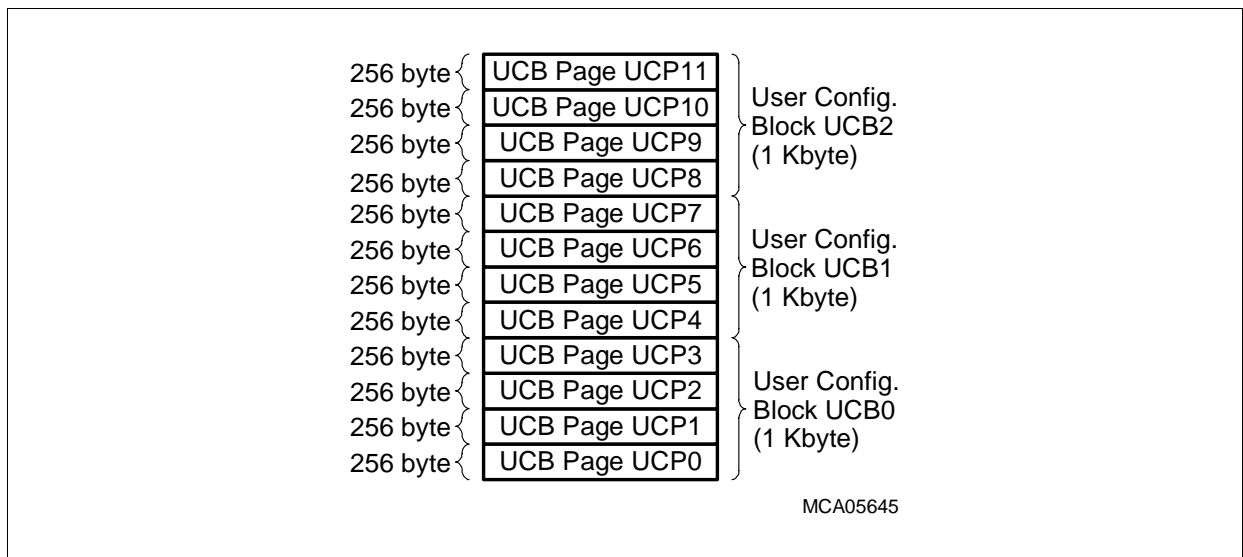


Figure 7-5 UCB Flash Structure

More details on UCBs are described on [Page 7-27](#).

Table 7-3 User Configuration Block Definitions

Numbering		Size	Address Range
UCB0	UCP0	256 byte	A000 0000 _H - A000 00FF _H
	UCP1	256 byte	A000 0100 _H - A000 01FF _H
	UCP2	256 byte	A000 0200 _H - A000 02FF _H
	UCP3	256 byte	A000 0300 _H - A000 03FF _H
UCB1	UCP4	256 byte	A000 0400 _H - A000 04FF _H
	UCP5	256 byte	A000 0500 _H - A000 05FF _H
	UCP6	256 byte	A000 0600 _H - A000 06FF _H
	UCP7	256 byte	A000 0700 _H - A000 07FF _H
UCB2	UCP8	256 byte	A000 0800 _H - A000 08FF _H
	UCP9	256 byte	A000 0900 _H - A000 09FF _H
	UCP10	256 byte	A000 0A00 _H - A000 0AFF _H
	UCP11	256 byte	A000 0B00 _H - A000 0BFF _H

7.2.4 Basic Flash Operating Modes

The basic operating modes are Flash-bank specific. Generally, there are two basic operating modes of the Flash banks:

- Read Mode (optionally with Page Mode activated)
- Command Mode

Since the TC1796 Flash array has three autonomous Flash banks, one PFLASH bank and two DFLASH banks, parallel write command execution (programming one bank while erasing the other bank) is supported for the two DFLASH banks, but not for the PFLASH bank and one DFLASH bank.

Flash register accesses are independent of the operating modes and allowed in any state. After any reset operation, the Flash banks are in Read Mode.

The following sections describe the operating modes specifically for the PFLASH. If there is any difference for the DFLASH, additional hints are given.

7.2.4.1 Read Mode

A Flash bank enters or remains in the standard operating mode, the Read Mode

- After a Reset-to-Read command, if no programming or erase operation is active
- After any completed Erase command
- After a completed Write Page (programming) command
- After every other completed command execution
- During ramp-up time after the deactivation of any reset
- After incorrect address/data values or wrong command sequence
- After incorrect requests (password failure) to program or erase a locked sector
- After an incorrect write access to a read-protected Flash memory

In Read Mode, command sequences are allowed. The Read Mode remains active until the last cycle of a command sequence is executed. In case of a write or erase command, Read Mode is terminated at the end of the command sequence.

If Page Mode is active in Read Mode, the page assembly buffer can be loaded (written) with data for the next write page (program) command while Flash read operations occur in parallel. As a special case, it is even allowed to place instructions in the PFLASH, used to load the page assembly buffer or to write command sequences to the DFLASH.

Read accesses from a Flash array are always 64-bit aligned. During these 64-bit wide read accesses from the Flash array, automatic error detection and - if a single-bit error is detected - an error correction can be executed. Bit errors are reported by separate single-bit error flags and double-bit error flags that are located in the FLASH_FSR register. Setting of single-bit or/and double-bit error flags can generate a Flash interrupt if enabled (see [Page 7-37](#)). In case of a double-bit error, the CPU is interrupted by a bus error trap per default which further results in an interrupt from the PLMB Bus Control Unit PBCU.

For verify operations, the standard read can be combined with a margin check to find problematic bits (a 0 is read instead of a programmed 1) in advance. The change of margins is controlled via the margin registers (see [Page 7-35](#)).

7.2.4.2 Command Mode

Every write operation to the Flash memory space is interpreted as a command cycle, belonging to a command sequence. A command sequence is composed of one to at maximum six command cycles (write operations) with well defined address and data values. After the last command cycle of a correct command sequence, the Command Mode is entered. The Command Mode remains active during the whole command execution. The state of a command execution is indicated by several status flags in the FSR status register.

Some command sequences that do not affect the Flash banks, e.g. the Enter Page Mode command or the Clear Status command, are immediately executed and finished after the last command cycle of the command. For such type of command sequences, the Command Mode is terminated immediately after its execution. After all other command sequences which activate a Flash bank operation such as erasing or programming, the Command Mode and the related status flags remain active until the command is really finished.

Note that all write operations to the Flash memory space are handled by the FCS.

Writing incorrect addresses or data values within a command sequence, or writing them in an wrong sequence, generates a sequence error (FSR.SQER is set) and terminates Command Mode.

If one DFLASH bank is busy with erasing, a programming command sequence for the other DFLASH bank is accepted immediately and the execution of the erase operation is interrupted until the programming operation is terminated (automatic suspend/resume operation).

When accessing a busy PFLASH or DFLASH bank by a data read operation (e.g. by a CPU read in a user program), the read operation is blocked (halted) until the related Flash bank is no more busy. Therefore, it is recommended to access a PFLASH or DFLASH bank by data read operations only when they are no longer busy (check the corresponding busy flags first).

The cycle definitions of command sequences are based on the JEDEC standard. The different write cycles of command sequences are not only used for operation definitions such as a sector-erase command, but also as fail-safe and unlock cycles in order to protect the Flash against inadvertent changes.

Note: The write cycles to the Flash belonging to a command sequence are/may be buffered in the DMI in store/write buffers. To maintain data coherency (defined sequence of command cycles is mandatory) and to guarantee immediate transfer of the command sequence to the PMU, all write cycles to the Flash must access

Program Memory Unit

the Flash in its non-cached address space. Additionally, it is recommended to include a dummy read cycle to a PMU register after the last write cycle of a command sequence.

Note: User code (command sequences) that programs or erase Program Flash should not be executed from internal program flash, but from other internal or external memory (e.g. from SPRAM).

7.2.4.3 Page Mode

With the Enter Page Mode command, Page Mode is entered for one of the three Flash banks and status flag FSR.PFPAGE (for PFLASH) or FSR.DFPAGE (for DFLASH) is set. In Page Mode, the assembly buffer is ready to be filled with data in preparation for a subsequent PFLASH or DFLASH programming operation. The width of the page assembly buffer is 128-byte for DFLASH and 256-byte for PFLASH. Page Mode can be entered only if the related Flash bank is not busy (if it does not execute program or erase operations).

7.2.5 Command Sequence Definitions

Flash commands are executed by writing specific data to dedicated addresses in a well defined command specific sequence. The data to be transmitted within a command sequence must be transmitted right-aligned on data bus lines D[7:0] as byte (exception: 32-bit data, 64-bit data, 32-bit password). Data lines D[31:8] are ignored in a command sequence (indicated in the “Data” columns of the command tables by a “X”). Addresses in a command sequence always refer to the non-cached address ranges. The available command sequences are shown in [Figure 7-4](#).

Table 7-4 Flash Command Overview

Command	Description	Details see
Reset-to-Read	Resetting Flash State Machine to Read Mode	Page 7-15
Enter Page Mode	Initiate Page Mode	Page 7-15
Load Page Buffer	Loading page assembly buffer with 32-bit or 64-bit data	Page 7-16
Write Page	Programming a Flash page with assembly buffer content	Page 7-18
Write User Configuration Page	Programming a user configuration page with assembly buffer content	Page 7-19
Erase Sector	Erasing a PFLASH or DFLASH sector	Page 7-20
Erase User Configuration Block	Erasing a UCB	Page 7-22
Disable Write Protection ¹⁾	Temporarily unlocking write protection	Page 7-23
Disable Read Protection	Temporarily unlocking read protection	Page 7-24
Resume Protection	Resumption of disabled read or write protection	Page 7-24
Clear Status	Resetting status register flags	Page 7-25

1) This command is not available for the DFLASH.

Note: During a programming or erasing operation, a minimum CPU clock (f_{CPU}) frequency of 1 MHz must be provided.

7.2.5.1 Reset-to-Read Command

With the one-cycle Reset-to-Read command, the internal command state machine is reset to its initial state. This command can be issued at any time during a command sequence.

Table 7-5 Reset-to-Read Command

Cycle No.	Address	Data
Cycle 1	A000 5554_H	XXXX XXF0_H

A running programming or erase operation of a Flash bank is not affected by a Reset-to-Read command and will be continued and finished. All error flags in the Flash Status Register FSR are cleared, and an active Page Mode is aborted. The busy state of the Flash bank (write operation or voltage ramp-up) is not aborted and the busy flags in the FSR are not affected. A Reset-to-Read command during a command sequence does **not** generate a sequence error (FSR.SQER is not set).

7.2.5.2 Enter Page Mode Command

The Page Mode is entered for a Flash bank with the one-cycle Enter Page Mode command, indicating that the page assembly buffer is ready to be filled with data for the related Flash bank in preparation for a subsequent Flash programming operation. The Page Mode can only be assigned to one of the PFLASH/DFLASH banks by executing the address/data information shown in [Table 7-6](#). The width of the page assembly buffer is 128-byte for DFLASH and 256-byte for PFLASH. Further, the Page Mode can only be entered if the related Flash bank is not busy (if it does not execute program or erase operations). However, an erase or program operation can be active in another Flash bank than the one that is in Page Mode.

Table 7-6 Enter Page Mode Command

Cycle No.	PFLASH		DFLASH		
	Address	Data	Bank	Address	Data
Cycle 1	A000 5554_H	XXXX XX50_H	DB0	AFE0 5554_H	XXXX XX5D_H
			DB1	AFE1 5554_H	XXXX XX5D_H

When Page Mode is entered, the pointer to the page assembly buffer is set to its first word location. Its base address has to point to the addressed Flash bank.

An active Page Mode is indicated when bit FSR.PFPAGE (for PFLASH) or FSR.DFPAGE (for DFLASH) is set.

The Page Mode and the Read Mode are allowed in parallel at the same time and in the same Flash memory bank. A new Enter Page Mode command during Page Mode aborts

the actual Page Mode, indicated by sequence error flag FSR.SQER set, and restarts a new page operation. The selected assembly register remains unchanged (not cleared) with a new Enter Page Mode command.

7.2.5.3 Load Page Buffer Command

There are basically two types of Load Page Buffer command:

- Load Page Buffer command with 32-bit data (two-cycle command)
- Load Page Buffer command with 64-bit data (one-cycle command)

Table 7-7 Load Page Buffer Command

Cycle No.	PFLASH		DFLASH		
	Address	Data	Bank	Address	Data
32-Bit Load Page Buffer Command					
Cycle 1	A000 55F0_H	32-bit data	DB0	AFE0 55F0_H	32-bit data
			DB1	AFE1 55F0_H	
Cycle 2	A000 55F4_H	32-bit data	DB0	AFE0 55F0_H	32-bit data
			DB1	AFE1 55F4_H	
64-Bit Load Page Buffer Command					
Cycle 1	A000 55F0_H	64-bit data	DB0	AFE0 55F0_H	64-bit data
			DB1	AFE1 55F0_H	

The Load Page Buffer command loads 32-bit words or 64-bit double-words into the assembly buffer, starting from the lowest to the highest assembly buffer entry. Only one type of the Load Page Buffer commands (either with 32-bit or 64-bit data width) is allowed during one assembly buffer loading sequence. Mixing 64-bit and 32-bit write data width within one assembly buffer filling sequence is not allowed.

The data width for the assembly buffer filling sequence is selected by using instruction(s) with the corresponding data format. A single Load Page Buffer command consists of a store instruction to a specific address. In case of 64-bit data width, the same address A000 55F0_H always has to be used. In case of 32-bit data width, alternating addresses A000 55F0_H and A000 55F4_H must be used.

For loading of the 256-byte wide assembly buffer for PFLASH programming, thirty-two Load Page commands with 64-bit data or sixty-four Load Page Buffer commands with 32-bit data must be issued. For loading of the 128-byte wide assembly buffer for DFLASH programming, sixteen Load Page commands with 64-bit data or thirty-two Load Page Buffer commands with 32-bit data must be issued.

The following example shows a Load Page Buffer command sequence for the DFLASH bank 0 assembly buffer (128 bytes) with 32-bit data width:

Program Memory Unit

1. Write AFE0 55F0_H = word 0 (assembly buffer byte 3-0)
2. Write AFE0 55F4_H = word 1 (assembly buffer byte 7-4)
3. Write AFE0 55F0_H = word 2 (assembly buffer byte 11-8)
4. Write AFE0 55F4_H = word 3 (assembly buffer byte 15-12)

...

1. Write AFE0 55F0_H = word 30 (assembly buffer byte 123-120)
2. Write AFE0 55F4_H = word 31 (assembly buffer byte 127-124)

Sequence errors (bit FSR.SQER set) are generated by a Load Page Buffer command in the following cases:

- A mix of 64-bit and 32-bit write data width within one assembly buffer filling sequence has been detected.
- An assembly buffer overrun condition is detected. The write data causing the overrun is lost.

7.2.5.4 Write Page Command

With the four-cycle Write Page command, the complete contents of the assembly buffer (256 bytes for PFLASH, 128 bytes for DFLASH) are transferred (programmed) into one page of PFLASH or DFLASH. The write address of cycle 4 is the page start address of the page to be programmed with the assembly buffer content. Parameter “n” is the page number PPn for PFLASH or DPn for DFLASH (see also [Table 7-1](#) and [Table 7-2](#)).

Table 7-8 Write Page Command

Cycle No.	PFLASH		DFLASH		
	Address	Data	Bank	Address	Data
Cycle 1	A000 5554 _H	XXXX XXAA _H	DB0	AFE0 5554 _H	XXXX XXAA _H
			DB1	AFE1 5554 _H	
Cycle 2	A000 AAA8 _H	XXXX XX55 _H	DB0	AFE0 AAA8 _H	XXXX XX55 _H
			DB1	AFE1 AAA8 _H	
Cycle 3	A000 5554 _H	XXXX XXA0 _H	DB0	AFE0 5554 _H	XXXX XXA0 _H
			DB1	AFE1 5554 _H	
Cycle 4	A000 0000 _H + n × 100 _H	XXXX XXAA _H	DB0	AFE0 0000 _H + n × 80 _H	XXXX XXAA _H
			DB1	AFE1 0000 _H + n × 80 _H	

Address bits [31:16] used in the command cycles of the Write Page command must be the same as those used by the previous Enter Page Mode command. Otherwise, a command sequence error is generated (bit FSR.SQER set) and the execution of the command is aborted.

The Write Page command programs a specific Flash page with the complete content of the assembly buffer. It also programs invalid data of assembler buffer locations that have not been loaded by previous Load Page Buffer commands.

With the last cycle of the Write Page command, Page Mode is terminated and the following status flags are updated:

- FSR.PFPAGE or FSR.DFPAGE is cleared, indicating that the related Flash is no more in Page Mode.
- FSR.PROG is set, indicating that a programming operation is running.
- Either FSR.PBUSY or FSR.D0BUSY or FSR.D1BUSY is set, indicating that the programming operation is running in PFLASH or in DFLASH bank 0 or in DFLASH bank 1.

Program Memory Unit

The page programming algorithm includes a programming quality check that identifies and reprograms weak bits of a Flash page. If the reprogramming of weak bits is unsuccessful, the verify error flag FSR.VER is set.

If read protection is activated or if write protection is enabled for the sector that contains the page to be programmed, Page Mode is terminated, programming operation is not started, and the protection error flag FSR.PROER is set. Write protection must be disabled previously to a programming operation for a write-protected sector by issuing a Disable Read Protection or/and Disable Write Protection command.

7.2.5.5 Write User Configuration Page Command

The four-cycle Write User Configuration Page command is used to transfer (program) the contents of the 256 byte assembly buffer into one page of a user configuration block.

The Write User Configuration Page command sequence is identical with the Write Page command except cycle 3. Cycle 3 transmits different data than the Write Page command does. The address transmitted in cycle 4 of the Write User Configuration Page command is the start address of the corresponding User Configuration Page UCP[11:0] (see [Table 7-3](#)).

Table 7-9 Write User Configuration Page Command

Cycle No.	Address	Data
Cycle 1	A000 5554 _H	XXXX XXAA _H
Cycle 2	A000 AAA8 _H	XXXX XX55 _H
Cycle 3	A000 5554 _H	XXXX XXC0 _H
Cycle 4	Start address of UCP to be programmed (A000 0X00 _H with X = 0 _H -B _H)	XXXX XXAA _H

Only if protection is not installed (e.g. for the very first installation of protection), read/write protection need not be disabled. After writing the correct protection confirmation code in the respective user configuration page, the protection is installed and becomes active after the next reset.

If a user configuration page needs to be reprogrammed (not possible for pages within UCB2) and protection is installed (FSR.PROIN = 1), the protection must be temporarily disabled, and afterwards the UCB must be erased by the Erase User Configuration Block command. When no protection is installed (FSR.PROIN = 0), the user configuration page can be programmed without any restriction.

If a Write User Configuration Page command is issued and the start address of the UCP (cycle 4) belongs to a UCB that is read- or write-protected, the protection error flag FSR.PROER is set and programming operation is not started.

7.2.5.6 Erase Sector Command

The six-cycle Erase Sector command is used to erase a sector in PFLASH or DFLASH. In PFLASH, either one of the 13 sectors PS_x (x = 0-12) or one of the 2 physical sectors PPS0 or PPS1 that each include four sectors can be erased. In DFLASH, one of its two banks DB0 or DB1 (identical with sectors DS0 and DS1) can be erased.

The write address transmitted in the last cycle of the Erase Sector command is the sector start address of the sector PS_x that should be erased.

Table 7-10 Erase Sector Command

Cycle No.	PFLASH			DFLASH		
	Sector	Address	Data	Bank	Address	Data
Cycle 1	PS _x ¹⁾	A000 5554 _H	XXXX XXAA _H	DB0	AFE0 5554 _H	XXXX XXAA _H
	PPS0			DB1	AFE1 5554 _H	
	PPS1			–		
Cycle 2	PS _x ¹⁾	A000 AAA8 _H	XXXX XX55 _H	DB0	AFE0 AAA8 _H	XXXX XX55 _H
	PPS0			DB1	AFE1 AAA8 _H	
	PPS1			–		
Cycle 3	PS _x ¹⁾	A000 5554 _H	XXXX XX80 _H	DB0	AFE0 5554 _H	XXXX XX80 _H
	PPS0			DB1	AFE1 5554 _H	
	PPS1			–		
Cycle 4	PS _x ¹⁾	A000 5554 _H	XXXX XXAA _H	DB0	AFE0 5554 _H	XXXX XXAA _H
	PPS0			DB1	AFE1 5554 _H	
	PPS1			–		
Cycle 5	PS _x ¹⁾	A000 AAA8 _H	XXXX XX55 _H	DB0	AFE0 AAA8 _H	XXXX XX55 _H
	PPS0			DB1	AFE1 AAA8 _H	
	PPS1			–		
Cycle 6	PS _x ¹⁾	A0XX XXXX _H ²⁾	XXXX XX30 _H	DB0	AFE0 0000 _H	XXXX XX40 _H
	PPS0	A000 0000 _H	XXXX XX40 _H	DB1	AFE1 0000 _H	
	PPS1	A001 0000 _H		–		

1) "x" is the number of the PFLASH sector PS_x that is defined in [Table 7-1](#) (x = 0-12).

2) This is the complete 32-bit non-cached start address of the PFLASH sector PS_x (x = 0-12) to be erased.

Program Memory Unit

With the last cycle of the Erase Sector command, Command Mode is entered and the following status flags are updated:

- FSR.ERASE is set, indicating that a erase operation is running.
- Either FSR.PBUSY or FSR.D0BUSY or FSR.D1BUSY is set, indicating that an erase operation is running in PFLASH or in DFLASH bank 0 or in DFLASH bank 1.

The start of sector erase operation is delayed if the Erase Sector command was issued to one DFLASH bank while the other DFLASH bank is busy with a program operation.

Read accesses to a DFLASH bank that is currently not erased or programmed are possible while the other DFLASH bank is erased or programmed. A read access to a busy (erasing or programming) Flash bank is allowed but delayed until the corresponding Flash bank is no longer busy. Note that in this case the read path (PLMB) is completely locked. Therefore, reading from a Flash bank in erase or program state should be avoided by first polling the busy flags in the FSR register.

If the Erase Sector operation is used to erase a physical sector of the program Flash (PPS0 or PPS1), this operation is executed only if none of the four sectors within the related physical sector is write-protected, and if no read protection is installed, or if Flash protection is disabled.

If read protection and/or write protection for the sector to be erased is enabled, or if one or more of the sectors within the physical sector to be erased is OTP-protected, the protection error flag FSR.PROER is set, Command Mode is not entered, and the erase operation is not started.

7.2.5.7 Erase User Configuration Block Command

This six-cycle Erase User Configuration Block command is used to erase a UCB.

The Erase User Configuration Block command sequence is identical with the Erase Sector command sequence except cycle 6. Cycle 6 of the Erase User Configuration Block command provides the base address of the UCB to be erased.

Table 7-11 Erase User Configuration Block Command

Cycle No.	Address		Data
Cycle 1	A000 5554 _H		XXXX XXAA _H
Cycle 2	A000 AAA8 _H		XXXX XX55 _H
Cycle 3	A000 5554 _H		XXXX XX80 _H
Cycle 4	A000 5554 _H		XXXX XXAA _H
Cycle 5	A000 AAA8 _H		XXXX XX55 _H
Cycle 6	UCB0	A000 0000 _H	XXXX XXC0 _H
	UCB1	A000 0400 _H	
	UCB2	A000 0800 _H	

With the last cycle of the Erase User Configuration command, Command Mode is entered and the following status flags are updated:

- FSR.ERASE is set, indicating that a erase operation is running.
- FSR.PBUSY is set, indicating that the erase operation is running in the UCB.

This command can only be executed after read protection and/or sector write protection has been disabled. If protection is not disabled when the Erase User Configuration Block command is received, the protection error flag FSR.PROER is set, the Command Mode is not entered, and the erase operation is not started.

After a Erase User Configuration Block command, a new protection configuration (including keywords and protection confirmation code) can be written to the pages of the user configuration block. But note that the user configuration blocks UCB0 and UCB1 can only be modified up to 4 times during the lifetime of a TC1796 device. UCB2 can be programmed only once.

Note: UCB2 is OneTime-Programmable (OTP). If sector write protection is installed and confirmed in UCB2, this block can never be erased again. UCB2 can only be erased before the confirmation of OTP write protection.

7.2.5.8 Disable Write Protection Command

The six-cycle Disable Write Protection command temporarily disables the write protection of all protected PFLASH sectors as defined for user 0 (indicated in register PROCON0) or user 1 (indicated in register PROCON1).

The Disable Write Protection command is not accepted for UCB2 because UCB2 is a OTP write-protected UCB.

Table 7-12 Disable Write Protection Command

Cycle No.	Address	Data
Cycle 1	A000 5554 _H	XXXX XX AA _H
Cycle 2	A000 AAA8 _H	XXXX XX 55 _H
Cycle 3	A000 553C _H	XXXX XX 00 _H (user 0, UCB0) or XXXX XX 01 _H (user 1, UCB1)
Cycle 4	A000 AAA8 _H	First 32-bit password
Cycle 5	A000 AAA8 _H	Second 32-bit password
Cycle 6	A000 5558 _H	XXXX XX 05 _H

The Disable Write Protection command is a protected command sequence, meaning that two passwords (data in cycle 4 and 5) must be issued within the command. The first and second 32-bit passwords are internally compared with the first and second 32-bit keyword as defined in the related UCB0 or UCB1 (see [Table 7-16](#)). If one or both passwords are not identical to their related keywords, the sectors remain protected and the protection error flag FSR.PROER is set. In this case, a new Disable Write Protection command is only accepted after the next reset operation.

After the correct execution of the Disable Write Protection command, all protected sectors as defined for UCB0 or UCB1 are unlocked (if no read protection is additionally installed and active) and flag FSR.WPRODIS0 or FSR.WPRODIS1 is set. Erase and write operations to temporarily unlocked sectors are now possible, until

- A Resume Protection command is executed, or
- The next reset operation (hardware or software reset) is executed.

7.2.5.9 Disable Read Protection Command

The six-cycle Disable Read Protection command temporarily disables an installed Flash read protection.

Table 7-13 Disable Read Protection Command

Cycle No.	Address	Data
Cycle 1	A000 5554 _H	XXXX XX AA _H
Cycle 2	A000 AAA8 _H	XXXX XX 55 _H
Cycle 3	A000 553C _H	XXXX XX 00 _H
Cycle 4	A000 AAA8 _H	First 32-bit password
Cycle 5	A000 AAA8 _H	Second 32-bit password
Cycle 6	A000 5558 _H	XXXX XX 08 _H

The Disable Read Protection command is a protected command sequence, meaning that two passwords (data in cycle 4 and 5) must be issued within the command. The first and second 32-bit passwords are internally compared to the first and second 32-bit keyword as defined in the related user configuration block UCB0 (see [Table 7-16](#)).

If one or both passwords are not identical to their related keywords, the sectors remain protected and the protection error flag FSR.PROER is set. In this case, a new Disable Read Protection command is only accepted after the next reset operation.

After the correct execution of this command, the PFLASH and both DFLASH banks are temporarily unlocked and flag FSR.RPRODIS is set. Read, erase, and write operations to all unlocked (not separately write-protected) sectors are now possible, until

- A Resume Protection command is executed, or
- The next reset operation (hardware or software reset) is executed.

7.2.5.10 Resume Protection Command

With the one-cycle Resume Protection Command, a temporarily unlocked write protection and/or read protection is terminated and the protection state is resumed. This command resumes both kinds of temporarily disabled protection, write and read protection.

Table 7-14 Resume Protection Command

Cycle No.	Address	Data
Cycle 1	A000 5554 _H	XXXX XX 5E _H

7.2.5.11 Clear Status Command

The one-cycle Clear Status command clears/resets the following flags of the Flash status register FSR:

- Error flags:
PFOPER, DFOPER, SQER, PROER, PFSBER, DFSBER, PFDBER, and DFDBER
- Status flags:
PROG, ERASE

The one-cycle Clear Status command is accepted only in Read Mode. Otherwise a command sequence error occurs (FSR.SQER is set). Read Mode is entered if Flash is not busy and after every error detection and indication change in FSR.

Table 7-15 Clear Status Command

Cycle No.	Address	Data
Cycle 1	A000 5554 _H	XXXX XXF5 _H

7.2.6 Data Flash and EEPROM Emulation

The 128 Kbyte Data Flash is able to support an emulation of an EEPROM. This EEPROM emulation is fully based on a software administration by a user program. The only hardware feature that supports EEPROM emulation is the ability to program a page in one DFLASH bank while the other DFLASH bank is erased, and to read data from one bank while the other bank is busy with programming or erasing operation.

The main difference between a Flash memory and an EEPROM is its endurance. For EEPROMs, the endurance is typically 120.000 write/erase cycles. The typical endurance of a Flash memory is in the range of 1.000 to 10.000 write/erase cycles. The following example discusses the basic principles of the EEPROM emulation as supported in the TC1796.

Example: 16 Kbyte EEPROM emulation, using the complete 128 Kbyte DFLASH

The DFLASH is logically divided into eight 16 Kbyte regions that operate as a circular buffer memory. Each of these regions has 128 pages, the smallest DFLASH entity that can be programmed by one programming/write command. At a time, one of the eight 16 Kbyte regions is always regarded as active EEPROM region. The other regions are within a loop in which all regions within one DFLASH bank can be erased together and programmed (page-wise) consecutively in a fixed order.

The active EEPROM region is held as a mirror memory in an on-chip RAM area. After a reset operation, the active EEPROM region is copied into the RAM and the related 16 Kbyte region may be marked (e.g. with all-one programming of a page) to be invalid. Now the next consecutive 16 Kbyte region within the DFLASH circular buffer becomes the active EEPROM region.

After the copy event, the user program is able to read/write data from/to the mirrored RAM area very fast. The user program also has the task of tracking the changes in the RAM and to decide when and which part of the mirrored EEPROM region must be written back (programmed) to the actual active (and erased) EEPROM region of the DFLASH. This decision can be, for example, an upcoming system switch-off event for which the EEPROM data must be saved.

As a result of the continuously changing assignment of the active EEPROM region in a circular buffer, the DFLASH memory cells of one EEPROM region can be erased/programmed 8-times as often as one physical 16 Kbyte region, resulting in an 8-fold endurance for one EEPROM region. Additionally, a reduced retention is assumed for EEPROM data. Thus, for the above described example with 16 Kbyte regions and with a retention of 5 years, the endurance for an emulated EEPROM region is increased to 120.000 write/erase cycles. As a result, the endurance of an emulated EEPROM region can be increased by raising the software overhead and DFLASH memory area.

For marking of invalid word-lines (two consecutive pages) it is allowed to over-program with all-one data. This is the only kind of twofold programming of a page which is allowed and can only be performed in DFLASH.

7.2.7 Read and Write Protection

In the TC1796, three different types of protection are possible:

- Read protection for PFLASH and DFLASH (always implies also a global write protection for PFLASH and DFLASH)
- Sector write protection for PFLASH
- Sector OTP write protection for PFLASH

In general, three protection configurations, assigned to so-called “users”, are defined in UCBs.

- **User 0 = User Configuration Block 0:**
This is the master user. It is able to install read protection for the whole PFLASH and DFLASH. It can also install sector write protection for the PFLASH sectors. User 0 controls its protection configuration and its keywords via UCB0.
- **User 1 = User Configuration Block 1:**
User 1 is able to install sector write protection for the PFLASH sectors with lower priority than user 0. User 1 controls its protection configuration and its keywords via UCB1.
- **User 2 = User Configuration Block 2:**
User 2 is able to install sector OTP write protection for the PFLASH sectors. OTP write-protected sectors are locked for ever, and are never re-programmable.

If read or write protection is installed and activated once, changing the read/write protection configuration parameters in the UCBs is password-protected and can only be executed if the passwords are known.

Note: If any PFLASH sector is locked for ever (OTP installed), PFLASH and DFLASH related testing investigations for Failure Analysis Requests (FARs) are no more possible.

7.2.7.1 User Configuration Block Definitions

The three 1 Kbyte user configuration blocks UCB_x (x = 0-2) contain four 256 byte pages each. The contents of the three user configuration blocks determine the read, write, and OTP protection configuration as assigned for user 0, user 1, and user 2.

The UCBs are Flash memory locations that can be programmed page-wise with the Write User Configuration Page command, and erased block-wise by the Erase User Configuration Block command. User configuration blocks UCB0 and UCB1 can be programmed/erased during TC1796 device life-time only up to 4 times. UCB2 can be programmed only once.

Table 7-16 defines the contents of the three UCBs.

Table 7-16 Layout of User Configuration Blocks

UCB	Page	Address	Byte(s)	Content	
UCB0	UCP0	A000 0000 _H	[1:0]	Protection configuration bits (content as defined for PROCON0)	
		A000 0008 _H	[9:8]	Copy of bytes [1:0]	
		A000 0010 _H	[19:16]	First 32-bit keyword of user 0	
		A000 0014 _H	[23:20]	Second 32-bit keyword of user 0	
		A000 0018 _H	[27:24]	Copy of first 32-bit keyword of user 0	
		A000 001C _H	[31:28]	Copy of second 32-bit keyword of user 0	
		–	others	Must be programmed to 00 _H	
	UCP1	–	all	This page is reserved for future purposes; must be programmed to 00 _H	
	UCP2	A000 0200 _H	[3:0]	32-bit confirmation code: 8AFE15C3 _H	
		A000 0208 _H	[11:8]	Copy of 32-bit confirmation code	
		–	others	Must be programmed to 00 _H	
	UCP3	–	all	This page is reserved for future purposes; must be programmed to 00 _H	
	UCB1	UCP4	A000 0400 _H	[1:0]	Protection configuration bits (content as defined for PROCON1)
			A000 0408 _H	[9:8]	Copy of bytes [1:0]
A000 0410 _H			[19:16]	First 32-bit keyword of user 1	
A000 0414 _H			[23:20]	Second 32-bit keyword of user 1	
A000 0418 _H			[27:24]	Copy of first 32-bit keyword of user 1	
A000 041C _H			[31:28]	Copy of second 32-bit keyword of user 1	
–			others	Must be programmed to 00 _H	
UCP5		–	all	This page is reserved for future purposes; must be programmed to 00 _H	
UCP6		A000 0600 _H	[3:0]	32-bit confirmation code: 8AFE15C3 _H	
		A000 0608 _H	[11:8]	Copy of 32-bit confirmation code	
		–	others	Must be programmed to 00 _H	
UCP7		–	all	This page is reserved for future purposes; must be programmed to 00 _H	

Table 7-16 Layout of User Configuration Blocks (cont'd)

UCB	Page	Address	Byte(s)	Content
UCB2	UCP8	A000 0800 _H	[1:0]	Protection configuration bits (content as defined for PROCON2)
		A000 0808 _H	[9:8]	Copy of bytes [1:0]
		–	others	Must be programmed to 00 _H
	UCP9	–	all	This page is reserved for future purposes; must be programmed to 00 _H
	UCP10	A000 0600 _H	[3:0]	32-bit confirmation code: 8AFE15C3 _H
		A000 0608 _H	[11:8]	Copy of 32-bit confirmation code
		–	others	Must be programmed to 00 _H
UCP11	–	all	This page is reserved for future purposes; must be programmed to 00 _H	

The protection configuration bits that are located in the first two bytes of the UCBs determine the requested protection configuration as it is defined for bits [15:0] of the Flash Protection Configuration registers PROCON0, PROCON1, and PROCON2. When a read/write protection has been installed for an user configuration block, the content of its first two bytes is copied into the corresponding Flash Protection Configuration register.

UCB Configuration, Confirmation and Activation

In order to set up a UCB correctly, several steps must be done to avoid incorrect and inoperable UCB contents and, as a result, unreparable read/write protection.

There are three main tasks to execute for UCB setup:

1. Configuration of a UCB

This step includes the programming of the first page of a UCB by executing a User Configuration Page command. This first page determines the protection type and the two 32-bit keywords. Unused bytes in the first page of the UCB must be programmed with 00_H.

2. Confirmation of the Keywords

The 32-bit confirmation code word, which is located in the third page of a UCB, should be programmed only after a check of the correct programming of the two 32-bit keywords. Reason: wrong keywords in a UCB can never be retrieved (because UCBs are not readable), and a confirmed read or write protection cannot be disabled and changed anymore when the password check (see [Page 7-33](#)) always fails.

The check for correct keywords in a UCB requires to execute a reset operation (e.g. software reset) after the configuration has been setup as described under point 1. After the reset, the protection is not fully activated because the confirmation code in

the UCB is still not valid but it is already configured. After issuing a Disable Read Protection or Disable Write Protection command (depending on the configured protection type) with the expected passwords, the status flag FSR.PROER indicates, whether password checking was ok or not.

If PROER = 0 after the Disable Read Protection or Disable Write Protection command, the password check was ok, meaning that the keywords in the UCB are identical with the passwords that have been transmitted with the command. Now the 32-bit confirmation code word must be still programmed into the third page of the UCB by the Write User Configuration Page command. Unused bytes of the third page of the UCB should be programmed with 00_H. After that operation, the selected protection is defined to be “installed”.

If PROER = 1 after the Disable Read Protection or Disable Write Protection command, the password check was negative. In this case (and if the correct passwords are not known), the related UCB has to be erased again and the UCB setup must be repeated as described under point 1.

3. Activation of an Installed Protection

When a read or write protection has been installed, it can only be activated by executing any reset operation. After this reset operation, an installed protection becomes “active”.

7.2.7.2 Write and OTP Protection for PFLASH

Write protection is a feature that must be installed by the user of the TC1796 device. In the delivery state of the TC1796, no write protection is installed meaning that the UCBs are in erased state. If sector write protection is active for a PFLASH sector, erasing and programming of this sector is only possible if the corresponding UCB keywords are known.

OTP write protection can be installed and enabled for a PFLASH sector only once after the TC1796 delivery state. Write protection configuration for a PFLASH sector can be modified by erasing and re-programming of the related UCB.

The sector write protection configuration must be initially programmed into one of the three UCBs by using the Write User Configuration Page command. With this command, the user determines the PFLASH sector(s) to be write-protected and two 32-bit keywords which are required to temporarily disable an already installed write protection configuration or to temporarily disable an active sector write protection. Erasing and reprogramming of UCB0 or UCB1 can be performed up to 4 times during TC1796 device lifetime.

As described above on [Page 7-29](#), sector write protection remains active as long as no Disable Write Protection command is issued. Within this command sequence, the user has to identify itself by its passwords and its user level (UL, see command sequence definition). After the Disable Write Protection command, sector write protection is temporarily disabled for all sectors that belong to the user. Thereby, disabling of write

Program Memory Unit

protection is hierarchically controlled. This means, if user 0 (assigned to UCB0) disables write protection for his sector(s), also write protection for user 1 (assigned to UCB1) is disabled but not vice versa (user 1 can only disable his own protected sectors).

Note: Sector specific write protection may be combined with read protection. In this case, after execution of the Disable Sector Write Protection command the protected sectors are only unlocked if read protection is also disabled.

Resumption of the temporarily disabled write protection (and read protection) is done by sending the Resume Protection command or by executing a reset operation. For UCB2, disabling write protection and thus re-programming is not possible.

The configuration of an installed write protection is indicated by:

- Three status flags FSR.WPROIN x ($x = 0-2$) that indicate whether sector write protection is installed for UCB x or not
- Status flag FSR.PROIN = 1; this bit is set coincidentally with FSR.WPROIN x
- Status flags SnL ($n = 0-12$) in the three Protection Configuration registers PROCON x ($x = 0-2$) that indicate which Flash sectors are write-protected by UCB x
- The state of a write protection (enabled or temporarily disabled) is indicated by bits FSR.WPRODIS0 (for UCB0) and FSR.WPRODIS1 (for UCB1).

After the execution of an Erase User Configuration Block command, which requires the preceding disabling of an active write protection by the Disable Write Protection command, all keywords and all protection parameters in the UCB are erased. Thus, the UCB is totally unprotected until it becomes re-programmed. The only exception is UCB2, which can never be erased after installation of OTP write protection.

If global write protection is additionally installed (implicitly with an installed read protection), a Disable Read Protection command must be issued before the write protection configuration parameters in UCB0 can be modified by user 0.

Note: All PFLASH sectors can be write-protected or OTP-protected, separately for all three users. DFLASH sectors cannot be separately write-protected (only generally via Read Protection).

7.2.7.3 Read Protection for PFLASH and DFLASH

When read protection is active, read accesses from PFLASH and DFLASH memory locations are generally disabled, if code execution is not started from internal Flash after reset. Additionally, a global write protection is always active for a read-protected PFLASH and DFLASH. This feature supports a protection against trojan horse programs. Note that read protection cannot be activated separately for PFLASH and DFLASH.

Read protection is installed only via UCB0. At programming of UCB0, the highest bit of its second byte (corresponds to bit PROCON0.RPRO, see [Page 7-59](#)) must be set.

After the configuration and confirmation of UCB0 (see [Page 7-29](#)), read protection is installed but still not active. After the following reset operation, the installed read protection becomes really active. It remains active as long as no Disable Read Protection command is issued. This command is password-protected and the user must provide the two passwords for temporarily disabling the read protection. This mechanism assures, that an installed read protection configuration can only be changed (e.g. disabled) by an user which has knowledge about the two keywords that have been initially programmed into the UCB. After the Disable Read Protection command, the read protection configuration as defined in UCB0 can be changed (after erasing UCB0), if not coincidentally sector write protection is installed by the user 0 (in this case, also sector write protection must be disabled). A temporarily disabled read protection can be re-enabled by sending the Resume Protection command, or by executing a reset operation.

Read protection can be combined with sector specific write protection. In this case, after execution of the Disable Read Protection command, only those sectors are unlocked for write accesses, which are not separately write-protected.

The status of a correctly installed read protection is indicated by:

- Status flag FSR.RPROIN = 1; this bit becomes updated after a reset operation (when the Boot ROM code has been left) and when read protection has been configured.
- Status flag PROCON0.RPRO = 1; indicates an installed (and confirmed) read protection as programmed in the protection configuration bits of UCB0.
- Read protection active flag FCON.RPA; indicates the state of an installed read protection (active or temporarily disabled).

There are also two Flash access disable bits in register FCON, DCF (= Disable Code Fetch) and DDF (= Disable Data Fetch), which control the Flash access during an active read protection. During execution of the Boot ROM program (see [Page 4-18](#)) with read protection, the following three locations for program code execution can be selected:

- Case 1: code execution from internal PFLASH
- Case 2: code execution from internal program memory after execution of a bootstrap loader
- Case 3: code execution from external program memory via EBU

Program Memory Unit

With a reset operation, the FCON register bits DCF and DDF are set. When starting code execution from internal PFLASH, these flags become cleared by the Boot ROM program before Boot ROM exit. This means that code execution from PFLASH and data read accesses from PFLASH or DFLASH is generally allowed. The program code, which executes the data read accesses from Flash, can be located in any program memory.

When starting from internal or external program RAM (case 2 and 3), flags DCF and DDF remain set at Boot ROM exit. This means, that code execution from PFLASH and data read access from PFLASH or DFLASH is disabled. In this case, code or data accesses from PFLASH or DFLASH are only possible while read protection is temporarily disabled by the password protected Disable Read Protection command (FCON.RPA is cleared). In this disable state, it is also possible to clear the DCF/DDF flags of register FCON.

Flash data accesses from dedicated bus masters others than the CPU/DMI can be disabled separately with FCON register bits DDFDBG, DDFDMA, and DDFPCP of register FCON. When such a bit is set, the corresponding bus master (Debug system, DMA controller, or PCP) is not allowed to access PFLASH or DFLASH memory. When these bits are set once, they can only be cleared again when read protection is not selected at all (inactive), or temporarily disabled.

Note: The debug interface is disabled after Boot ROM exit with read protection.

7.2.7.4 Password Check Control

The Disable Write Protection command and the Disable Read Protection command provide a protected command sequence, meaning that two 32-bit passwords must be issued within the command. Both commands are executed only if the two passwords are identical with the two keywords that are stored in the corresponding user configuration block. If one or both passwords are not identical to their related keywords, the protected sectors remain in the locked state (read- and/or write-protected) and the protection error flag FSR.PROER is set. In this case, a new Disable Write Protection command or a Disable Read Protection command is only accepted after the next TC1796 reset operation.

Note that the Disable Write Protection command can be applied for user 0 (UCB0) or user 1 (UCB1). The Disable Read Protection command can be applied only for user 0 (UCB0).

7.2.8 Error Correction and Margin Control

7.2.8.1 Dynamic Error Correction

Error detection and correction are controlled in the Flash module by using a SECDED algorithm. This algorithm calculates an 8-bit error correction code (ECC) for every 64-bit data portion in PFLASH and DFLASH. This 8-bit ECC is generated during write operations to the assembly buffer and programmed into the Flash array together with the assembly buffer data during the execution of the Write Page command. With every 64-bit read access from PFLASH and DFLASH, the associated 8-bit ECC is fetched and checked whether it is correct or not.

The ECC is defined such that an 8-bit ECC of 00_H is generated for 64-bit data portions with all bits set at 0. Therefore, after an erase operation all Flash locations including the ECCs are at logical 0, meaning that the Flash is erased with correct ECC information. On the other hand, an ECC of FF_H is generated when all bits of a 64-bit data portion are programmed with a 1.

In general, the TC1796 Flash module supports the following error detection and correction functionality for PFLASH and DFLASH read accesses:

- Single-bit error detection within 64-bit read data with on-the-fly correction
 - Status flag indication for PFLASH (FSR.PFSBER) and DFLASH (FSR.DFSBER) single-bit errors
 - Optional single-bit error Flash interrupt generation for PFLASH and DFLASH
- Double-bit error detection (no correction)
 - Status flag indication for PFLASH (FSR.PFDBER) and DFLASH (FSR.DFDBER) double-bit errors
 - Optional double-bit error Flash interrupt generation for PFLASH and DFLASH
 - In double-bit error case, the CPU/PCP can be interrupted by a PLMB bus error which further results in an interrupt from the PLMB bus control unit PBCU. This PLMB bus error interrupt can be disabled e.g. for margin check purposes.

A single-bit or a double-bit error may also be caused by a disturbed ECC with correct 64-bit data, or by a wrong selection of internal Flash access time with wait states.

Note: After the detection of a single-bit error it is generally recommended to execute a margin check procedure (described in the next section) and, as a result of this check, to erase the violated sector and reprogram it completely new (to avoid double-bit errors).

7.2.8.2 Margin Check Control

The margin control feature of the TC1796 Flash module makes it possible to change the sensing levels of the sense amplifiers of the Flash array bit lines for read operations. With this feature, problematic Flash array bits can be found in advance, before they convert to stable erratic bits. Margin control is supported separately for PFLASH and DFLASH.

Two Margin Control Registers, MARP for PFLASH and MARD for DFLASH, are provided to adapt the margin levels. Two margin levels, standard or high margin level, are selectable for 0 or 1 level detection. Standard margin levels are selected as default after reset.

Since problematic Flash array bits always change their value from a valid 1 to an invalid 0 level, these bits can be identified and corrected in advance by executing the following steps:

1. Checking the high-level margin by setting bit field MARGIN1 in the MARP or MARD register to 01_B (high margin selected) and MARGIN0 to 00_B (standard margin selected).
2. Reading related Flash memory locations while observing the single-bit and double-bit Flash interrupts; if an error occurs, executing an erase and reprogramming of the corresponding sector.

A zero check (analogous procedure as shown in steps 1 and 2 for programmed cells) is normally not necessary, because the energetically stabilized state of Flash cells is close to the zero-state.

Only one high margin level (high or standard level, either for PFLASH or DFLASH) is allowed at a time. During erase or programming operations only the standard margin selections are allowed.

Attention: *To increase the security and to inhibit unintended write accesses to the MARP register, the standard “ENDINIT” protection feature (including watchdog password access control) should be used for write accesses to the MARP register (for PFLASH). Using this mechanism, it is possible to change the read margins of the PFLASH only before End-of-initialization or after ENDINIT with valid password access to register WDT_CON0. The MARD register for Data Flash is not especially protected.*

Note: After changing a margin level in the MARP or MARD registers, a wait time of 10 μ s must elapse before Flash read operations with the modified margin can be executed.

7.2.9 Flash Interrupt Generation and Control

One interrupt request can be issued by the Flash module. The related interrupt control register is located in the DMA controller.

The Flash interrupt can be issued when any of the following events occur:

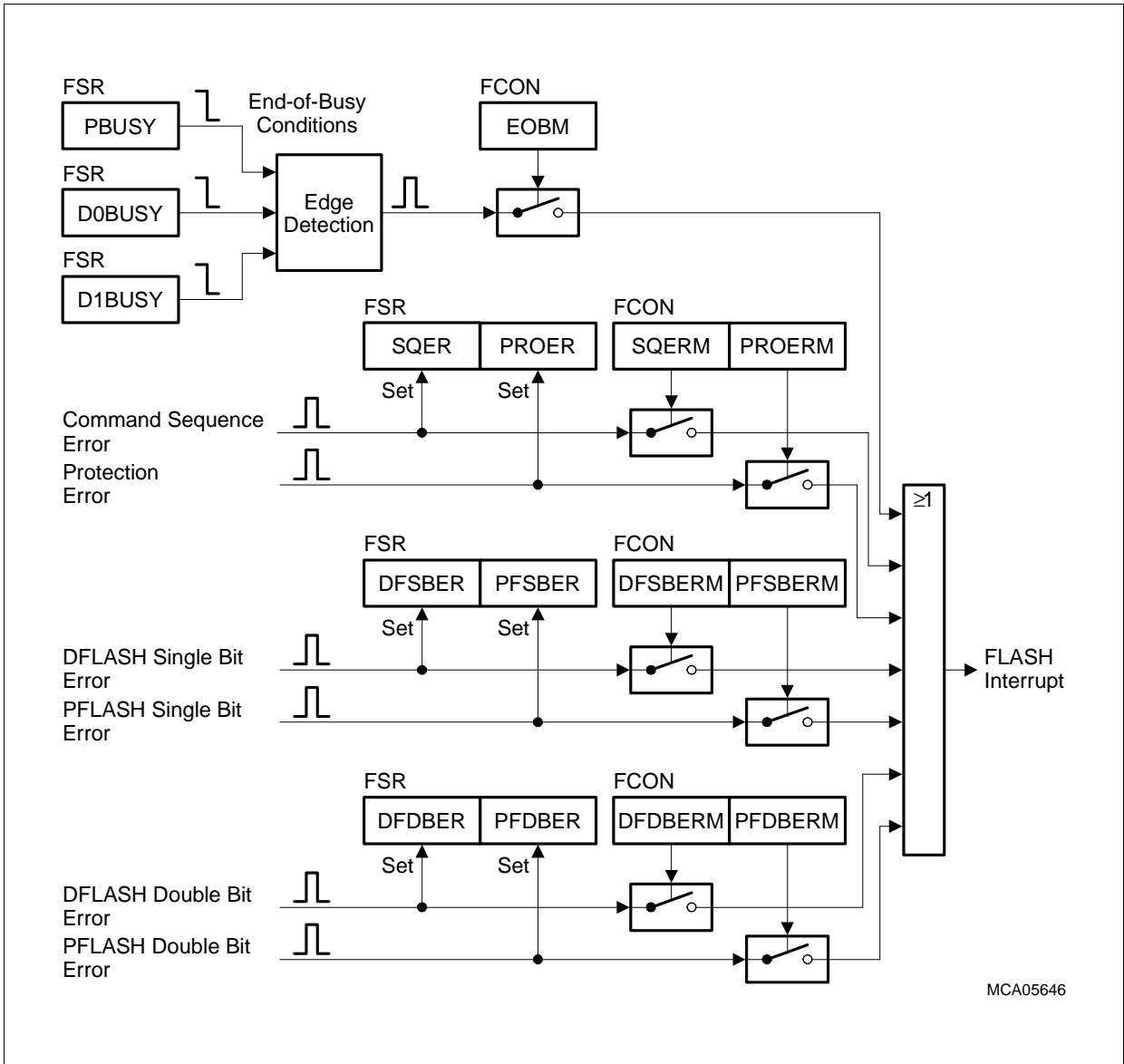
- End-of-busy: programming or erase operation of Flash bank finished
- Command sequence error
- Protection error
- Single-bit error in PFLASH
- Single-bit error in DFLASH
- Double-bit error in PFLASH
- Double-bit error in DFLASH

The source of the Flash interrupt is indicated in the Flash Status Register FSR by the corresponding busy and error flags. Every interrupt source is disabled after reset, and can be enabled individually via dedicated enable bits in the Flash Configuration Register FCON.

The end-of-busy interrupt becomes active whenever a programming or erasing operation is finished. At this event, one of the three FSR busy flags PBUSY, DOBUSY, or D1BUSY is cleared from 1 to 0. End-of-busy interrupts are enabled by setting the FCON.EOBM bit.

The error flags in the Flash Status Register are controlled independently of the interrupt configuration as defined in the FCON register. Thus, they may be polled without interrupt support. When set, an error flag has to be cleared by the user with the Clear Status command. All error flags are also cleared with any reset.

Errors, which are caused by a reset during program or erase operation, are not indicate, but its detection is supported via the PROG and ERASE flags in the Flash Status Register FSR. These two flags are cleared only with a power-on reset. Therefore, it is possible to detect an aborted Flash operation, if after every terminated Flash operation these flags are immediately cleared by the user with the Clear Status command, and if these flags are checked with every user-boot after reset (all except power-on reset).



MCA05646

Figure 7-6 Flash Service Request Control

7.2.10 Flash Power Supply, Power Saving and Reset

This section describes some power supply, power saving mode, and reset related issues of the Flash modules.

7.2.10.1 Power Supply

The TC1796 Flash module has a separate 3.3 V power supply line V_{DDFL3} . It is used to generate on-chip regulated voltages for the program and erase operations, as well as for read operations.

If a power-fail occurs during a Flash erase or programming operation, a proper control of the PORST input is required to avoid a possibly destructive operation (see Data Sheet for details of power-fail control).

7.2.10.2 Sleep Mode

The TC1796 Flash module provides a sleep mode which makes it possible to reduce the power consumption of the Flash module. Flash sleep mode is only entered if the Flash module is in idle state with all pending or active operations processed and terminated. Only in this case, the Flash module executes a ramp-down into Flash sleep mode. In Flash sleep mode, all sense amplifiers are switched off, the internal voltages are ramped down, and the Flash array oscillator is switched off. Next, status flag FSR.SLM is set indicating an active Flash sleep mode.

Flash sleep mode can be initiated by software (separately from the other modules in the TC1796) by setting bit FCON.SLEEP.

Flash sleep mode can also be initiated by hardware when the sleep mode is initiated by the power management system of the TC1796 (see [Section 5.1](#) on [Page 5-2](#)). A TC1796 device sleep mode request affects the Flash module only if the external sleep request disable bit FSR.ESLDIS is not set.

Wake-up from Flash sleep mode is initiated by clearing bit FCON.SLEEP or when the device sleep mode signal from the power management system is released. In both cases, the wake-up from sleep mode is controlled in the Flash module by starting the Flash array oscillator again, ramping-up the internal voltages, and returning to read mode. When again in read mode, also status flag FSR.SLM becomes cleared.

During ramp-down, active sleep mode, and wake-up from Flash sleep mode to active mode, the Flash reports to be busy also by setting the bits FSR.PBUSY, FSR.D0BUSY, and FSR.D1BUSY.

7.2.10.3 Shut-Down Mode

The Flash module can also be put into a shut-down mode in which it is switched off completely and cannot be used at all. This shut-down mode is not available for a user

program but controlled by the Boot ROM startup procedure that is executed after each reset operation.

7.2.10.4 Reset Control

The Flash module uses

- The system reset, which is represented by the fast LMB reset; this reset includes all reset sources (power-on, hardware, software and watchdog reset), and
- The power-on reset.

The Flash will be automatically reset to Read Mode within 900 μ sec (including Flash ramp-up) after every reset represented by the system reset. When that happens, any erase or programming operation that is running is aborted, and the Flash might have erroneous data in the location being operated on (thus, the aborted operation must be repeated). Such an aborted state can be detected by careful handling of the PROG and ERASE flags in FSR.

Note: The startup time after reset until the first user instruction includes the Flash ramp-up and the Boot ROM startup, and depends on the reset type (cold or warm) and on the clock frequency (PLL free-running or application frequency). But for all cases and $f_{CPU} > 12$ MHz, it is < 1.5 msec.

7.2.11 Flash Registers

FLASH Register Overview

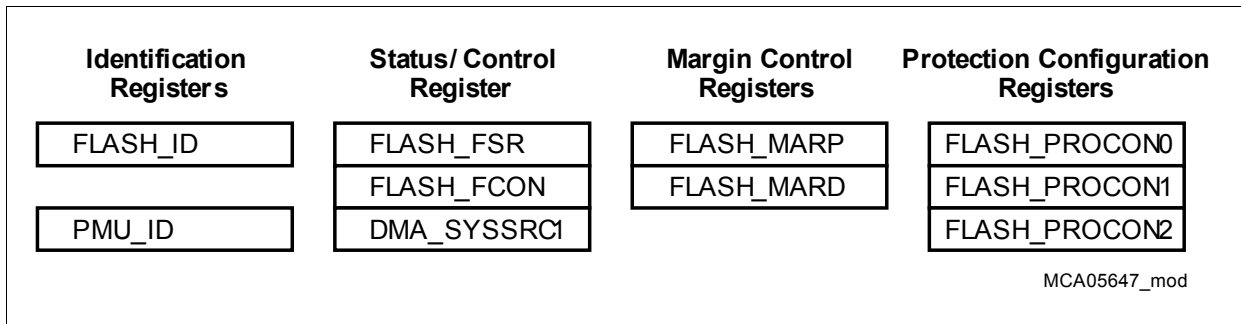


Figure 7-7 Flash Registers

The complete and detailed address map of the FLASH registers is described in [Table 18-36](#) on [Page 18-117](#) of the TC1796 User's Manual System Units part (Volume 1).

Table 7-17 Registers Address Space -Flash Registers

Module	Base Address	End Address	Note
FLASH	F800 1000 _H	F800 23FF _H	–
PMU	F800 0500 _H	F800 05FF _H	–

Table 7-18 Registers Overview - Flash Registers

Register Short Name ¹⁾	Register Long Name	Offset Address	Description see
FLASH_ID	Flash Module Identification Register	1008 _H	Page 7-41
FLASH_FSR	Flash Status Register	1010 _H	Page 7-43
FLASH_FCON	Flash Configuration Register	1014 _H	Page 7-53
FLASH_MARP	Flash Margin Control Register PFLASH	1018 _H	Page 7-51
FLASH_MARD	Flash Margin Control Register DFLASH	101C _H	Page 7-52
FLASH_PROCON0	Flash Protection Config. Reg. User 0	1020 _H	Page 7-59
FLASH_PROCON1	Flash Protection Config. Reg. User 1	1024 _H	Page 7-60
FLASH_PROCON2	Flash Protection Config. Reg. User 2	1028 _H	Page 7-61
DMA_SYSSRC1	DMA System Interrupt Service Request Control Register 1	²⁾	Page 12-108

Program Memory Unit

- 1) The Flash register short names are also referenced in other parts of this chapter without the module prefix name "FLASH_".
- 2) This register is located in the address range for the DMA controller.

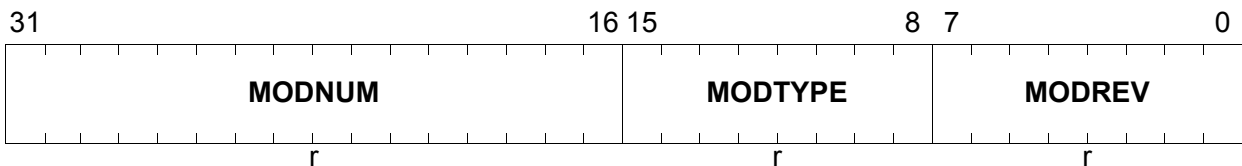
7.2.11.1 Flash and PMU Module Identification Registers

The Flash Module Identification Register ID contains read-only information about the Flash module version.

FLASH_ID

FLASH Module Identification Register(1008_H)

Reset Value: 0031 C0XX_H



Field	Bits	Type	Description
MODREV	[7:0]	r	Module Revision Number MODREV defines the module revision number. The value of a module revision starts with 01 _H (first revision).
MODTYPE	[15:8]	r	Module Type This bit field defines the module as a 32-bit module: C0 _H
MODNUM	[31:16]	r	Module Number Value This bit field defines the module identification number for the FLASH: 0031 _H

7.2.11.2 Flash Status Register

The read-only Flash Status Register indicates the overall status of the Flash module. It includes busy, program, erase, error, protection, and sleep mode flags.

FLASH_FSR

Flash Status Register

(1010_H)

Reset Value: 0XXX X0X0_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VER	0	SLM	0	W PRO DIS1	W PRO DIS0	0	W PRO IN2	W PRO IN1	W PRO IN0	0	R PRO DIS	R PRO IN	0	PRO IN	
rh	r	rh	r	rh	rh	r	rh	rh	rh	r	rh	rh	r	rh	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DF DB ER	PF DB ER	DF SB ER	PF SB ER	PRO ER	SQ ER	DF OP ER	PF OP ER	DF PA GE	PF PA GE	ERA SE	PR OG	D1 BU SY	D0 BU SY	FA BU SY	P BU SY
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
PBUSY	0	rh	<p>PFLASH Busy</p> <p>This flag indicates the busy state of the PFLASH during program or erase operations. It also indicates when the PFLASH is not in Read Mode, e.g. in ramp-up state. PBUSY is cleared by any reset operation.</p> <p>0_B PFLASH is ready and in Read Mode. (default after reset)</p> <p>1_B PFLASH is busy (program, erase, ramp-up, or in sleep mode) and it is not in Read Mode.</p>
FABUSY	1	rh	<p>Flash Array Busy</p> <p>This status flag is a flag for test purposes that should not be used by software drivers. It indicates whether any of the Flash arrays is in busy state. FABUSY is cleared by any reset operation.</p> <p>0_B Flash array banks are not busy and are in Read Mode (default after reset).</p> <p>1_B At least one of the Flash array banks or the whole array is busy (program, erase, ramp-up or sleep mode). Flash read accesses are not possible.</p>

Program Memory Unit

Field	Bits	Type	Description
D0BUSY	2	rh	<p>DFLASH Bank 0 Busy This flag indicates the busy state of DFLASH bank 0 during program or erase operations. It also indicates when the DFLASH bank 0 is not in Read Mode, e.g. in ramp-up state. D0BUSY is cleared by any reset operation.</p> <p>0_B DFLASH bank 0 is ready and in Read Mode. (default after reset)</p> <p>1_B DFLASH bank 0 is busy (program, erase, ramp-up or sleep mode) and it is not in Read Mode.</p>
D1BUSY	3	rh	<p>DFLASH Bank 1 Busy This flag indicates the busy state of DFLASH bank 1 during program or erase operations. It also indicates when the DFLASH bank 1 is not in Read Mode, e.g. in ramp-up state. D0BUSY is cleared by any reset operation.</p> <p>0_B DFLASH bank 1 is ready and in Read Mode. (default after reset)</p> <p>1_B DFLASH bank 1 is busy (program, erase, ramp-up or sleep mode) and it is not in Read Mode.</p>
PROG	4	rh	<p>Programming State This flag indicates whether PFLASH or DFLASH is currently programmed or have been programmed. PROG is set with the last cycle of a write page command. After a programming operation, this bit remains set. PROG is cleared by a Clear Status command or by a power-on reset.</p> <p>0_B There is no programming operation running, requested or finished.</p> <p>1_B A programming operation (write page) is running, requested or finished.</p>

Program Memory Unit

Field	Bits	Type	Description
ERASE	5	rh	<p>Erase State This flag indicates whether PFLASH or DFLASH is currently erased or have been erased. ERASE is set with the last cycle of an erase command sequence. After an erase operation this bit remains set. ERASE is cleared by a Clear Status command or by a power-on reset.</p> <p>0_B There is no erase operation running, requested, or finished.</p> <p>1_B An erase operation is running, requested, or finished.</p>
PFPAGE	6	rh	<p>PFLASH in Page Mode This flag indicates whether the PFLASH is in Page Mode or not. It is set with the enter Page Mode command and cleared by a Write Page command, or by a Reset-to-Read command, or by any reset operation.</p> <p>0_B PFLASH is not in Page Mode. (default after reset)</p> <p>1_B PFLASH is in Page Mode. The page assembly buffer of the PFLASH is ready to be filled.</p>
DFPAGE	7	rh	<p>DFLASH in Page Mode This flag indicates whether the DFLASH is in Page Mode or not. It is set with the enter Page Mode command and cleared by a write Page Mode command, or by a Reset-to-Read command, or by any reset operation.</p> <p>0_B DFLASH is not in Page Mode. (default after reset)</p> <p>1_B DFLASH is in Page Mode. The page assembly buffer of the DFLASH is ready to be filled.</p>
PFOPER	8	rh	<p>PFLASH Operation Error This flag indicates a flash array error during PFLASH operation. PFOPER is cleared by a reset operation, by a Reset-to-Read command, or by a Clear Status command.</p> <p>0_B PFLASH operation is finished or currently running without error.</p> <p>1_B PFLASH operation is aborted caused by a PFLASH array failure.</p>

Program Memory Unit

Field	Bits	Type	Description
DFOPER	9	rh	<p>DFLASH Operation Error</p> <p>This flag indicates a flash array error during DFLASH operation. DFOPER is cleared by a reset operation, by a Reset-to-Read command, or by a Clear Status command.</p> <p>0_B DFLASH operation is finished or currently running without error.</p> <p>1_B DFLASH operation is aborted caused by a DFLASH array failure.</p>
SQER	10	rh	<p>Command Sequence Error</p> <p>This flag indicates a command sequence error. SQER is cleared by a Reset-to-Read command, or by a Clear Status command, or by any reset operation. SQER is not set when a command sequence is aborted by a Reset-to-Read command.</p> <p>0_B No command sequence error has occurred. (default after reset)</p> <p>1_B An invalid command sequence has been detected (command is not executed).</p>
PROER	11	rh	<p>Protection Error</p> <p>This flag indicates a protection error. Such an error is a command that is not allowed, for example an erase sector or write page command addressing a locked sector. It also indicates when wrong passwords are used in disable read or disable write protection commands. PROER is cleared by a Reset-to-Read command, or by a Clear Status command, or by any reset operation.</p> <p>0_B No protection error has occurred. (default after reset)</p> <p>1_B A protection error has occurred.</p>
PFSBER	12	rh	<p>PFLASH Single-Bit Error and Correction</p> <p>This flag indicates the occurrence of a single-bit error in the PFLASH that has been corrected. PFSBER is cleared by a Reset-to-Read command, or by a Clear Status command, or by any reset operation.</p> <p>0_B No error has been detected. (default after reset)</p> <p>1_B A single-bit error in PFLASH has been detected and corrected.</p>

Program Memory Unit

Field	Bits	Type	Description
DFSBER	13	rh	<p>DFLASH Single-Bit Error and Correction</p> <p>This flag indicates the occurrence of a single-bit error in the DFLASH that has been corrected. DFSBER is cleared by a Reset-to-Read command, or by a Clear Status command, or by any reset operation.</p> <p>0_B No error has been detected. (default after reset)</p> <p>1_B A single-bit error in DFLASH has been detected and corrected.</p>
PFDBER	14	rh	<p>PFLASH Double-Bit Error</p> <p>This flag indicates the occurrence of a double-bit error in the PFLASH. PFDBER is cleared by a Reset-to-Read command, or by a Clear Status command, or by any reset operation.</p> <p>0_B No error has been detected. (default after reset)</p> <p>1_B A double-bit error in PFLASH has been detected.</p>
DFDBER	15	rh	<p>DFLASH Double-Bit Error</p> <p>This flag indicates the occurrence of a double-bit error in the DFLASH. DFDBER is cleared by a Reset-to-Read command, or by a Clear Status command, or by any reset operation.</p> <p>0_B No error has been detected. (default after reset)</p> <p>1_B A double-bit error in DFLASH has been detected.</p>
PROIN	16	rh	<p>Protection Installed</p> <p>This flag indicates whether read or/and write protection is correctly installed. This bit is updated only if a modified protection installation is detected at a reset operation.</p> <p>0_B No protection is installed.</p> <p>1_B A read or/and write protection is correctly installed.</p>

Program Memory Unit

Field	Bits	Type	Description
RPROIN	18	rh	<p>Read Protection Installed</p> <p>This flag indicates whether read protection is correctly installed. This bit is updated only if a modified read protection installation is detected at a reset operation.</p> <p>0_B No read protection installed. 1_B Read protection is correctly installed.</p>
RPRODIS	19	rh	<p>Read Protection Disable State</p> <p>This flag indicates whether read protection is temporarily disabled. RPRODIS is cleared by any reset operation and by the Resume Protection command.</p> <p>0_B The read protection is installed and enabled, or it is not installed (default after reset). 1_B The read and global write protection is temporarily disabled. External Flash read as well as programming or erase on not separately write-protected sectors is possible.</p>
WPROIN0	21	rh	<p>UCB0 Write Protection Installed</p> <p>This flag indicates whether sector write protection in UCB0 is correctly installed and confirmed. This bit is updated only if a modified UCB0 protection installation is detected at a reset operation.</p> <p>0_B No write protection installed in UCB0. 1_B Sector write protection is correctly installed in UCB0.</p>
WPROIN1	22	rh	<p>UCB1 Write Protection Installed</p> <p>This flag indicates whether sector write protection in UCB1 is correctly installed and confirmed. This bit is updated only if a modified UCB1 protection installation is detected at a reset operation.</p> <p>0_B No write protection installed in UCB1. 1_B Sector write protection is correctly installed in UCB1.</p>

Program Memory Unit

Field	Bits	Type	Description
WPROIN2	23	rh	<p>UCB2 Write Protection (OTP) Installed This flag indicates whether sector write protection (OTP) in user configuration block UCB2 is correctly installed and confirmed.</p> <p>0_B No OTP write protection installed in UCB2. 1_B Sector write protection correctly installed in UCB2. The protection is permanently locked.</p>
WPRODIS0	25	rh	<p>UCB0 Write Protection Disabled This flag indicates whether PFLASH sectors that are write-protected by user configuration block UCB0 are temporarily unlocked. WPRODIS0 is cleared by any reset operation and by the Resume Protection command.</p> <p>0_B All write-protected sectors of UCB0 are locked or write protection is not installed in UCB0. 1_B All write-protected sectors of UCB0 and UCB1 are temporarily unlocked. The programming or erasing of UCB0 is possible if no read protection is installed.</p> <p>Hierarchical protection levels: Unlock state of user-level zero also includes sectors, that are protected by user 1.</p>
WPRODIS1	26	rh	<p>UCB1 Write Protection Disabled This flag indicates whether PFLASH sectors that are write-protected by UCB1 are temporarily unlocked. WPRODIS1 is cleared by any reset operation and by the Resume Protection command.</p> <p>0_B All write-protected sectors of UCB1 are locked or write protection is not installed in UCB1. 1_B All write-protected sectors of UCB1 are temporarily unlocked if not globally (read protection) or concurrently (write protection) locked by UCB0/UCB2. The programming or erasing of UCB1 is possible.</p>
SLM	28	rh	<p>Flash Sleep Mode This flag indicates whether the Flash module is in sleep mode. SLM is cleared by any reset operation or when Flash sleep mode is left.</p> <p>0_B Flash is not in sleep mode (default after reset). 1_B Flash is in sleep mode.</p>

Program Memory Unit

Field	Bits	Type	Description
VER	31	rh	<p>Verify Error This flag indicates whether a Flash page has been correctly programmed. VER is cleared by any reset operation and by the Clear Status command.</p> <p>0_B A Flash page is correctly programmed. All programmed 1 bits have a full quality.</p> <p>1_B A Flash page programming verify error has been detected. A correction of weak bits by the programming algorithm was unsuccessful during the last programming operation.</p> <p><i>Note: There is no interrupt generated when VER is set.</i></p>
0	17, 20, 24, 27, [30:29]	r	<p>Reserved Read as 0.</p>

7.2.11.3 Margin Control Registers

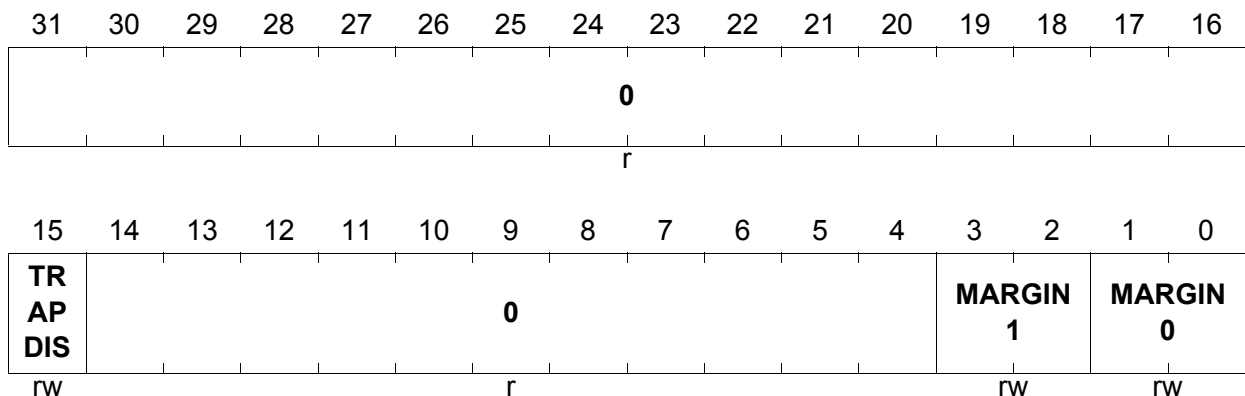
The Margin Control Registers for Program Flash (MARP) and for Data Flash (MARD) are defined as follows:

FLASH_MARP

Flash Margin Control Register PFLASH

(1018_H)

Reset Value: 0000 8000_H



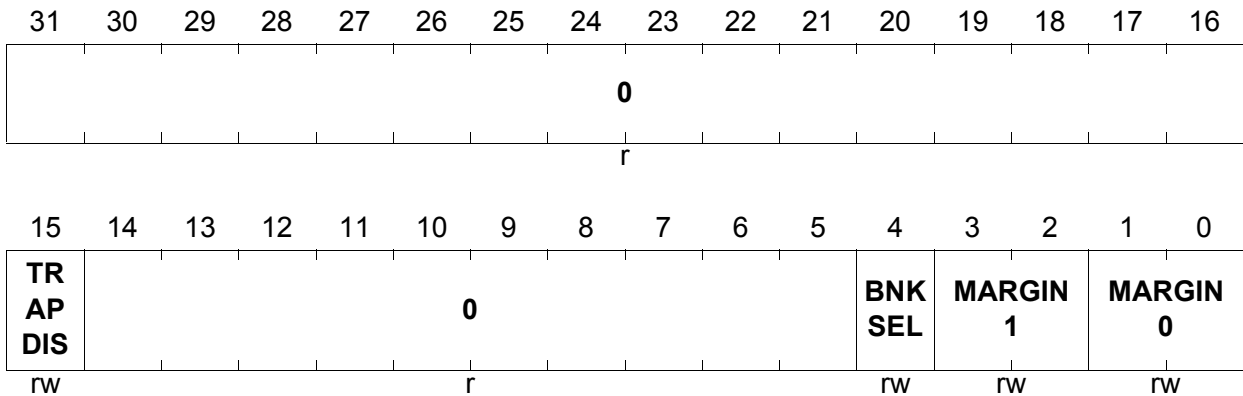
Field	Bits	Type	Description
MARGIN0	[1:0]	rw	PFLASH Margin Selection for Low Level 00 _B Standard margin selected 01 _B High margin for 0 (low) level selected 10 _B Reserved 11 _B Reserved
MARGIN1	[3:2]	rw	PFLASH Margin Selection for High Level 00 _B Standard margin selected 01 _B High margin for 1 (high) level selected 10 _B Reserved 11 _B Reserved
TRAPDIS	15	rw	PFLASH Double-Bit Error LMB Bus Error Disable 0 _B If a double-bit error occurs in PFLASH, a bus error at PLMB or DLMB is generated. 1 _B If a double-bit error occurs in PFLASH, no LMB bus error is generated. After Boot ROM exit, double-bit error traps are enabled (TRAPDIS = 0).
0	[14:4], [31:16]	r	Reserved Read as 0; should be written with 0.

FLASH_MARD

Flash Margin Control Register DFLASH

(101C_H)

Reset Value: 0000 8000_H



Field	Bits	Type	Description
MARGIN0	[1:0]	rw	DFLASH Margin Selection for Low Level 00 _B Standard margin selected 01 _B High margin for 0 (low) level selected 10 _B Reserved 11 _B Reserved
MARGIN1	[3:2]	rw	DFLASH Margin Selection for High Level 00 _B Standard margin selected 01 _B High margin for 1 (high) level selected 10 _B Reserved 11 _B Reserved
BNKSEL	4	rw	DFLASH Bank Selection 0 _B DFLASH bank 0 selected for margin control 1 _B DFLASH bank 1 selected for margin control
TRAPDIS	15	rw	DFLASH Double-Bit Error DLMB Bus Error Disable 0 _B If a double-bit error occurs in DFLASH, a DLMB bus error is generated. 1 _B If a double-bit error occurs in DFLASH, no DLMB bus error is generated. After Boot ROM exit, double-bit error traps are enabled (TRAPDIS = 0).
0	[14:5], [31:16]	r	Reserved Read as 0; should be written with 0.

7.2.11.4 Flash Configuration Register

The Flash Configuration Register FCON controls general Flash configuration functions:

- Number of internal wait states for Flash accesses (without or with word-line hit)
- Indication of installed and active read protection
- Instruction and data access control for read protection
- Interrupt enable/mask bits
- Power reduction and shut-down control

FCON is an Endinit-protected register. The reset value of register FCON as indicated below shows the value after ramp-up and after execution of the startup procedure out of Boot ROM.

FLASH_FCON

Flash Configuration Register

(1014_H)

Reset Value: 000X 0636_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EOB M	DF DB ERM	PF DB ERM	DF SB ERM	PF SB ERM	PRO ERM	SQ ERM		0		DDF PCP	DDF DMA	DDF DBG	DDF	DCF	RPA
rw	rw	rw	rw	rw	rw	rw		r		rw	rw	rw	rwh	rwh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SL EEP	ESL DIS	0		WS EC DF		WS DFLASH		0		WS WLHIT		WS EC PF		WS PFLASH	
rw	rw	r		rw		rw		r		rw		rw		rw	

Field	Bits	Type	Description
WSPFLASH ¹⁾	[2:0]	rw	<p>Wait States for PFLASH Read Access</p> <p>This bit field determines the number of internal wait states that are used for an initial PFLASH read access.</p> <p>000_B 1 clock cycle wait state selected</p> <p>001_B 1 clock cycle wait state selected</p> <p>010_B 2 clock cycles wait state selected</p> <p>011_B 3 clock cycles wait state selected</p> <p>100_B 4 clock cycles wait state selected</p> <p>101_B 5 clock cycles wait state selected</p> <p>110_B 6 clock cycles wait state selected (default after reset)</p> <p>111_B 7 clock cycles wait state selected</p>

Program Memory Unit

Field	Bits	Type	Description
WSECPF ¹⁾	3	rw	<p>Wait State for PFLASH Error Correction</p> <p>This bit determines whether an additional wait state is inserted for error correction during read accesses to PFLASH.</p> <p>0_B No additional wait state inserted. 1_B One additional wait state inserted for PFLASH error correction.</p> <p>If enabled, this wait state is only used for the first transfer of a burst transfer.</p>
WSWLHIT ¹⁾	[6:4]	rw	<p>Wait States for PFLASH Read Access with Wordline Hit</p> <p>This bit field determines the number of internal wait states that are used for an initial read access to the same word-line (512 byte) of the PFLASH memory as the last PFLASH access.</p> <p>000_B Reserved 001_B Flash read hit access in one clock cycle 010_B Hit access in 2 clock cycles 011_B Hit access in 3 clock cycles (default after reset) 100_B Hit access in 4 clock cycles 101_B Flash access in 5 clock cycles 110_B Flash access in 6 clock cycles (default after Boot ROM exit; see Table 4-4 on Page 4-19) 111_B Reserved</p>
WSDFLASH ¹⁾	[10:8]	rw	<p>Wait States for DFLASH Read Access</p> <p>This bit field determines the number of internal wait states that are used for a DFLASH read access.</p> <p>000_B 1 clock cycle wait state selected 001_B 1 clock cycle wait state selected 010_B 2 clock cycles wait state selected 011_B 3 clock cycles wait state selected 100_B 4 clock cycles wait state selected 101_B 5 clock cycles wait state selected 110_B 6 clock cycles wait state selected (default after reset) 111_B 7 clock cycles wait state selected</p>

Program Memory Unit

Field	Bits	Type	Description
WSECDF ¹⁾	11	rw	<p>Wait State for DFLASH Error Correction</p> <p>This bit determines whether an additional wait state is inserted for error correction during read accesses to DFLASH.</p> <p>0_B No additional wait state inserted. 1_B One additional wait state inserted for DFLASH error correction.</p>
ESLDIS	14	rw	<p>External Sleep Request Disable</p> <p>0_B External sleep mode request also requests the Flash sleep mode. 1_B External sleep mode request does not request the Flash sleep mode.</p> <p>The external sleep mode is controlled and requested by the Power Management System of the SCU.</p>
SLEEP	15	rw	<p>Flash Sleep Mode Control</p> <p>This bit controls the Flash sleep mode of the Flash module.</p> <p>0_B Normal Flash mode or wake-up mode is active. 1_B Flash sleep mode is selected.</p> <p>Flash sleep mode is left (wake-up from sleep) by clearing the SLEEP bit.</p>
RPA	16	rh	<p>Read Protection Activated</p> <p>This flag indicates the status of the read protection.</p> <p>0_B The read protection is inactive or temporarily disabled. Bits DCF and DDF can be cleared or set by software. DCF and DDF are not evaluated. 1_B The read protection is active. Bits DCF and DDF are evaluated and cannot be cleared by software.</p>

Program Memory Unit

Field	Bits	Type	Description
DCF	17	rwh	<p>Disable Code Fetch from Flash Memory</p> <p>This bit enables/disables the code fetches from the internal PFLASH memory when read protection is active. Once set, this bit can only be cleared when read protection is inactive (RPA = 0). DCF is automatically set after a reset operation. It is cleared by hardware in case of internal program start out of the PFLASH.</p> <p>0_B Code fetches from PFLASH are allowed. 1_B Code fetches from PFLASH are not allowed. DCF is not evaluated when read protection is inactive (RPA = 0).</p>
DDF	18	rwh	<p>Disable Data Fetch from Flash Memory</p> <p>This bit enables/disables the data read access from the PFLASH and DFLASH memory when read protection is active. Once set, this bit can only be cleared when read protection is inactive (RPA = 0). DDF is automatically set after a reset operation. It is cleared by hardware in case of internal program start out of the PFLASH.</p> <p>0_B Data read access to PFLASH and DFLASH is allowed. 1_B Data read access to PFLASH and DFLASH is not allowed. DDF is not evaluated when read protection is inactive (RPA = 0).</p>
DDFDBG	19	rw	<p>Disable Data Fetch from Debug System</p> <p>This bit enables/disables PFLASH and DFLASH data read accesses that are initiated by the on-chip debug system (Cerberus). Once set, this bit can only be cleared when RPA = 0.</p> <p>0_B Data read accesses from PFLASH/DFLASH initiated by the debug system are enabled. 1_B Data read accesses from PFLASH/DFLASH initiated by the debug system are disabled.</p>

Program Memory Unit

Field	Bits	Type	Description
DDFDMA	20	rw	<p>Disable Data Fetch from DMA Controller This bit enables/disables PFLASH and DFLASH data read accesses that are initiated by the DMA controller. Once set, this bit can only be cleared when RPA = 0.</p> <p>0_B Data read accesses from PFLASH/DFLASH initiated by the DMA controller are enabled. 1_B Data read accesses from PFLASH/DFLASH initiated by the DMA controller are disabled.</p>
DDFPCP	21	rw	<p>Disable Data Fetch from PCP This bit enables/disables PFLASH and DFLASH data read accesses that are initiated by the PCP. Once set, this bit can only be cleared when RPA = 0.</p> <p>0_B Data read accesses from PFLASH/DFLASH initiated by the PCP are enabled. 1_B Data read accesses from PFLASH/DFLASH initiated by the PCP are disabled.</p>
SQERM	25	rw	<p>Command Sequence Error Interrupt Mask This bit disables/enables the command sequence error interrupt.</p> <p>0_B Command sequence error interrupt is disabled. 1_B Command sequence error interrupt is enabled.</p>
PROERM	26	rw	<p>Protection Error Interrupt Mask This bit disables/enables the protection error interrupt.</p> <p>0_B Protection error interrupt is disabled. 1_B Protection error interrupt is enabled.</p>
PFSBERM	27	rw	<p>PFLASH Single-Bit Error Interrupt Mask This bit disables/enables the PFLASH single-bit error interrupt.</p> <p>0_B PFLASH single-bit error interrupt is disabled. 1_B PFLASH single-bit error interrupt is enabled.</p>
DFSBERM	28	rw	<p>DFLASH Single-Bit Error Interrupt Mask This bit disables/enables the DFLASH single-bit error interrupt.</p> <p>0_B DFLASH single-bit error interrupt is disabled. 1_B DFLASH single-bit error interrupt is enabled.</p>

Program Memory Unit

Field	Bits	Type	Description
PFDBERM	29	rw	PFLASH Double-Bit Error Interrupt Mask This bit disables/enables the PFLASH double-bit error interrupt. 0 _B PFLASH double-bit error interrupt is disabled. 1 _B PFLASH double-bit error interrupt is enabled.
DFDBERM	30	rw	DFLASH Double-Bit Error Interrupt Mask This bit disables/enables the DFLASH double-bit error interrupt. 0 _B DFLASH single-bit error interrupt is disabled. 1 _B DFLASH single-bit error interrupt is enabled.
EOBM	31	rw	End-of-Busy Interrupt Mask This bit enables the end-of-busy interrupt. 0 _B End-of-busy interrupt is disabled. 1 _B End-of-busy interrupt is enabled.
0	7, [13:12], [24:22]	r	Reserved Read as 0; should be written with 0.

1) These bits and bit fields can be changed at any time, also with code fetched from Program Flash. A modified wait state parameter will be taken into account with the next corresponding access.

7.2.11.5 Protection Configuration Registers

The configuration of read/write protection and OTP functionality is indicated by three Protection Configuration Registers:

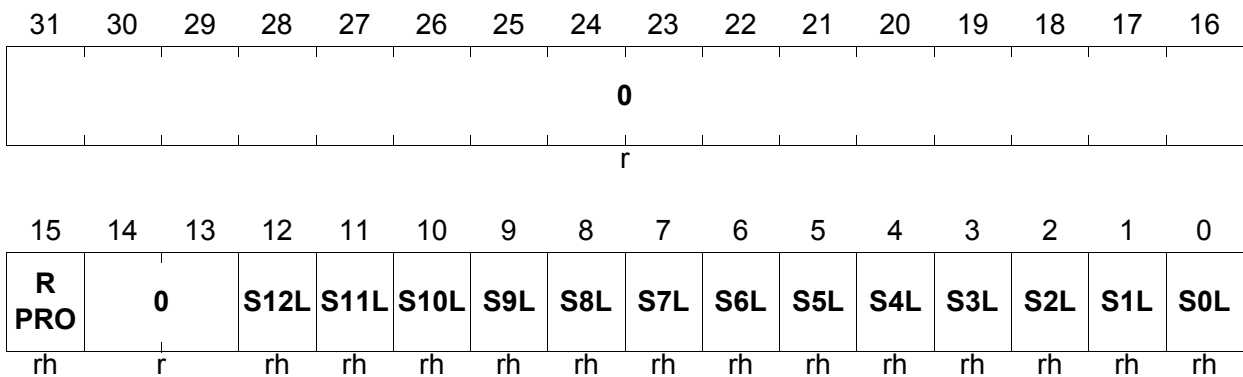
- FLASH_PROCON0 indicates the read/write protection configuration for user 0 (UCB0)
- FLASH_PROCON1 indicates the write protection configuration for user 1 (UCB1)
- FLASH_PROCON2 indicates the OTP write protection for user 2 (UCB2)

FLASH_PROCON0

Flash Protection Configuration Register User 0

(1020_H)

Reset Value: 0000 XXXX_H



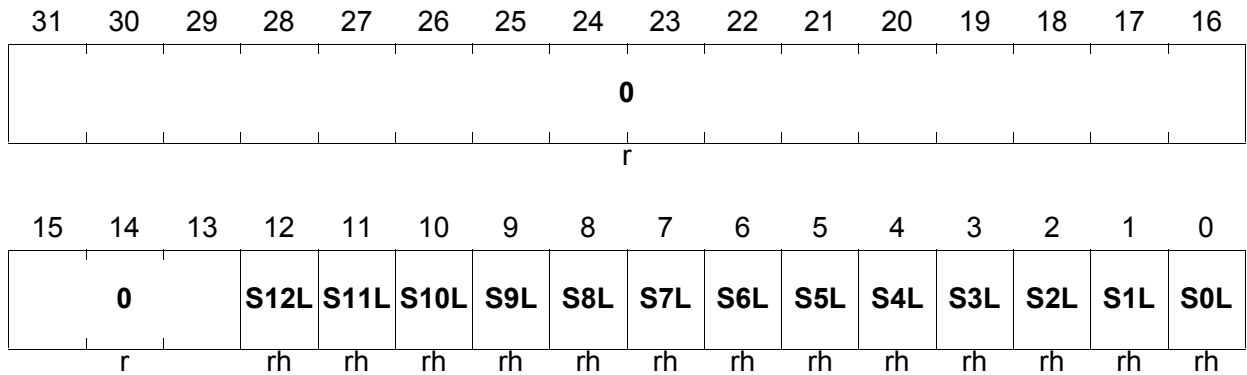
Field	Bits	Type	Description
SnL (n = 0-12)	n	rh	Sector Locked for Write Protection by User 0 These bits indicate whether PFLASH sector PSn is write-protected by user 0 or not. 0 _B No write protection is installed for PFLASH sector PSn. 1 _B Write protection is installed for PFLASH sector PSn.
RPRO	15	rh	Read Protection Configuration This bit indicates whether read protection is installed for user 0. 0 _B No read protection installed 1 _B Read protection and global write protection is installed
0	[14:13], [31:16]	r	Reserved Read as 0.

FLASH_PROCON1

Flash Protection Configuration Register User 1

(1024_H)

Reset Value: 0000 XXXX_H



Field	Bits	Type	Description
SnL (n = 0-12)	n	rh	Sector Locked for Write Protection by User 1 These bits indicate whether sector n is write-protected by user 1 or not. 0 _B No write protection installed for sector n. 1 _B Write protection installed for sector n.
0	[31:13]	r	Reserved Read as 0.

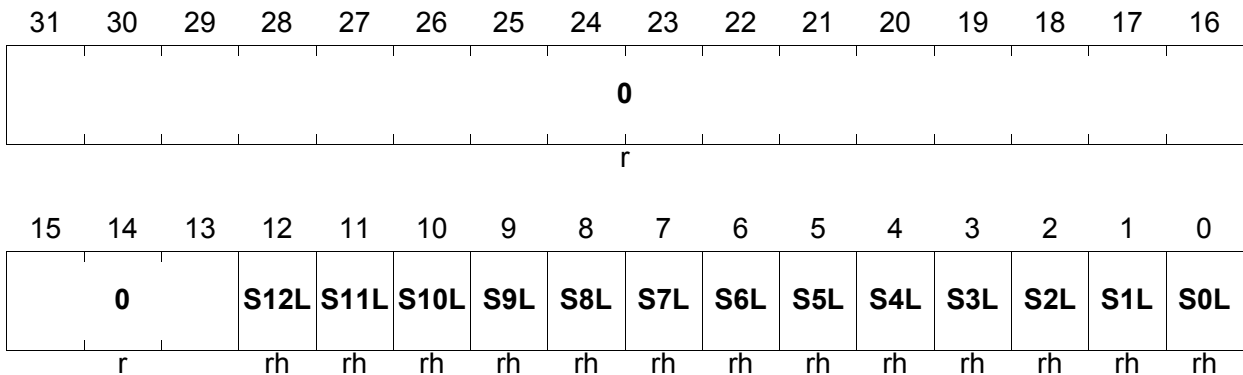
Program Memory Unit

FLASH_PROCON2

Flash Protection Configuration Register User 2

(1028_H)

Reset Value: 0000 XXXX_H



Field	Bits	Type	Description
SnL (n = 0-12)	n	rh	Sector with OTP Protection (User 2) These bits indicate whether OTP protection is installed and locked for sector n. 0 _B No OTP protection installed for sector n. 1 _B OTP protection installed for sector n. Reprogramming of this sector is no longer possible.
0	[31:13]	r	Reserved Read as 0.

7.3 Emulation Interface

The emulation memory interface shown in [Figure 7-1](#) is always disabled in the TC1796 device. This interface is a 64-bit wide memory interface that controls an emulation memory in the TC1796 emulation device, called TC1796ED. Segment 8 and 10 reserves 512 Kbyte for the emulation memory interface.

In the TC1796, a CPU read access from the emulation memory region causes a DSE trap, a PLMB and a DLMB bus error. If the emulation memory region read access is initiated by a SPB master (e.g. PCP or DMA), additionally a SPB error is generated. Write accesses to the emulation memory region by any master causes a PLMB bus error.

8 Data Memory Unit

The Data Memory Unit (DMU) shown in **Figure 8-1** contains a total of 80 Kbyte of fast static RAM memory:

- 64 Kbyte SRAM memory that can be used as
 - Normal data storage
 - Overlay memory
 - Data Flash mirror memory
- 16 Kbyte of standby memory (SBRAM)
- SRAM and SBRAM are parity protected
- With overlay address generation and control logic, that makes it possible to redirect data accesses to the PMU-based memories into the DMU or emulation memory.

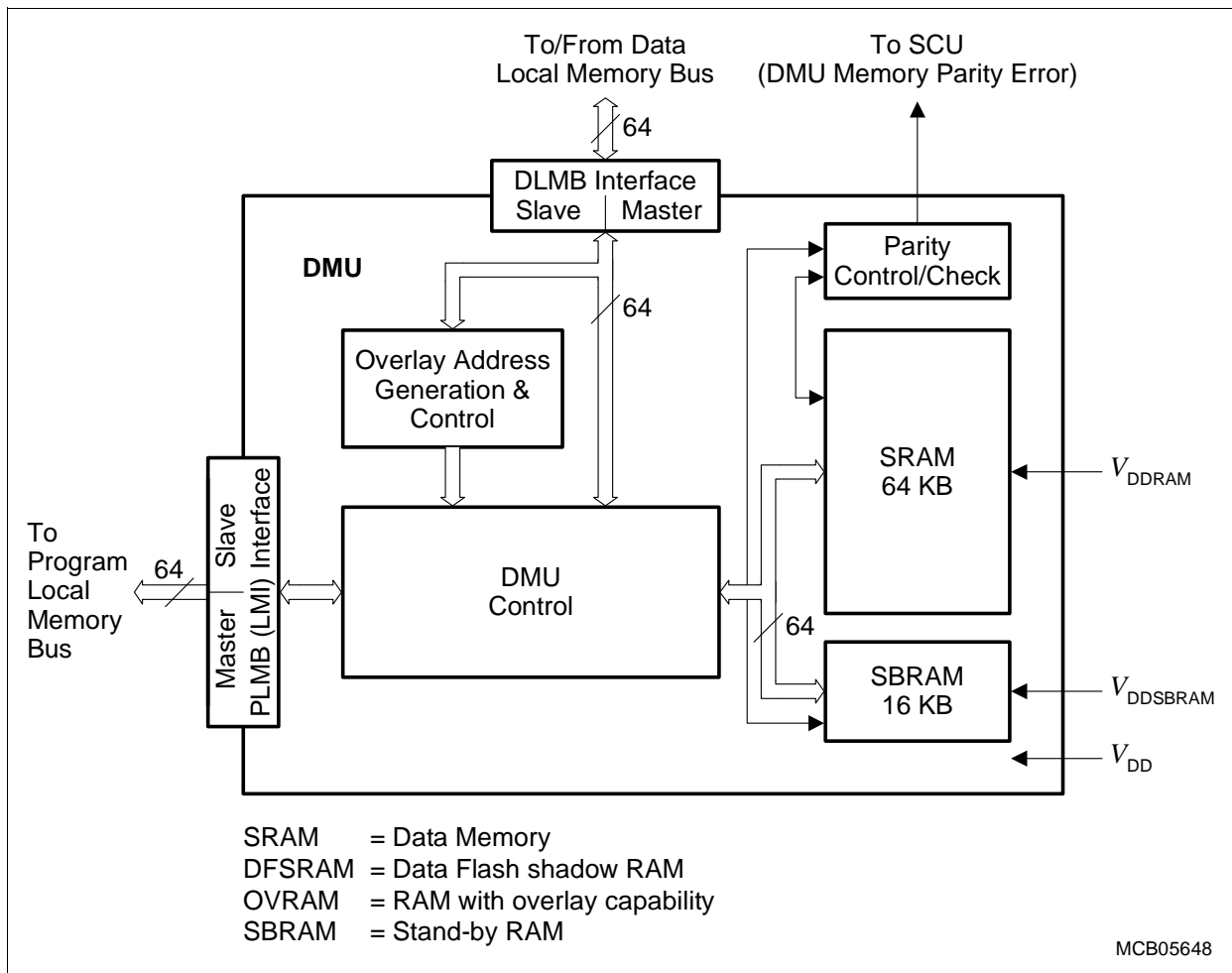


Figure 8-1 Block Diagram of the Data Memory Unit (DMU)

Note: DMU memories cannot be accessed by the PCP using burst transfers (BCOPY instruction).

8.1 DLMB/PLMB Interfaces

The DLMB interface is used for execution of data read write operations to/from the DMU memories and registers.

The PLMB interface is used when data accesses from internal or external code memory are redirected to data read accesses from the SRAM memory (data access overlay capability for calibration purposes).

8.2 SBRAM

The 16 Kbyte SBRAM provides stand-by functionality. It is connected to a separate power supply pin $V_{DD\text{SBRAM}}$. This power supply pin provides the power for normal operation of SBRAM, and during power-off state of the TC1796. If $V_{DD\text{SBRAM}}$ does not drop below a specified voltage level in power-down mode (specified in Data Sheet), the SBRAM will remain its memory content. The switch between regular supply voltage and power-down mode supply voltage must be done externally.

In order to maintain the memory contents when switching into power-down mode and coming back from power-down mode, the SBRAM must be locked. This SBRAM lock mechanism is controlled by the Stand-by SRAM Control Register SBRCTR. When writing a dedicated pattern into bit fields STBULK and STBSLK, the SBRAM will be locked. A status flag STBLOCK indicates whether the SBRAM is locked or not. To unlock the SBRAM, register SBRCTR must be written by three consecutive write operations (for a detailed description see [Page 8-12](#)).

).

If an access to the SBRAM is performed while the SBRAM is locked, an error acknowledge will be issued as response to the access.

8.3 SRAM

64 Kbyte of fast SRAM are available in the DMU. Besides being used for normal data storage, this memory can be used to shadow the active portions of the Data Flash and as overlay RAM for calibration/instrumentation support.

The SRAM can operate as fast mirror memory during Data Flash EEPROM emulation. In this application, a copy of the active DFLASH region (typically 16 Kbyte) is stored in the SRAM memory and can be accessed (read/written) very fast. More details about Data Flash EEPROM emulation are described on [Page 7-26](#).

The SRAM also provides an overlay functionality. During calibration operations, SRAM memory can be the target of redirected data accesses from internal or external code memories. The redirection process is described in the following section.

8.4 Parity Protection for DMU Memories

In the TC1796, the SBRAM and SRAM memory blocks of the DMU are equipped with a parity error detection logic that makes it possible to detect parity errors. In case of a parity error a NMI is generated.

Note that before using parity protection for SBRAM and SRAM the first time after a power-on reset operation (before setting the corresponding parity error enable bits), the corresponding memories must be completely initialized by a user program that writes every memory location of it once.

More details about the parity control for on-chip memories are described in [Section 5.5](#) on [Page 5-37](#).

8.5 Data Access Overlay Functionality

The DMU overlay functionality provides the capability to redirect data read accesses from internal or external code memory to data accesses from the DMU SRAM. This functionality makes it possible, for example, to modify program parameters (which are typically stored in the code memory) during run time of a program. Instruction fetches are not affected by the PMU data read access redirection capability. Note that read and write data accesses from/to code memory are redirected.

The basic overlay scheme is shown in [Figure 8-2](#).

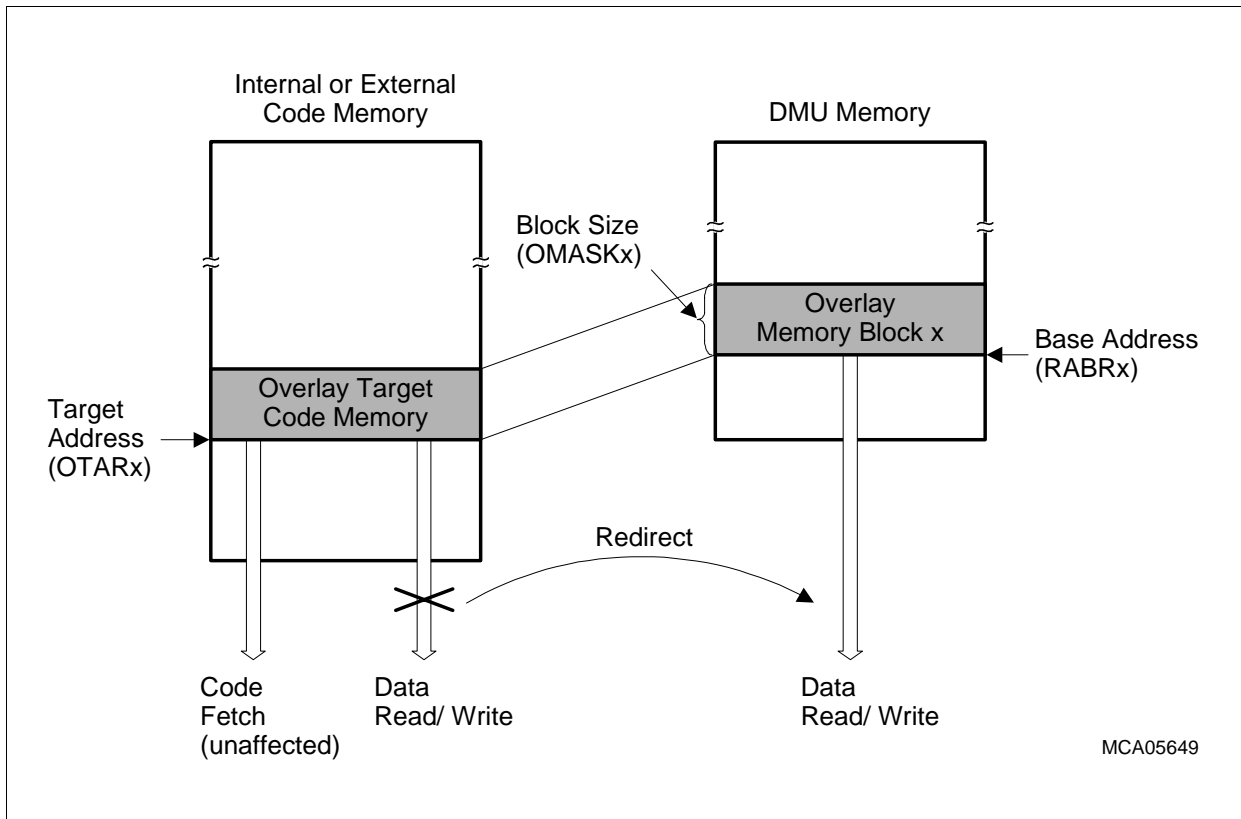


Figure 8-2 Redirection of Data Read Accesses from Code Memory to Internal Data Memory

In the TC1796, the complete 64 Kbyte SRAM of the DMU can be used as overlay memory. Sixteen overlay memory blocks within the SRAM with programmable base address and block sizes are supported, and can be individually enabled for overlay functionality. The block size of each overlay memory block can be in the range of 2 bytes up to 512 bytes.

Attention: In the TC1796ED emulation device, it is possible in addition to redirect code memory data accesses to the emulation memory of the TC1796ED.

There are four overlay RAM control registers (DMU_IOCRn, n = 0-3) assigned to control the internal overlay functionality. Each register specifies the start address of an overlaid 2 Kbyte block within the lower 128 Mbytes of segment 10 and 11. This start address can be placed on any 2 Kbyte boundary within the external code memory, using bit field OVPTR.

The principal operation of the address translation process is shown in [Figure 8-3](#).

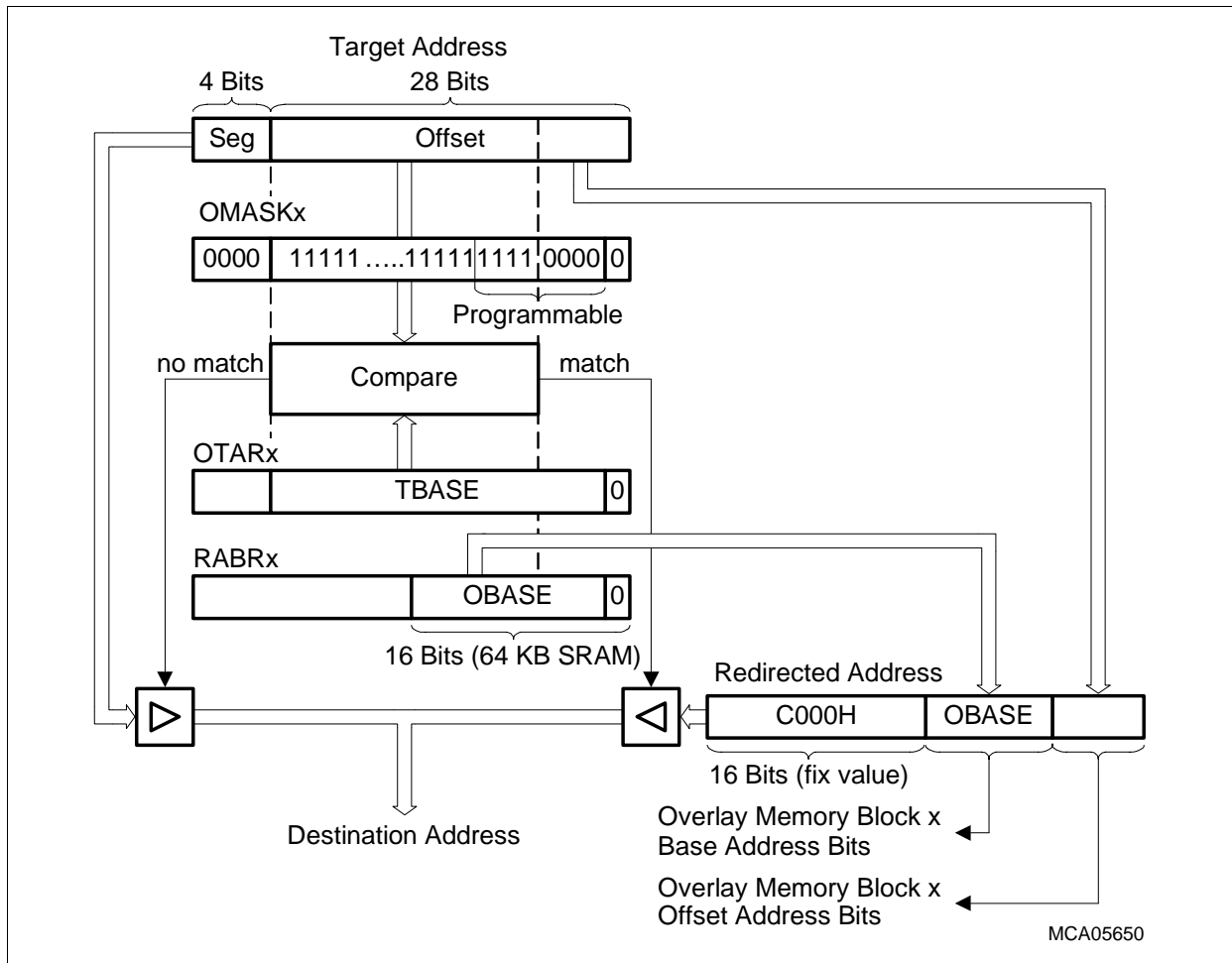


Figure 8-3 Address Translation Process

In each enabled overlay block control logic, three registers hold the information to control the overlay functionality:

- The overlay target address in the Overlay Target Address Register OTARx, which determines the base address of the code memory data block x to be redirected
- The base address of the overlay memory block in the DMU SRAM in the Redirected Address Base Register RABRx
- A mask in the Overlay Mask Register OMASKx, defining the size of the block, the address bits to check for an address match, and which bits are used from the target address and which from the redirected address base

The size of the overlay memory blocks can be 2^n times the minimal block size (2 byte or 1 Kbyte). The start address of the block can be a integer multiple of the programmed block size (natural page boundary).

If the overlay block is enabled and the segment address is 8_H or A_H , the segment offset of the target address is compared with the target block address of the overlay target address. This bit-wise comparison is qualified by the content of the mask, ignoring the address bits that form the offset into the overlaid block.

If there is no match, the original target address is taken to perform the access.

If there is a match:

- The most significant part of the segment offset, that addresses memory sizes beyond the maximum overlay memory size (64 Kbyte for DMU, 1024 Kbyte for TC1796ED emulation memory), is set to predefined values according to the address map (fixed bits in registers RABRx).
- The part of the target block address, that corresponds to the overlay block base address and is masked, is replaced by the respective overlay block base address bits (bits OBASE in RABRx, where the corresponding mask bits OMASK in registers OMASKx are set to "1").
- The address is completed by the original offset into the block; the number of bits used are determined by the bits set to "0" in the mask OMASK.

Minimum Block Sizes for DLMB Transactions

The overlay functionality provides the possibility to redirect data accesses with block sizes of 2 bytes or higher. The redirection is done by a modification of the target address as shown in [Figure 8-3](#). If the data size of a DLMB transaction in TC1796 is greater than the selected overlay memory block size, the following behavior will be observed:

- If the address of the DLMB transaction matches the overlay target address, the complete transaction is redirected, and the complete data (exceeding the block size) is taken from the overlay memory.
- If the address of the DLMB transaction does not match the overlay target address, the transaction is not redirected even if the overlay block resides within the transaction data block.

Therefore, it is recommended not to use DLMB transactions which have a greater data size as the programmed minimum overlay memory block size (OMASK bit field in Register OMASKx). This recommendation can, for example, be guaranteed by using a minimum overlay memory block size of 8 byte.

8.5.1 Internal Overlay

For internal overlay, the size of the blocks can be 2^n , with $n = 1$ to 9 (2 to 512 byte). Thus the maximum memory size needed for overlay memory (if all blocks are used with maximum size) is 8 Kbyte. Only the 64 Kbyte DMU SRAM (not the Stand-by SBSRAM) can be used as internal overlay memory.

During address translation, the segment number is set to C_H .

8.5.2 Emulation Memory Overlay

For Emulation Memory Overlay (only available in the emulation device TC1796ED), the size of the blocks can be $2^n \times 1$ Kbyte, with $n = 1$ to 5 (2 Kbyte to 32 Kbyte). Thus, the maximum memory size needed for overlay memory (if all blocks are used with maximum size) is 512 Kbyte.

8.5.3 Switching between Internal and Emulation Memory Overlay

When switching a region between internal and external overlay, it has to be ensured that the respective OVEN bit is cleared, all region parameters are set properly, and then the region is enabled again. Other wise, unintended access re-directions may occur.

8.5.4 Region Priority

If concurrent matches in more than one region occur, the region with the lowest order number will win and perform the address translation.

8.5.5 Access Performance

Write Accesses

Write accesses are performed with zero wait states.

If no overlay is enabled, the accesses to the DMU memories take one cycle (1 wait state). If a pipelined read follows immediately a write, the read will be delayed by an additional wait state.

If an overlay is enabled, the access to the DMU overlay memory block may be prolonged by one cycle.

The access to the Emulation Memory will not be more than 10 cycles. Details can be found in TC1796ED specific documents.

8.6 Program Local Memory Bus Interface (LMI)

The PLMB (LMI) interface is a unidirectional interface, allowing the DMI/DMU to do data reads and writes into the PLMB-based modules such as e.g. Flash or EBU space.

Accepted DLMB-to-PLMB bus write transactions are never error acknowledged by the LMI.

The data phase of a PLMB read transaction will be extended until the data is available. The LMI will reject any legal transaction that it cannot handle at that particular time (e.g. when the LMI is busy and a second LMI-related transaction request is requested). Other DMU memory accesses can be performed in parallel.

LMI transactions may receive error acknowledgments to any transaction. In every case the LMI responds to error acknowledgments by halting the ongoing transaction that received the error acknowledge, and dropping this transaction from the LMI PLMB bus master transaction queue. The PLMB bus is released completely (including LOCK) for a single cycle before further transactions are processed.

8.6.1 Data Read Buffer

The LMI contains a data read buffer which makes it possible to cache one line of 16 bytes (= 128 bits) of data read from specific memory areas on PMU side.

Read Transactions

When the first read access to cachable memory areas is done, the LMI initiates a 2-beat burst access on PLMB, requesting 128 bit of data. The start address for the burst is generated by setting the last 4 bits of the address to 0.

Cachable memory regions for data accesses are:

Segment 8: 8000 0000_H - 8FFF FFFF_H

In all other memory regions mapped via the LMI, normal non-burst accesses of the requested data size are initiated and the read data are not stored in the read buffer. The read data cache buffer content remains unmodified.

For subsequent accesses into cachable memory regions, it is checked, whether the data is completely contained in the buffer:

- If this is true, the data is extracted out of the buffer without initiating an access to the PLMB.
- If the data is not contained in the buffer, a new 2-beat burst access is performed to the PLMB as previously described, and the read data is stored in the buffer.

The requested data can only be extracted after all 128 bits have been read; thus a cached access will take 1 cycle longer than non-cached access. As the cachable memory regions are also visible in non-cachable areas (see address map), the user can map the data dependent on his application to optimize between additional latency and

subsequent fast accesses for data with high locality, and un-cached faster accesses for data with low locality.

Write Transactions

When a write transaction hits into the read buffer, the buffer content is invalidated. This feature also can be used to flush the buffer contents.

Read-Modify-Write Transactions

Read-modify-write transaction always will be forwarded to the PLMB. They will not effect the buffer content at all (no caching, no invalidation). This implies that read-modify-write operations should be done to non-cached areas only.

8.7 DMU Registers

Figure 8-4 shows all registers associated with the DMU.

DMU Registers Overview

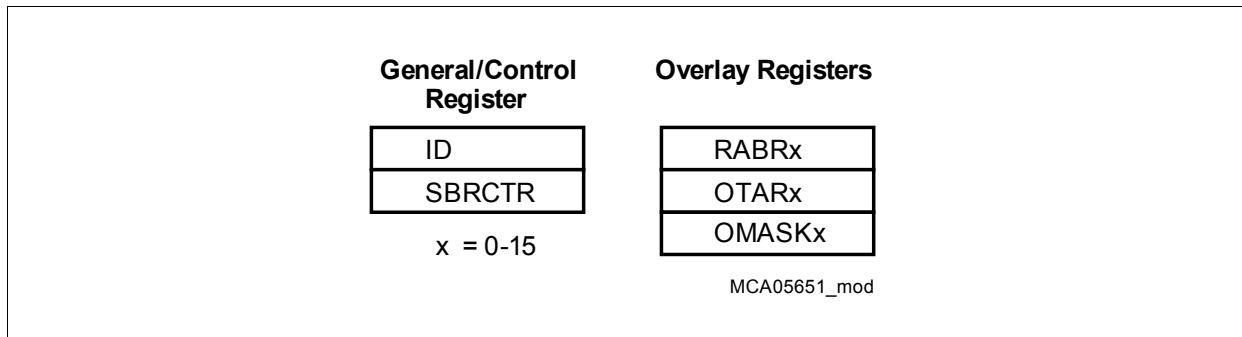


Figure 8-4 DMU Overlay Control Registers

The complete and detailed address map of the PMU module with its registers is shown in Table 18-37 on Page 18-118 of this TC1796 System Units (Vol. 1 of 2) User's Manual.

Table 8-1 Registers Address Space - DMU Registers

Module	Base Address	End Address	Note
DMU	F801 0100 _H	F801 01FF _H	—

Table 8-2 Registers Overview PMU Overlay Control Registers

Register ¹⁾ Short Name	Register Long Name	Offset Address	Description see
ID	Module Identification Register	0008 _H	Page 8-11
RABRx	Redirected Address Base Register x (x = 0-15)	0020 _H + x × C _H	Page 8-13
OTARx	Overlay Target Address Register x (x = 0-15)	0020 _H + x × C _H + 4	Page 8-15
OMASKx	Overlay Mask Register x (x = 0-15)	0020 _H + x × C _H + 8	Page 8-16
SBRCTR	Stand-by SRAM Control Register	00E0 _H	Page 8-12

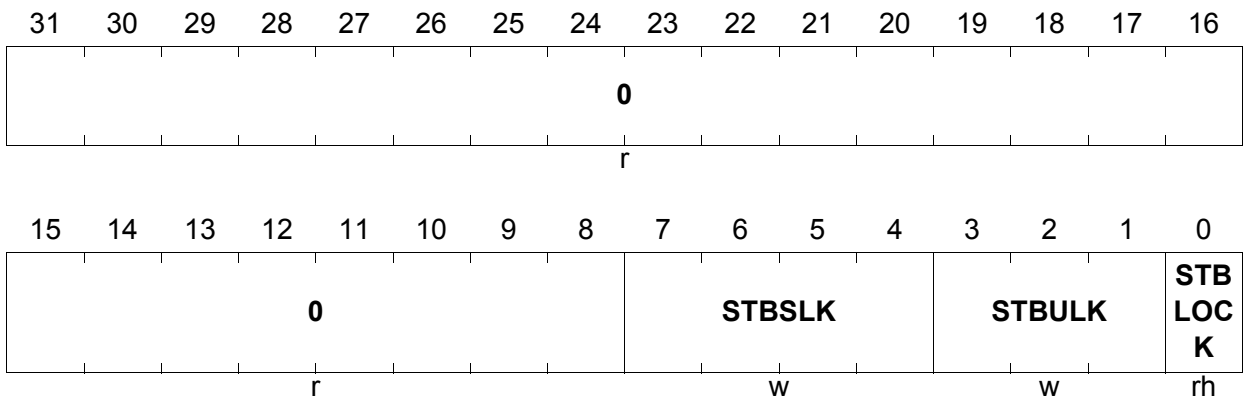
1) The PMU register short names are extended and referenced in the other parts of the TC1796 User's Manual with the module name prefix "PMU_".

Data Memory Unit

The Stand-by SRAM Control Register SBRCTR controls the locking and unlocking of the DMU stand-by memory (SBRAM).

SBRCTR

Stand-by SRAM Control Register (E0_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
STBLOCK	0	rh	<p>Stand-by Lock Flag Shows the current lock state of the SBRAM.</p> <p>0_B SBRAM is locked 1_B SBRAM is unlocked</p>
STBULK	[3:1]	w	<p>Unlock Stand-by Lock Flag In order to unlock the SBRAM, three consecutive write cycles must be written into STBULK with the following pattern:</p> <ol style="list-style-type: none"> 1. Write STBULK = 001_B 2. Write STBULK = 011_B 3. Write STBULK = 111_B <p>During the three consecutive write operations, STBSLK must be written with 0000_B. If any bit of STBSLK is set when writing a non-zero pattern to STBULK, this is treated as invalid pattern and the SBRAM will not be unlocked. Reading STBULK always return 000_B.</p>
STBSLK	[7:4]	w	<p>Set Stand-by Lock Flag In order to lock the SBRAM, the value 1001_B must be written into STBSLK. At the same time, the value 000_B must be written into STBULK. If any bit of STBULK is set when writing 1001_B to STBSLK, this is treated as invalid pattern and the SBRAM will not be locked. Reading STBSLK always returns 0000_B.</p>

Data Memory Unit

Field	Bits	Type	Description
IEMS = 0	30	r	<p>Internal/Emulation Memory Select</p> <p>This bit indicates the location of the overlay memory. In the TC1796, this bit is always read as 0, indicating that the DMU SRAM is used as overlay memory.</p> <p>0_B DMU SRAM is selected as overlay memory; Block sizes are 2ⁿ bytes, n = 1-9; Segment address bits [31:28] are set to 1100_B.</p> <p><i>Note: In an TC1796ED emulation device, this bit can be written, too. When set, an emulation memory is selected as overlay memory (not applicable in the TC1796).</i></p>
OVEN	31	rw	<p>Overlay Enabled</p> <p>This bit controls whether or not the overlay function of overlay block x is enabled.</p> <p>0_B Overlay function of block x is disabled. 1_B Overlay function of block x is enabled.</p>
0	0, [27:16]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

1) The block size is determined by the mask register OMASK.

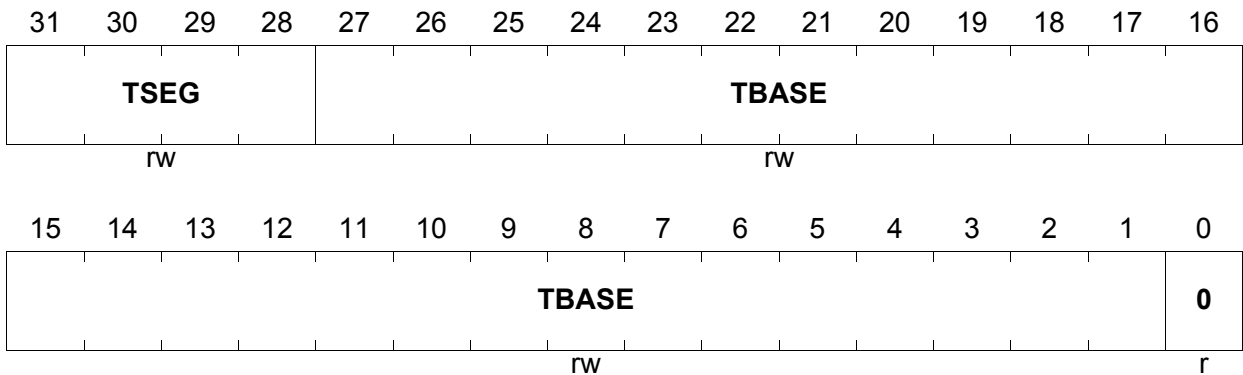
Data Memory Unit

OTARx (x = 0-15)

Overlay Target Address Register x

$$(20_H + x * C_H + 4_H)$$

Reset Value: 0000 0000_H



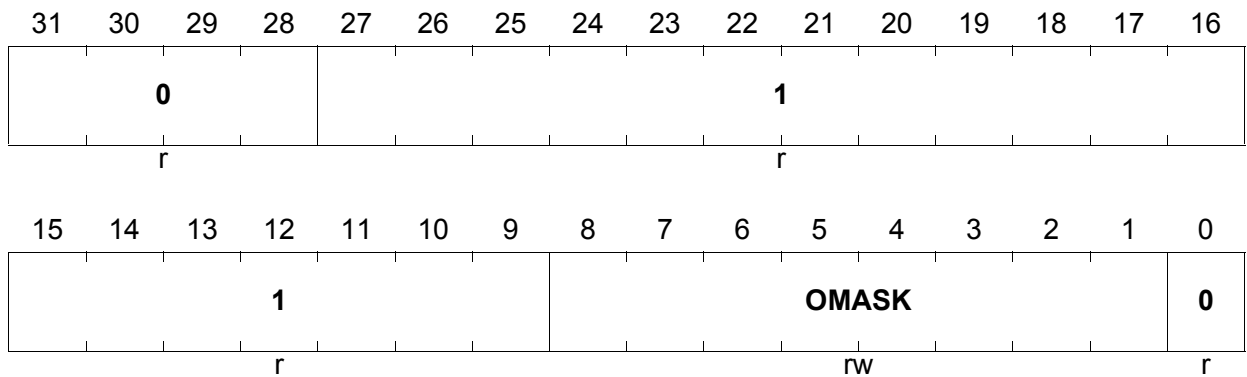
Field	Bits	Type	Description
TBASE	[27:1]	rw	Target Base Holds the base address of the overlay memory block in the target memory. If IEMS is set, bits [9:1] will be forced to 0 and cannot be modified.
TSEG	[31:28]	rw	Target Segment (reserved) This bit field is reserved for future use, to select a segment. In TC1796 implementation, any access to segments 8_H , or A_H will be checked for a valid base address; return 0 if read; should be written with 0.
0	0	r	Reserved Read as 0; should be written with 0.

Data Memory Unit

The Overlay Mask Register x determines the size of the overlay memory block x. It also determines which address bits will participate in the address compare for a block base address and which bits are used from the original target address and which bits are taken from RABRx.OBASE.

OMASKx (x = 0-15)

Overlay Mask Register x **(20_H+x*C_H+8_H)** **Reset Value: 0FFF FE00_H**



Field	Bits	Type	Description																		
OMASK	[8:1]	rw	<p>Overlay Address Mask</p> <p>This bit field determines two parameters:</p> <ul style="list-style-type: none"> • Block size of the overlay memory block x • Address bits taken for address generation <p>The mask bits determine which address bits are taken to check if a target address is overlaid, and determines which address bits are taken from the original target address as offset, and which are taken from RABRx.OBASE. Thus the mask also determines the block size of the overlay block.</p> <p>0_B Corresponding address bits are not used in the address comparison Corresponding final address bits are derived from the original address Block size is 2ⁿ bytes, n = number of trailing zeros in the register</p> <p>1_B Corresponding address bits are used in the address comparison Corresponding final address bits are derived from RABR_OBASE Block size is 2ⁿ bytes, n = number of trailing zeros in the register</p> <p>All OMASK bits located right of the most significant mask bit which is set to zero, are treated as zeros as well, independent from their actual value. They will be read back as 0.</p> <p>Selected overlay memory block size:</p> <table> <tr> <td>00000000_B</td> <td>512 bytes</td> </tr> <tr> <td>10000000_B</td> <td>256 bytes</td> </tr> <tr> <td>11000000_B</td> <td>128 bytes</td> </tr> <tr> <td>11100000_B</td> <td>64 bytes</td> </tr> <tr> <td>11110000_B</td> <td>32 bytes</td> </tr> <tr> <td>11111000_B</td> <td>16 bytes</td> </tr> <tr> <td>11111100_B</td> <td>8 bytes</td> </tr> <tr> <td>11111110_B</td> <td>4 bytes</td> </tr> <tr> <td>11111111_B</td> <td>2 bytes</td> </tr> </table>	00000000 _B	512 bytes	10000000 _B	256 bytes	11000000 _B	128 bytes	11100000 _B	64 bytes	11110000 _B	32 bytes	11111000 _B	16 bytes	11111100 _B	8 bytes	11111110 _B	4 bytes	11111111 _B	2 bytes
00000000 _B	512 bytes																				
10000000 _B	256 bytes																				
11000000 _B	128 bytes																				
11100000 _B	64 bytes																				
11110000 _B	32 bytes																				
11111000 _B	16 bytes																				
11111100 _B	8 bytes																				
11111110 _B	4 bytes																				
11111111 _B	2 bytes																				

Data Memory Unit

Field	Bits	Type	Description
0	0, [31:28]	r	Fixed Values Corresponding address bits are not used in the address comparison. Corresponding final address bits are taken from the original address.
1	[27:9]	r	Fixed Values Corresponding address bits are participating in the address comparison. Corresponding final address bits are taken from RABR.

9 Memory Maps

This chapter gives an overview of the TC1796 memory map, and describes the address locations and access possibilities for the units, memories, and reserved areas as “seen” from the three different on-chip buses’ points of view.

The TC1796 has the following memories:

- Data Memory Unit (DMU) with
 - 64 Kbyte of Data Memory (SRAM)
 - 16 Kbyte of Stand-by Data Memory (SBRAM)
- Program Memory Unit (PMU) with
 - 2 Mbyte of Program Flash Memory (PFLASH)
 - 128 Kbyte of Data Flash Memory (DFLASH)
 - 8 Kbyte of Boot ROM (BROM)
 - 8 Kbyte of Test ROM (TROM)
- Program Memory Interface (PMI)
 - 48 Kbyte of Scratch-Pad RAM (SPRAM)
 - 16 Kbyte of Instruction Cache (ICACHE)
- Data Memory Interface (DMI)
 - 56 Kbyte of Local Data RAM (LDRAM)
 - 8 Kbyte of Dual-Port RAM (DPRAM)
- PCP memory
 - 32 Kbyte of PCP Code Memory (CMEM)
 - 16 Kbyte of PCP Data Memory (PRAM)

Furthermore, the TC1796 has four on-chip buses:

- System Peripheral Bus (SPB)
- Remote Peripheral Bus (RPB)
- Program Local Memory Bus (PLMB)
- Data Local Memory Bus (DLMB)

9.1 How to Read the Address Maps

The bus-specific address maps describe how the different bus master devices react on accesses to on-chip memories and modules, and which address ranges are valid or invalid for the corresponding buses.

The FPI Bus address map shows the system addresses from the point of view of the SPB and RPB master agents. SPB master agents are PCP2, OCDS, and DMA (BI0). The RPB is a single master bus with DMA (BI1) as the only master agent. Note that SPB and RPB are combined in the single FPI Bus address map for simplification.

The PLMB address map shows the system addresses from the point of view of the PLMB master (PMI).

The DLMB address map shows the system addresses from the point of view of the DLMB master (DMI).

Table 9-1 defines the acronyms and other terms that are used in the address maps (**Table 9-2** to **Table 9-5**).

Table 9-1 Definition of Acronyms and Terms

Term	Description
...BE	Means “Bus error” generation
...BET	Means “Bus error & trap” generation
RPBBE	A bus access is terminated with a bus error on the RPB.
RPBBET	A bus access is terminated with a bus error on the RPB and a DSE trap (read access) or DAE trap (write access).
SPBBE	A bus access is terminated with a bus error on the SPB.
SPBBET	A bus access is terminated with a bus error on the SPB and a DSE trap (read access) or DAE trap (write access).
PLMBBE	A bus access is terminated with a bus error on the PLMB.
PLMBBET	A bus access is terminated with a bus error on the PLMB and a DSE trap (read access) or DAE trap (write access).
DLMBBE	A bus access is terminated with a bus error on the DLMB.
DLMBBET	A bus access is terminated with a bus error on the DLMB and a DSE trap (read access) or DAE trap (write access).
EBUBET	If the requested address is configured by the EBU registers and exist externally, the access is executed. In case of an address error, a DLMBBET or PLMBBET is generated for a read access. A PLMBBE is generated for a write access.
EBU access	If the requested address is configured by the EBU registers, the access is executed. In case of an un-configured address, an “ignore” is applicable for a write access. For read access, a SPBBE, DLMBBE, and PLMBBE are generated for PCP2, DMA, and OCDS. DLMBBE and PLMBBE are generated for DMI. PLMBBE is generated for PMI.
access	A bus access is allowed and is executed.
ignore	A bus access is ignored and is not executed. No bus error is generated.
trap	A DSE trap (read access) or DAE trap (write access) is generated.

Table 9-1 Definition of Acronyms and Terms (cont'd)

Term	Description
32	Only 32-bit word bus accesses are permitted to that register/address range.
nE	A bus access generates no bus error, although the bus access points to an undefined address or address range. This is valid e.g. for CPU accesses (MTCR/MFCR) to undefined addresses in the CSFR range.

9.2 Contents of the Segments

This section summarizes the contents of the segments.

Segments 0-7

These segments are reserved segments in the TC1796.

Segment 8

From the SPB point of view (PCP, DMA, and Cerberus), this memory segment allows accesses to all PMU memories (PFLASH, DFLASH, BROM, and TROM) and the external EBU space.

From the CPU point of view (PMI and DMI), this memory segment allows cached accesses to all PMU memories (PFLASH, DFLASH, BROM, and TROM) and the external EBU space.

Segment 9

This memory segment is reserved in the TC1796.

Segment 10

From the SPB point of view (PCP, DMA, and Cerberus), this memory segment allows accesses to all PMU memories (PFLASH, DFLASH, BROM, and TROM) and the external EBU space.

From the CPU point of view (PMI and DMI), this memory segment allows non-cached accesses to all PMU memories (PFLASH, DFLASH, BROM, and TROM) and the external EBU space.

Segment 11

This memory segment is reserved in the TC1796 (comparable to segment 9).

Segment 12

From the SPB point of view (PCP, DMA, and Cerberus), this memory segment is reserved in the TC1796.

From the PLMB point of view (PMI), this memory segment is reserved in the TC1796.

From the DLMB point of view (DMI), this memory segment allows cached accesses to all DMU memories (SRAM and SBRAM).

Segment 13

From the SPB point of view (PCP, DMA, and Cerberus), this memory segment allows accesses to the external peripheral and emulator space.

Memory Maps

From the PLMB point of view (PMI), this memory segment allows non-cached accesses to the external peripheral and emulator space, the PMI scratch-pad RAM and read access to the boot ROM and test ROM (BROM and TROM).

From the DLMB point of view (DMI), this memory segment allows non-cached accesses to the external peripheral and emulator space and to the DMI memories (LDRAM and DPRAM).

Segment 14

From the SPB point of view (PCP, DMA, and Cerberus), this memory segment allows accesses to the external peripheral space, the PMU data memory (SRAM), the DMI memories (LDRAM and DPRAM), and the PMI scratch-pad memory (SPRAM).

From the CPU point of view (PMI and DMI), this memory segment allows non-cached accesses to the external peripheral space.

Segment 15

From the SPB point of view (PCP, DMA, and Cerberus), this memory segment allows accesses to all SFRs and CSFRs, the PCP memories, and the MLI transfer windows.

From the CPU point of view (PMI and DMI), this memory segment allows accesses to all SFRs and CSFRs, the PCP memories, and the MLI transfer windows.

9.3 Address Map of the FPI Bus System

9.3.1 Segments 0 to 14

Table 9-2 shows the address map of segments 0 to 14 as it is seen from the SPB and RPB bus masters PCP, DMA, and OCDS.

Table 9-2 SPB/RPB Address Map of Segment 0 to 14

Segment	Address Range	Size	Description	Access Type		
				Read	Write	
0-7	0000 0000 _H - 0000 0007 _H	8 byte	Reserved (virtual address space)	MPN trap	MPN trap	
	0000 0008 _H - 7FFF FFFF _H	8 × 256 Mbyte		SPBBE	SPBBE	
8	8000 0000 _H - 801F FFFF _H	2 Mbyte	Program Flash (PFLASH)	access	access ¹⁾	
	8020 0000 _H - 807F FFFF _H	6 Mbyte	Reserved	PLMBBE & DLMBBE & SPBBE	PLMBBE	
	8080 0000 _H - 8FDF FFFF _H	246 Mbyte	External EBU space	EBU access	EBU access	
	8FE0 0000 _H - 8FE1 FFFF _H	128 Kbyte	Data Flash (DFLASH)	access	access ¹⁾	
	8FE2 0000 _H - 8FEF FFFF _H	896 Kbyte	Reserved	PLMBBE & DLMBBE & SPBBE	PLMBBE	
	8FF0 0000 _H - 8FF7 FFFF _H	512 Kbyte	Reserved for TC1796 emulation device memory			
	8FF8 0000 _H - 8FFF BFFF _H	496 Kbyte	Reserved			
		8FFF C000 _H - 8FFF DFFF _H	8 Kbyte	Boot ROM (BROM)	access	
		8FFF E000 _H - 8FFF FFFF _H	8 Kbyte	Test ROM (TROM)		
9	9000 0000 _H - 9FFF FFFF _H	256 Mbyte	Reserved	SPBBE	SPBBE	

Table 9-2 SPB/RPB Address Map of Segment 0 to 14 (cont'd)

Segment	Address Range	Size	Description	Access Type	
				Read	Write
10	A000 0000 _H - A01F FFFF _H	2 Mbyte	Program Flash (PFLASH)	access	access ¹⁾
	A020 0000 _H - A07F FFFF _H	6 Mbyte	Reserved	PLMBBE & DLMBBE & SPBBE	PLMBBE
	A080 0000 _H - AFDF FFFF _H	246 Mbyte	External EBU space	EBU access	EBU access
	AFE0 0000 _H - AFE1 FFFF _H	128 Kbyte	Data Flash (DFLASH)	access	access ¹⁾
	AFE2 0000 _H - AFEF FFFF _H	896 Kbyte	Reserved	PLMBBE & DLMBBE & SPBBE	PLMBBE
	AFF0 0000 _H - AFF7 FFFF _H	512 Kbyte	Reserved for TC1796 emulation device memory		
	AFF8 0000 _H - AFFF BFFF _H	496 Kbyte	Reserved		
	AFFF C000 _H - AFFF DFFF _H	8 Kbyte	Boot ROM (BROM)	access	
	AFFF E000 _H - AFFF FFFF _H	8 Kbyte	Test ROM (TROM)		
11	B000 0000 _H - BFFF FFFF _H	256 Mbyte	Reserved	SPBBE	SPBBE
12	C000 0000 _H - C000 FFFF _H	64 Kbyte	DMU data memory (SRAM)	SPBBE	SPBBE
	C001 0000 _H - C03F BFFF _H	3.92 Mbyte	Reserved	SPBBE	SPBBE
	C03F C000 _H - C03F FFFF _H	16 Kbyte	DMU stand-by data memory (SBRAM)	SPBBE	SPBBE
	C040 0000 _H - CFFF FFFF _H	251 Mbyte	Reserved		

Table 9-2 SPB/RPB Address Map of Segment 0 to 14 (cont'd)

Segment	Address Range	Size	Description	Access Type	
				Read	Write
13	D000 0000 _H - D000 DFFF _H	56 Kbyte	DMI Local Data RAM (LDRAM)	SPBBE	SPBBE
	D000 E000 _H - D000 FFFF _H	8 Kbyte	DMI Dual-Port RAM (DPRAM)		
	D001 0000 _H - D3FF FFFF _H	≈ 64 Mbyte	Reserved		
	D400 0000 _H - D400 BFFF _H	48 Kbyte	PMI Scratch-Pad RAM (SPRAM)		
	D400 C000 _H - D7FF FFFF _H	≈ 64 Mbyte	Reserved		
	D800 0000 _H - DDFF FFFF _H	96 Mbyte	External Peripheral Space	compare	compare
	DE00 0000 _H - DEFF FFFF _H	16 Mbyte	External Emulator Space		
	DF00 0000 _H - DFFF BFFF _H	≈ 16 Mbyte	Reserved	SPBBE	SPBBE
	DFFF C000 _H - DFFF DFFF _H	8 Kbyte	Boot ROM (BROM)		
	DFFF E000 _H - DFFF FFFF _H	8 Kbyte	Test ROM (TROM)		

Memory Maps

Table 9-2 SPB/RPB Address Map of Segment 0 to 14 (cont'd)

Segment	Address Range	Size	Description	Access Type	
				Read	Write
14	E000 0000 _H - E7FF FFFF _H	128 Mbyte	External Peripheral Space	EBU access	EBU access
	E800 0000 _H - E800 FFFF _H	64 Kbyte	PMU data memory (SRAM))	access	access
	E801 0000 _H - E83F FFFF _H	3.92 Mbyte	Reserved	DLMBBE & SPBBE	DLMBBE
	E840 0000 _H - E840 DFFF _H	56 Kbyte	DMI Local Data RAM (LDRAM)	access	access
	E840 E000 _H - E840 FFFF _H	8 Kbyte	DMI Dual-Port RAM (DPRAM)		
	E841 0000 _H - E84F FFFF _H	≈ 1 Mbyte	Reserved	DLMBBE & SPBBE	DLMBBE
	E850 0000 _H - E850 BFFF _H	48 Kbyte	PMI Scratch-Pad SRAM (SPRAM)	access	access
	E850 C000 _H - E87F FFFF _H	≈ 3 Mbyte	Reserved	DLMBBE & SPBBE	PLMBBE
	E880 C000 _H - EFFF FFFF _H	≈ 120 Mbyte	Reserved	SPBBE	PLMBBE
15	F000 0000 _H - FFFF FFFF _H	256 Mbyte	see Table 9-3		

1) Only applicable when writing Flash command sequences.

9.3.2 Segment 15

Table 9-3 shows the address map of segment 15 as seen from the SPB and RPB bus masters PCP, DMA, and OCDS. Please note that access in **Table 9-3** means only that an access to an address within the defined address range is not automatically incorrect or ignored. If an access is really addressing a correct address, it can be found in the detailed tables in **Chapter 18**.

Table 9-3 SPB/RPB Address Map of Segment 15

Unit	Unit	Unit	Access Type	
			Read	Write
System Control Unit (SCU) and Watchdog Timer (WDT)	F000 0000 _H - F000 00FF _H	256 byte	access	access
System Peripheral Bus Control Unit (SBCU)	F000 0100 _H - F000 01FF _H	256 byte	access	access
System Timer (STM)	F000 0200 _H - F000 02FF _H	256 byte	access	access
Reserved	F000 0300 _H - F000 03FF _H	–	SPBBE	SPBBE
On-Chip Debug Support (Cerberus)	F000 0400 _H - F000 04FF _H	256 byte	access	access
Reserved	F000 0500 _H - F000 07FF _H	–	SPBBE	SPBBE
MicroSecond Bus Controller 0 (MSC0)	F000 0800 _H - F000 08FF _H	256 byte	access	access
MicroSecond Bus Controller 1 (MSC1)	F000 0900 _H - F000 09FF _H	256 byte	access	access
Async./Sync. Serial Interface 0 (ASC0)	F000 0A00 _H - F000 0AFF _H	256 byte	access	access
Async./Sync. Serial Interface 1 (ASC1)	F000 0B00 _H - F000 0BFF _H	256 byte	access	access
Port 0	F000 0C00 _H - F000 0CFF _H	256 byte	access	access
Port 1	F000 0D00 _H - F000 0DFF _H	256 byte	access	access
Port 2	F000 0E00 _H - F000 0EFF _H	256 byte	access	access

Table 9-3 SPB/RPB Address Map of Segment 15 (cont'd)

Unit	Unit	Unit	Access Type	
			Read	Write
Port 3	F000 0F00 _H - F000 0FFF _H	256 byte	access	access
Port 4	F000 1000 _H - F000 10FF _H	256 byte	access	access
Port 5	F000 1100 _H - F000 11FF _H	256 byte	access	access
Port 6	F000 1200 _H - F000 12FF _H	256 byte	access	access
Port 7	F000 1300 _H - F000 13FF _H	256 byte	access	access
Port 8	F000 1400 _H - F000 14FF _H	256 byte	access	access
Port 9	F000 1500 _H - F000 15FF _H	256 byte	access	access
Port 10	F000 1600 _H - F000 16FF _H	256 byte	access	access
Reserved	F000 1700 _H - F000 17FF _H	–	SPBBE	SPBBE
General Purpose Timer Array 0 (GPTA0)	F000 1800 _H - F000 1FFF _H	8 × 256 byte	access	access
General Purpose Timer Array 1 (GPTA1)	F000 2000 _H - F000 27FF _H	8 × 256 byte	access	access
Local Timer Cell Array 2 (LTCA2)	F000 2800 _H - F000 2FFF _H	8 × 256 byte	access	access
Reserved	F000 3000 _H - F000 3BFF _H	–	SPBBE	SPBBE
Direct Memory Access Controller (DMA)	F000 3C00 _H - F000 3EFF _H	3 × 256 byte	access	access
Reserved	F000 3F00 _H - F000 3FFF _H	–	SPBBE	SPBBE
MultiCAN Controller (CAN)	F000 4000 _H - F000 5FFF _H	8 Kbyte	access	access

Table 9-3 SPB/RPB Address Map of Segment 15 (cont'd)

Unit		Unit	Unit	Access Type	
				Read	Write
Reserved		F000 6000 _H - F003 FFFF _H	–	SPBBE	SPBBE
PCP	Reserved	F004 0000 _H - F004 3EFF _H	–	SPBBE	SPBBE
	PCP Registers	F004 3F00 _H - F004 3FFF _H	256 byte	access	access
	Reserved	F004 4000 _H - F004 FFFF _H	–	SPBBE	SPBBE
	PCP Data Memory (PRAM)	F005 0000 _H - F005 3FFF _H	16 Kbyte	nE, 32	nE, 32
	Reserved	F005 4000 _H - F005 FFFF _H	–	SPBBE	SPBBE
	PCP Code Memory (CMEM)	F006 0000 _H - F006 7FFF _H	32 Kbyte	nE, 32	nE, 32
	Reserved	F006 8000 _H - F007 FFFF _H	–	SPBBE	SPBBE
Reserved		F008 0000 _H - F00F FFFF _H	–	SPBBE	SPBBE
Remote Peripheral Bus Control Unit (RBCU)		F010 0000 _H - F010 00FF _H	256 byte	access	access
Synchronous Serial Interface 0 (SSC0)		F010 0100 _H - F010 01FF _H	256 byte	access	access
Synchronous Serial Interface 1 (SSC1)		F010 0200 _H - F010 02FF _H	256 byte	access	access
Fast Analog-to-Digital Converter (FADC)		F010 0300 _H - F010 03FF _H	256 byte	access	access
Analog-to-Digital Converter 0 (ADC0)		F010 0400 _H - F010 05FF _H	2 × 256 byte	access	access
Analog-to-Digital Converter 1 (ADC1)		F010 0600 _H - F010 07FF _H	2 × 256 byte	access	access
Reserved		F010 0800 _H - F010 9FFF _H	–	RPBBE & SPBBE ¹⁾	RPBBE

Table 9-3 SPB/RPB Address Map of Segment 15 (cont'd)

Unit		Unit	Unit	Access Type	
				Read	Write
Dual-Port RAM		F010 A000 _H - F010 BFFF _H	8 Kbyte	access	access
Micro Link Interface 0 (MLI0)		F010 C000 _H - F010 C0FF _H	256 byte	access	access
Micro Link Interface 1 (MLI1)		F010 C100 _H - F010 C1FF _H	256 byte	access	access
Memory Checker (MCHK)		F010 C200 _H - F010 C2FF _H	256 byte	access	access
Reserved		F010 C300 _H - F01D FFFF _H	–	SPBBE	SPBBE
MLI0 Small Transfer Windows		F01E 0000 _H - F01E 7FFF _H	4 × 8 Kbyte	access	access
MLI1 Small Transfer Windows		F01E 8000 _H - F01E FFFF _H	4 × 8 Kbyte	access	access
Reserved		F01F 0000 _H - F01F FFFF _H	–	SPBBE	SPBBE
MLI0 Large Transfer Windows		F020 0000 _H - F023FFFF _H	4 × 64 Kbyte	access	access
MLI1 Large Transfer Windows		F024 0000 _H - F027 FFFF _H	4 × 64 Kbyte	access	access
Reserved		F028 0000 _H - F7E0 FEFF _H	–	SPBBE	SPBBE
CPU	CPU Slave Interface Registers (CPS)	F7E0 FF00 _H - F7E0 FFFF _H	256 byte	access	access
	CPU Core SFRs & GPRs	F7E1 0000 _H - F7E1 FFFF _H	64 Kbyte	access	access
Reserved		F7E2 0000 _H - F7FF FFFF _H	–	SPBBE	SPBBE
External Bus Interface Unit (EBU)		F800 0000 _H - F800 03FF _H	4 × 256 byte	access	access
Reserved		F800 0400 _H - F800 04FF _H	–	PLMBBE & DLMBBE & SPBBE	PLMBBE

Table 9-3 SPB/RPB Address Map of Segment 15 (cont'd)

Unit		Unit	Unit	Access Type	
				Read	Write
Program Memory Unit (PMU)		F800 0500 _H - F800 05FF _H	256 byte	access	access
Reserved		F800 0600 _H - F800 0FFF _H	–	PLMBBE & DLMBBE & SPBBE	PLMBBE
Flash Register		F800 1000 _H - F800 23FF _H	5 Kbyte	access	access
Reserved		F800 2400 _H - F801 00FF _H	–	DLMBBE & SPBBE	DLMBBE
Data Memory Unit (DMU)		F801 0100 _H - F801 01FF _H	256 byte	access	access
Reserved		F801 0200 _H - F87F F9FF _H	–	DLMBBE & SPBBE	DLMBBE
Data Local Memory Bus Control Unit (DBCUC)		F87F FA00 _H - F87F FAFF _H	256 byte	access	access
Reserved		F87F FB00 _H - F87F FBFF _H	–	DLMBBE & SPBBE	DLMBBE
CPU	DMI Registers	F87F FC00 _H - F87F FCFF _H	256 byte	access	access
	PMI Registers	F87F FD00 _H - F87F FDFF _H	256 byte	access	access
Program Local Memory Bus Control Unit (PBCUC)		F87F FE00 _H - F87F FEFF _H	256 byte	access	access
LFI Bridge		F87F FF00 _H - F87F FFFF _H	256 byte	access	access
Reserved		F880 0000 _H - FFFF FFFF _H	–	DLMBBE & SPBBE	DLMBBE

1) The SPBBE is not generated for accesses from the DMA.

9.4 Address Map of the Program Local Memory Bus (PLMB)

Table 9-4 shows the address map as seen from the PLMB bus masters (PMI and DMU).

Table 9-4 PLMB Address Map

Segment	Address Range	Size	Description	Action	
				Read	Write
0-7 ¹⁾	0000 0000 _H - 0000 0007 _H	8 byte	Reserved (virtual address space)	MPN trap	MPN trap
	0000 0008 _H - 7FFF FFFF _H	8 × 256 Mbyte		PLMBBET	PLMBBET
8 ¹⁾	8000 0000 _H - 801F FFFF _H	2 Mbyte	Program Flash (PFLASH)	access	access ²⁾
	8020 0000 _H - 807F FFFF _H	6 Mbyte	Reserved	PLMBBET	PLMBBET
	8080 0000 _H - 8FDF FFFF _H	246 Mbyte	External EBU Space	EBU access	EBU access
	8FE0 0000 _H - 8FE1 FFFF _H	128 Kbyte	Data Flash (DFLASH)	PLMBBET	access ²⁾
	8FE2 0000 _H - 8FEF FFFF _H	896 Kbyte	Reserved	PLMBBET	PLMBBET
	8FF0 0000 _H - 8FF7 FFFF _H	512 Kbyte	Reserved for TC1796 emulation device memory		
	8FF8 0000 _H - 8FFF BFFF _H	496 Kbyte	Reserved		
	8FFF C000 _H - 8FFF DFFF _H	8 Kbyte	Boot ROM (BROM)	access	
	8FFF E000 _H - 8FFF FFFF _H	8 Kbyte	Test ROM (TROM)		
9 ¹⁾	9000 0000 _H - 9FFF FFFF _H	256 Mbyte	Reserved	PLMBBET	PLMBBET

Table 9-4 PLMB Address Map (cont'd)

Segment	Address Range	Size	Description	Action	
				Read	Write
10 ³⁾	A000 0000 _H - A01F FFFF _H	2 Mbyte	Program Flash (PFLASH)	access	access ²⁾
	A020 0000 _H - A07F FFFF _H	6 Mbyte	Reserved	PLMBBET	PLMBBET
	A080 0000 _H - AFDF FFFF _H	246 Mbyte	External EBU Space	EBUBET	EBUBET
	AFE0 0000 _H - AFE1 FFFF _H	128 Kbyte	Data Flash (DFLASH)	PLMBBET	access ²⁾
	AFE2 0000 _H - AFEF FFFF _H	896 Kbyte	Reserved	PLMBBET	PLMBBET
	AFF0 0000 _H - AFF7 FFFF _H	512 Kbyte	Reserved for TC1796 emulation device memory	access	access
	AFF8 0000 _H - AFFF BFFF _H	496 Kbyte	Reserved	PLMBBET	PLMBBET
	AFFF C000 _H - AFFF DFFF _H	8 Kbyte	Boot ROM (BROM)	access	
	AFFF E000 _H - AFFF FFFF _H	8 Kbyte	Test ROM (TROM)		
11 ³⁾	B000 0000 _H - BFFF FFFF _H	256 Mbyte	Reserved	PLMBBET	PLMBBET
12 ¹⁾	C000 0000 _H - CFFF FFFF _H	256 Mbyte	Reserved	PLMBBET	PLMBBET

Table 9-4 PLMB Address Map (cont'd)

Segment	Address Range	Size	Description	Action	
				Read	Write
13 ³⁾	D000 0000 _H - D000 DFFF _H	56 Kbyte	Reserved (DMI Local Data RAM)	PLMBBET	PLMBBET
	D000 E000 _H - D000 FFFF _H	8 Kbyte	Reserved (DMI Dual-Port RAM))		
	D001 0000 _H - D3FF FFFF _H	≈ 64 Mbyte	Reserved		
	D400 0000 _H - D400 BFFF _H	48 Kbyte	Reserved (PMI Scratch-Pad RAM)	ignore	ignore
	D400 C000 _H - D7FF FFFF _H	≈ 64 Mbyte	Reserved	PLMBBET	PLMBBET
	D800 0000 _H - DDFF FFFF _H	96 Mbyte	External Peripheral Space	EBU access	EBU access
	DE00 0000 _H - DEFF FFFF _H	16 Mbyte	External Emulator Space		
	DF00 0000 _H - DFFF BFFF _H	≈ 16 Mbyte	Reserved	PLMBBET	PLMBBET
	DFFF C000 _H - DFFF DFFF _H	8 Kbyte	Boot ROM (BROM)	access	PLMBBET
	DFFF E000 _H - DFFF FFFF _H	8 Kbyte	Test ROM (TROM)		
14 ³⁾	E000 0000 _H - E7FF FFFF _H	128 Mbyte	External Peripheral Space	EBU access	EBU access
	E800 0000 _H - EFFF FFFF _H	128 Mbyte	Reserved	PLMBBET	PLMBBET

Table 9-4 PLMB Address Map (cont'd)

Segment	Address Range	Size	Description	Action	
				Read	Write
15	F000 0000 _H - F7FF FFFF _H	128 Mbyte	Reserved	PLMBBET	PLMBBET
	F800 0000 _H - F800 03FF _H	1 Kbyte	External Bus Interface Unit (EBU)	access	access
	F800 0400 _H - F800 04FF _H	256 byte	Reserved	PLMBBET	PLMBBET
	F800 0500 _H - F800 05FF _H	256 byte	Program Memory Unit (PMU)	access	access
	F800 0600 _H - F800 0FFF _H	≈ 2 Kbyte	Reserved	PLMBBET	PLMBBET
	F800 1000 _H - F800 23FF _H	5 Kbyte	Reserved (Flash Registers)	ignore	ignore
	F800 2400 _H - F87F FCFF _H	≈ 8 Mbyte	Reserved	PLMBBET	PLMBBET
	F87F FD00 _H - F87F FFFF _H	768 byte	Reserved (PMI, PBCU and LFI)	ignore	ignore
	F880 0000 _H - FFFF FFFF _H	≈ 119 Mbyte	Reserved	PLMBBET	PLMBBET

- 1) Cached area
- 2) Only applicable when writing Flash command sequences
- 3) Non-cached area

9.5 Address Map of the Data Local Memory Bus (DLMB)

Table 9-4 shows the address map of as it is seen from the DLMB bus masters (DMI and LFI Bridge).

Table 9-5 DLMB Address Map

Segment	Addresses	Size	Description	Action	
				Read	Write
0-7 ¹⁾	0000 0000 _H - 0000 0007 _H	8 byte	Reserved (virtual address space)	MPN Trap	MPN Trap
	0000 0008 _H - 7FFF FFFF _H	8 × 256 Mbyte		DLMBBET & SPBBET	SPBBE
8 ¹⁾	8000 0000 _H - 801F FFFF _H	2 Mbyte	Program Flash (PFLASH)	access	access ²⁾
	8020 0000 _H - 807F FFFF _H	6 Mbyte	Reserved	PLMBBET & DLMBBET	PLMBBE
	8080 0000 _H - 8FDF FFFF _H	246 Mbyte	External EBU space	EBU access	EBU access
	8FE0 0000 _H - 8FE1 FFFF _H	128 Kbyte	Data Flash (DFLASH)	access	access ²⁾
	8FE2 0000 _H - 8FEF FFFF _H	896 Kbyte	Reserved	PLMBBET & DLMBBET	PLMBBE
	8FF0 0000 _H - 8FF7 FFFF _H	512 Kbyte	Reserved for TC1796 emulation device memory		
	8FF8 0000 _H - 8FFF BFFF _H	496 Kbyte	Reserved		
	8FFF C000 _H - 8FFF DFFF _H	8 Kbyte	Boot ROM (BROM)	access	
	8FFF E000 _H - 8FFF FFFF _H	8 Kbyte	Test ROM (TROM)		
9 ¹⁾	9000 0000 _H - 9FFF FFFF _H	256 Mbyte	Reserved	DLMBBET & SPBBET	SPBBE

Table 9-5 DLMB Address Map (cont'd)

Segment	Addresses	Size	Description	Action	
				Read	Write
10 ³⁾	A000 0000 _H - A01F FFFF _H	2 Mbyte	Program Flash (PFLASH)	access	access ²⁾
	A020 0000 _H - A07F FFFF _H	6 Mbyte	Reserved	PLMBBET & DLMBBET	PLMBBE
	A080 0000 _H - AFDF FFFF _H	246 Mbyte	External EBU space	compare	compare
	AFE0 0000 _H - AFE1 FFFF _H	128 Kbyte	Data Flash (DFLASH)	access	access ²⁾
	AFE2 0000 _H - AFEF FFFF _H	896 Kbyte	Reserved	PLMBBET & DLMBBET	PLMBBE
	AFF0 0000 _H - AFF7 FFFF _H	512 Kbyte	Reserved for TC1796 emulation device memory		
	AFF8 0000 _H - AFFF BFFF _H	496 Kbyte	Reserved		
	AFFF C000 _H - AFFF DFFF _H	8 Kbyte	Boot ROM (BROM)	access	
	AFFF E000 _H - AFFF FFFF _H	8 Kbyte	Test ROM (TROM)		
11 ³⁾	B000 0000 _H - BFFF FFFF _H	256 Mbyte	Reserved	DLMBBET & SPBBET	SPBBE
12 ¹⁾	C000 0000 _H - C000 FFFF _H	64 Kbyte	DMU data memory (SRAM)	access	access
	C001 0000 _H - C03F BFFF _H	3.92 Mbyte	Reserved	DLMBBET	DLMBBET
	C03F C000 _H - C03F FFFF _H	16 Kbyte	DMU Stand-by data memory (SBRAM)	access	access
	C040 0000 _H - CFFF FFFF _H	251 Mbyte	Reserved	DLMBBET	DLMBBET

Table 9-5 DLMB Address Map (cont'd)

Segment	Addresses	Size	Description	Action	
				Read	Write
13 ³⁾	D000 0000 _H - D000 DFFF _H	56 Kbyte	DMI Local Data RAM (LDRAM)	ignore	ignore
	D000 E000 _H - D000 FFFF _H	8 Kbyte	DMI Dual-Port RAM (DPRAM)		
	D001 0000 _H - D3FF FFFF _H	≈ 64 Mbyte	Reserved	Trap	Trap
	D400 0000 _H - D400 BFFF _H	48 Kbyte	Reserved (PMI Scratch-Pad RAM)	ignore	ignore ⁴⁾
	D400 C000 _H - D7FF FFFF _H	≈ 64 Mbyte	Reserved	PLMBBET & DLMBBET	PLMBBE
	D800 0000 _H - DDFF FFFF _H	96 Mbyte	External Peripheral Space	EBU access	EBU access
	DE00 0000 _H - DEFF FFFF _H	16 Mbyte	External Emulator Space	EBU access	EBU access
	DF00 0000 _H - DFFF BFFF _H	≈ 16 Mbyte	Reserved	DLMBBET & SPEBET	SPEBE
	DFFF C000 _H - DFFF DFFF _H	8 Kbyte	Boot ROM (BROM)		
	DFFF E000 _H - DFFF FFFF _H	8 Kbyte	Test ROM (TROM)		
14 ³⁾	E000 0000 _H - E7FF FFFF _H	128 Mbyte	External Peripheral Space	EBU access	EBU access
	E800 0000 _H - EFFF FFFF _H	128 Mbyte	Reserved	DLMBBE	DLMBBE
15 ³⁾	F000 0000 _H - FFFF FFFF _H	256 Mbyte	Address map is identical to FPI Bus segment 15 address map (see Table 9-3).		

- 1) Cached area
- 2) Only applicable when writing Flash command sequences
- 3) Non-cached area
- 4) In case of a byte access, a PLMBBET is generated

9.6 Memory Module Access Restrictions

Table 9-6 describes which type of accesses are possible to the different memories in the TC1796.

Table 9-6 Possible Memory Accesses

Memory		Bit	Byte		Half-word		Word		Double-word	
		rmw	r	w	r	w	r	w	r	w
PMI	SPRAM	✓	✓	–	✓	✓	✓	✓	✓	✓
DMI	LDRAM	✓	✓	✓	✓	✓	✓	✓	✓	✓
	DPRAM	✓	✓	✓	✓	✓	✓	✓	✓	✓
PMU	ROM	–	✓	–	✓	–	✓	–	✓	–
	PFLASH	–	✓	–	✓	–	✓	✓	✓	✓
	DFLASH	–	✓	–	✓	–	✓	✓	✓	✓
DMU	SRAM	✓	✓	✓	✓	✓	✓	✓	✓	✓
	SBRAM	✓	✓	✓	✓	✓	✓	✓	✓	✓
PCP	CMEM	–	–	–	–	–	✓	✓	–	–
	PRAM	–	–	–	–	–	✓	✓	–	–

General Purpose I/O Ports and Peripheral I/O Lines

10 General Purpose I/O Ports and Peripheral I/O Lines

The TC1796 has 127 digital General Purpose Input/Output (GPIO) port lines which are connected to the on-chip peripheral units. They are divided into:

- Four 16-bit GPIO ports (Ports 0, 1, 3, and 4)
- One 14-bit GPIO port (Port 2)
- One 12-bit GPIO port (Port 6)
- One 9-bit GPIO port (Port 9)
- Three 8-bit GPIO ports (Port 5, 7, and 8)
- One 4-bit input only port (Port 10)

Additionally, fourteen dedicated input/output lines without GPIO functionality are provided. The External Bus Interface (EBU) has its own set of dedicated signal lines.

Figure 10-1 is an overview of the I/O lines with its general port-to-peripheral unit assignments. Further details are described in the port-specific sections of this chapter.

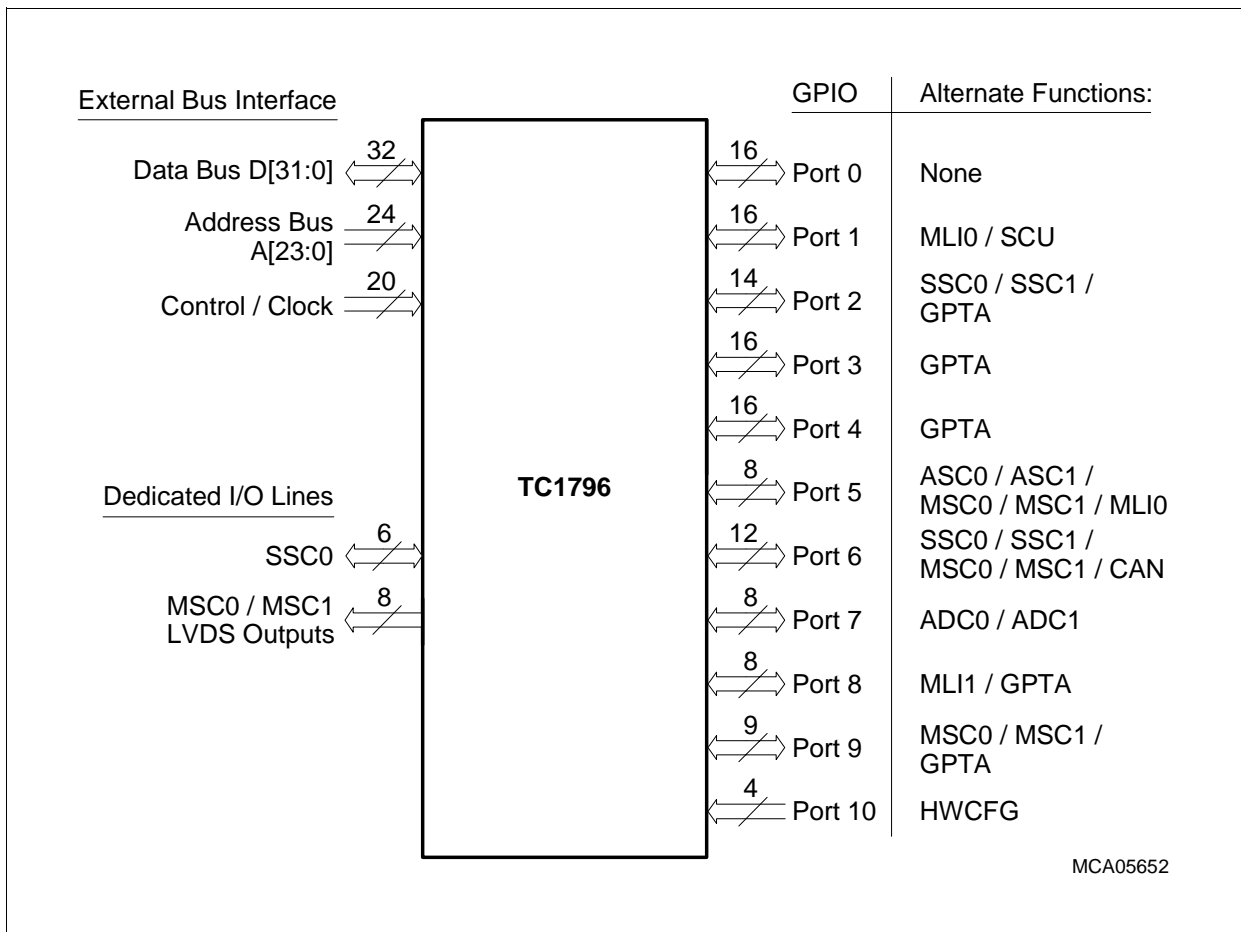


Figure 10-1 Peripheral/GPIO Lines of the TC1796

10.1 Basic Port Operation

Figure 10-2 is a general block diagram of a TC1796 GPIO port line.

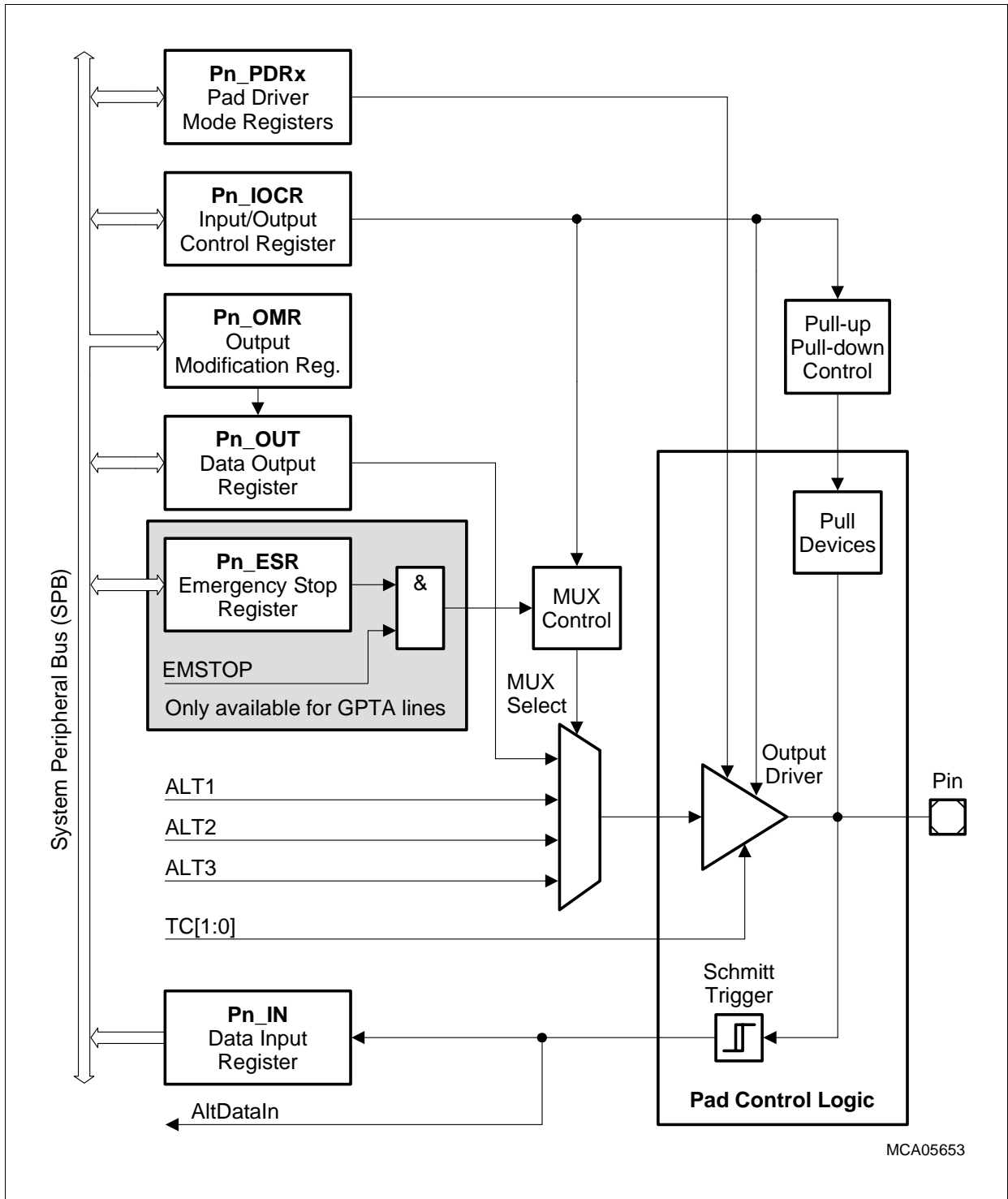


Figure 10-2 General Structure of a Port Pin

General Purpose I/O Ports and Peripheral I/O Lines

Each port line has a number of control and data bits, enabling very flexible usage of the line. Each port pin (except Port 10) can be configured for input or output operation. In input mode (default after reset), the output driver is switched off (high-impedance). The actual voltage level present at the port pin is translated into a logical 0 or 1 via a Schmitt-Trigger device and can be read via the read-only register Pn_IN. An input signal can also be connected directly to the various inputs of the peripheral units (AltDataIn). The function of the input line from the pin to the input register Pn_IN and to AltDataIn is independent of whether the port pin operates as input or output. This means that when the port is in output mode, the level of the pin can be read by software via Pn_IN or a peripheral can use the pin level as an input.

In output mode, the output driver is activated and drives the value supplied through the multiplexer to the port pin. Switching between input and output mode is accomplished through the Pn_IOCR register, which enables or disables the output driver. If a peripheral unit uses a GPIO port line as a bi-directional I/O line, register Pn_IOCR has to be written for input or output selection. The Pn_IOCR register further controls the driver type of the output driver, and determines whether an internal weak pull-up or pull-down device is alternatively connected to the pin when used as an input. This offers additional advantages in an application.

The output multiplexer in front of the output driver selects the signal source for the GPIO line when used as output. If the pin is used as general-purpose output, the multiplexer is switched by software (Pn_IOCR register) to the Output Data Register Pn_OUT. Software can set or clear the bit in Pn_OUT, and therefore it can directly influence the state of the port pin. If the on-chip peripheral units use the pin for output signals, the alternate output lines ALT1 to ALT3 can be switched via the multiplexer to the output driver. The data written into the output register Pn_OUT by software can be used as input data to an on-chip peripheral. This enables, for example, peripheral tests via software without external circuitry.

When selected as general-purpose output line, the logic state of each port pin can be changed individually by programming the pin-related bits in the Output Modification Register Pn_OMR. The bits in Pn_OMR make it possible to set, clear, toggle, or leave the bits in the Pn_OUT register unchanged.

When selected as general-purpose output line, the actual logic level at the pin can be examined through reading latch Pn_IN and compared against the applied output level (either applied through software via the output register Pn_OUT, or via an alternate output function of a peripheral unit). This can be used to detect some electrical failures at the pin caused through external circuitry. In addition, software-supported arbitration schemes can be implemented in this way using the open-drain configuration and an external wired-And circuitry. Collisions on the external communication lines can be detected when a high level (1) is output, but a low level (0) is seen when reading the pin value via the input register Pn_IN.

General Purpose I/O Ports and Peripheral I/O Lines

The temperature compensation feature for pad output drivers (two inputs lines TC[1:0]) makes it possible to control the output driver characteristics automatically within operating temperature range. The temperature compensation can be controlled for two separate groups: for all high speed (class A2) outputs at GPIO ports and all EBU output lines.

All GPIO lines of the TC1796 that are used by the GPTA modules (GPTA0, GPTA1, LTCA2) have an emergency stop logic. This logic makes it possible to individually disconnect GPTA outputs from the driving GPTA module outputs and to put them onto a well defined logic state in an emergency case. In an emergency case, the content of the port output register Pn_OUT is driven at the output pin instead of the GPTA module output. The Emergency Stop Register Pn_EMR determines whether a GPTA output is enabled or disabled in an emergency case.

General Purpose I/O Ports and Peripheral I/O Lines

10.2 Port Register Description

The individual control and data bits of each GPIO port are implemented in a number of registers. Bits with the same meaning and function are assembled together in the same register. The registers are used to configure and use the port as general-purpose I/O or alternate function input/output. For some ports, not all registers are implemented. The availability of the registers in the specific ports is described in [Table 10-6](#) on [Page 10-19](#) up to [Table 10-25](#) on [Page 10-76](#).

Port Register Overview

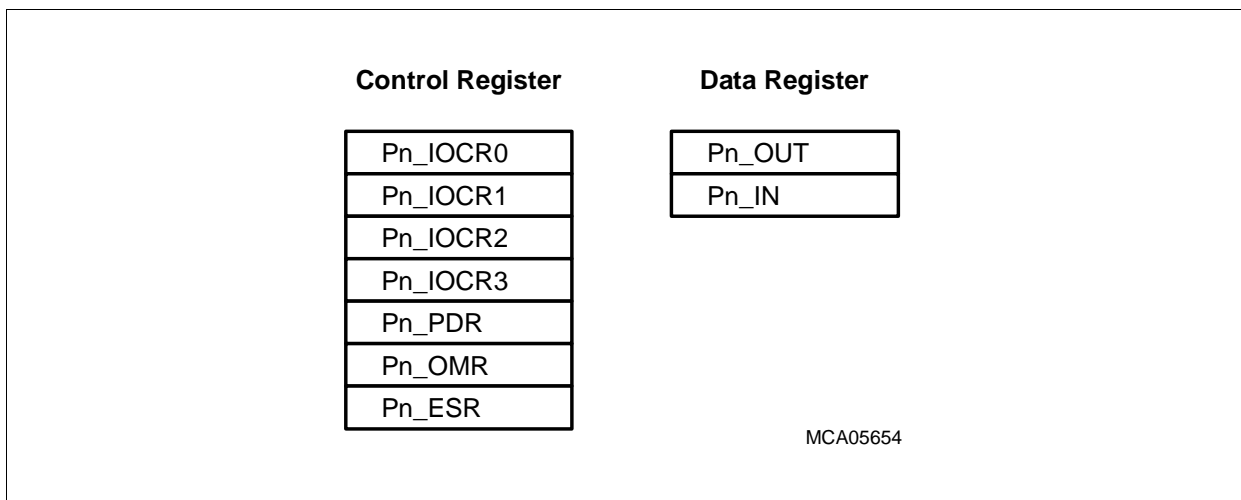


Figure 10-3 Port Registers

Note: The complete address map of the GPIO ports is described in the [Chapter 18](#) “Register Overview” of this TC1796 System Units (Vol. 1 of 2) User’s Manual.

Table 10-1 Registers Address Space

Module	Base Address	End Address	Note
P0	F000 0C00 _H	F000 0CFF _H	–
P1	F000 0D00 _H	F000 0DFF _H	–
P2	F000 0E00 _H	F000 0EFF _H	–
P3	F000 0F00 _H	F000 0FFF _H	–
P4	F000 1000 _H	F000 10FF _H	–
P5	F000 1100 _H	F000 11FF _H	–
P6	F000 1200 _H	F000 12FF _H	–
P7	F000 1300 _H	F000 13FF _H	–
P8	F000 1400 _H	F000 14FF _H	–

General Purpose I/O Ports and Peripheral I/O Lines

Table 10-1 Registers Address Space

Module	Base Address	End Address	Note
P9	F000 1500 _H	F000 15FF _H	–
P10	F000 1600 _H	F000 16FF _H	–

Table 10-2 Registers Overview

Register Short Name	Register Long Name	Offset Address	Description see
Pn_OUT	Port n Output Register	00 _H	Page 10-13
Pn_OMR	Port n Output Modification Register	04 _H	Page 10-14
Pn_IOCR0	Port n Input/Output Control Register 0	10 _H	Page 10-7
Pn_IOCR4	Port n Input/Output Control Register 4	14 _H	Page 10-8
Pn_IOCR8	Port n Input/Output Control Register 8	18 _H	Page 10-8
Pn_IOCR12	Port n Input/Output Control Register 12	1C _H	Page 10-9
Pn_IN	Port n Input Register	24 _H	Page 10-17
Pn_PDR	Port n Pad Driver Mode Register	40 _H	Page 10-10
Pn_ESR	Port n Emergency Stop Register	50 _H	Page 10-16

General Purpose I/O Ports and Peripheral I/O Lines

10.2.1 Port Input/Output Control Registers

The port input/output control registers select the digital output and input driver functionality and characteristics of a GPIO port pin. Port direction (input or output), pull-up or pull-down devices for inputs, and push-pull or open-drain functionality for outputs can be selected by the corresponding bit fields PCx (x = 0-15). Each 32-bit wide port input/output control register controls four GPIO port lines:

- Register Pn_IOCR0 controls the Pn.[3:0] port lines
- Register Pn_IOCR4 controls the Pn.[7:4] port lines
- Register Pn_IOCR8 controls the Pn.[11:8] port lines
- Register Pn_IOCR12 controls the Pn.[15:12] port lines

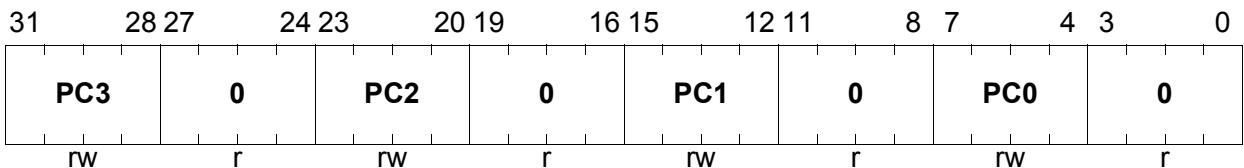
The diagrams below show the register layouts of the port input/output control registers with the PCx bit fields. One PCx bit field controls exactly one port line Pn.x.

Pn_IOCR0

Port n Input/Output Control Register 0

(10_H)

Reset Value: 2020 2020_H



Field	Bits	Type	Description
PC0, PC1, PC2, PC3	[7:4], [15:12], [23:20], [31:28]	rw	Port Control for Port n Pin 0 to 3 This bit field determines the Port n line x functionality (x = 0-3) according to the coding table (see Table 10-3).
0	[3:0], [11:8], [19:16], [27:24]	r	Reserved Read as 0; should be written with 0.

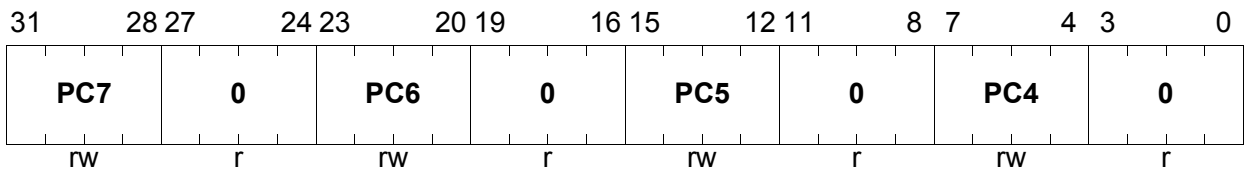
General Purpose I/O Ports and Peripheral I/O Lines

Pn_IOCRA

Port n Input/Output Control Register 4

(14_H)

Reset Value: 2020 2020_H



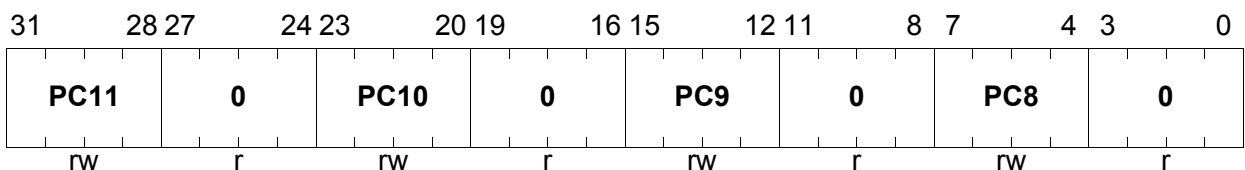
Field	Bits	Type	Description
PC4, PC5, PC6, PC7	[7:4], [15:12], [23:20], [31:28]	rw	Port Control for Port n Pin 4 to 7 This bit field determines the Port n line x functionality (x = 4-7) according to the coding table (see Table 10-3).
0	[3:0], [11:8], [19:16], [27:24]	r	Reserved Read as 0; should be written with 0.

Pn_IOCRA8

Port n Input/Output Control Register 8

(18_H)

Reset Value: 2020 2020_H



Field	Bits	Type	Description
PC8, PC9, PC10, PC11	[7:4], [15:12], [23:20], [31:28]	rw	Port Control for Port n Pin 8 to 11 This bit field determines the Port n line x functionality (x = 8-11) according to the coding table (see Table 10-3).
0	[3:0], [11:8], [19:16], [27:24]	r	Reserved Read as 0; should be written with 0.

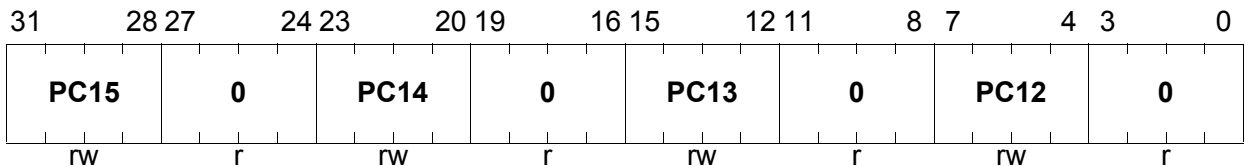
General Purpose I/O Ports and Peripheral I/O Lines

Pn_IOCRR12

Port n Input/Output Control Register 12

(1C_H)

Reset Value: 2020 2020_H



Field	Bits	Type	Description
PC12, PC13, PC14, PC15	[7:4], [15:12], [23:20], [31:28]	rw	Port Control for Port n Pin 12 to 15 This bit field determines the Port n line x functionality (x = 12-15) according to the coding table (see Table 10-3).
0	[3:0], [11:8], [19:16], [27:24]	r	Reserved Read as 0; should be written with 0.

Depending on the GPIO port functionality (number of GPIO lines of a port), not all of the port input/output control registers are implemented.

The structure with one control bit field for each port pin located in different register bytes offers the possibility to configure the port pin functionality of a single pin with byte-oriented accesses without accessing the other PCx bit fields.

General Purpose I/O Ports and Peripheral I/O Lines

Port Control Coding

Table 10-3 describes the coding of the PCx bit fields that determine the port line functionality.

Table 10-3 PCx Coding (except Port 4)

PCx[3:0]	I/O	Output Characteristics	Selected Pull-up / Pull-down / Selected Output Function
0X00 _B	Input	–	No input pull device connected
0X01 _B			Input pull-down device connected
0X10 _B ¹⁾			Input pull-up device connected
0X11 _B			No input pull device connected
1000 _B	Output	Push-pull	General-purpose output
1001 _B			Alternate output function 1
1010 _B			Alternate output function 2
1011 _B			Alternate output function 3
1100 _B		Open-drain	General-purpose output
1101 _B			Alternate output function 1
1110 _B			Alternate output function 2
1111 _B			Alternate output function 3

1) This bit field value is default after reset.

10.2.2 Pad Driver Mode Register

Overview

The pad structure of the TC1796 GPIO lines offers the possibility to select the output driver strength and the slew rate. These two parameters are controlled by the bit fields in the pad driver mode register Pn_PDR, independently from input/output and pull-up/pull-down control functionality as programmed in the Pn_IOCR register. One Pn_PDR register is assigned to each port.

The GPIO port lines consist of by two classes of pads:

- Class A1 pins (low speed 3.3V LVTTTL outputs)
- Class A2 pins (high speed 3.3V LVTTTL outputs. e.g. for serial outputs)

The assignment of each port pin to one of these pad classes is shown in the port configuration figures (**Figure 10-4** to **Figure 10-13**). Further details about pad driver classes that are available in the TC1796 are summarized in **Table 1-5** on **Page 1-57**.

General Purpose I/O Ports and Peripheral I/O Lines

Depending on the assigned pad class, the 3-bit wide pad driver mode selection bit fields PDx in the pad driver mode registers Pn_PDR make it possible to select the port line functionality as shown in [Table 10-4](#). Note that the pad driver mode registers are specific for each port. Therefore, the Pn_PDR layout is described for each port in the port-specific sections.

Class A1 pins make it possible to select between medium and weak output drivers. Class A2 pins make it possible to select between strong/medium/weak output drivers. In the case of strong driver type, the signal transition edge can be additionally selected as sharp/medium/soft.

Table 10-4 Pad Driver Mode Selection

Pad Class	PDx.2	PDx.1	PDx.0	Functionality
A1	X	X	0	Medium driver selected
			1	Weak driver selected
A2	0	0	0	Strong driver, sharp edge selected ¹⁾
	0	0	1	Strong driver, medium edge selected ¹⁾
	0	1	0	Strong driver, soft edge selected ¹⁾
	0	1	1	Weak driver selected
	1	0	0	Medium driver selected
	1	0	1	
	1	1	0	
	1	1	1	Weak driver selected

1) In strong driver mode, the output driver characteristics of class A2 pads can be additionally controlled by a temperature compensation logic (see [Page 5-41](#)).

Note: Please refer to the TC1796 Data Sheet for detailed DC characteristics of class A1 and class A2 pads.

In addition to the pad driver mode register control selections, a temperature compensation logic, which is a part of the system control unit (SCU), makes it possible to adjust the edges of the high-speed class A2 GPIO output lines depending on the die temperature. The temperature compensation logic affects all class A2 GPIO output lines that are used simultaneously as output.

The fully programmable temperature compensation logic provides four types of output driver levels:

- Maximum level (low temperature)
- High level
- Low level
- Minimum level (high temperature)

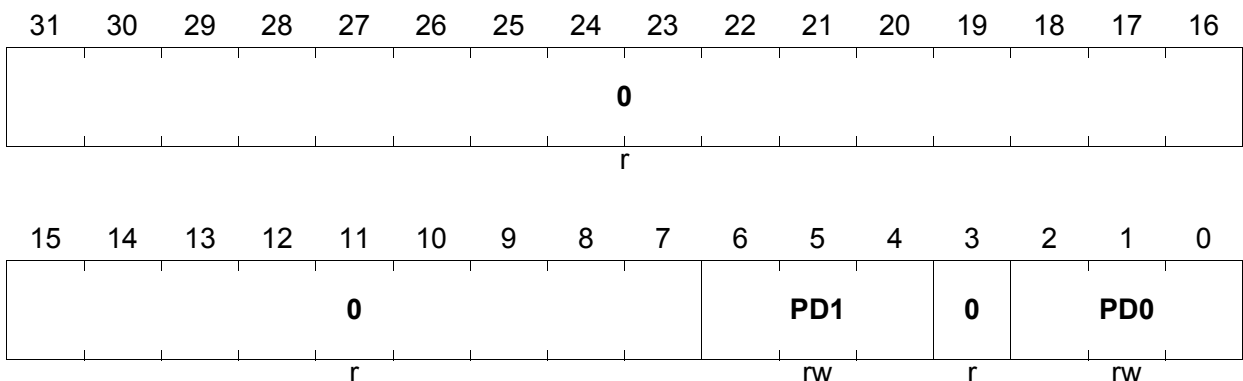
General Purpose I/O Ports and Peripheral I/O Lines

Further details of the temperature compensation logic are described in [Section 5.6 on Page 5-41](#) (SCU) of this User’s Manual.

Pad Driver Mode Register Example: Port 0 Pad Driver Mode Register

P0_PDR

Port 0 Pad Driver Mode Register (40_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
PD0	[2:0]	rw	Pad Driver Mode for P0.[7:0] (Class A1 pads; coding see Page 10-11)
PD1	[6:4]	rw	Pad Driver Mode for P0.[15:8] (Class A1 pads; coding see Page 10-11)
0	3, [31:7]	r	Reserved Read as 0; should be written with 0.

In addition to the pad driver mode register control selections, a temperature compensation logic, which is a part of the system control unit (SCU), makes it possible to adjust the edges of the high-speed class A2 GPIO output lines depending on the die temperature. The temperature compensation logic affects all class A2 GPIO output lines that are used simultaneously as output.

The fully programmable temperature compensation logic provides four types of output driver levels:

- Maximum level (low temperature)
- High level
- Low level
- Minimum level (high temperature)

General Purpose I/O Ports and Peripheral I/O Lines

Further details of the temperature compensation logic are described in [Section 5.6](#) on [Page 5-41](#) (SCU) of this User's Manual.

10.2.3 Port Output Register

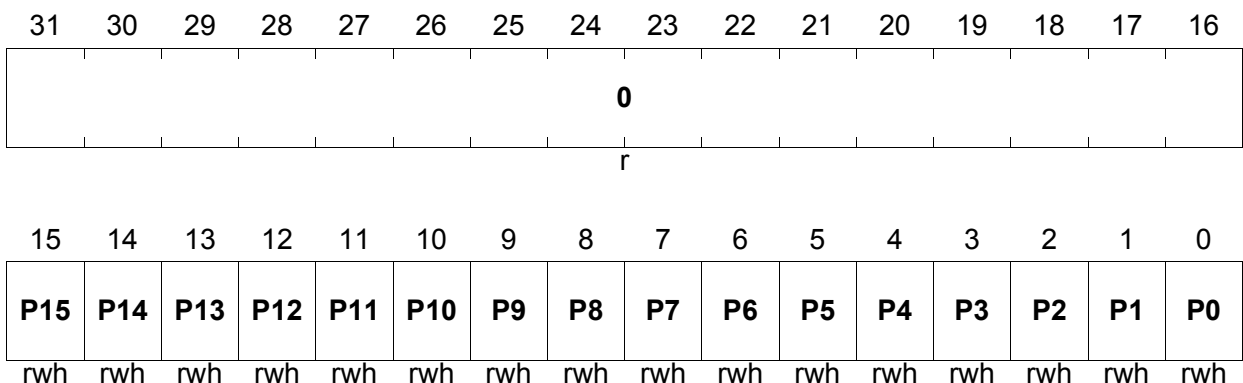
The port output register determines the value of a GPIO pin when it is selected by Pn_IOCRx as output. Writing a 0 to a Pn_OUT.Px (x = 0-15) bit position delivers a low level at the corresponding output pin. A high level is output when the corresponding bit is written with a 1. Note that the bits of Pn_OUT.Px can also be set/cleared by writing appropriate values into the port output modification register Pn_OMR.

Pn_OUT

Port n Output Register

(00_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
Px (x = 0-15)	x	rwh	Port n Output Bit x This bit determines the level at the output pin Pn.x if the output is selected as GPIO output. 0 _B The output level of Pn.x is 0. 1 _B The output level of Pn.x is 1. Pn.x can also be set/cleared by control bits of the Pn_OMR register.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

Note: Only Port 0, 1, 3, and 4 are 16-bit wide ports. The Pn_OUT registers of the other ports have a reduced number of Px bits (see Pn_OUT register descriptions in the corresponding port sections).

General Purpose I/O Ports and Peripheral I/O Lines

10.2.4 Port Output Modification Register

The port output modification register contains control bits that make it possible to individually set, clear, or toggle the logic state of a single port line by manipulating the output register.

Pn_OMR

Port n Output Modification Register (04_H)

Reset Value: 0000 XXXX_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PR	PR	PR	PR	PR	PR	PR	PR	PR	PR	PR	PR	PR	PR	PR	PR
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS	PS	PS	PS	PS	PS	PS	PS	PS	PS	PS	PS	PS	PS	PS	PS
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Field	Bits	Type	Description
PSx (x = 0-15)	x	w	Port n Set Bit x Setting this bit will set or toggle the corresponding bit in the port output register Pn_OUT. The function of this bit is shown in Table 10-5 .
PRx (x = 0-15)	x + 16	w	Port n Reset Bit x Setting this bit will clear or toggle the corresponding bit in the port output register Pn_OUT. The function of this bit is shown in Table 10-5 .

*Note: Register Pn_OMR is virtual and does not contain any flip-flop. A read action delivers the value of the corresponding port input register Pn_IN. Therefore, reset values of Pn_OMR and Pn_IN registers are also identical.
The Pn_OMR registers should always be written using word accesses.*

General Purpose I/O Ports and Peripheral I/O Lines**Table 10-5 Function of the Bits PRx and PSx**

PRx	PSx	Function
0	0	Bit Pn_OUT.Px is not changed.
0	1	Bit Pn_OUT.Px is set.
1	0	Bit Pn_OUT.Px is cleared.
1	1	Bit Pn_OUT.Px is toggled.

General Purpose I/O Ports and Peripheral I/O Lines

10.2.5 Emergency Stop Register

All GPIO lines which are used by the GPTA modules (GPTA0, GPTA1, LTCA2) have an emergency stop logic implemented (see [Figure 10-2](#)). These GPTA related GPIO lines are:

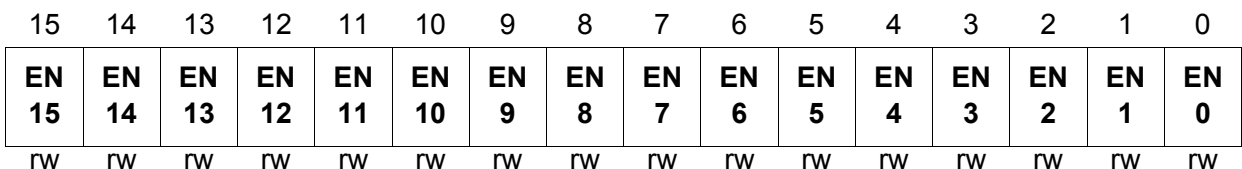
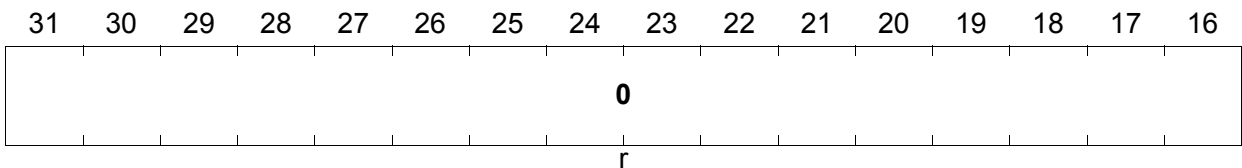
- P2.[15:8], P3.[15:0], P4.[15:0], P8.[7:0], and P9.[7:0]

Each of these GPIO lines has its own emergency stop enable bit EN_x that is located in the emergency stop register P_n_ESR of Port n. If the emergency stop signal becomes active, one of two states can be selected:

- Emergency stop function disabled (EN_x = 0):
A GPTA output line remains connected to the GPTA module (alternate function).
- Emergency stop function enabled (EN_x = 1):
A GPTA output line is disconnected from the GPTA module (alternate function) and connected to the corresponding bit of the P_n_OUT output register (the content of the corresponding PC_x bit fields in register P_n_IOCR is discarded).

P_n_ESR

Port n Emergency Stop Register (50_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
EN_x (x = 0-15)	x	rW	Emergency Stop Enable for Port n Pin x This bit enables the emergency stop function for GPIO lines used as GPTA outputs. If the emergency stop condition is met and enabled, the output selection is automatically switched from alternate (GPTA output) function to GPIO output function. 0 _B Emergency stop function for P _n .x is disabled. 1 _B Emergency stop function for P _n .x is enabled.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

General Purpose I/O Ports and Peripheral I/O Lines

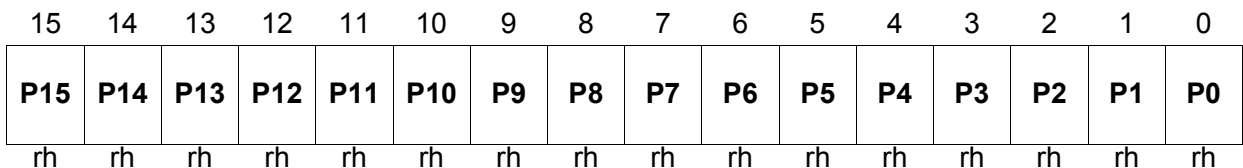
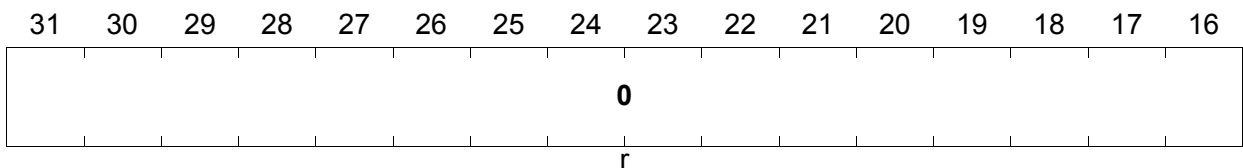
Note: Only Port 3 and 4 are 16-bit wide ports used by the GPTA modules. The Pn_ESR registers of the other ports have a reduced number of bits (see Pn_ESR register descriptions in the corresponding port sections).

10.2.6 Port Input Register

The logic level of a GPIO pin can be read via the read-only port input register Pn_IN. When a read of Pn_IN occurs, the values at the GPIO pins are latched in Pn_IN independently whether the GPIO pin is selected as input or output.

Pn_IN

Port n Input Register **(24_H)** **Reset Value: 0000 XXXX_H**



Field	Bits	Type	Description
Px (x = 0-15)	x	rh	Port n Input Bit x This bit indicates the level at the input pinPn.x. 0 _B The input level of Pn.x is 0. 1 _B The input level of Pn.x is 1.
0	[31:16]	r	Reserved Read as 0.

Note: Only Port 0, 1, 3, and 4 are 16-bit wide ports. The Pn_IN registers of the other ports have a reduced number of Px bits (see Pn_IN register descriptions in the corresponding port sections).

General Purpose I/O Ports and Peripheral I/O Lines

10.3 Port 0

This section describes the Port 0 functionality in detail.

10.3.1 Port 0 Configuration

Port 0 is a general-purpose 16-bit bi-directional port. It serves as GPIO lines without secondary functions. The logic state of the sixteen Port 0 lines is stored in the SCU Software Configuration Latched Input Register SCU_SCLIR at the rising edge of HDRST and can be read by software afterwards. Note that some of the P0.[7:0] lines are used for configuration purposes, too (see [Page 10-25](#)).

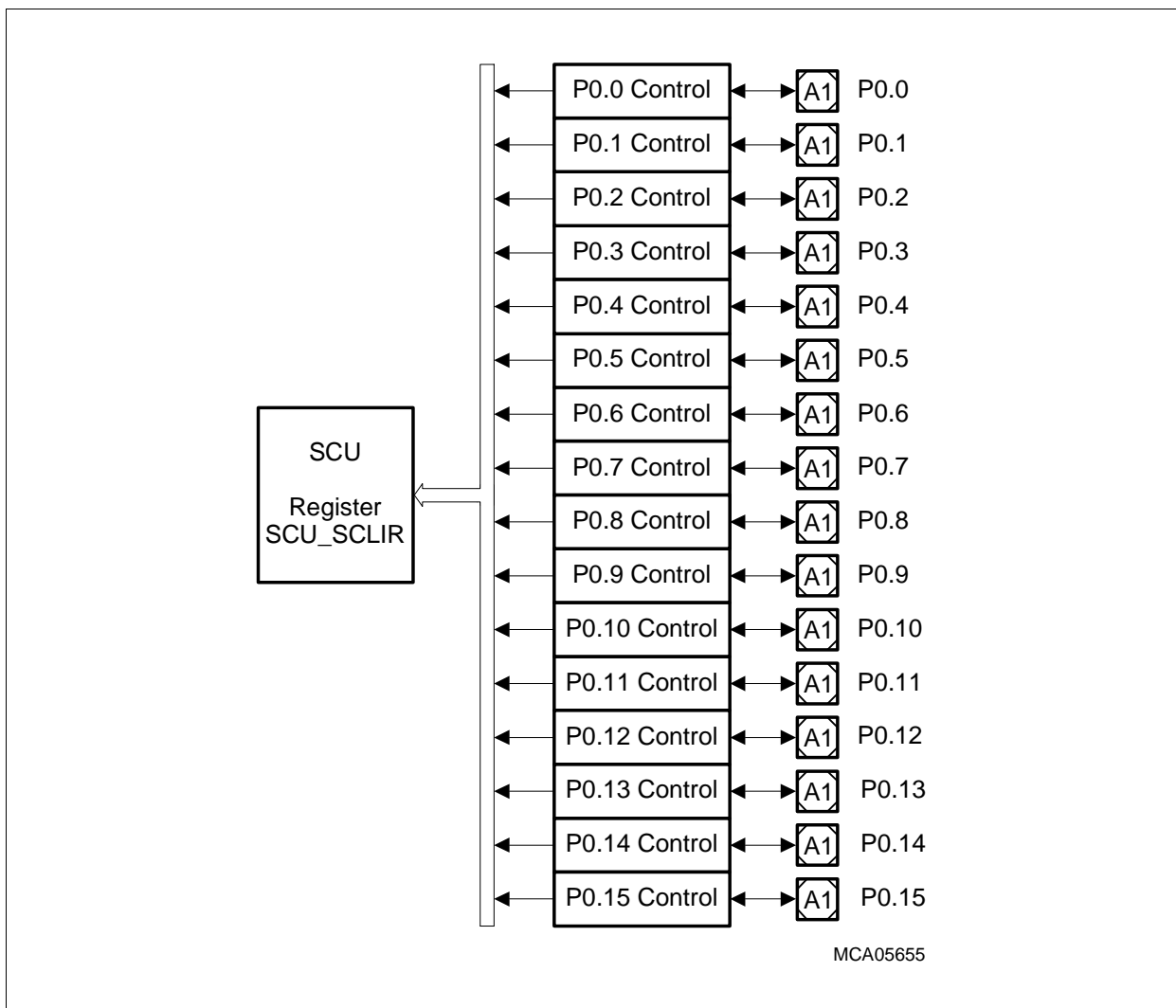


Figure 10-4 Port 0 Configuration Diagram

[Table 10-6](#) summarizes the I/O control selection functions of each Port 0 line.

General Purpose I/O Ports and Peripheral I/O Lines

10.3.2 Port 0 Function Table

Table 10-6 summarizes the I/O control selection functions of each Port 0 line.

Table 10-6 Port 0 Functions

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P0.0	I	General-purpose input	P0_IN.P0	P0_IOCRO.PC0	0XXX _B
	O	General-purpose output	P0_OUT.P0		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B
P0.1	I	General-purpose input	P0_IN.P1	P0_IOCRO.PC1	0XXX _B
	O	General-purpose output	P0_OUT.P1		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B
P0.2	I	General-purpose input	P0_IN.P2	P0_IOCRO.PC2	0XXX _B
	O	General-purpose output	P0_OUT.P2		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B
P0.3	I	General-purpose input	P0_IN.P3	P0_IOCRO.PC3	0XXX _B
	O	General-purpose output	P0_OUT.P3		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B
P0.4	I	General-purpose input	P0_IN.P4	P0_IOCRO4.PC4	0XXX _B
	O	General-purpose output	P0_OUT.P4		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B
P0.5	I	General-purpose input	P0_IN.P5	P0_IOCRO4.PC5	0XXX _B
	O	General-purpose output	P0_OUT.P5		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B

General Purpose I/O Ports and Peripheral I/O Lines

Table 10-6 Port 0 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P0.6	I	General-purpose input	P0_IN.P6	P0_IOCR4.PC6	0XXX _B
	O	General-purpose output	P0_OUT.P6		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B
P0.7	I	General-purpose input	P0_IN.P7	P0_IOCR4.PC7	0XXX _B
	O	General-purpose output	P0_OUT.P7		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B
P0.8	I	General-purpose input	P0_IN.P8	P0_IOCR8.PC8	0XXX _B
	O	General-purpose output	P0_OUT.P8		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B
P0.9	I	General-purpose input	P0_IN.P9	P0_IOCR8.PC9	0XXX _B
	O	General-purpose output	P0_OUT.P9		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B
P0.10	I	General-purpose input	P0_IN.P10	P0_IOCR8. PC10	0XXX _B
	O	General-purpose output	P0_OUT.P10		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B
P0.11	I	General-purpose input	P0_IN.P11	P0_IOCR8. PC11	0XXX _B
	O	General-purpose output	P0_OUT.P11		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B

General Purpose I/O Ports and Peripheral I/O Lines

Table 10-6 Port 0 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P0.12	I	General-purpose input	P0_IN.P12	P0_IOCRR12. PC12	0XXX _B
	O	General-purpose output	P0_OUT.P12		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B
P0.13	I	General-purpose input	P0_IN.P13	P0_IOCRR12. PC13	0XXX _B
	O	General-purpose output	P0_OUT.P13		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B
P0.14	I	General-purpose input	P0_IN.P14	P0_IOCRR12. PC14	0XXX _B
	O	General-purpose output	P0_OUT.P14		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B
P0.15	I	General-purpose input	P0_IN.P15	P0_IOCRR12. PC15	0XXX _B
	O	General-purpose output	P0_OUT.P15		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B

1) The port I/O control values P0_IOCRRx.Py that are assigned to this reserved alternate output control selection should not be used. Otherwise, unpredictable output port line behavior may occur.

General Purpose I/O Ports and Peripheral I/O Lines

10.3.3 Port 0 Registers

The following registers are available on Port 0:

Table 10-7 Port 0 Registers

Register Short Name	Register Long Name	Offset Address	Description see
P0_OUT	Port 0 Output Register	0000 _H	Page 10-13
P0_OMR	Port 0 Output Modification Register	0004 _H	Page 10-14
P0_IOCR0	Port 0 Input/Output Control Register 0	0010 _H	Page 10-7
P0_IOCR4	Port 0 Input/Output Control Register 4	0014 _H	Page 10-8
P0_IOCR8	Port 0 Input/Output Control Register 8	0018 _H	Page 10-8
P0_IOCR12	Port 0 Input/Output Control Register 12	001C _H	Page 10-9
P0_IN	Port 0 Input Register	0024 _H	Page 10-17
P0_PDR	Port 0 Pad Driver Mode Register	0040 _H	Page 10-23 ¹⁾
SCU_SCLIR	SCU Software Configuration Latched Inputs Register	²⁾	Page 10-24

1) This register is listed here in the Port 0 section because they differ from the general port register description given in [Section 10.2](#).

2) This register is located in the address range of the SCU.

Note: The complete address map of Port 0 is described in [Table 18-9](#) on [Page 18-22](#) of this TC1796 System Units (Vol. 1 of 2) User's Manual.

General Purpose I/O Ports and Peripheral I/O Lines

10.3.3.2 Port 0 Software Configuration Selection

The logic levels of the sixteen Port 0 lines are latched into the register SCU_SCLIR (SCU Software Configuration Latched Inputs Register) when the hardware reset goes inactive (at the rising edge of \overline{HDRST}). This feature makes it possible to use Port 0 lines for software configuration selection purposes.

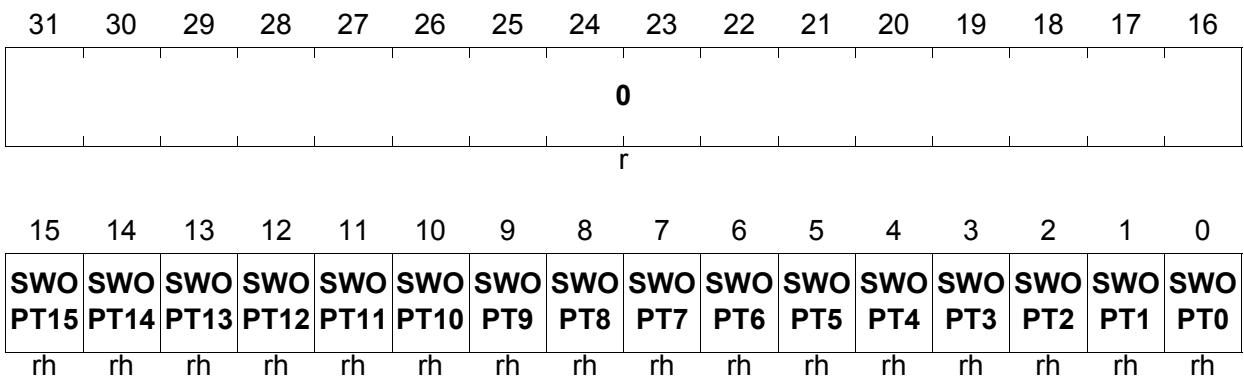
Depending on the TC1796 device used, several of the SWOPT bits (meaning several of the P0 lines) are reserved and may not be used by a user program for software configuration selection purposes. Detail see [Section 10.3.3.3](#) on the next page.

SCU_SCLIR

SCU Software Configuration Latched Inputs Register

(F000038_H)

Reset Value: 0000 XXXX_H



Field	Bits	Type	Function
SWOPT_x (x = 0-15)	x	rh	Software Configuration Bits These bits show the state of pin P0.x that was latched with the last rising edge of \overline{HDRST} . 0 _B P0.x state/logic level latched by \overline{HDRST} is 0. 1 _B P0.x state/logic level latched by \overline{HDRST} is 1.
0	[31:16]	r	Reserved Read as 0.

Note: The reset value (bits "X") of the register SCU_SCLIR is defined by the circuitry connected to Port 0 at the rising edge of \overline{HDRST} . Port 0 lines are set to inputs with pull-up devices connected (reset values of port input/output control registers).

General Purpose I/O Ports and Peripheral I/O Lines

10.3.3.3 Reserved SWOPT Bits of SCU_SCLIR Register

Depending on the TC1796 device version used in an application, several Port 0 lines (meaning several SWOPT bits) are reserved and cannot be used for user system purposes during a HDRST reset operation. [Table 10-8](#) defines the reserved Port 0 lines (indicated by 0 or 1) as well as the Port 0 lines that can be used by a user program (indicated by “user”) for software configuration selection (or as GPIO pins) depending on the specific TC1796 device version.

Table 10-8 Reserved SWOPT Bits in TC1796 Devices

TC1796 Device Version	SWOPTx Bits (x = 0-15)						
	P0. [15:8]	P0. [7:6]	P0.5	P0.4	P0.3	P0. [2:0]	P0. [15:8]
TC1796	user	user	user	1	user	1))	user
TC1796ED (Emulation device)		user	0 or 1 ²⁾	1			
TC1796-L (Flash-Not-Available version)		user	11 _B ³⁾	0			
TC1796ED in TC1796-L socket		11 _B ³⁾	0 or 1 ²⁾	0			

- 1) The P0.[2:0] bits are only used in alternate boot modes (see [Page 4-28](#)). If alternate boot modes are not required or used in an application, P0.[2:0] can also be used for user program software configuration selection purposes during a hardware reset operation or as GPIO pins.
- 2) 0: Emulation device functionality is not available.
1: Emulation device functionality is fully supported.
- 3) In Flash-Not-Available devices/sockets with P0.4 = 0, P0.[7:6] are used for test purposes and must always be set to 11_B during a hardware reset operation.

General Purpose I/O Ports and Peripheral I/O Lines

10.4 Port 1

This section describes the Port 1 functionality in detail.

10.4.1 Port 1 Configuration

Port 1 is a 16-bit bi-directional general-purpose I/O port that can be alternatively used for the MLI0 I/O lines or for the external trigger inputs REQ[3:0] of the CPU. Furthermore, the system clock output SYSCLK is provided at Port 1.

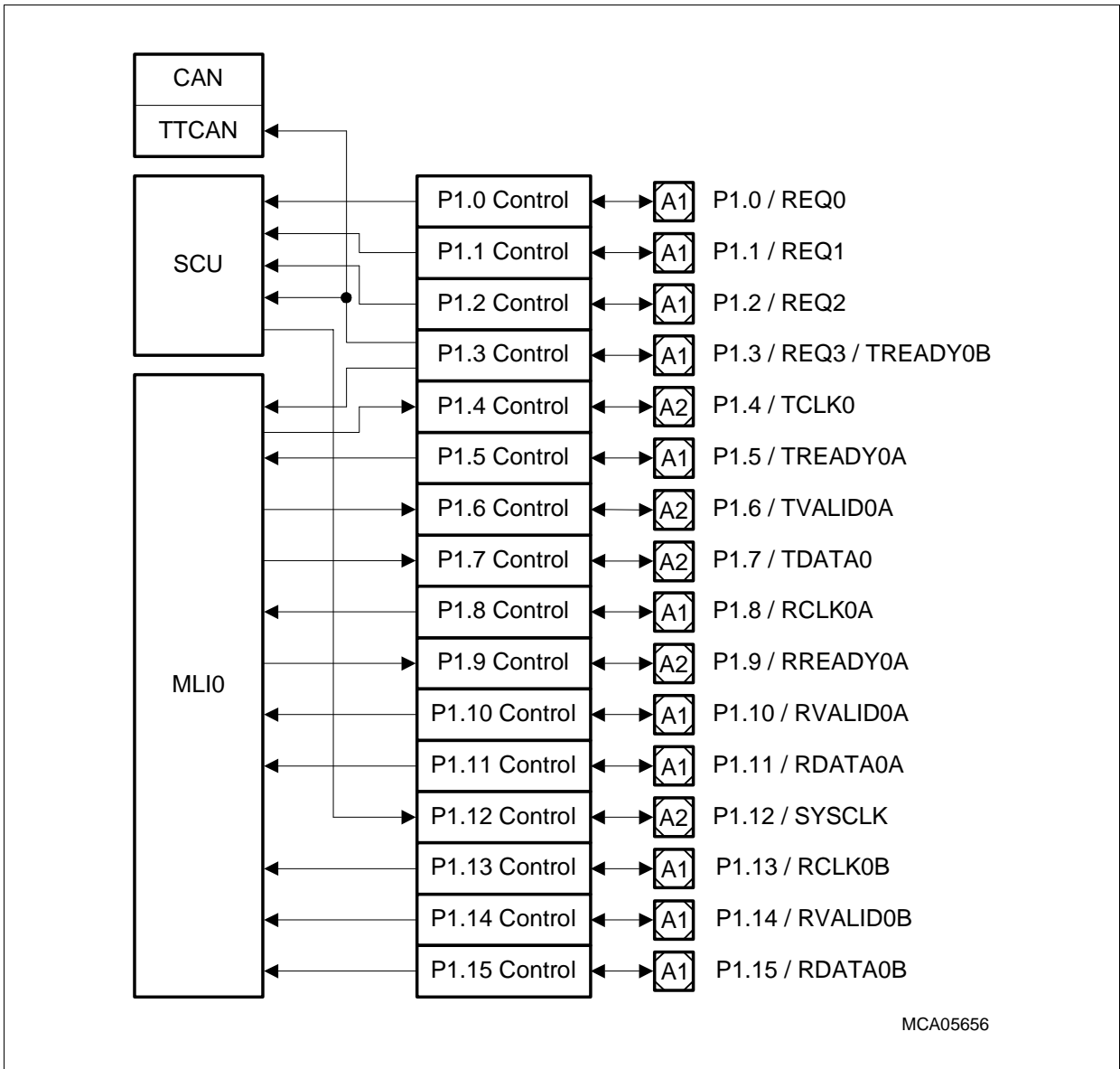


Figure 10-5 Port 1 Configuration Diagram

General Purpose I/O Ports and Peripheral I/O Lines

10.4.2 Port 1 Function Table

Table 10-9 summarizes the I/O control selection functions of each Port 1 line.

Table 10-9 Port 1 Functions

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P1.0	I	General-purpose input	P1_IN.P0	P1_IOCRO.PC0	0XXX _B
		SCU input	REQ0		
	O	General-purpose output	P1_OUT.P0		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B
P1.1	I	General-purpose input	P1_IN.P1	P1_IOCRO.PC1	0XXX _B
			REQ1		
	O	General-purpose output	P1_OUT.P1		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B
P1.2	I	General-purpose input	P1_IN.P2	P1_IOCRO.PC2	0XXX _B
			REQ2		
	O	General-purpose output	P1_OUT.P2		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B
P1.3	I	General-purpose input	P1_IN.P3	P1_IOCRO.PC3	0XXX _B
		SCU input	REQ3		
		TTCAN input ECTT1			
		MLI0 input	TREADY0B		
	O	General-purpose output	P1_OUT.P3		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B

General Purpose I/O Ports and Peripheral I/O Lines

Table 10-9 Port 1 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P1.4	I	General-purpose input	P1_IN.P4	P1_IOCR4.PC4	0XXX _B
	O	General-purpose output	P1_OUT.P4		1X00 _B
		MLI0 output	TCLK0		1X01 _B
		Reserved ¹⁾	–		1X10 _B 1X11 _B
P1.5	I	General-purpose input	P1_IN.P5	P1_IOCR4.PC5	0XXX _B
		MLI0 input	TREADY0A		
	O	General-purpose output	P1_OUT.P5		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B
P1.6	I	General-purpose input	P1_IN.P6	P1_IOCR4.PC6	0XXX _B
	O	General-purpose output	P1_OUT.P6		1X00 _B
		MLI0 output	TVALID0A		1X01 _B
		Reserved ¹⁾	–		1X10 _B 1X11 _B
P1.7	I	General-purpose input	P1_IN.P7	P1_IOCR4.PC7	0XXX _B
	O	General-purpose output	P1_OUT.P7		1X00 _B
		MLI0 output	TDATA0		1X01 _B
		Reserved ¹⁾	–		1X10 _B 1X11 _B
P1.8	I	General-purpose input	P1_IN.P8	P1_IOCR8.PC8	0XXX _B
		MLI0 input	RCLK0A		
	O	General-purpose output	P1_OUT.P8		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B

General Purpose I/O Ports and Peripheral I/O Lines

Table 10-9 Port 1 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P1.9	I	General-purpose input	P1_IN.P9	P1_IOCR8.PC9	0XXX _B
	O	General-purpose output	P1_OUT.P9		1X00 _B
		MLI0 output	RREADY0A		1X01 _B
		Reserved ¹⁾	–		1X10 _B 1X11 _B
P1.10	I	General-purpose input	P1_IN.P10	P1_IOCR8.PC10	0XXX _B
		MLI0 input	RVALID0A		
	O	General-purpose output	P1_OUT.P10		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B
P1.11	I	General-purpose input	P1_IN.P11	P1_IOCR8.PC11	0XXX _B
		MLI0 input	RDATA0A		
	O	General-purpose output	P1_OUT.P11		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B
P1.12		General-purpose input	P1_IN.P12	P1_IOCR12. PC12	0XXX _B
	O	General-purpose output	P1_OUT.P12		1X00 _B
		System clock output	SYSCCLK		1X01 _B
		Reserved ¹⁾	–		1X10 _B 1X11 _B
P1.13	I	General-purpose input	P1_IN.P13	P1_IOCR12. PC13	0XXX _B
		MLI0 input	RCLK0B		
	O	General-purpose output	P1_OUT.P13		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B

General Purpose I/O Ports and Peripheral I/O Lines

Table 10-9 Port 1 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P1.14	I	General-purpose input	P1_IN.P14	P1_IOCR12. PC14	0XXX _B
		MLI0 input	RVALID0B		
	O	General-purpose output	P1_OUT.P14		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B
P1.15	I	General-purpose input	P1_IN.P15	P1_IOCR12. PC15	0XXX _B
		MLI0 input	RDATA0B		
	O	General-purpose output	P1_OUT.P15		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B

1) The port I/O control values P1_IOCRx.Py that are assigned to this reserved alternate output control selection should not be used. Otherwise, unpredictable output port line behavior may occur.

10.4.3 Port 1 Registers

The following registers are available on Port 1:

Table 10-10 Port 1 Registers

Register Short Name	Register Long Name	Offset Address	Description see
P1_OUT	Port 1 Output Register	0000 _H	Page 10-13
P1_OMR	Port 1 Output Modification Register	0004 _H	Page 10-14
P1_IOCR0	Port 1 Input/Output Control Register 0	0010 _H	Page 10-7
P1_IOCR4	Port 1 Input/Output Control Register 4	0014 _H	Page 10-8
P1_IOCR8	Port 1 Input/Output Control Register 8	0018 _H	Page 10-8
P1_IOCR12	Port 1 Input/Output Control Register 12	001C _H	Page 10-9
P1_IN	Port 1 Input Register	0024 _H	Page 10-17
P1_PDR	Port 1 Pad Driver Mode Register	0040 _H	Page 10-31 ¹⁾

1) This register is listed here in the Port 1 section because they differ from the general port register description given in [Section 10.2](#).

General Purpose I/O Ports and Peripheral I/O Lines

Note: The complete address map of Port 1 is described in [Table 18-10](#) on [Page 18-23](#) of this TC1796 System Units (Vol. 1 of 2) User's Manual.

10.4.3.1 Port 1 Pad Driver Mode Register and Pad Classes

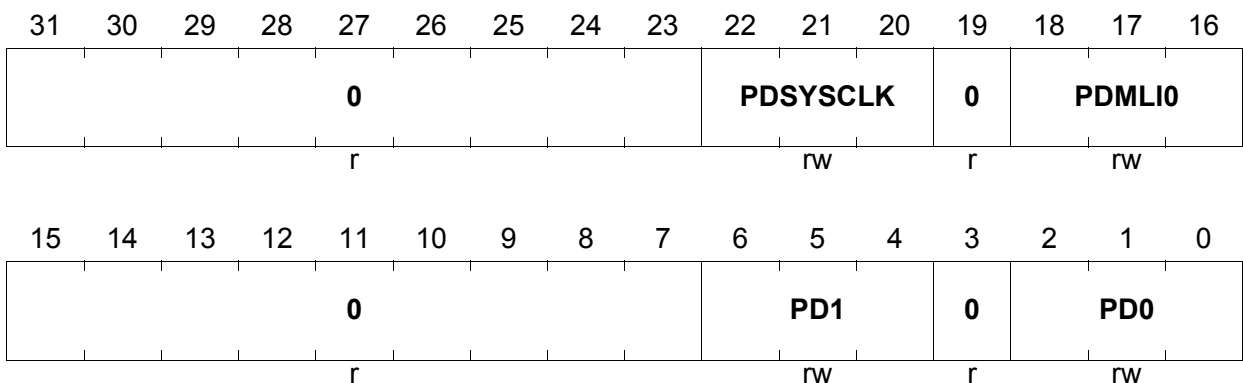
The Port 1 pad driver mode register contains four bit fields that determine the pad driver mode (output driver strength and slew rate) of Port 1 lines and line groups. The Port 1 port lines are assigned to A1 and A2 pad classes (see also [Figure 10-5](#)).

P1_PDR

Port 1 Pad Driver Mode Register

(40_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PD0	[2:0]	rw	Pad Driver Mode for P1.[3:0] and P1.[15:13] (Class A1 pads; for coding see Page 10-11)
PD1	[6:4]	rw	Pad Driver Mode for P1.5, P1.8, P1.[11:10] (Class A1 pads; for coding see Page 10-11)
PDMLI0	[18:16]	rw	Pad Driver Mode for P1.4, P1.[7:6], P1.9 (Class A2 pads; for coding see Page 10-11)
PDSYSCLK	[22:20]	rw	Pad Driver Mode for P1.12 (Class A2 pads; for coding see Page 10-11)
0	3, 19, [15:7], [31:23]	r	Reserved Read as 0; should be written with 0.

General Purpose I/O Ports and Peripheral I/O Lines

10.5 Port 2

This section describes the Port 2 functionality in detail.

10.5.1 Port 2 Configuration

Port 2 is a 14-bit bi-directional general-purpose I/O port that can be used either for the SSC0/SSC1 chip select output lines or for MSC0/MSC1 or GPTA0/GPTA1/LTCA2 I/O lines.

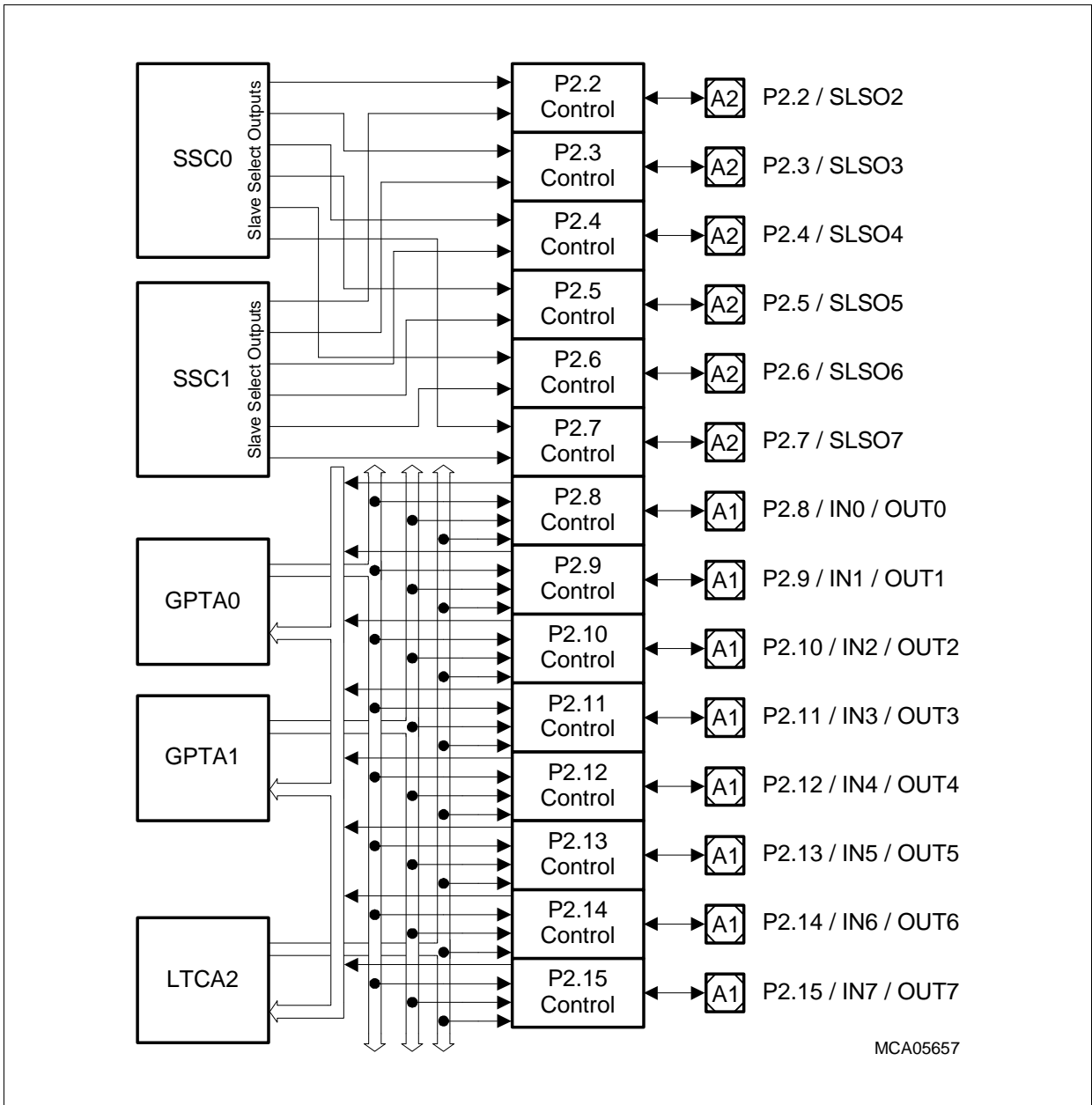


Figure 10-6 Port 2 Configuration Diagram

General Purpose I/O Ports and Peripheral I/O Lines

10.5.2 Port 2 Function Table

Table 10-11 summarizes the I/O control selection functions of each Port 2 line.

Table 10-11 Port 2 Functions

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P2.2	I	General-purpose input	P2_IN.P2	P2_IOCR0.PC2	0XXX _B
	O	General-purpose output	P2_OUT.P2		1X00 _B
		SSC0 output	SLSO02		1X01 _B
		SSC1 output	SLSO12		1X10 _B
		SSC0/SSC1 output	SLSO02 AND SLSO12		1X11 _B
P2.3	I	General-purpose input	P2_IN.P3	P2_IOCR0.PC3	0XXX _B
	O	General-purpose output	P2_OUT.P3		1X00 _B
		SSC0 output	SLSO03		1X01 _B
		SSC1 output	SLSO13		1X10 _B
		SSC0/SSC1 output	SLSO03 AND SLSO13		1X11 _B
P2.4	I	General-purpose input	P2_IN.P4	P2_IOCR4.PC4	0XXX _B
	O	General-purpose output	P2_OUT.P4		1X00 _B
		SSC0 output	SLSO04		1X01 _B
		SSC1 output	SLSO14		1X10 _B
		SSC0/SSC1 output	SLSO04 AND SLSO14		1X11 _B
P2.5	I	General-purpose input	P2_IN.P5	P2_IOCR4.PC5	0XXX _B
	O	General-purpose output	P2_OUT.P5		1X00 _B
		SSC0 output	SLSO05		1X01 _B
		SSC1 output	SLSO15		1X10 _B
		SSC0/SSC1 output	SLSO05 AND SLSO15		1X11 _B

General Purpose I/O Ports and Peripheral I/O Lines

Table 10-11 Port 2 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P2.6	I	General-purpose input	P2_IN.P6	P2_IOCR4.PC6	0XXX _B
	O	General-purpose output	P2_OUT.P6		1X00 _B
		SSC0 output	SLSO06		1X01 _B
		SSC1 output	SLSO16		1X10 _B
		SSC0/SSC1 output	SLSO06 AND SLSO16		1X11 _B
P2.7	I	General-purpose input	P2_IN.P7	P2_IOCR4.PC7	0XXX _B
	O	General-purpose output	P2_OUT.P7		1X00 _B
		SSC0 output	SLSO07		1X01 _B
		SSC1 output	SLSO17		1X10 _B
		SSC0/SSC1 output	SLSO07 AND SLSO17		1X11 _B
P2.8	I	General-purpose input	P2_IN.P8	P2_IOCR8.PC8	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN0		
	O	General-purpose output	P2_OUT.P8		1X00 _B
		GPTA0 output	OUT0		1X01 _B
		GPTA1 output	OUT0		1X10 _B
		LTCA2 output	OUT0		1X11 _B
	P2.9	I	General-purpose input		P2_IN.P9
GPTA0/GPTA1/LTCA2 input			IN1		
O		General-purpose output	P2_OUT.P9	1X00 _B	
		GPTA0 output	OUT1	1X01 _B	
		GPTA1 output	OUT1	1X10 _B	
		LTCA2 output	OUT1	1X11 _B	

General Purpose I/O Ports and Peripheral I/O Lines

Table 10-11 Port 2 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P2.10	I	General-purpose input	P2_IN.P10	P2_IOCR8. PC10	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN2		
	O	General-purpose output	P2_OUT.P10		1X00 _B
		GPTA0 output	OUT2		1X01 _B
		GPTA1 output	OUT2		1X10 _B
LTCA2 output	OUT2	1X11 _B			
P2.11	I	General-purpose input	P2_IN.P11	P2_IOCR8. PC11	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN3		
	O	General-purpose output	P2_OUT.P11		1X00 _B
		GPTA0 output	OUT3		1X01 _B
		GPTA1 output	OUT3		1X10 _B
LTCA2 output	OUT3	1X11 _B			
P2.12	I	General-purpose input	P2_IN.P12	P2_IOCR12. PC12	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN4		
	O	General-purpose output	P2_OUT.P12		1X00 _B
		GPTA0 output	OUT4		1X01 _B
		GPTA1 output	OUT4		1X10 _B
LTCA2 output	OUT4	1X11 _B			
P2.13	I	General-purpose input	P2_IN.P13	P2_IOCR12. PC13	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN5		
	O	General-purpose output	P2_OUT.P13		1X00 _B
		GPTA0 output	OUT5		1X01 _B
		GPTA1 output	OUT5		1X10 _B
LTCA2 output	OUT5	1X11 _B			

General Purpose I/O Ports and Peripheral I/O Lines

Table 10-11 Port 2 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P2.14	I	General-purpose input	P2_IN.P14	P2_IOCR12. PC14	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN6		
	O	General-purpose output	P2_OUT.P14		1X00 _B
		GPTA0 output	OUT6		1X01 _B
		GPTA1 output	OUT6		1X10 _B
LTCA2 output	OUT6	1X11 _B			
P2.15	I	General-purpose input	P2_IN.P15	P2_IOCR12. PC15	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN7		
	O	General-purpose output	P2_OUT.P15		1X00 _B
		GPTA0 output	OUT7		1X01 _B
		GPTA1 output	OUT7		1X10 _B
LTCA2 output	OUT7	1X11 _B			

General Purpose I/O Ports and Peripheral I/O Lines

10.5.3 Port 2 Registers

The following registers are available on Port 2:

Table 10-12 Port 2 Registers

Register Short Name	Register Long Name	Offset Address	Description see
P2_OUT	Port 2 Output Register	0000 _H	below ¹⁾
P2_OMR	Port 2 Output Modification Register	0004 _H	
P2_IOCRO	Port 2 Input/Output Control Register 0	0010 _H	Page 10-38 ¹⁾
P2_IOCRA	Port 2 Input/Output Control Register 4	0014 _H	Page 10-8
P2_IOCRA	Port 2 Input/Output Control Register 8	0018 _H	Page 10-8
P2_IOCRA	Port 2 Input/Output Control Register 12	001C _H	Page 10-9
P2_IN	Port 2 Input Register	0024 _H	Page 10-38 ¹⁾
P2_PDR	Port 2 Pad Driver Mode Register	0040 _H	Page 10-39 ¹⁾
P2_ESR	Port 2 Emergency Stop Register	0050 _H	Page 10-39 ¹⁾

1) These registers are listed and noted here in the Port 2 section because they differ from the general port register description given in [Section 10.2](#).

Note: The complete address map of Port 2 is described in [Table 18-11](#) on [Page 18-24](#) of this TC1796 System Units (Vol. 1 of 2) User's Manual.

10.5.3.1 Port 2 Output Register

The basic P2_OUT register functionality is described on [Page 10-13](#). Port lines P2.[1:0] are not connected to port lines. Therefore, reading the P2_OUT bits P[1:0] returns the value that was last written (0 after reset). These bits can be also set/cleared by the corresponding P2_OMR bits.

10.5.3.2 Port 2 Output Modification Register

The basic P2_OMR register functionality is described on [Page 10-14](#). However, port lines P2.0 and P2.1 are not available. Therefore, the P2_OMR bits PS[1:0] and PR[1:0] have no direct effect on port lines but only on register bits P2_OUT.P[1:0].

General Purpose I/O Ports and Peripheral I/O Lines

10.5.3.3 Port 2 Input/Output Control Register 0

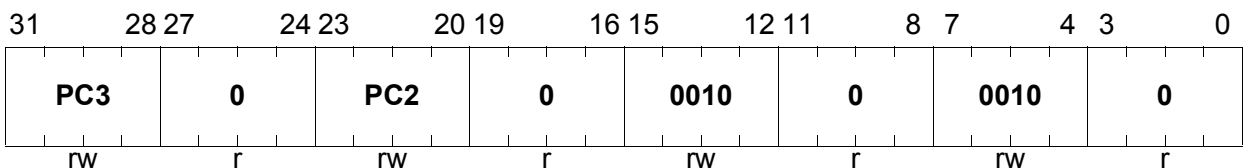
Port lines P2.0 and P2.1 are not available. Therefore, the PC0 and PC1 bit fields in register P2_IOCRO are not connected to any port lines.

P2_IOCRO

Port 2 Input/Output Control Register 0

(10_H)

Reset Value: 2020 2020_H



Field	Bits	Type	Description
–	[7:4], [15:12]	rw	Reserved Read as 0010 _B after reset; returns value that was written.
PC2	[23:20]	rw	Port Control for Port 2.2 (coding see Table 10-3 on Page 10-10)
PC3	[31:28]	rw	Port Control for Port 2.3 (coding see Table 10-3 on Page 10-10)
0	[3:0], [11:8], [19:16], [27:24]	r	Reserved Read as 0; should be written with 0.

10.5.3.4 Port 2 Input Register

The basic P2_IN register functionality is described on [Page 10-17](#). However, port lines P2.0 and P2.1 are not available. Therefore, bits P0 and P1 in register P2_IN are always read as 0.

10.5.3.5 Port 2 Emergency Stop Register

The basic P2_ESR register functionality is described on [Page 10-16](#). At Port 2, only port lines P2.[15:8] are connected to GPTA I/O lines. Therefore, only these port lines of Port 2 can be controlled for the emergency stop function. The P2_ESR bits EN[7:0] are not implemented. They are always read as 0 and should be written with 0.

General Purpose I/O Ports and Peripheral I/O Lines

10.5.3.6 Port 2 Pad Driver Mode Register and Pad Classes

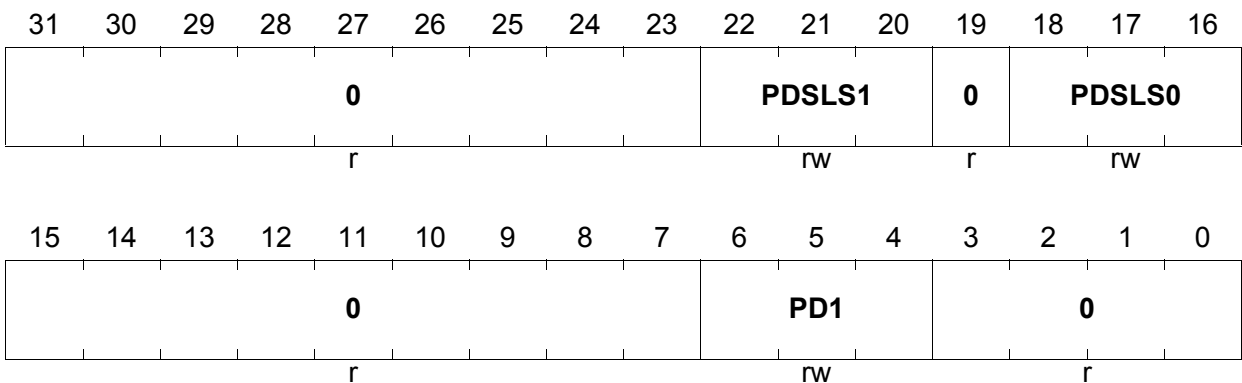
The Port 2 pad driver mode register contains four bit fields that determine the pad driver mode (output driver strength and slew rate) of Port 2 line groups. The Port 2 port lines are assigned to A1 and A2 pad classes (see also [Figure 10-6](#)).

P2_PDR

Port 2 Pad Driver Mode Register

(40_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PD1	[6:4]	rw	Pad Driver Mode for P2.[15:8] (Class A1 pads; for coding see Page 10-11)
PDSLS0	[18:16]	rw	Pad Driver Mode for P2.[3:2] (Class A2 pads; for coding see Page 10-11)
PDSLS1	[22:20]	rw	Pad Driver Mode for P2.[7:4] (Class A2 pads; for coding see Page 10-11)
0	[3:0], [15:7], 19, [31:23]	r	Reserved Read as 0; should be written with 0.

General Purpose I/O Ports and Peripheral I/O Lines

10.6 Port 3

This section describes the Port 3 functionality in detail.

10.6.1 Port 3 Configuration

Port 3 is a 16-bit bi-directional general-purpose I/O port that can be used for the GPTA0/GPTA1/LTCA2 I/O lines.

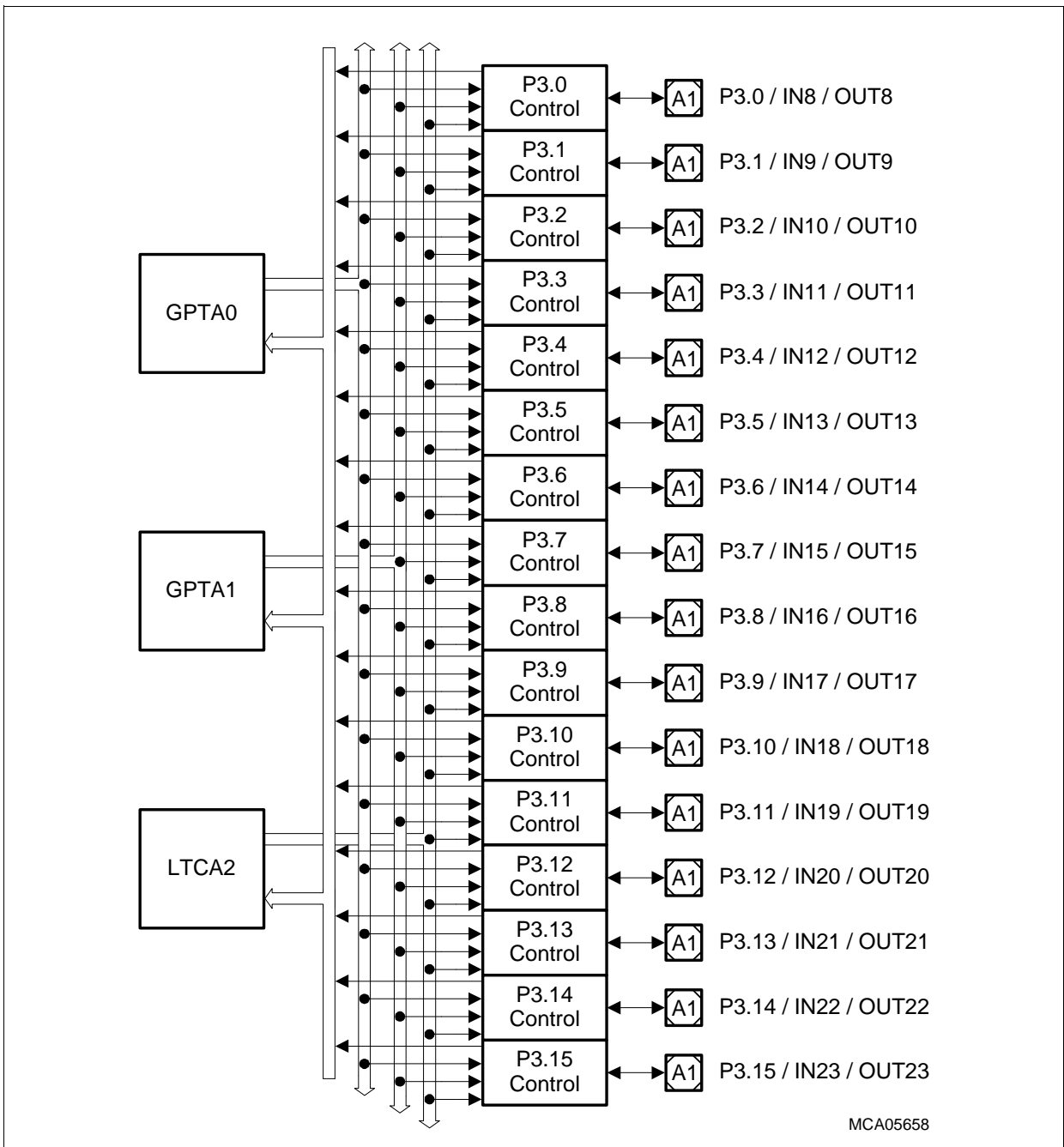


Figure 10-7 Port 3 Configuration Diagram

General Purpose I/O Ports and Peripheral I/O Lines

10.6.2 Port 3 Function Table

Table 10-13 summarizes the I/O control selection functions of each Port 3 line.

Table 10-13 Port 3 Functions

Port Pin	I/O	Pin Functionality	Associated Reg./I/O Line	Port I/O Control Select.			
				Reg./Bit Field	Value		
P3.0	I	General-purpose input	P3_IN.P0	P3_IOCR0.PC0	0XXX _B		
		GPTA0/GPTA1/LTCA2 input	IN8				
	O	General-purpose output	P3_OUT.P0		1X00 _B		
		GPTA0 output	OUT8		1X01 _B		
		GPTA1 output	OUT8		1X10 _B		
		LTCA2 output	OUT8		1X11 _B		
	P3.1	I	General-purpose input		P3_IN.P1	P3_IOCR0.PC1	0XXX _B
			GPTA0/GPTA1/LTCA2 input		IN9		
O		General-purpose output	P3_OUT.P1	1X00 _B			
		GPTA0 output	OUT9	1X01 _B			
		GPTA1 output	OUT9	1X10 _B			
		LTCA2 output	OUT9	1X11 _B			
P3.2		I	General-purpose input	P3_IN.P2	P3_IOCR0.PC2		0XXX _B
			GPTA0/GPTA1/LTCA2 input	IN10			
	O	General-purpose output	P3_OUT.P2	1X00 _B			
		GPTA0 output	OUT10	1X01 _B			
		GPTA1 output	OUT10	1X10 _B			
		LTCA2 output	OUT10	1X11 _B			
	P3.3	I	General-purpose input	P3_IN.P3		P3_IOCR0.PC3	0XXX _B
			GPTA0/GPTA1/LTCA2 input	IN11			
O		General-purpose output	P3_OUT.P3	1X00 _B			
		GPTA0 output	OUT11	1X01 _B			
		GPTA1 output	OUT11	1X10 _B			
		LTCA2 output	OUT11	1X11 _B			

General Purpose I/O Ports and Peripheral I/O Lines

Table 10-13 Port 3 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P3.4	I	General-purpose input	P3_IN.P4	P3_IOCR4.PC4	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN12		
	O	General-purpose output	P3_OUT.P4		1X00 _B
		GPTA0 output	OUT12		1X01 _B
		GPTA1 output	OUT12		1X10 _B
		LTCA2 output	OUT12		1X11 _B
	P3.5	I	General-purpose input		P3_IN.P5
GPTA0/GPTA1/LTCA2 input			IN13		
O		General-purpose output	P3_OUT.P5	1X00 _B	
		GPTA0 output	OUT13	1X01 _B	
		GPTA1 output	OUT13	1X10 _B	
		LTCA2 output	OUT13	1X11 _B	
P3.6		I	General-purpose input	P3_IN.P6	P3_IOCR4.PC6
	GPTA0/GPTA1/LTCA2 input		IN14		
	O	General-purpose output	P3_OUT.P6	1X00 _B	
		GPTA0 output	OUT14	1X01 _B	
		GPTA1 output	OUT14	1X10 _B	
		LTCA2 output	OUT14	1X11 _B	
	P3.7	I	General-purpose input	P3_IN.P7	
GPTA0/GPTA1/LTCA2 input			IN15		
O		General-purpose output	P3_OUT.P7	1X00 _B	
		GPTA0 output	OUT15	1X01 _B	
		GPTA1 output	OUT15	1X10 _B	
		LTCA2 output	OUT15	1X11 _B	

General Purpose I/O Ports and Peripheral I/O Lines

Table 10-13 Port 3 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P3.8	I	General-purpose input	P3_IN.P8	P3_IOCR8.PC8	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN16		
	O	General-purpose output	P3_OUT.P8		1X00 _B
		GPTA0 output	OUT16		1X01 _B
	GPTA1 output	OUT16	1X10 _B		
	LTCA2 output	OUT16	1X11 _B		
P3.9	I	General-purpose input	P3_IN.P9	P3_IOCR8.PC9	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN17		
	O	General-purpose output	P3_OUT.P9		1X00 _B
		GPTA0 output	OUT17		1X01 _B
	GPTA1 output	OUT17	1X10 _B		
	LTCA2 output	OUT17	1X11 _B		
P3.10	I	General-purpose input	P3_IN.P10	P3_IOCR8.PC10	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN18		
	O	General-purpose output	P3_OUT.P10		1X00 _B
		GPTA0 output	OUT18		1X01 _B
	GPTA1 output	OUT18	1X10 _B		
	LTCA2 output	OUT18	1X11 _B		
P3.11	I	General-purpose input	P3_IN.P11	P3_IOCR8.PC11	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN19		
	O	General-purpose output	P3_OUT.P11		1X00 _B
		GPTA0 output	OUT19		1X01 _B
	GPTA1 output	OUT19	1X10 _B		
	LTCA2 output	OUT19	1X11 _B		

General Purpose I/O Ports and Peripheral I/O Lines

Table 10-13 Port 3 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P3.12	I	General-purpose input	P3_IN.P12	P3_IOC12. PC12	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN20		
	O	General-purpose output	P3_OUT.P12		1X00 _B
		GPTA0 output	OUT20		1X01 _B
		GPTA1 output	OUT20		1X10 _B
LTCA2 output	OUT20	1X11 _B			
P3.13	I	General-purpose input	P3_IN.P13	P3_IOC12. PC13	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN21		
	O	General-purpose output	P3_OUT.P13		1X00 _B
		GPTA0 output	OUT21		1X01 _B
		GPTA1 output	OUT21		1X10 _B
LTCA2 output	OUT21	1X11 _B			
P3.14	I	General-purpose input	P3_IN.P14	P3_IOC12. PC14	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN22		
	O	General-purpose output	P3_OUT.P14		1X00 _B
		GPTA0 output	OUT22		1X01 _B
		GPTA1 output	OUT22		1X10 _B
LTCA2 output	OUT22	1X11 _B			
P3.15	I	General-purpose input	P3_IN.P15	P3_IOC12. PC15	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN23		
	O	General-purpose output	P3_OUT.P15		1X00 _B
		GPTA0 output	OUT23		1X01 _B
		GPTA1 output	OUT23		1X10 _B
LTCA2 output	OUT23	1X11 _B			

General Purpose I/O Ports and Peripheral I/O Lines

10.6.3 Port 3 Registers

The following registers are available on Port 3:

Table 10-14 Port 3 Registers

Register Short Name	Register Long Name	Offset Address	Description see
P3_OUT	Port 3 Output Register	0000 _H	Page 10-13
P3_OMR	Port 3 Output Modification Register	0004 _H	Page 10-14
P3_IOCRO	Port 3 Input/Output Control Register 0	0010 _H	Page 10-7
P3_IOCRA	Port 3 Input/Output Control Register 4	0014 _H	Page 10-8
P3_IOCRA	Port 3 Input/Output Control Register 8	0018 _H	Page 10-8
P3_IOCRA	Port 3 Input/Output Control Register 12	001C _H	Page 10-9
P3_IN	Port 3 Input Register	0024 _H	Page 10-17
P3_PDR	Port 3 Pad Driver Mode Register	0040 _H	Page 10-46 ¹⁾
P3_ESR	Port 3 Emergency Stop Register	0050 _H	Page 10-16

1) This register is listed here in the Port 3 section because they differ from the general port register description given in [Section 10.2](#).

Note: The complete address map of Port 3 is described in [Table 18-12](#) on [Page 18-25](#) of this TC1796 System Units (Vol. 1 of 2) User's Manual.

General Purpose I/O Ports and Peripheral I/O Lines

10.7 Port 4

This section describes the Port 4 functionality in detail.

10.7.1 Port 4 Configuration

Port 4 is a 16-bit bi-directional general-purpose I/O port that can be used for the GPTA0/GPTA1/LTCA2 I/O lines.

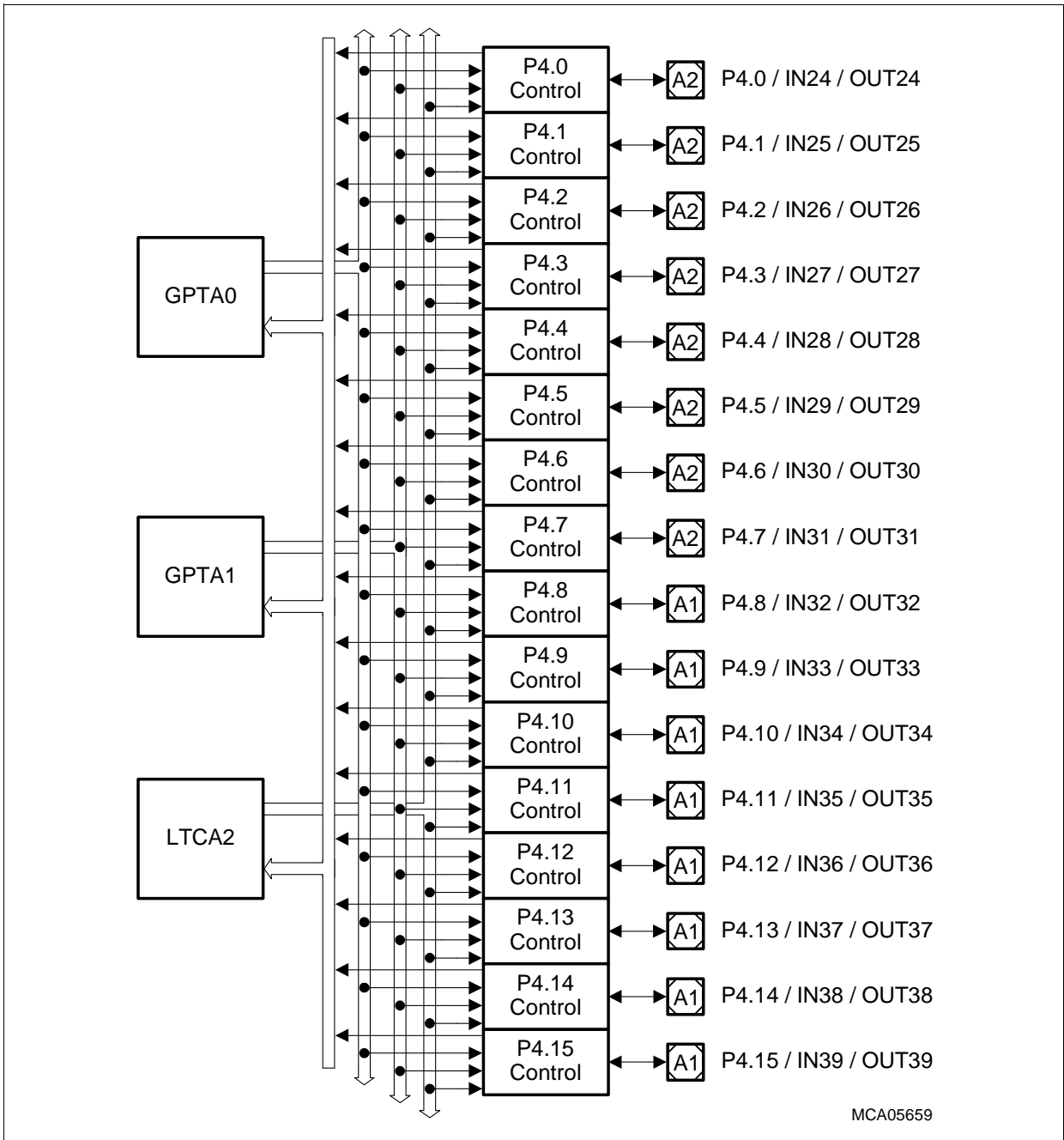


Figure 10-8 Port 4 Configuration Diagram

General Purpose I/O Ports and Peripheral I/O Lines

10.7.2 Port 4 Function Table

Table 10-15 summarizes the I/O control selection functions of each Port 4 line.

Table 10-15 Port 4 Functions

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P4.0	I	General-purpose input	P4_IN.P0	P4_IOCRO.PC0	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN24		
	O	General-purpose output	P4_OUT.P0		1X00 _B
		GPTA0 output	OUT24		1X01 _B
		GPTA1 output	OUT24		1X10 _B
		LTCA2 output	OUT24		1X11 _B
	P4.1	I	General-purpose input		P4_IN.P1
GPTA0/GPTA1/LTCA2 input			IN25		
O		General-purpose output	P4_OUT.P1	1X00 _B	
		GPTA0 output	OUT25	1X01 _B	
		GPTA1 output	OUT25	1X10 _B	
		LTCA2 output	OUT25	1X11 _B	
P4.2		I	General-purpose input	P4_IN.P2	P4_IOCRO.PC2
	GPTA0/GPTA1/LTCA2 input		IN26		
	O	General-purpose output	P4_OUT.P2	1X00 _B	
		GPTA0 output	OUT26	1X01 _B	
		GPTA1 output	OUT26	1X10 _B	
	LTCA2 output	OUT26	1X11 _B		
P4.3	I	General-purpose input	P4_IN.P3	P4_IOCRO.PC3	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN27		
	O	General-purpose output	P4_OUT.P3		1X00 _B
		GPTA0 output	OUT27		1X01 _B
		GPTA1 output	OUT27		1X10 _B
	LTCA2 output	OUT27	1X11 _B		

General Purpose I/O Ports and Peripheral I/O Lines

Table 10-15 Port 4 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P4.4	I	General-purpose input	P4_IN.P4	P4_IOCR4.PC4	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN28		
	O	General-purpose output	P4_OUT.P4		1X00 _B
		GPTA0 output	OUT28		1X01 _B
GPTA1 output	OUT28	1X10 _B			
LTCA2 output	OUT28	1X11 _B			
P4.5	I	General-purpose input	P4_IN.P5	P4_IOCR4.PC5	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN29		
	O	General-purpose output	P4_OUT.P5		1X00 _B
		GPTA0 output	OUT29		1X01 _B
GPTA1 output	OUT29	1X10 _B			
LTCA2 output	OUT29	1X11 _B			
P4.6	I	General-purpose input	P4_IN.P6	P4_IOCR4.PC6	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN30		
	O	General-purpose output	P4_OUT.P6		1X00 _B
		GPTA0 output	OUT30		1X01 _B
GPTA1 output	OUT30	1X10 _B			
LTCA2 output	OUT30	1X11 _B			
P4.7	I	General-purpose input	P4_IN.P7	P4_IOCR4.PC7	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN31		
	O	General-purpose output	P4_OUT.P7		1X00 _B
		GPTA0 output	OUT31		1X01 _B
GPTA1 output	OUT31	1X10 _B			
LTCA2 output	OUT31	1X11 _B			

General Purpose I/O Ports and Peripheral I/O Lines

Table 10-15 Port 4 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P4.8	I	General-purpose input	P4_IN.P8	P4_IOCR8.PC8	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN32		
	O	General-purpose output	P4_OUT.P8		1X00 _B
		GPTA0 output	OUT32		1X01 _B
		GPTA1 output	OUT32		1X10 _B
LTCA2 output	OUT32	1X11 _B			
P4.9	I	General-purpose input	P4_IN.P9	P4_IOCR8.PC9	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN33		
	O	General-purpose output	P4_OUT.P9		1X00 _B
		GPTA0 output	OUT33		1X01 _B
		GPTA1 output	OUT33		1X10 _B
LTCA2 output	OUT33	1X11 _B			
P4.10	I	General-purpose input	P4_IN.P10	P4_IOCR8.PC10	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN34		
	O	General-purpose output	P4_OUT.P10		1X00 _B
		GPTA0 output	OUT34		1X01 _B
		GPTA1 output	OUT34		1X10 _B
LTCA2 output	OUT34	1X11 _B			
P4.11	I	General-purpose input	P4_IN.P11	P4_IOCR8.PC11	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN35		
	O	General-purpose output	P4_OUT.P11		1X00 _B
		GPTA0 output	OUT35		1X01 _B
		GPTA1 output	OUT35		1X10 _B
LTCA2 output	OUT35	1X11 _B			

General Purpose I/O Ports and Peripheral I/O Lines

Table 10-15 Port 4 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P4.12	I	General-purpose input	P4_IN.P12	P4_IOCR12. PC12	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN36		
	O	General-purpose output	P4_OUT.P12		1X00 _B
		GPTA1 output	OUT36		1X10 _B
	GPTA0 output	OUT36	1X01 _B		
	LTCA2 output	OUT36	1X11 _B		
P4.13	I	General-purpose input	P4_IN.P13	P4_IOCR12. PC13	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN37		
	O	General-purpose output	P4_OUT.P13		1X00 _B
		GPTA0 output	OUT37		1X01 _B
	GPTA1 output	OUT37	1X10 _B		
	LTCA2 output	OUT37	1X11 _B		
P4.14	I	General-purpose input	P4_IN.P14	P4_IOCR12. PC14	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN38		
	O	General-purpose output	P4_OUT.P14		1X00 _B
		GPTA0 output	OUT38		1X01 _B
	GPTA1 output	OUT38	1X10 _B		
	LTCA2 output	OUT38	1X11 _B		
P4.15	I	General-purpose input	P4_IN.P15	P4_IOCR12. PC15	0XXX _B
		GPTA0/GPTA1/LTCA2 input	IN39		
	O	General-purpose output	P4_OUT.P15		1X00 _B
		GPTA0 output	OUT39		1X01 _B
	GPTA1 output	OUT39	1X10 _B		
	LTCA2 output	OUT39	1X11 _B		

10.7.3 Port 4 Registers

The following registers are available on Port 4:

General Purpose I/O Ports and Peripheral I/O Lines

Table 10-16 Port 4 Registers

Register Short Name	Register Long Name	Offset Address	Description see
P4_OUT	Port 4 Output Register	0000 _H	Page 10-13
P4_OMR	Port 4 Output Modification Register	0004 _H	Page 10-14
P4_IOCR0	Port 4 Input/Output Control Register 0	0010 _H	Page 10-7
P4_IOCR4	Port 4 Input/Output Control Register 4	0014 _H	Page 10-8
P4_IOCR8	Port 4 Input/Output Control Register 8	0018 _H	Page 10-8
P4_IOCR12	Port 4 Input/Output Control Register 12	001C _H	Page 10-9
P4_IN	Port 4 Input Register	0024 _H	Page 10-17
P4_PDR	Port 4 Pad Driver Mode Register	0040 _H	Page 10-53 ¹⁾
P4_ESR	Port 4 Emergency Stop Register	0050 _H	Page 10-16

1) This register is listed here in the Port 4 section because they differ from the general port register description given in [Section 10.2](#).

Note: The complete address map of Port 4 is described in [Table 18-13](#) on [Page 18-26](#) of this TC1796 System Units (Vol. 1 of 2) User's Manual.

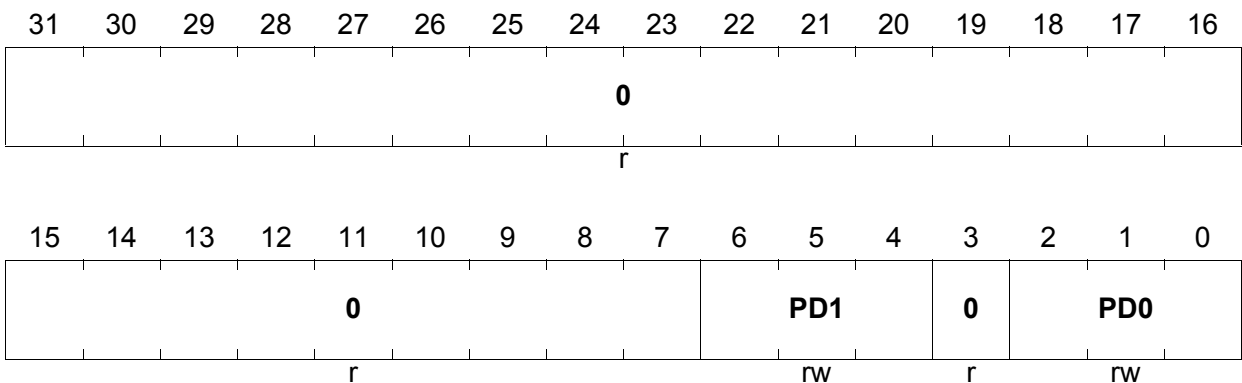
General Purpose I/O Ports and Peripheral I/O Lines

10.7.3.1 Port 4 Pad Driver Mode Register and Pad Classes

The Port 4 pad driver mode register contains two bit fields that determine the pad driver mode (output driver strength and slew rate) of Port 4 lines and line groups. The Port 4 port lines are assigned to A1 and A2 pad classes (see also [Figure 10-8](#)).

P4_PDR

Port 4 Pad Driver Mode Register (40_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
PD0	[2:0]	rw	Pad Driver Mode for P4.[7:0] (Class A2 pads; for coding see Page 10-11)
PD1	[6:4]	rw	Pad Driver Mode for P4.[15:8] (Class A1 pads; for coding see Page 10-11)
0	3, [31:7]	r	Reserved Read as 0; should be written with 0.

General Purpose I/O Ports and Peripheral I/O Lines

10.8 Port 5

This section describes the Port 5 functionality in detail.

10.8.1 Port 5 Configuration

Port 5 is an 8-bit bi-directional general-purpose I/O port which can be used for the ASC0/ASC1, MSC0/MS1, or MLI0 interface I/O lines.

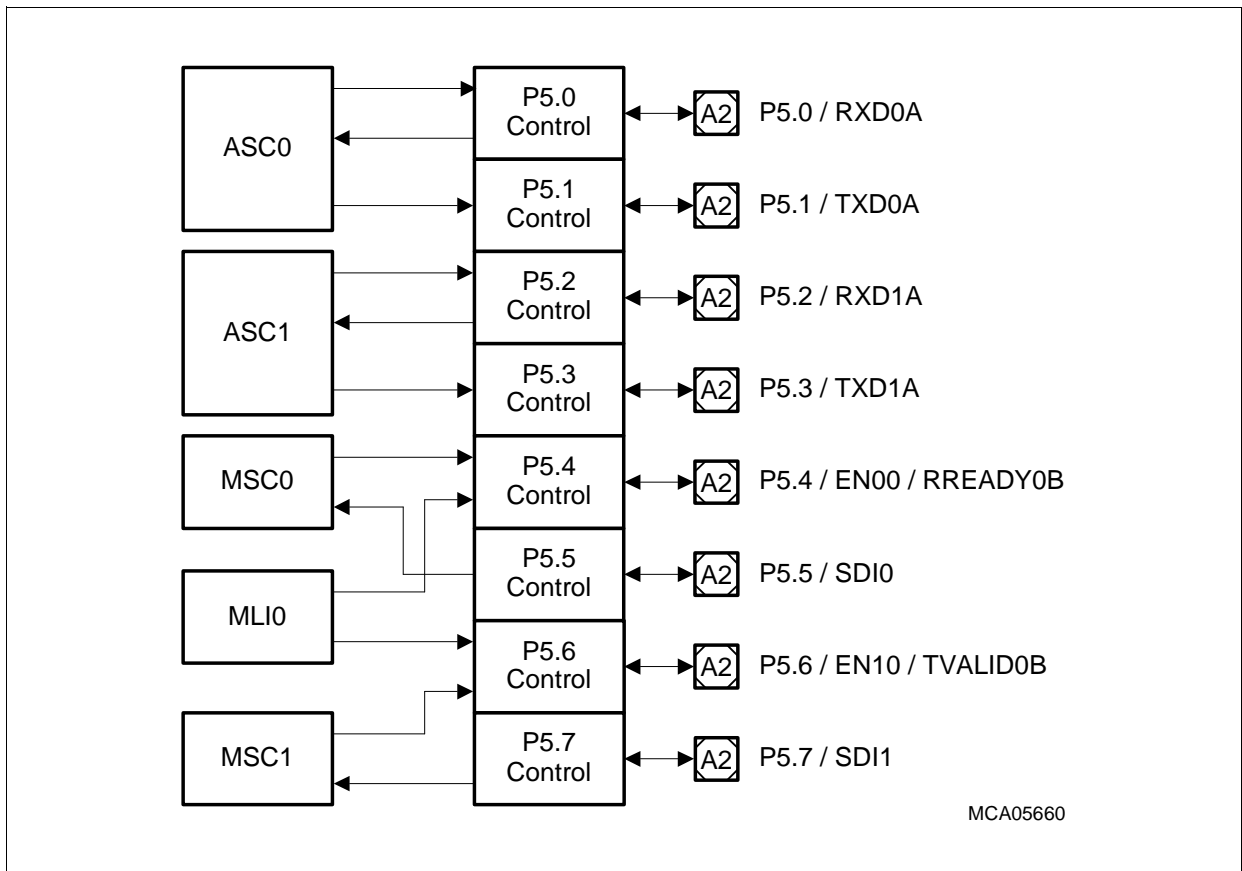


Figure 10-9 Port 5 Configuration Diagram

General Purpose I/O Ports and Peripheral I/O Lines

10.8.2 Port 5 Function Table

Table 10-17 summarizes the I/O control selection functions of each Port 5 line.

Table 10-17 Port 5 Functions

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P5.0	I	General-purpose input	P5_IN.P0	P5_IOCR0.PC0	0XXX _B
		ASC0 input	RXD0A		
	O	General-purpose output	P5_OUT.P0		1X00 _B
		ASC0 output (sync. mode)	RXD0A		1X01 _B
	Reserved ¹⁾	–	1X10 _B 1X11 _B		
P5.1	I	General-purpose input	P5_IN.P1	P5_IOCR0.PC1	0XXX _B
		General-purpose output	P5_OUT.P1		
	O	ASC0 output	TXD0A		1X01 _B
		Reserved ¹⁾	–		1X10 _B 1X11 _B
P5.2	I	General-purpose input	P5_IN.P2	P5_IOCR0.PC2	0XXX _B
		ASC1 input	RXD1A		
	O	General-purpose output	P5_OUT.P2		1X00 _B
		ASC1 output (sync. mode)	RXD1A		1X01 _B
	Reserved ¹⁾	–	1X10 _B 1X11 _B		
P5.3	I	General-purpose input	P5_IN.P3	P5_IOCR0.PC3	0XXX _B
		General-purpose output	P5_OUT.P3		
	O	ASC1 output	TXD1A		1X01 _B
		Reserved ¹⁾	–		1X10 _B 1X11 _B
P5.4	I	General-purpose input	P5_IN.P4	P5_IOCR4.PC4	0XXX _B
		General-purpose output	P5_OUT.P4		
	O	MSC0 output	EN00		1X01 _B 1X11 _B
		MLI0 output	RREADY0B		1X10 _B

General Purpose I/O Ports and Peripheral I/O Lines

Table 10-17 Port 5 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P5.5	I	General-purpose input	P5_IN.P5	P5_IOCR4.PC5	0XXX _B
		MSC0 input	SDI0		
	O	General-purpose output	P5_OUT.P5		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B
P5.6	I	General-purpose input	P5_IN.P6	P5_IOCR4.PC6	0XXX _B
		General-purpose output	P5_OUT.P6		1X00 _B
	O	MSC1 output	EN10		1X01 _B 1X11 _B
		MLI0 output	TVALID0B		1X10 _B
P5.7	I	General-purpose input	P5_IN.P7	P5_IOCR4.PC7	0XXX _B
		MSC1 input	SDI1		
	O	General-purpose output	P5_OUT.P7		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B

1) The port I/O control values P5_IOCRx.Py that are assigned to this reserved alternate output control selection should not be used. Otherwise, unpredictable output port line behavior may occur.

General Purpose I/O Ports and Peripheral I/O Lines

10.8.3 Port 5 Registers

The following registers are available on Port 5:

Table 10-18 Port 5 Registers

Register Short Name	Register Long Name	Offset Address	Description see
P5_OUT	Port 5 Output Register	0000 _H	below ¹⁾
P5_OMR	Port 5 Output Modification Register	0004 _H	
P5_IOCRO	Port 5 Input/Output Control Register 0	0010 _H	Page 10-8
P5_IOCRA	Port 5 Input/Output Control Register 4	0014 _H	Page 10-8
P5_IN	Port 5 Input Register	0024 _H	below ¹⁾
P5_PDR	Port 5 Pad Driver Mode Register	0040 _H	Page 10-58 ¹⁾

1) These registers are noted here in the Port 5 section because they differ from the general port register description given in [Section 10.2](#).

Note: The complete address map of Port 5 is described in [Table 18-14](#) on [Page 18-27](#) of this TC1796 System Units (Vol. 1 of 2) User's Manual.

10.8.3.1 Port 5 Output Register

The basic P5_OUT register functionality is described on [Page 10-13](#). Port lines P5.[15:8] are not available. Therefore, the P5_OUT bits P[15:8] should be written with 0, and are always read as 0.

10.8.3.2 Port 5 Output Modification Register

The basic P5_OMR register functionality is described on [Page 10-14](#). Port lines P5.[15:8] are not available. Therefore, the P5_OMR bits PS[15:8] and PR[15:8] are not implemented. These bits should always be written with 0.

10.8.3.3 Port 5 Input Register

The basic P5_IN register functionality is described on [Page 10-17](#). Port lines P5.[15:8] are not available. Therefore, the P5_IN bits P[15:8] are always read as 0.

General Purpose I/O Ports and Peripheral I/O Lines

10.8.3.4 Port 5 Pad Driver Mode Register and Pad Classes

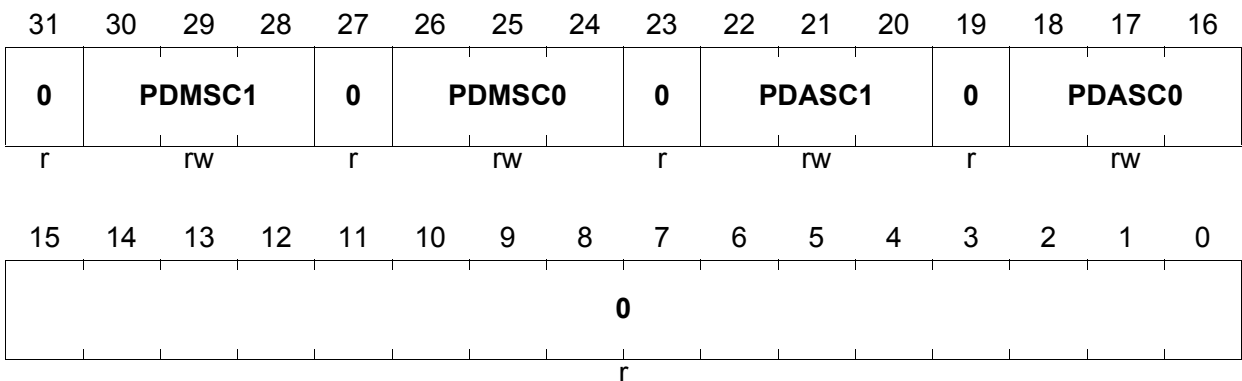
The Port 5 pad driver mode register contains four bit fields that determine the pad driver mode (output driver strength and slew rate) of Port 5 line groups. The Port 5 port lines are all class A2 pads (see also [Figure 10-9](#)).

P5_PDR

Port 5 Pad Driver Mode Register

(40_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PDASC0	[18:16]	rw	Pad Driver Mode for P5.[1:0] (Class A2 pads; for coding see Page 10-11)
PDASC1	[22:20]	rw	Pad Driver Mode for P5.[3:2] (Class A2 pads; for coding see Page 10-11)
PDMSC0	[26:24]	rw	Pad Driver Mode for P5.[5:4] (Class A2 pads; for coding see Page 10-11)
PDMSC1	[30:28]	rw	Pad Driver Mode for P5.[7:6] (Class A2 pads; for coding see Page 10-11)
0	[15:0], 19, 23, 27, 31	r	Reserved Read as 0; should be written with 0.

General Purpose I/O Ports and Peripheral I/O Lines

10.9 Port 6

This section describes the Port 6 functionality in detail.

10.9.1 Port 6 Configuration

Port 6 is a 12-bit bi-directional general-purpose I/O port which can be used for the SSC1, ASC0/ASC1, or for the MultiCAN controller I/O lines.

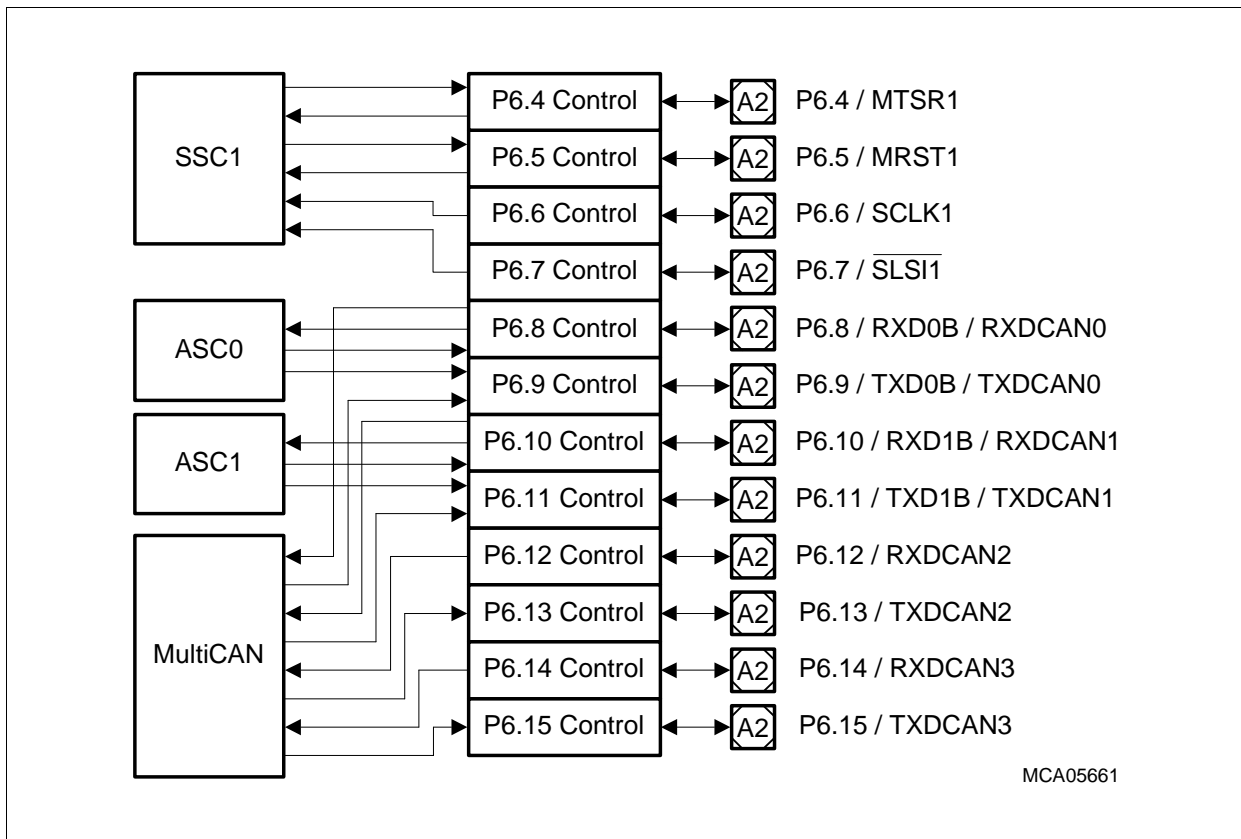


Figure 10-10 Port 6 Configuration Diagram

General Purpose I/O Ports and Peripheral I/O Lines

10.9.2 Port 6 Function Table

Table 10-19 summarizes the I/O control selection functions of each Port 6 line.

Table 10-19 Port 6 Functions

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P6.4	I	General-purpose input	P6_IN.P4	P6_IOCR4.PC4	0XXX _B
		SSC1 input (slave mode)	MTSR1		
	O	General-purpose output	P6_OUT.P4		1X00 _B
		SSC1 output (master mode)	MTSR1		1X01 _B
	Reserved ¹⁾	–	1X10 _B 1X11 _B		
P6.5	I	General-purpose input	P6_IN.P5	P6_IOCR4.PC5	0XXX _B
		SSC1 input (master mode)	MRST1		
	O	General-purpose output	P6_OUT.P5		1X00 _B
		SSC1 output (slave mode)	MRST1		1X01 _B
		Reserved ¹⁾	–		1X10 _B 1X11 _B
P6.6	I	General-purpose input	P6_IN.P6	P6_IOCR4.PC6	0XXX _B
		SSC1 input (slave mode)	SCLK1		
	O	General-purpose output	P6_OUT.P6		1X00 _B
		SSC1 output (master mode)	SCLK1		1X01 _B
	Reserved ¹⁾	–	1X10 _B 1X11 _B		
P6.7	I	General-purpose input	P6_IN.P7	P6_IOCR4.PC7	0XXX _B
		SSC1 input	SLSI1		
	O	General-purpose output	P6_OUT.P7		1X00 _B
	Reserved ¹⁾	–	1X01 _B 1X10 _B 1X11 _B		

General Purpose I/O Ports and Peripheral I/O Lines

Table 10-19 Port 6 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P6.8	I	General-purpose input	P6_IN.P8	P6_IOCR8.PC8	0XXX _B
		CAN node 0 rec. input 0 CAN node 3 rec. input 1	RXDCAN0		
		ASC0 input	RXD0B		
	O	General-purpose output	P6_OUT.P8		1X00 _B 1X01 _B
		ASC0 output (sync. mode)	RXD0B		1X10 _B
		Reserved ¹⁾	–		1X11 _B
P6.9	I	General-purpose input	P6_IN.P9	P6_IOCR8.PC9	0XXX _B
	O	General-purpose output	P6_OUT.P9		1X00 _B
		CAN node 0 output	TXDCAN0		1X01 _B
		ASC0 output	TXD0B		1X10 _B
		Reserved ¹⁾	–		1X11 _B
P6.10	I	General-purpose input	P6_IN.P10	P6_IOCR8. PC10	0XXX _B
		CAN node 1 rec. input 0 CAN node 0 rec. input 1	RXDCAN1		
		ASC1 input	RXD1B		
	O	General-purpose output	P6_OUT.P10		1X00 _B 1X01 _B
		ASC1 output (sync. mode)	RXD1B		1X01 _B
		Reserved ¹⁾	–		1X11 _B
P6.11	I	General-purpose input	P6_IN.P11	P6_IOCR8. PC11	0XXX _B
	O	General-purpose output	P6_OUT.P11		1X00 _B
		CAN node 1 output	TXDCAN1		1X01 _B
		ASC1 output	TXD1B		1X10 _B
		Reserved ¹⁾	–		1X11 _B

General Purpose I/O Ports and Peripheral I/O Lines

Table 10-19 Port 6 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P6.12	I	General-purpose input	P6_IN.P12	P6_IOCR12. PC12	0XXX _B
		CAN node 2 rec. input 0 CAN node 1 rec. input 1	RXDCAN2		
	O	General-purpose output	P6_OUT.P12		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B
P6.13	I	General-purpose input	P6_IN.P13	P6_IOCR12. PC13	0XXX _B
	O	General-purpose output	P6_OUT.P13		1X00 _B
		CAN node 2 output	TXDCAN2		1X01 _B
		Reserved ¹⁾	–		1X10 _B 1X11 _B
P6.14	I	General-purpose input	P6_IN.P14	P6_IOCR12. PC14	0XXX _B
		CAN node 3 rec. input 0 CAN node 2 rec. input 1	RXDCAN3		
	O	General-purpose output	P6_OUT.P14		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B
P6.15	I	General-purpose input	P6_IN.P15	P6_IOCR12. PC15	0XXX _B
	O	General-purpose output	P6_OUT.P15		1X00 _B
		CAN node 3 output	TXDCAN3		1X01 _B
		Reserved ¹⁾	–		1X10 _B 1X11 _B

1) The port I/O control values P6_IOCRx.Py that are assigned to this reserved alternate output control selection should not be used. Otherwise, unpredictable output port line behavior may occur.

General Purpose I/O Ports and Peripheral I/O Lines

10.9.3 Port 6 Registers

The following registers are available on Port 6:

Table 10-20 Port 6 Registers

Register Short Name	Register Long Name	Offset Address	Description see
P6_OUT	Port 6 Output Register	0000 _H	below ¹⁾
P6_OMR	Port 6 Output Modification Register	0004 _H	
P6_IOC4	Port 6 Input/Output Control Register 4	0014 _H	Page 10-8
P6_IOC8	Port 6 Input/Output Control Register 8	0018 _H	Page 10-8
P6_IOC12	Port 6 Input/Output Control Register 12	001C _H	Page 10-9
P6_IN	Port 6 Input Register	0024 _H	below ¹⁾
P6_PDR	Port 6 Pad Driver Mode Register	0040 _H	Page 10-64 ¹⁾

1) These registers are listed and noted here in the Port 6 section because they differ from the general port register description given in [Section 10.2](#).

Note: The complete address map of Port 6 is described in [Table 18-15](#) on [Page 18-28](#) of this TC1796 System Units (Vol. 1 of 2) User's Manual.

10.9.3.1 Port 6 Output Register

The basic P6_OUT register functionality is described on [Page 10-13](#). Port lines P6.[3:0] are not connected to port lines. Therefore, reading the P6_OUT bits P[3:0] returns the value that was last written (0 after reset). These bits can be also set/cleared by the corresponding P6_OMR bits.

10.9.3.2 Port 6 Output Modification Register

The basic P6_OMR register functionality is described on [Page 10-14](#). Port lines P6.[3:0] are not available. Therefore, they are not implemented. These bits should always be written with 0.

10.9.3.3 Port 6 Input Register

The basic P6_IN register functionality is described on [Page 10-17](#). Port lines P6.[3:0] are not available. Therefore, the P6_IN bits P[3:0] are always read as 0.

General Purpose I/O Ports and Peripheral I/O Lines

10.9.3.4 Port 6 Pad Driver Mode Register and Pad Classes

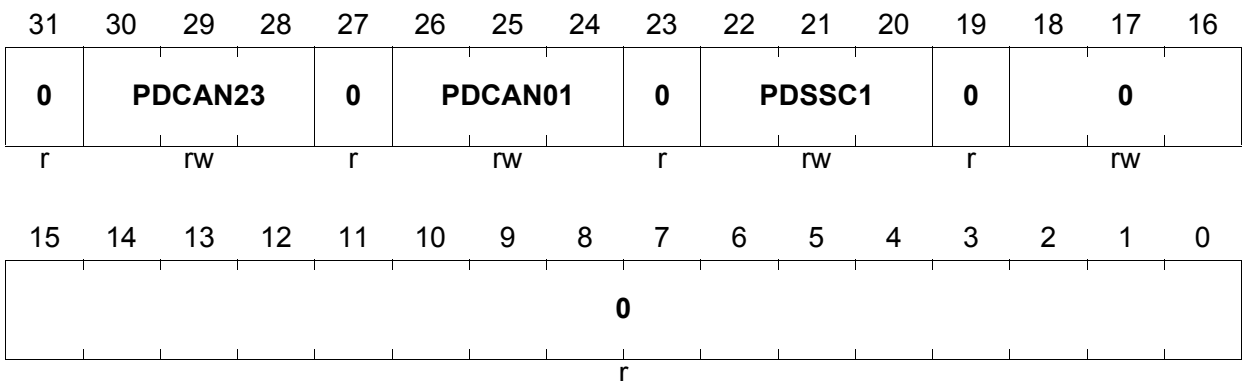
The Port 6 pad driver mode register contains three bit fields that determine the pad driver mode (output driver strength and slew rate) of Port 6 line groups. The Port 6 port lines are all class A2 pads (see also [Figure 10-10](#)).

P6_PDR

Port 6 Pad Driver Mode Register

(40_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PDSSC1	[22:20]	rw	Pad Driver Mode for P6.[7:4] (Class A2 pads; for coding see Page 10-11)
PDCAN01	[26:24]	rw	Pad Driver Mode for P6.[11:8] (Class A2 pads; for coding see Page 10-11)
PDCAN23	[30:28]	rw	Pad Driver Mode for P6.[15:12] (Class A2 pads; for coding see Page 10-11)
0	[18:16]	rw	Reserved Read as 000 _B after reset; returns value that was written.
0	[15:0], 19, 23, 27, 31	r	Reserved Read as 0; should be written with 0.

General Purpose I/O Ports and Peripheral I/O Lines

10.10 Port 7

This section describes the Port 7 functionality in detail.

10.10.1 Port 7 Configuration

Port 7 is an 8-bit bi-directional general-purpose I/O port that can be used for the external trigger input lines REQ[7:4] or for the ADC0/ADC1 external multiplexer control output lines.

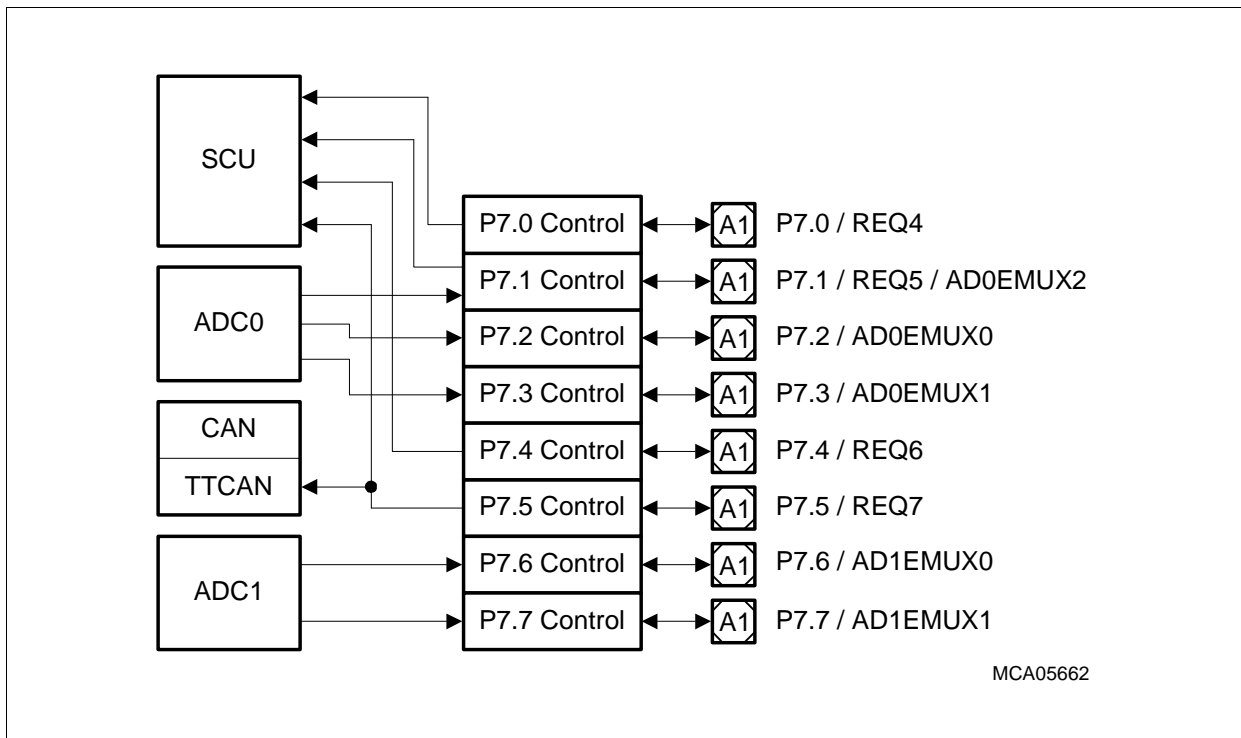


Figure 10-11 Port 7 Configuration Diagram

General Purpose I/O Ports and Peripheral I/O Lines

10.10.2 Port 7 Function Table

Table 10-21 summarizes the I/O control selection functions of each Port 7 line.

Table 10-21 Port 7 Functions

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P7.0	I	General-purpose input	P7_IN.P0	P7_IOCRO.PC0	0XXX _B
		SCU input	REQ4		
	O	General-purpose output	P7_OUT.P0		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B
P7.1	I	General-purpose input	P7_IN.P1	P7_IOCRO.PC1	0XXX _B
		SCU input	REQ5		
	O	General-purpose output	P7_OUT.P1		1X00 _B
		ADC0 output	AD0EMUX2		1X01 _B 1X10 _B 1X11 _B
P7.2	I	General-purpose input	P7_IN.P2	P7_IOCRO.PC2	0XXX _B
		SCU input	REQ6		
	O	General-purpose output	P7_OUT.P2		1X00 _B
		ADC0 output	AD0EMUX0		1X01 _B 1X10 _B 1X11 _B
P7.3	I	General-purpose input	P7_IN.P3	P7_IOCRO.PC3	0XXX _B
		SCU input	REQ7		
	O	General-purpose output	P7_OUT.P3		1X00 _B
		ADC0 output	AD0EMUX1		1X01 _B 1X10 _B 1X11 _B
P7.4	I	General-purpose input	P7_IN.P4	P7_IOCRO4.PC4	0XXX _B
		SCU input	REQ8		
	O	General-purpose output	P7_OUT.P4		1X00 _B
		Reserved ¹⁾	–		1X01 _B 1X10 _B 1X11 _B

General Purpose I/O Ports and Peripheral I/O Lines

Table 10-21 Port 7 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P7.5	I	General-purpose input	P7_IN.P5	P7_IOCR4.PC5	0XXX _B
		SCU input	REQ7		
		TTCAN input ECTT2			
	O	General-purpose output	P7_OUT.P5		1X00 _B
	Reserved ¹⁾	–	1X01 _B 1X10 _B 1X11 _B		
P7.6	I	General-purpose input	P7_IN.P6	P7_IOCR4.PC6	0XXX _B
	O	General-purpose output	P7_OUT.P6		1X00 _B
		ADC1 output	AD1EMUX0		1X01 _B
		Reserved ¹⁾	–		1X10 _B 1X11 _B
P7.7	I	General-purpose input	P7_IN.P7	P7_IOCR4.PC7	0XXX _B
	O	General-purpose output	P7_OUT.P7		1X00 _B
		ADC1 output	AD1EMUX1		1X01 _B
		Reserved ¹⁾	–		1X10 _B 1X11 _B

1) The port I/O control values P7_IOCRx.Py that are assigned to this reserved alternate output control selection should not be used. Otherwise, unpredictable output port line behavior may occur.

General Purpose I/O Ports and Peripheral I/O Lines

10.10.3 Port 7 Registers

The following registers are available on Port 7:

Table 10-22 Port 7 Registers

Register Short Name	Register Long Name	Offset Address	Description see
P7_OUT	Port 7 Output Register	0000 _H	below ¹⁾
P7_OMR	Port 7 Output Modification Register	0004 _H	
P7_IOCRO	Port 7 Input/Output Control Register 0	0010 _H	Page 10-7
P7_IOCRA	Port 7 Input/Output Control Register 4	0014 _H	Page 10-8
P7_IN	Port 7 Input Register	0024 _H	below ¹⁾
P7_PDR	Port 7 Pad Driver Mode Register	0040 _H	Page 10-69 ¹⁾

1) These registers are listed and noted here in the Port 7 section because they differ from the general port register description given in [Section 10.2](#).

Note: The complete address map of Port 7 is described in [Table 18-16](#) on [Page 18-29](#) of this TC1796 System Units (Vol. 1 of 2) User's Manual.

10.10.3.1 Port 7 Output Register

The basic P7_OUT register functionality is described on [Page 10-13](#). Port lines P7.[15:8] are not available. Therefore, the P7_OUT bits P[15:8] should be written with 0, and are always read as 0.

10.10.3.2 Port 7 Output Modification Register

The basic P7_OMR register functionality is described on [Page 10-14](#). Port lines P7.[15:8] are not available. Therefore, the P7_OMR bits PS[15:8] and PR[15:8] are not implemented. These bits should always be written with 0.

10.10.3.3 Port 7 Input Register

The basic P7_IN register functionality is described on [Page 10-17](#). Port lines P7.[15:8] are not available. Therefore, the P7_IN bits P[15:8] are always read as 0.

General Purpose I/O Ports and Peripheral I/O Lines

10.11 Port 8

This section describes the Port 8 functionality in detail.

10.11.1 Port 8 Configuration

Port 8 is an 8-bit bi-directional general-purpose I/O port which can be used for the MLI1 interface lines or for the GPTA0/GPTA1 I/O lines.

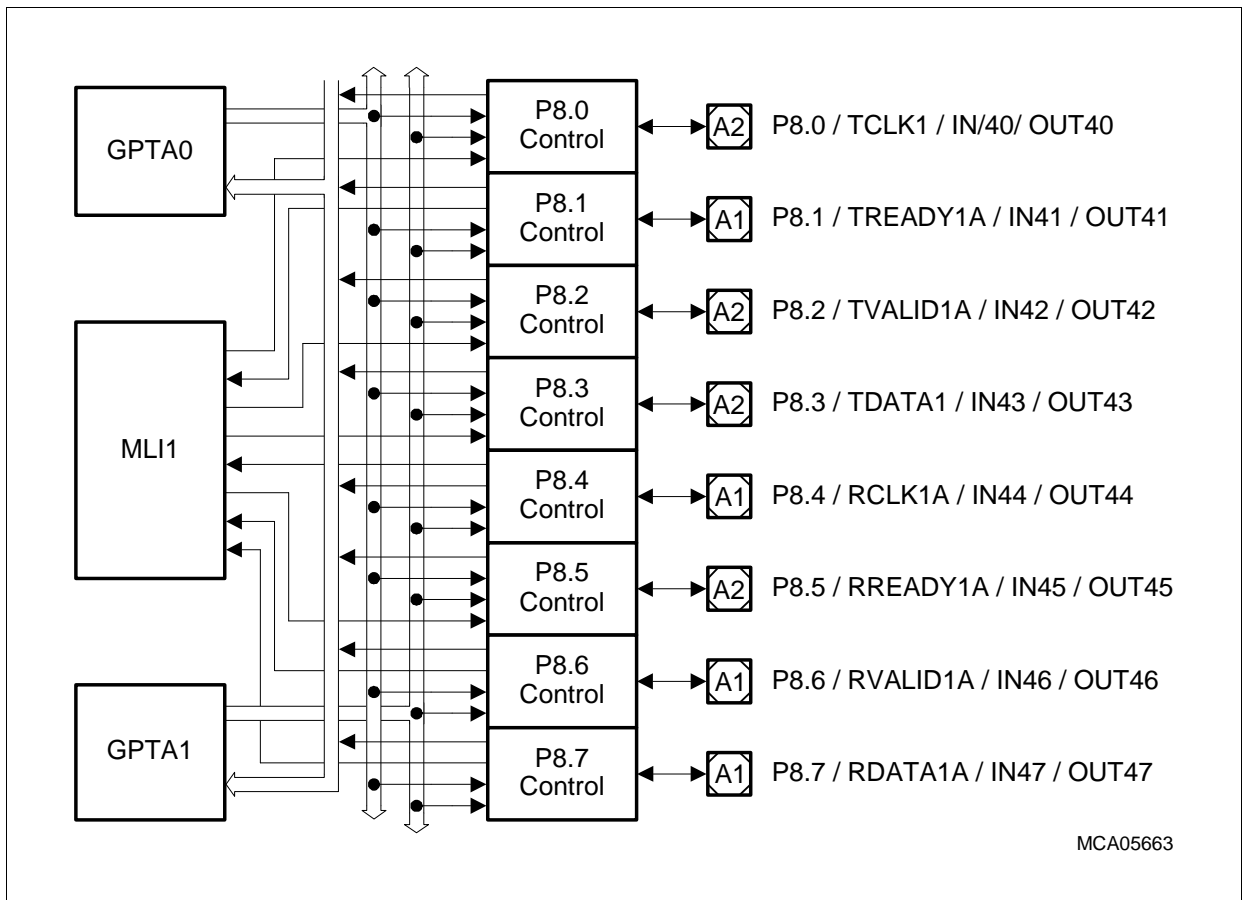


Figure 10-12 Port 8 Configuration Diagram

General Purpose I/O Ports and Peripheral I/O Lines

10.11.2 Port 8 Function Table

Table 10-23 summarizes the I/O control selection functions of each Port 8 line.

Table 10-23 Port 8 Functions

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P8.0	I	General-purpose input	P8_IN.P0	P8_IOCRO.PC0	0XXX _B
		GPTA0/GPTA1 input	IN40		
	O	General-purpose output	P8_OUT.P0		1X00 _B
		GPTA0 output	OUT40		1X01 _B
		GPTA1 output	OUT40		1X10 _B
MLI1 output	TCLK1	1X11 _B			
P8.1	I	General-purpose input	P8_IN.P1	P8_IOCRO.PC1	0XXX _B
		GPTA0/GPTA1 input	IN41		
		MLI1 input	TREADY1A		
	O	General-purpose output	P8_OUT.P1		1X00 _B
		GPTA0 output	OUT41		1X01 _B
		GPTA1 output	OUT41		1X10 _B
		Reserved ¹⁾	–		1X11 _B
P8.2	I	General-purpose input	P8_IN.P2	P8_IOCRO.PC2	0XXX _B
		GPTA0/GPTA1 input	IN42		
	O	General-purpose output	P8_OUT.P2		1X00 _B
		GPTA0 output	OUT42		1X01 _B
		GPTA1 output	OUT42		1X10 _B
MLI1 output	TVALID1A	1X11 _B			
P8.3	I	General-purpose input	P8_IN.P3	P8_IOCRO.PC3	0XXX _B
		GPTA0/GPTA1 input	IN43		
	O	General-purpose output	P8_OUT.P3		1X00 _B
		GPTA0 output	OUT43		1X01 _B
		GPTA1 output	OUT43		1X10 _B
MLI1 output	TDATA1	1X11 _B			

General Purpose I/O Ports and Peripheral I/O Lines

Table 10-23 Port 8 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P8.4	I	General-purpose input	P8_IN.P4	P8_IOCR4.PC4	0XXX _B
		GPTA0/GPTA1 input	IN44		
		MLI1 input	RCLK1A		
	O	General-purpose output	P8_OUT.P4		1X00 _B
		GPTA0 output	OUT44		1X01 _B
		GPTA1 output	OUT44		1X10 _B
		Reserved ¹⁾	–		1X11 _B
P8.5	I	General-purpose input	P8_IN.P5	P8_IOCR4.PC5	0XXX _B
		GPTA0/GPTA1 input	IN45		
	O	General-purpose output	P8_OUT.P5		1X00 _B
		GPTA0 output	OUT45		1X01 _B
		GPTA1 output	OUT45		1X10 _B
		MLI1 output	RREADY1A		1X11 _B
	P8.6	I	General-purpose input		P8_IN.P6
GPTA0/GPTA1 input			IN46		
MLI1 input			RVALID1A		
O		General-purpose output	P8_OUT.P6	1X00 _B	
		GPTA0 output	OUT46	1X01 _B	
		GPTA1 output	OUT46	1X10 _B	
		Reserved ¹⁾	–	1X11 _B	
P8.7	I	General-purpose input	P8_IN.P7	P8_IOCR4.PC7	0XXX _B
		GPTA0/GPTA1 input	IN47		
		MLI1 input	RDATA1A		
	O	General-purpose output	P8_OUT.P7		1X00 _B
		GPTA0 output	OUT47		1X01 _B
		GPTA1 output	OUT47		1X10 _B
		Reserved ¹⁾	–		1X11 _B

1) The port I/O control values P8_IOCRx.Py that are assigned to this reserved alternate output control selection should not be used. Otherwise, unpredictable output port line behavior may occur.

General Purpose I/O Ports and Peripheral I/O Lines

10.11.3 Port 8 Register

The following registers are available on Port 8:

Table 10-24 Port 8 Registers

Register Short Name	Register Long Name	Offset Address	Description see
P8_OUT	Port 8 Output Register	0000 _H	below ¹⁾
P8_OMR	Port 8 Output Modification Register	0004 _H	
P8_IOCRO	Port 8 Input/Output Control Register 0	0010 _H	Page 10-7
P8_IOCRA	Port 8 Input/Output Control Register 4	0014 _H	Page 10-8
P8_IN	Port 8 Input Register	0024 _H	below ¹⁾
P8_PDR	Port 8 Pad Driver Mode Register	0040 _H	Page 10-74 ¹⁾
P8_ESR	Port 8 Emergency Stop Register	0050 _H	below ¹⁾

1) These registers are listed here in the Port 8 section because they differ from the general port register description given in [Section 10.2](#).

Note: The complete address map of Port 8 is described in [Table 18-17](#) on [Page 18-30](#) of this TC1796 System Units (Vol. 1 of 2) User's Manual.

10.11.3.1 Port 8 Output Register

The basic P8_OUT register functionality is described on [Page 10-13](#). Port lines P8.[15:8] are not available. Therefore, the P8_OUT bits P[15:8] should be written with 0 and are always read as 0.

10.11.3.2 Port 8 Output Modification Register

The basic P8_OMR register functionality is described on [Page 10-14](#). Port lines P8.[15:8] are not available. Therefore, the P8_OMR bits PS[15:8] and PR[15:8] are not implemented. These bits should always be written with 0.

10.11.3.3 Port 8 Input Register

The basic P8_IN register functionality is described on [Page 10-17](#). Port lines P8.[15:8] are not available. Therefore, the P8_IN bits P[15:8] are always read as 0.

10.11.3.4 Port 8 Emergency Stop Register

The basic P8_ESR register functionality is described on [Page 10-16](#). At Port 8, only port lines P8.[7:0] are implemented. Therefore, the P8_ESR bits EN[15:8] are not implemented. They are always read as 0 and should be written with 0.

General Purpose I/O Ports and Peripheral I/O Lines

10.11.3.5 Port 8 Pad Driver Mode Register and Pad Classes

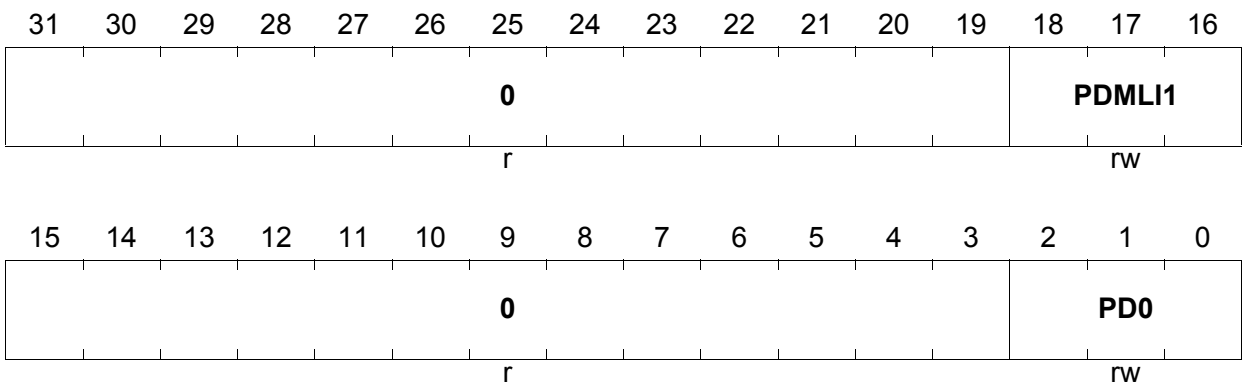
The Port 8 pad driver mode register contains two bit fields that determine the pad driver mode (output driver strength and slew rate) of Port 8 lines and line groups. The Port 8 port lines are assigned to A1 and A2 pad classes (see also [Figure 10-12](#)).

P8_PDR

Port 8 Pad Driver Mode Register

(40_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PD0	[2:0]	rw	Pad Driver Mode for P8.1, P8.4, P8.6, and P8.7 (Class A1 pads; for coding see Page 10-11)
PDMLI1	[18:16]	rw	Pad Driver Mode for P8.0, P8.2, P8.3, and P8.5 (Class A2 pads; for coding see Page 10-11)
0	[15:3], [31:19]	r	Reserved Read as 0; should be written with 0.

General Purpose I/O Ports and Peripheral I/O Lines

10.12 Port 9

This section describes the Port 9 functionality in detail.

10.12.1 Port 9 Configuration

Port 9 is a 9-bit bi-directional general-purpose I/O port which can be used for the MSC0/MSC1 interface output lines or for the GPTA0/GPTA1 I/O lines.

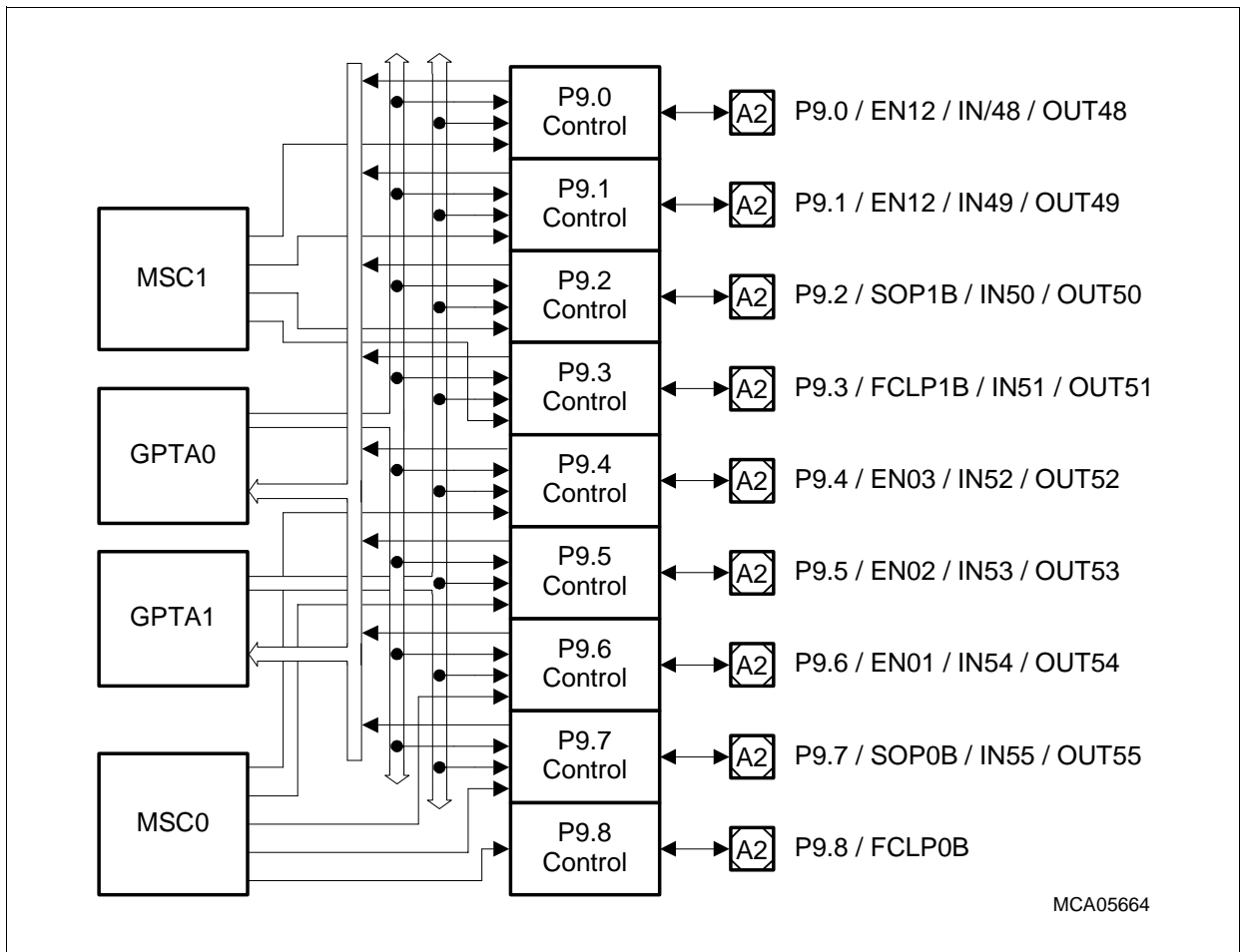


Figure 10-13 Port 9 Configuration Diagram

General Purpose I/O Ports and Peripheral I/O Lines

10.12.2 Port 9 Function Table

Table 10-25 summarizes the I/O control selection functions of each Port 9 line.

Table 10-25 Port 9 Functions

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P9.0	I	General-purpose input	P9_IN.P0	P9_IOCR0.PC0	0XXX _B
		GPTA0/GPTA1 input	IN48		
	O	General-purpose output	P9_OUT.P0		1X00 _B
		GPTA0 output	OUT48		1X01 _B
		GPTA1 output	OUT48		1X10 _B
MSC1 output	EN12	1X11 _B			
P9.1	I	General-purpose input	P9_IN.P1	P9_IOCR0.PC1	0XXX _B
		GPTA0/GPTA1 input	IN49		
	O	General-purpose output	P9_OUT.P1		1X00 _B
		GPTA0 output	OUT49		1X01 _B
		GPTA1 output	OUT49		1X10 _B
MSC1 output	EN11	1X11 _B			
P9.2	I	General-purpose input	P9_IN.P2	P9_IOCR0.PC2	0XXX _B
		GPTA0/GPTA1 input	IN50		
	O	General-purpose output	P9_OUT.P2		1X00 _B
		GPTA0 output	OUT50		1X01 _B
		GPTA1 output	OUT50		1X10 _B
MSC1 output	SOP1B	1X11 _B			
P9.3	I	General-purpose input	P9_IN.P3	P9_IOCR0.PC3	0XXX _B
		GPTA0/GPTA1 input	IN51		
	O	General-purpose output	P9_OUT.P3		1X00 _B
		GPTA0 output	OUT51		1X01 _B
		GPTA1 output	OUT51		1X10 _B
MSC1 output	FCLP1B	1X11 _B			

General Purpose I/O Ports and Peripheral I/O Lines

Table 10-25 Port 9 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P9.4	I	General-purpose input	P9_IN.P4	P9_IOCR4.PC4	0XXX _B
		GPTA0/GPTA1 input	IN52		
	O	General-purpose output	P9_OUT.P4		1X00 _B
		GPTA0 output	OUT52		1X01 _B
		GPTA1 output	OUT52		1X10 _B
MSC0 output	EN03	1X11 _B			
P9.5	I	General-purpose input	P9_IN.P5	P9_IOCR4.PC5	0XXX _B
		GPTA0/GPTA1 input	IN53		
	O	General-purpose output	P9_OUT.P5		1X00 _B
		GPTA0 output	OUT53		1X01 _B
		GPTA1 output	OUT53		1X10 _B
MSC0 output	EN02	1X11 _B			
P9.6	I	General-purpose input	P9_IN.P6	P9_IOCR4.PC6	0XXX _B
		GPTA0/GPTA1 input	IN54		
	O	General-purpose output	P9_OUT.P6		1X00 _B
		GPTA0 output	OUT54		1X01 _B
		GPTA1 output	OUT54		1X10 _B
MSC0 output	EN01	1X11 _B			
P9.7	I	General-purpose input	P9_IN.P7	P9_IOCR4.PC7	0XXX _B
		GPTA0/GPTA1 input	IN55		
	O	General-purpose output	P9_OUT.P7		1X00 _B
		GPTA0 output	OUT55		1X01 _B
		GPTA1 output	OUT55		1X10 _B
MSC0 output	SOP0B	1X11 _B			
P9.8	I	General-purpose input	P9_IN.P8	P9_IOCR8.PC8	0XXX _B
	O	General-purpose output	P9_OUT.P8		1X00 _B
		MSC0 output	FCLP0B		1X01 _B 1X10 _B 1X11 _B

General Purpose I/O Ports and Peripheral I/O Lines

10.12.3 Port 9 Register

The following registers are available on Port 8:

Table 10-26 Port 9 Registers

Register Short Name	Register Long Name	Offset Address	Description see
P9_OUT	Port 9 Output Register	0000 _H	below ¹⁾
P9_OMR	Port 9 Output Modification Register	0004 _H	
P9_IOCRO	Port 9 Input/Output Control Register 0	0010 _H	Page 10-7
P9_IOCRA	Port 9 Input/Output Control Register 4	0014 _H	Page 10-8
P9_IOCRA8	Port 9 Input/Output Control Register 8	0018 _H	Page 10-79 ¹⁾
P9_IN	Port 9 Input Register	0024 _H	Page 10-79 ¹⁾
P9_PDR	Port 9 Pad Driver Mode Register	0040 _H	Page 10-80 ¹⁾
P9_ESR	Port 9 Emergency Stop Register	0050 _H	Page 10-80 ¹⁾

1) These registers are listed and noted here in the Port 9 section because they differ from the general port register description given in [Section 10.2](#).

Note: The complete address map of Port 9 is described in [Table 18-18](#) on [Page 18-31](#) of this TC1796 System Units (Vol. 1 of 2) User's Manual.

10.12.3.1 Port 9 Output Register

The basic P9_OUT register functionality is described on [Page 10-13](#). Port lines P9.[15:9] are not available. Therefore, the P9_OUT bits P[15:9] should be written with 0 and are always read as 0.

10.12.3.2 Port 9 Output Modification Register

The basic P9_OMR register functionality is described on [Page 10-14](#). Port lines P9.[15:9] are not available. Therefore, the P9_OMR bits PS[15:9] and PR[15:9] are not implemented. These bits should always be written with 0.

General Purpose I/O Ports and Peripheral I/O Lines

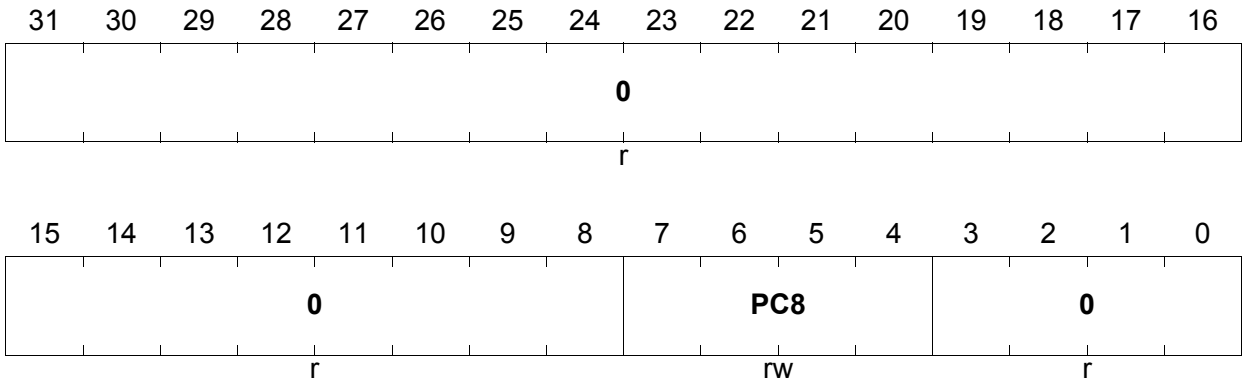
10.12.3.3 Port 9 Input/Output Control Register 8

P9_IOC8

Port 9 Input/Output Control Register 8

(14_H)

Reset Value: 0000 0020_H



Field	Bits	Type	Description
PC8	[7:4]	rw	Port Control for P9.8 This bit field determines the P9.8 functionality.
0	[3:0], [31:8]	r	Reserved Read as 0; should be written with 0.

10.12.3.4 Port 9 Input Register

The basic P9_IN register functionality is described on [Page 10-17](#). Port lines P9.[15:9] are not available. Therefore, the P9_IN bits P[15:9] are always read as 0.

10.12.3.5 Port 9 Emergency Stop Register

The basic P9_ESR register functionality is described on [Page 10-16](#). At Port 9, only port lines P9.[8:0] are implemented. Therefore, the P9_ESR bits EN[15:9] are not implemented. They are always read as 0 and should be written with 0.

General Purpose I/O Ports and Peripheral I/O Lines

10.12.3.6 Port 9 Pad Driver Mode Register and Pad Classes

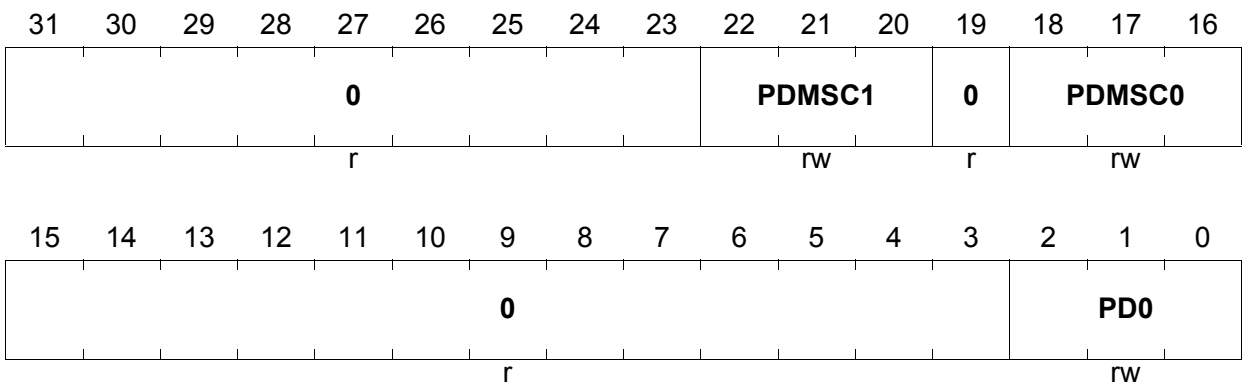
The Port 9 pad driver mode register contains three bit fields that determine the pad driver mode (output driver strength and slew rate) of Port 9 lines. The Port 9 port lines are all class A2 pads (see also [Figure 10-13](#)).

P9_PDR

Port 9 Pad Driver Mode Register

(40_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PD0	[2:0]	rw	Pad Driver Mode for MSC1/GPIO P9.8 (Class A2 pads; for coding see Page 10-11)
PDMSC0	[18:16]	rw	Pad Driver Mode for GPTA/MSC0 Outputs P9.[7:4] (Class A2 pads; for coding see Page 10-11)
PDMSC1	[22:20]	rw	Pad Driver Mode for GPTA/MSC1 Outputs P9.[3:0] (Class A2 pads; for coding see Page 10-11)
0	[15:3], 19, [31:23]	r	Reserved Read as 0; should be written with 0.

General Purpose I/O Ports and Peripheral I/O Lines

10.13 Port 10 (Hardware Select Inputs)

This section describes the Port 10 functionality in detail.

10.13.1 Port 10 Configuration

Port 10 is a 4-bit input-only port. Its pins are used for hardware configuration as HWCFG[3:0] lines. The logic levels of the four Port 10 lines are latched when the hardware reset goes inactive (at the rising edge of $\overline{\text{HDRST}}$) and can be checked by software by reading bit field HWCFG in the reset status register RTS_SR. The four input lines of Port 10 have weak pull-up devices connected (always active).

Inside the TC1796, the HWCFG[3:0] lines are used for boot configuration selection. Input P10.1 (HWCFG1) operates as emergency stop input for the GPTA modules. The emergency stop control logic is described in detail in the “System Control Unit” chapter (Page 5-57) of this User’s Manual.

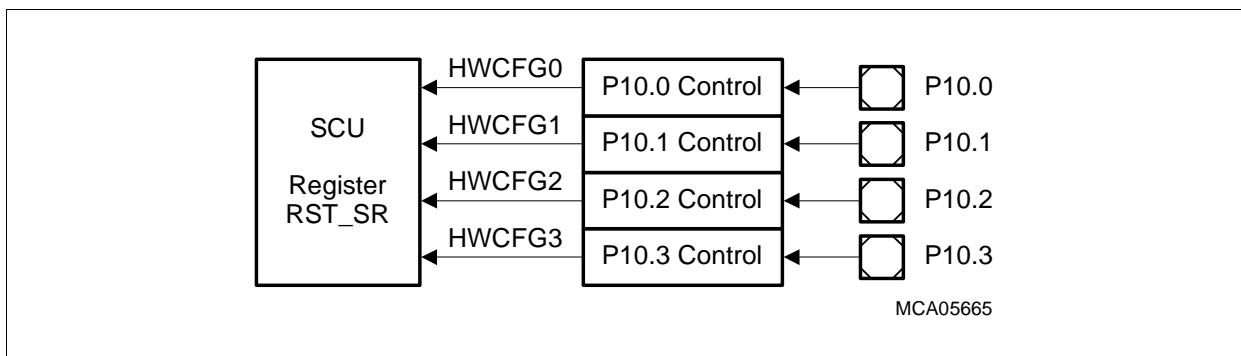


Figure 10-14 Port 10 Configuration Diagram

General Purpose I/O Ports and Peripheral I/O Lines

10.13.2 Port 10 Registers

The following registers are available on Port 10:

Table 10-27 Port 10 Registers

Register Short Name	Register Long Name	Offset Address ¹⁾	Description see
P10_IN	Port 10 Input Register	0024 _H	Page 10-82 ²⁾
RST_SR	Reset Status Register	–	Page 4-3

- 1) The absolute addresses are calculated by adding the offset address to the module base address (see [Table 10-2](#))
- 2) This register is listed here in the Port 10 section because it differs from the general port register description given in [Section 10.2](#).

Note: The complete address map of Port 10 is described in [Table 18-19](#) on [Page 18-32](#) of this TC1796 System Units (Vol. 1 of 2) User's Manual.

Note: Bit field HWCFG in register RST_SR contains the latched logic levels of the Port 10 inputs that were detected at the last low-to-high transition of \overline{HDRST} . Register P10_IN makes it possible to read the actual logic levels of the Port 10 inputs.

10.13.2.1 Port 10 Input Register

The basic P10_IN register functionality is described on [Page 10-17](#). Port lines P10.[15:4] are not available. Therefore, the P10_IN bits P[3:0] are always read as 0.

General Purpose I/O Ports and Peripheral I/O Lines

10.14 Dedicated Peripheral I/O Lines

10.14.1 Dedicated I/O Lines for SSC0 and SSC1

In the TC1796 the Synchronous Serial interface SSC0 is directly connected to dedicated pins. Bits SSC0_CON.EN (signal “SSC Enabled”) enables/disables these dedicated pins. After a reset, SSC0_CON.EN is 0 and the dedicated SSC0 I/O pins are tri-stated. With SSC0_CON.EN = 1, the dedicated SSC0 I/O pins are enabled and the state of SSC0_CON.MS (signal “M/S Enabled”) determines whether it will be used as input or output. When the dedicated SSC0 I/O pins are enabled (SSC0_CON.EN = 1), changing SSC0_CON.MS directly affects the I/O direction of the three SSC0 I/O pins.

Two SLSOx outputs of SSC0, SLSO0 and SLSO1, are connected to dedicated output pins. When enabled (SSC0_CON.EN = 1), SLSO0 and SLSO1 are dedicated output pins. SLSO0 and SLSO1 are tri-stated with SSC0_CON.EN = 0 (default after reset).

When a dedicated SSC0 I/O line is used as outputs, its pad driver is controlled by two bits of the SCU_CON register (located in the SCU). SCU_CON.SSC0PDR controls the output drivers of MTRS0, MRST0, and SCLK0 while SCU_CON.SLSPDR controls the output drivers of SLSO0 and SLSO1. Two driver strengths can be selected with each bit. (see also SCU chapter [Page 5-66](#)).

Table 10-28 SSC0 Dedicated I/O Line Selection and Setup

Dedicated SSC0 Port Line	Input/Output	SSC0 Mode	Input/Output Selection	Enable Control	Pad Driver Control
Dedicated SSC0 I/O Lines					
MTRS0	Input	Slave	SSC0_CON.MS = 0	SSC0_CON.EN = 1	–
	Output	Master	SSC0_CON.MS = 1		SCU_CON.SSC0PDR ¹⁾
MRST0	Input	Master	SSC0_CON.MS = 1		–
	Output	Slave	SSC0_CON.MS = 0		SCU_CON.SSC0PDR ¹⁾
SCLK0	Input	Slave	SSC0_CON.MS = 0		–
	Output	Master	SSC0_CON.MS = 1		SCU_CON.SSC0PDR ¹⁾

General Purpose I/O Ports and Peripheral I/O Lines

Table 10-28 SSC0 Dedicated I/O Line Selection and Setup (cont'd)

Dedicated SSC0 Port Line	Input/Output	SSC0 Mode	Input/Output Selection	Enable Control	Pad Driver Control
MTSR0 MRST0 SCLK0	Tri-Stated	–	–	SSC0_CON.EN = 0	–
SLSI0	Input	–	–	–	–

Dedicated SSC0 Slave Select Outputs

SLSO0 SLSO1	Output	Master	–	SSC0_CON.EN = 1	SCU_CON.SLSPDR ²⁾
SLSO0 SLSO1	Tri-Stated			SSC0_CON.EN = 0	–

1) SSC0PDR = 0: strong driver / sharp edge selected; SSC0PDR = 1: strong driver / soft edge selected;

2) SLSPDR = 0: strong driver / sharp edge selected; SLSPDR = 1: strong driver / soft edge selected;

Note: The dedicated SSC0 output pins are class A2 pins (high-speed 3.3 V LVTTTL outputs). They do not provide open-drain and pull-up/pull-down capability!

General Purpose I/O Ports and Peripheral I/O Lines

10.14.2 LVDS Outputs of MSC0 and MSC1

The clock and data output lines of MSC0 and MSC1 are connected to dedicated differential output drivers, each of which has positive and negative signal polarity. These types of 3.3V LVDS pads are assigned as class C pads.

The LVDS outputs are controlled by a bit LDEN (LVDS Driver Enable) that is located in register SCU_CON. With LDEN = 0, all LVDS drivers are disabled and in power-down mode. With LDEN = 1 all LVDS drivers are enabled for operation.

Table 10-29 LVDS Outputs of MSC0/MSC1

Signal/Pin Short Name	Description
FCLP0A	MSC0 differential driver clock output positive A
FCLN0	MSC0 differential driver clock output negative
SOP0A	MSC0 differential driver serial data output positive A
SON0	MSC0 differential driver serial data output negative
FCLP1A	MSC1 differential driver clock output positive A
FCLN1	MSC1 differential driver clock output negative
SOP1A	MSC1 differential driver serial data output positive A
SON1	MSC1 differential driver serial data output negative

11 Peripheral Control Processor (PCP)

This chapter describes the Peripheral Control Processor (PCP), its architecture, programming model, registers, and instructions. The TC1796's PCP is an enhanced version 2 (PCP2) of the TC1775's PCP peripheral control processor.

[Section 11.1](#) to [Section 11.13](#) of this chapter describe the TC1796 PCP in general. TC1796 implementation-specific details are described in [Section 11.14](#).

11.1 Peripheral Control Processor Overview

The PCP in the TC1796 performs tasks that would normally be performed by the combination of a DMA controller and its supporting CPU interrupt service routines in a traditional computer system. It could easily be considered as the host processor's first line of defence as an interrupt-handling engine. The PCP can unload the CPU from having to service time-critical interrupts. This provides many benefits, including:

- Avoiding large interrupt-driven task context-switching latencies in the host processor
- Reducing the cost of interrupts in terms of processor register and memory overhead
- Improving the responsiveness of interrupt service routines to data-capture and data-transfer operations
- Easing the implementation of multitasking operating systems.

The PCP has an architecture that efficiently supports DMA-type transactions to and from arbitrary devices and memory addresses within the TC1796 and also has reasonable stand-alone computational capabilities.

The TC1796 contains an improved version of the TC1775's PCP with the following enhancements:

- Optimized context switching
- Support for nested interrupts
- Enhanced instruction set
- Enhanced instruction execution speed
- Enhanced interrupt queueing

11.2 PCP Architecture

The PCP is made up of several modular blocks as follows. Please refer to [Figure 11-1](#).

- PCP Processor Core
- Code Memory (CMEM) with parity protection
- Parameter Memory (PRAM) with parity protection
- PCP Interrupt Control Unit (PICU)
- PCP Service Request Nodes (PSRN)
- System bus interface to the Flexible Peripheral Interface (FPI Bus)

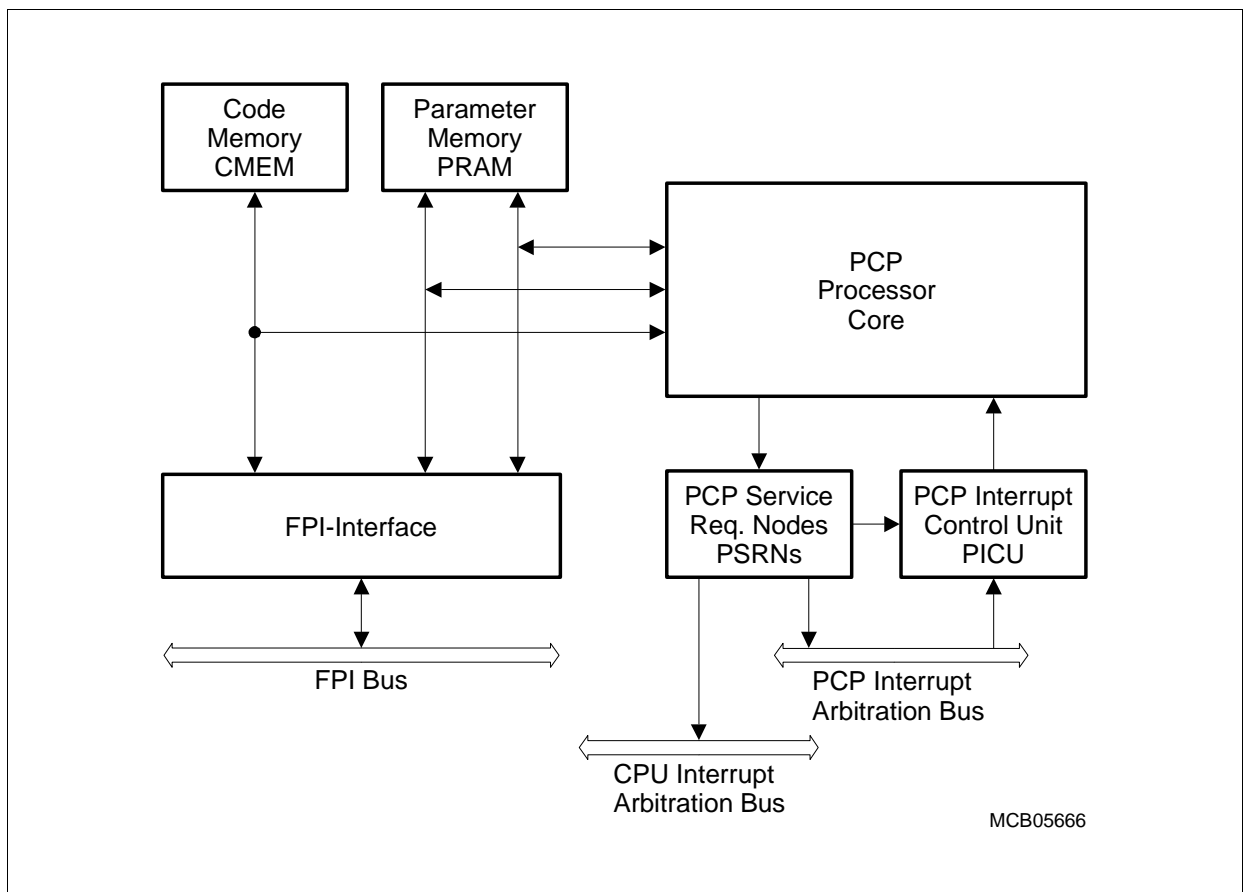


Figure 11-1 PCP Block Diagram

11.2.1 PCP Processor

The PCP Processor is the main engine of the PCP. It contains an instruction pipeline, a set of GPRs, an arithmetic/logic unit, as well as control and status registers and logic. Its instruction set is optimized especially for the tasks it has to perform. [Table 11-1](#) provides an overview of the PCP instruction set.

The PCP Processor Core receives service requests from peripherals or other modules in the system via its PCP Interrupt Control Unit (PICU), and executes a channel program (see [Page 11-6](#)) selected via the priority number of each service request. It first restores the channel program's context from the PRAM and then starts to execute the channel program's instructions stored in the Code Memory (CMEM). Upon an exit condition, it terminates the channel program and saves its context into PRAM. It is then ready to receive the next service request.

The PCP Processor Core is capable of suspending execution of a channel program on receipt of a service request with a higher priority than the channel currently being executed. The Core will automatically resume processing of the original channel program once the higher-priority request (or requests) has been processed. A channel that has been suspended in this way is termed a "Suspended Channel".

The PCP is fully interrupt-driven, meaning it is only activated through service requests; there is no main program running in the background as with a conventional processor.

Table 11-1 PCP Instruction Set Overview

Instruction Group	Description
DMA Primitives	Efficient DMA channel implementation
Load/Store	Transfer data between PRAM or FPI memory and the GPRs, as well as move or exchange values between registers
Arithmetic	Add, subtract, compare and complement
Divide/Multiply	Divide and multiply
Logical	And, Or, Exclusive Or, Negate
Shift	Shift right or left, rotate right or left, prioritize
Bit Manipulation	Set, clear, insert, and test bits
Flow Control	Jump conditionally, jump long, exit
Miscellaneous	No operation, Debug

11.2.2 PCP Code Memory

The Code Memory (CMEM) of the PCP holds the channel programs, consisting of PCP instructions. All instructions of the PCP are 16 bits long; thus, the PCP accesses its CMEM in 16-bit (half-word) quantities. With the 16-bit Program Counter (PC) of the PCP, a maximum of 64 K 16-bit instructions can be addressed. This results in a maximum size of the PCP code memory of 128 Kbytes. The actual type (Flash, ROM, SRAM, etc.) and size of the code memory is implementation-specific; see [Page 11-125](#) for the implemented type and size of the code memory in the TC1796.

The PCP CMEM is viewed from the FPI Bus as a 32-bit wide memory, that must be accessed with 32-bit (word) accesses, and is addressed with byte addresses. Thus, care has to be taken when calculating PCP instruction FPI addresses. See [Page 11-48](#) for details.

Note: The PCP has a “Harvard” architecture and therefore cannot directly access the CMEM other than reading instructions from it. It is recommended that the PCP should not access CMEM via the FPI Bus.

11.2.3 PCP Parameter RAM

The PCP Parameter RAM (PRAM) is the local holding place for each channel program’s context, and for general data storage. It is also an area that the PCP and the host processor or other FPI Bus masters can use to communicate and share data.

While a portion of the PRAM is always implicitly used for the Context Save Areas (CSAs) of the channel programs, the remaining area can be used for channel-specific or general data storage. A programmable 8-bit Data Pointer (DPTR), concatenated with a 6-bit offset, is provided for arbitrary access to the PRAM. The effective address is a 14-bit word address, allowing a PRAM size of up to 64 Kbytes. The actual type (SRAM, DRAM, etc.) and size of the parameter RAM is implementation-specific; see [Page 11-125](#) for the implemented size of the PRAM in this derivative.

Both the PCP and FPI Bus masters address the PRAM as 32-bit words. There is no concept of half-word or byte accesses to PRAM. FPI Bus masters must, however, use byte addresses in order to access the PRAM. As for the CMEM, care has to be taken when calculating PRAM FPI addresses. See [Page 11-48](#) for details.

11.2.4 FPI Bus Interface

The PCP can access all peripheral units on the FPI Bus and other resources through the FPI Bus interface. The PCP can become an FPI Bus slave, so that other FPI Bus master may access CMEM and PRAM and the control and status registers in the PCP.

The CMEM and PRAM blocks are visible to FPI Bus masters as a block of memory on the FPI Bus. If an FPI Bus master accesses CMEM or PRAM concurrently with the PCP, the external FPI Bus master is given precedence over the PCP to avoid deadlocks. The PCP access is stalled for several cycles until the FPI Bus master has completed its

Peripheral Control Processor (PCP)

access. If an FPI Bus master performs an atomic read-modify-write access to a PCP memory block, any concurrent PCP access to that block is stalled for the duration of the atomic operation.

11.2.5 PCP Interrupt Control Unit and Service Request Nodes

The PCP is activated in response to an interrupt request programmed for PCP service in one of the Service Request Nodes (SRNs) of the system (nodes associated with a peripheral, the CPU, external interrupts, etc.). The PCP Interrupt Control Unit (PICU) determines the request with the currently highest priority and routes the request together with its priority number to the PCP Processor Core. It also acknowledges the requesting source when the PCP starts the service of this interrupt.

The PCP itself can generate service requests to either the CPU or itself through a number of PCP Service Request Nodes (PSRNs). The PSRNs are also used to store all information required by the PCP Processor Core to allow the later restart of a channel program when it is suspended in favour of a higher-priority service request. Please refer to [Page 11-33](#) for more detailed information on the operation of these nodes.

11.3 PCP Programming Model

The PCP programming model can be viewed as a set of autonomous programs, or tasks, called channel programs, that share the processing resources of the PCP. channel programs may be short and simple, or very complex; but they can coexist persistently within the PCP.

From a programming point of view, the individual parts of a channel program are its instruction sequence in the CMEM and its context in the PRAM. It uses the instruction set and the GPRs (R0 - R7) of the PCP processor core to perform the necessary operations, and to communicate with the various resources of the on-chip and off-chip system depending on its task in the application.

These parts of the programming model are discussed in the following sections (with the obvious exception of the system environment outside of the scope of the PCP).

11.3.1 General Purpose Register Set of the PCP

The program-accessible register file of the PCP is composed of eight 32-bit General Purpose Registers (GPRs). These registers are all accessible by PCP programs directly as part of the PCP instruction set. Source and destination registers must be specified in most instructions. These registers are referenced to in this document as Rn or R[n], where n is in the range 0 to 7.

Table 11-2 Directly Accessible Registers

Register	Implicit Use	Description
R0	Accumulator	Implicit target for some arithmetic and logical instructions
R1	–	32-bit general-use register
R2	Return Address	32-bit general-use register
R3	–	32-bit general-use register
R4	SRC (Source)	Source Pointer for COPY instruction
R5	DST (Destination)	Destination Pointer for COPY instruction
R6	CPPN/SRPN/TOS/CNT1	CNT1: Transfer Count for COPY TOS: Type-of-Service SRPN: 8-bit field used for posting interrupt on EXIT instruction CPPN: Current PCP Priority Number
R7	DPTR/Flags	PRAM Data Pointer (DPTR) and Status Flags

Peripheral Control Processor (PCP)

R7 is the only one of the eight registers that may not be used as a full GPR. The most significant 16 bits of R7 may not be written, and will always read back as 0. However, no error will occur when writing to the most significant 16 bits.

Note: The GPRs of the PCP are not memory-mapped into the overall address space. They can only be directly accessed through PCP instructions. The contents of all or some of the registers are part of a channel program's context stored in the PRAM between executions of the channel program. This context is then accessible from outside the PCP.

11.3.1.1 Register R0

R0 is used as an implicit operand destination for some instructions. These are detailed in the individual instruction descriptions.

11.3.1.2 Registers R1, R2, and R3

R1, R2, and R3 are general-use registers. It is recommended that, by convention, R2 should be used as a return address register when call and return program structures are used.

11.3.1.3 Registers R4 and R5

Registers R4 and R5 are also general-use registers. However, the COPY instruction implicitly uses R4 and R5 as full 32-bit address pointers (R4 is used as the source address and R5 as the destination address). As the COPY instruction uses these registers to maintain the address pointers, either or both R4 and R5 values may or may not be modified by the COPY instruction, depending on the options used in the instruction.

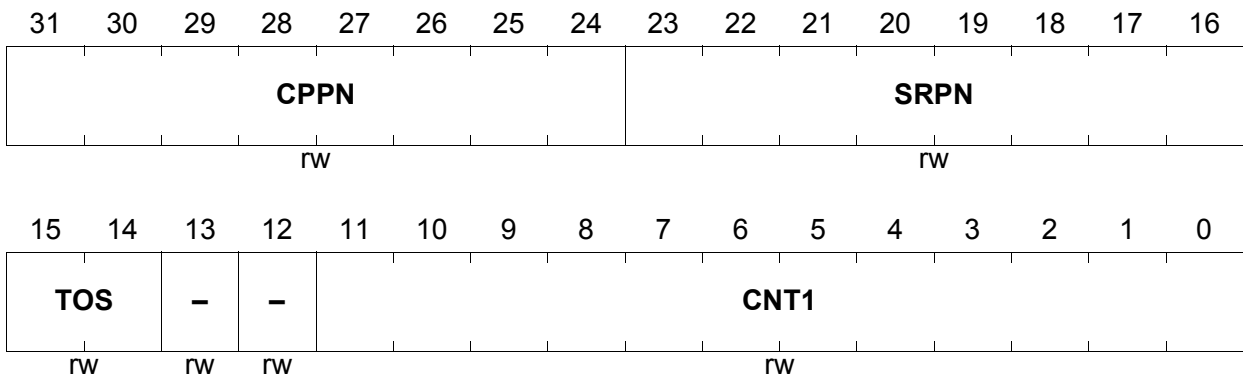
Peripheral Control Processor (PCP)

11.3.1.4 Register R6

Register R6 may also be used as a general-use register. Again however, there are some instructions that use fields within R6. If the COPY or EXIT instructions are used, then the field R6.CNT1 can optionally be used implicitly as a counter. If an EXIT instruction is used that causes an interrupt, R6.SRPN and R6.TOS must be configured properly prior to execution of the EXIT. If interrupt priority management is used, then R6.CPPN must be set to the priority level at which the channel shall run at its next invocation, before the EXIT is executed. The fields for R6 are shown below.

PCP Register R6

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CNT1	[11:0]	rw	General-use/Outer Loop count for COPY Instruction or EXIT Instruction
TOS	[15:14]	rw	General-use/Type-of-Service for EXIT Interrupt Upper bit of TOS is always forced to 0 when transferred into the PCP SRNs, regardless of the value specified in R6[15].
SRPN	[23:16]	rw	General-use/Service Request Priority Number for EXIT Interrupt
CPPN	[31:24]	rw	General-use/PCP Priority Number Posted to PICU
-	13, 12	rw	General-use

11.3.1.5 Register R7

Register R7 is an exception with respect to the other registers in that not all bits within the register can be written, and the implicit use of the remaining bits virtually excludes the use of R7 as a GPR. R7 serves purposes similar to those of a Program Status Word found in traditional processors.

R7 holds the flag bits, a channel enable/disable control bit, and the PRAM Data Pointer (DPTR). The upper 16 bits of R7 are reserved.

Most instructions of the PCP update the flags (CN1Z, V, C, N, Z) in R7 according to the result of their operation. See [Table 11-14](#) on [Page 11-107](#) for details on the flag updates of the individual instructions. The values of the flag bits in R7 maintain their state until another instruction that updates their state is executed.

Note: Implicit updates to the flags caused by instruction take precedence over any bits that are explicitly moved to R7. For example, if a MOV instruction is used to place 0000FF07_H in R7, then the bit positions for the C (carry), Z (zero) and N (negative) flags are being written with 1. The MOV instruction, however, implicitly updates the Z and N flag bits in R7 as a result of its operation. Because the number is not negative, and not zero, it will update the Z and N flags to 0. As a result, the value left in R7 after the MOV is complete will be 0000FF04_H (i.e C = 1, Z = 0, N = 0). It is recommended that only SET and CLR instructions should be used to explicitly modify flags in R7.

The Data Pointer, R7.DPTR, is the means of accessing PRAM variables programmatically. It points to a 64-word PRAM segment that may be addressed by instructions that can use the PRAM for source or destination operands (xx.P and xx.PI instructions). The 8 bits of the DPTR are concatenated with a 6-bit offset value (either specified in the instruction as an immediate value or contained in one of the registers) to give a 14 bit (word) address. A program is able to update the DPTR value dynamically, in order to index more than 64 words of PRAM.

Note: Care must be taken when updating R7.DPTR to ensure that other bits within R7 (e.g. R7.CEN) are not inadvertently corrupted.

The channel enable control bit, R7.CEN, allows the enabling or disabling of specific channel programs. If an interrupt request is received for a channel that is disabled, an error exit is forced, and an error interrupt to the CPU is activated.

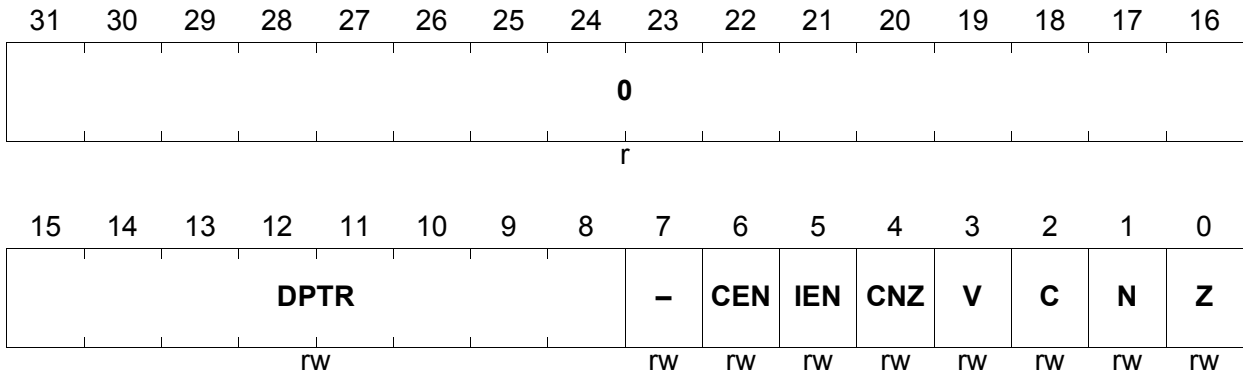
The interrupt enable control bit, R7.IEN, allows the enabling or disabling of channel interruption on a channel to channel basis. When R7.IEN is 0 the channel will continue execution regardless of the priority of any new service requests. When R7.IEN is '1' and conditions allow, the channel will be suspended on receipt of a higher-priority service request.

Note: See [Page 11-120](#) regarding the use of R7.IEN.

Peripheral Control Processor (PCP)

PCP Register R7

Reset Value: 0000 0000_H



Field	Bits	Type	Description
Z	0	rw	Zero
N	1	rw	Negative
C	2	rw	Carry
V	3	rw	Overflow
CNZ	4	rw	Outer Loop Counter 1 Zero Flag
IEN	5	rw	Interrupt Enable 0 _B Channel is not interruptible 1 _B Channel can be suspended in favor of a higher-priority service request
CEN	6	rw	Channel Enable Control Bit
DPTR	[15:8]	rw	Data Pointer Segment Address for PRAM accesses
-	7	rw	Reserved ; should always be written with 0.
0	[31:16]	r	Reserved ; read as 0; should be written with 0.

11.3.2 Contexts and Context Models

After initialization, the instruction sequence of a PCP channel program is permanently stored (i.e. usually at least as long as the application is running) in the CMEM, and data parameters are held in the PRAM. These will remain stored regardless of whether a particular channel program is currently idle or is executing (although, of course, the value of data variables in the PRAM might be modified by the PCP or external FPI Bus masters). The contents of the GPRs of the PCP (used as address pointers, data variables, intermediate results, etc.) however, are usually only valid for a given channel program as long as it is executing. If another channel program is executed, the channel program will re-use the GPRs according to its needs.

Thus, the state of the GPRs of a channel program (termed the “Context” of the channel) needs to be preserved while a channel program is not being executed. The content of the registers needs to be saved when execution of a channel program finishes, and needs to be restored before execution starts again.

The PCP implements automatic handling of these context save and restore operations. On termination of a channel program, the state of all or some of the GPRs is automatically copied to a defined area in the PRAM (Context Save). If the same channel program is re-activated, the contents of the registers are restored by copying the values from the same defined PRAM area into the appropriate registers (Context Restore).

The defined area in the PRAM for the context save and restore operations is called the CSA. Each channel program has its own individual, predefined region in the CSA. When a service request is accepted by the PCP, the service request priority number (SRPN) associated with the request is used to select the channel program and its respective CSA region.

11.3.2.1 Context Models

A Context Model is a means of selecting whether some or all of the registers are saved and restored when a context switch occurs. In order to serve different application needs in terms of PRAM space usage, the PCP offers a choice between three different Context Models:

- Full Context Model: Eight Registers (8×32 -bit words) are saved/restored per channel.
- Small Context Model: Four Registers (4×32 -bit words) are saved/restored per channel.
- Minimum Context Model: Two Registers (2×32 -bit words) are saved/restored.

As illustrated in [Figure 11-2](#), the contents of R0 through R7 constitute the Full Context of a channel program. A Small Context consists of R4 through R7. Use of the Small Context Model allows for correct operation of DMA channels, as well as channels which are not required to save large amounts of data in their contexts between invocations. A Minimum Context saves and restores only R6 and R7.

Peripheral Control Processor (PCP)

To distinguish the actual register contents from the copies stored in the PRAM context regions, the term CRx is used throughout the rest of this document to refer to the register values in the context regions. Registers R6 and R7 are always handled in a special way during context save and restore operations, this is described in detail in [Page 11-17](#).

The Context Model is selected via a bit field PCP_CS.CS, this is a global setting (i.e. the selected Context Model is used for all channels). Once a Context Model has been selected (during PCP configuration) and the PCP has been started the PCP must continue to use that Context Model. Attempting to change the Context Model in use during PCP operation will lead to invalid context restore operations which will in turn lead to invalid PCP operation.

In the case of Small and Minimum Context Models, the unsaved and unrestored registers (shaded in [Figure 11-2](#)) can be thought of as global registers that any channel program can use or change, or reference as constants – for example as base address pointers (see [Page 11-114](#) for more detail).

Note: Special care must be taken when using Minimum or Small Context Model with nested interrupts (see [Page 11-119](#)).

Peripheral Control Processor (PCP)

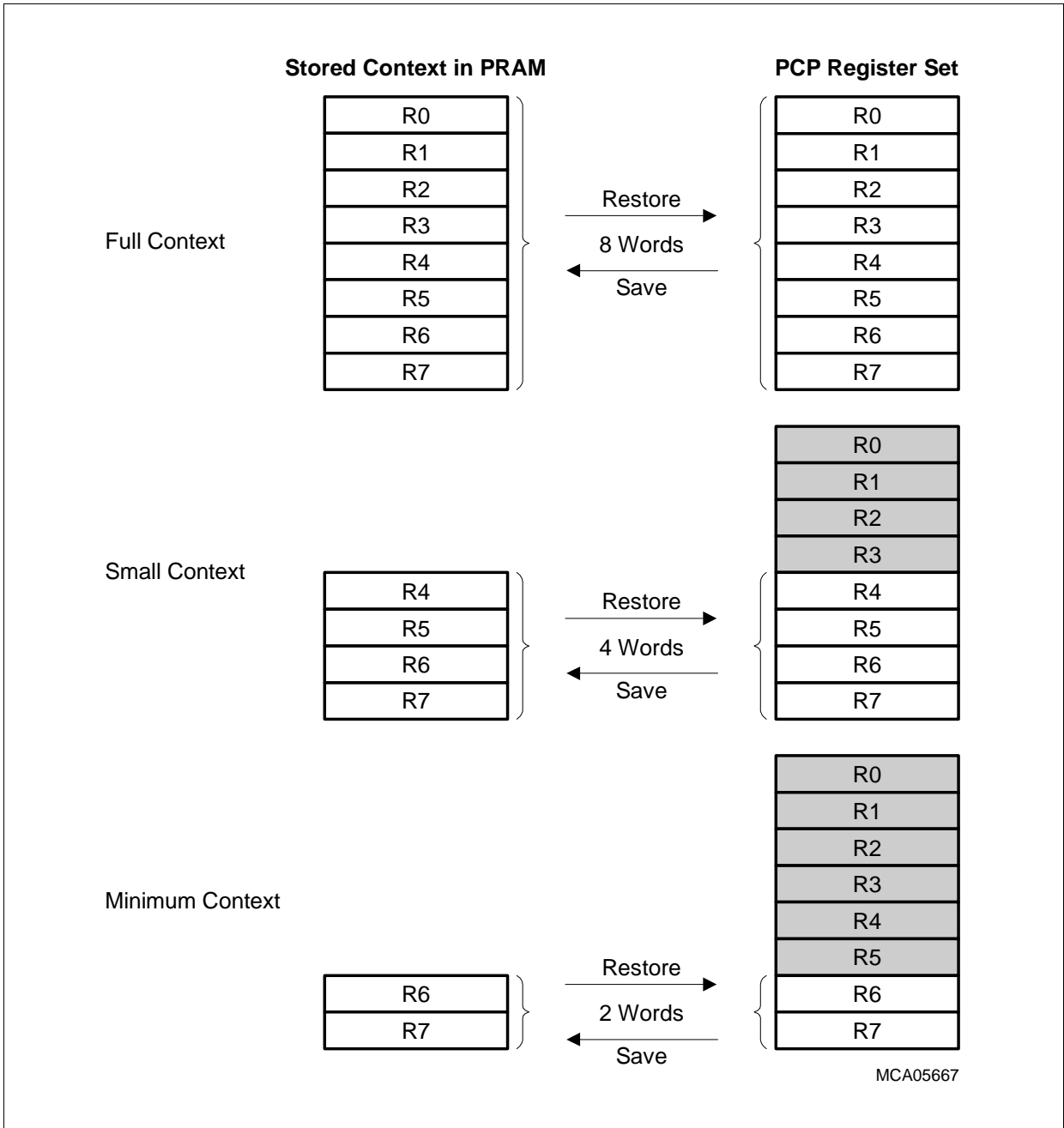


Figure 11-2 PCP Context Models

11.3.2.2 Context Save Area

The Context Save Area (CSA) is a region in PRAM reserved for storing the contexts for all channel programs (while any particular channel is not executing). Each channel's context is stored in a region of the CSA based on the channel number. The channel number is equal to the priority number (SRPN) of the service request. The PCP uses this number to calculate the start address of the context of the associated channel program. The size of a context is determined by the Context Model that the PCP has been initialized to use. As all channels use the same context size, the PRAM address (word address) of the context for a particular channel is simply calculated by multiplying the channel number by the number of registers in the context (8 for Full Context, 4 for Small Context and 2 for Minimum Context). [Figure 11-3](#) shows the resulting PRAM layout, and from this it can be seen that changing the Context Model also changes the base address for all regions within the CSA. Thus, the chosen Context Model may only be set when the PCP is initialized, and may not be changed during operation.

The CSA in the PRAM starts at address 00_H and grows upward. It is partitioned into equally sized regions, where the size of these regions is determined by the selected Context Model. The priority number (SRPN) of a service request is used to access the appropriate context region for the associated channel program. Since a request with an SRPN of 00_H is not considered as valid request in the TriCore Architecture, the bottom region (context region 0) of the CSA is never used for an actual context.

The total size of the CSA depends on the Context Model and the number of service request numbers used in a given system. Each priority number used in a service request node which can activate interrupts to the PCP must be represented through a dedicated context region in the PRAM. The highest address range in the PRAM used for a context region is determined by the highest priority number presented to the PCP with a service request.

The range of usable priority numbers is further determined by the size of the implemented PRAM and by the space required for other variables and global data located in the PRAM. See [Page 11-125](#) for the implemented size of the PRAM in the TC1796. As an example, a PRAM of 2 Kbytes, solely used for the CSA, can store up to 255 Minimum Contexts, allowing the highest SRPN used for a PCP service request to be 255 (remember, an SRPN of 0 and an associated context region is never used; the valid SRPNs and the context and channel numbers range from 1 to 255). With a Small Context Model, 127 contexts can be stored, resulting in 127 being the highest usable SRPN in this configuration. Finally, a Full Context Model allows 63 context areas, with 63 being the highest usable SRPN. Interrupt requests to the PCP with priority numbers that would cause loading of a context from outside the available PRAM area must not be generated. Invalid PCP operation will result should this situation be allowed to occur. The PCP can be optionally configured such that if an interrupt request is received that would cause loading of a context from outside the available PRAM area then an error exit is forced, and an error interrupt to the CPU is activated (see [Page 11-38](#)).

Peripheral Control Processor (PCP)

If portions of the PRAM are used for other variables and global data, the space available for the CSA and the range of valid SRPNs is reduced by the memory space required for this data. For best utilization of PRAM, it is advisable to have the CSA grow upwards as a contiguous area without any “holes”, meaning that all SRPNs in the range 1 ... max. are actually used to place interrupt requests on the PCP. Unused regions within the CSA (that is, the unused region at the base of the CSA and any context regions associated with unused channels) cannot be used for general variable storage.

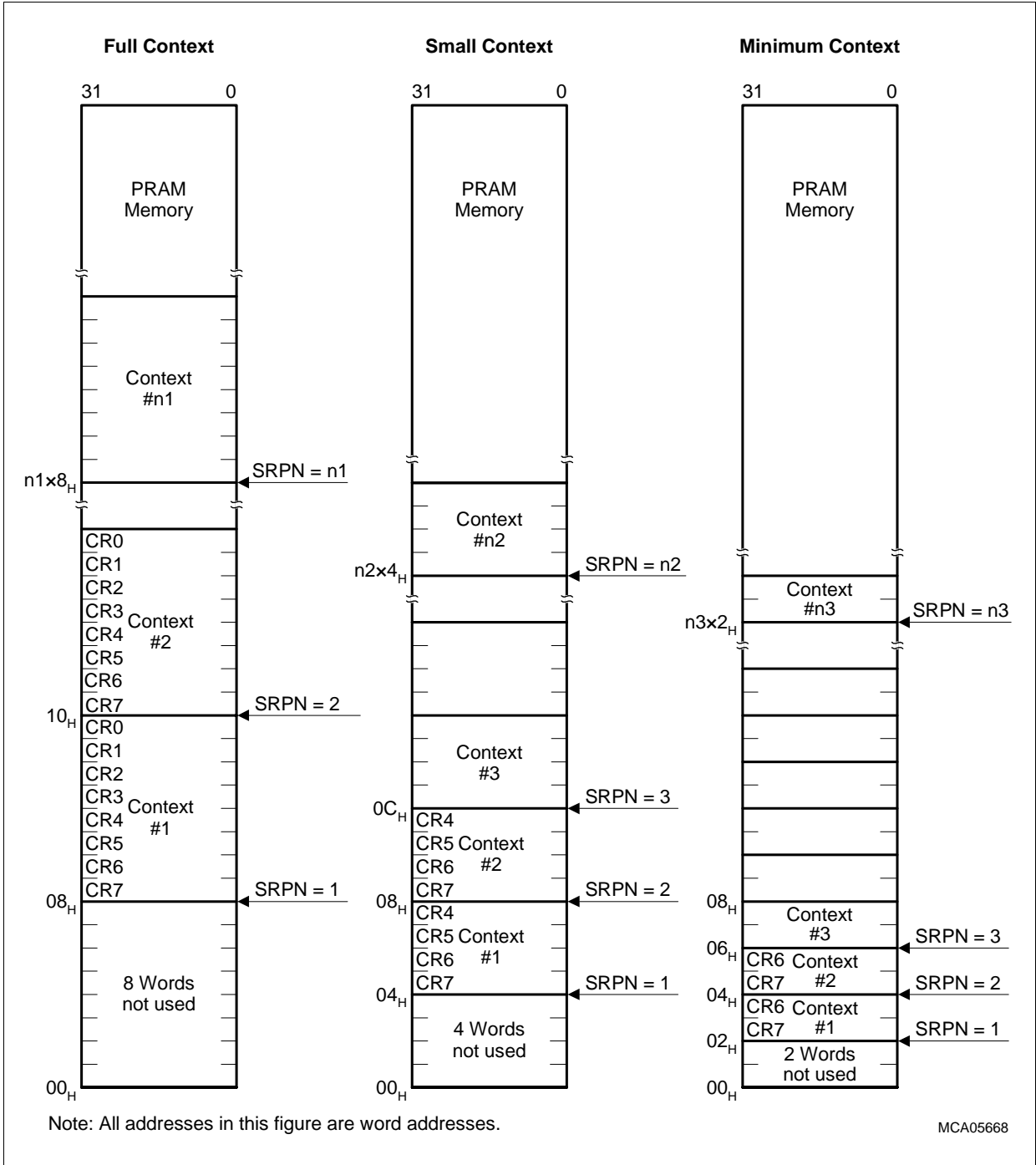


Figure 11-3 Context Storage in PRAM

When choosing the Context Model for a given application, the following considerations can be helpful. When choosing the small or the Minimum Context Models, save and restore operations for registers not handled in the automatic context operations can still be handled through explicit load and store instructions under control of the user. This may be advantageous for applications in which the majority of the channels do not need

Peripheral Control Processor (PCP)

the Full Context, and only some would require more context to be saved. In this case, a smaller Context Model can be used, and the channels which would require more register to be saved/restored would do this via explicit load and store instructions. This is especially advantageous if the channel program can be designed such that the initial real-time response operations can be executed using only the registers which have been automatically restored. Then, as the timing requirements of the service are relaxed, further register contents can be restored from PRAM through regular load instructions. Of course, the contents of these registers needs to be explicitly saved, through regular store instructions, before the exit of the channel program.

Note: Special care must be exercised when using Minimum or Small Context Models in conjunction with nested interrupts (see [Page 11-119](#)).

The criteria for choosing the Context Model are listed in the following:

- Size of PRAM implemented in a given derivative
- Amount of channels (= SRPNs) that need to be used in a system
- Amount of PRAM used for general variables and globals
- Amount of context (register content) which need to be saved and restored quickly by most of the most important channels

While registers R0 through R5 are always restored in a normal manner (according to the context size), registers R6 and R7 merit discussion regarding context restore operations. The memory location CR7 in a context region is used to hold two different pieces of information: The lower part of register R7, and the PC value of the channel. Similarly, the memory location CR6 in a context region can also be used to hold two different pieces of information: The value to be restored to register R6, and the Operating Priority (CPPN) value of the channel. This leads to the Restore/Save operations described in the following two sections.

11.3.2.3 Context Restore Operation for CR6 and CR7

The operation of R6 and R7 context restore varies according to whether the channel program that is starting is a “new” channel program (i.e. a channel program that is starting in response to the receipt of a new service request), or is a “suspended” channel program (i.e. a channel program that is re-starting after being suspended in favour of a higher-priority channel program). In addition, when a “new” channel program is starting, the context restore operation depends on the channel start mode that has been selected (see [Page 11-25](#)).

Channel Resume Mode

Figure 11-4 illustrates the operation of a context restore for a “new” channel program when Channel Resume Mode has been selected (see [Page 11-26](#)). The PC is loaded from CR7[31:16], and the lower half of R7 is loaded from CR7[15:0]. The operating priority of the channel is taken from CR6[31:24] and all of R6 is loaded from CR6.

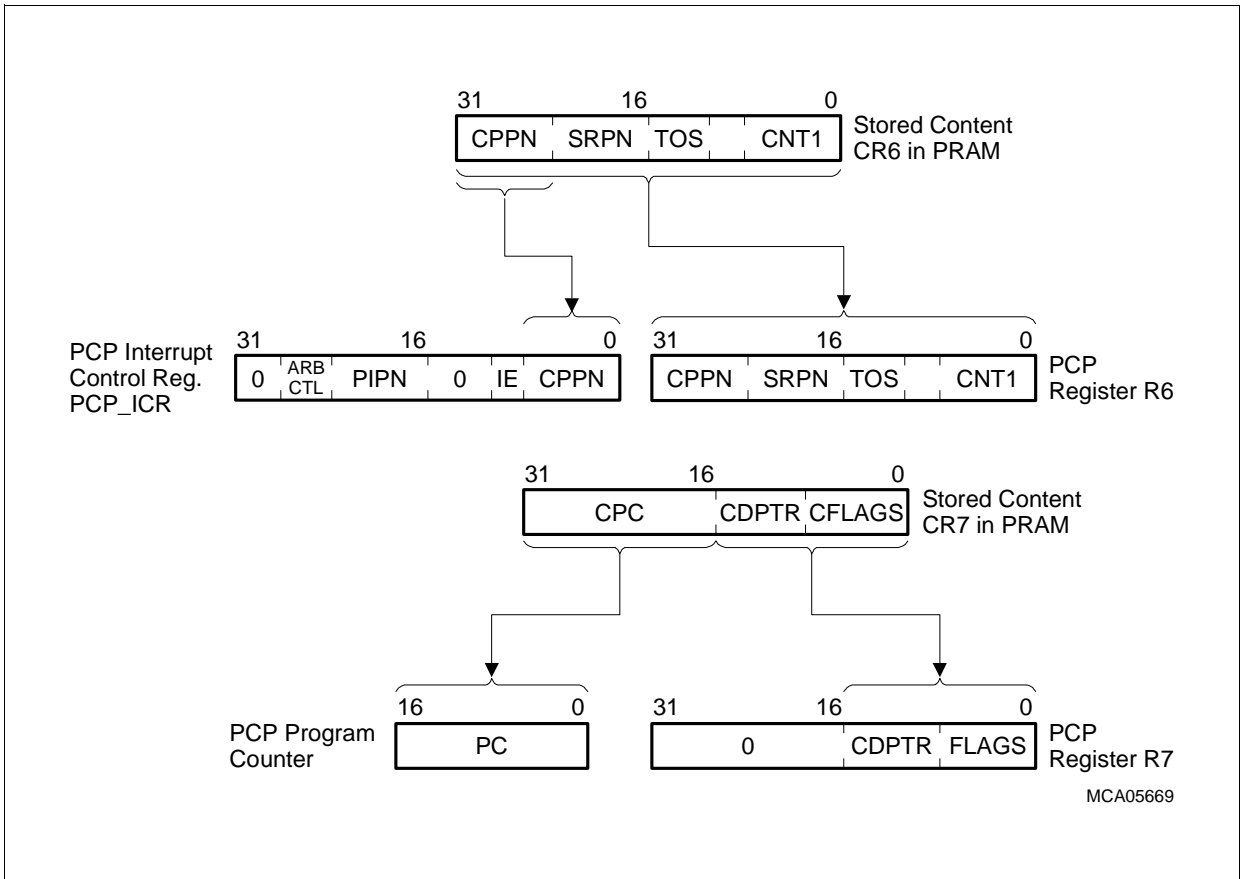


Figure 11-4 Context Restore: Channel Start in Channel Resume Mode

Channel Restart Mode

Figure 11-5 illustrates the operation of a context restore for a “new” channel program when Channel Restart Mode has been selected (see [Page 11-25](#)). The PC is loaded with the channel entry table address, and the lower half of R7 is loaded from CR7[15:0]. The upper half of CR6 is discarded. The operating priority of the channel is taken from CR6[31:24] and all of R6 is loaded from CR6.

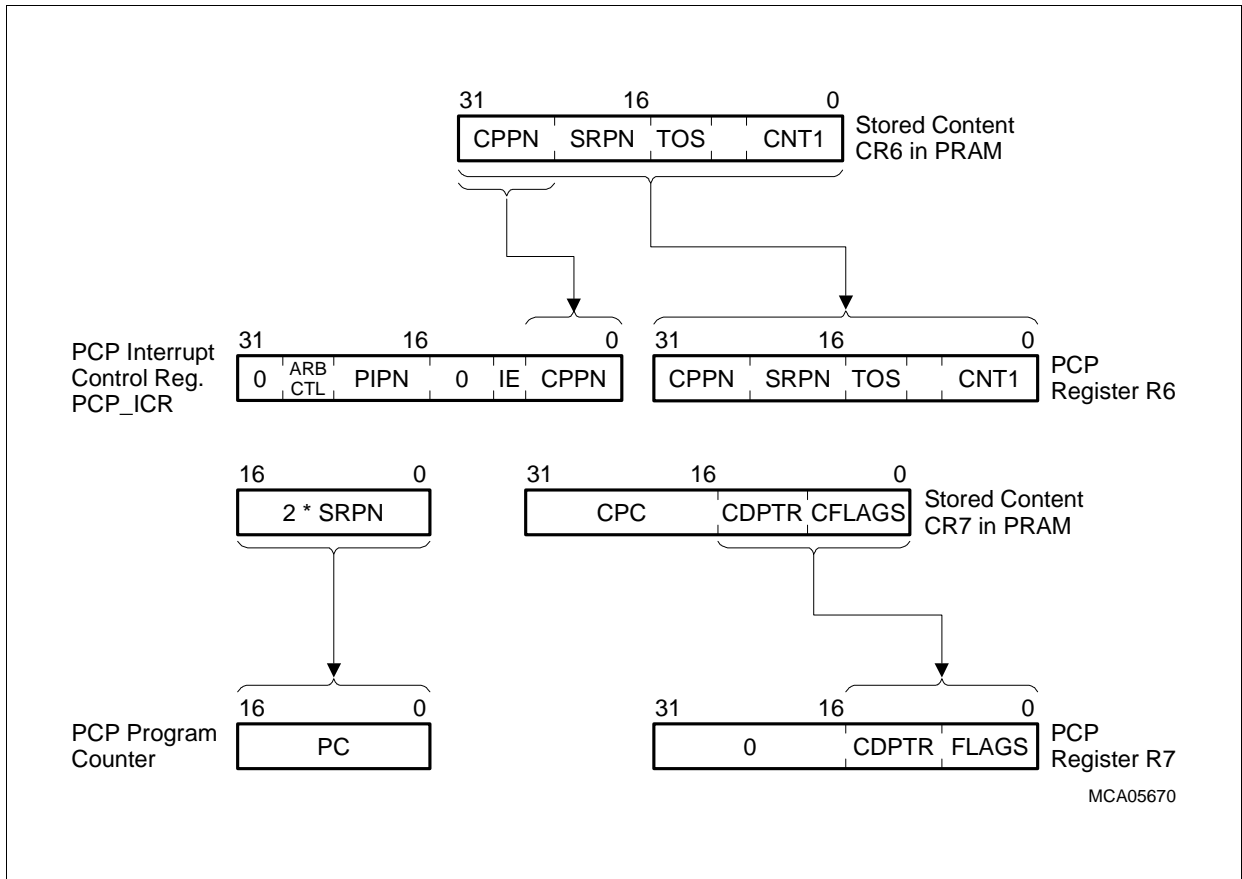


Figure 11-5 Context Restore: Channel Start in Channel Restart Mode

Suspended Channel Restart

Figure 11-6 illustrates the operation of a context restore for a “suspended” channel program. The PC is loaded from CR7[31:16] (regardless of the Channel Start Mode), and the lower half of R7 is loaded from CR7[15:0]. All of R6 is loaded from CR6. The figure also shows how the operating priority of the channel (PCP_IR.CPPN) is restored from the Service Request Node that was used to store the Suspended Interrupt Request (see [Page 11-70](#)).

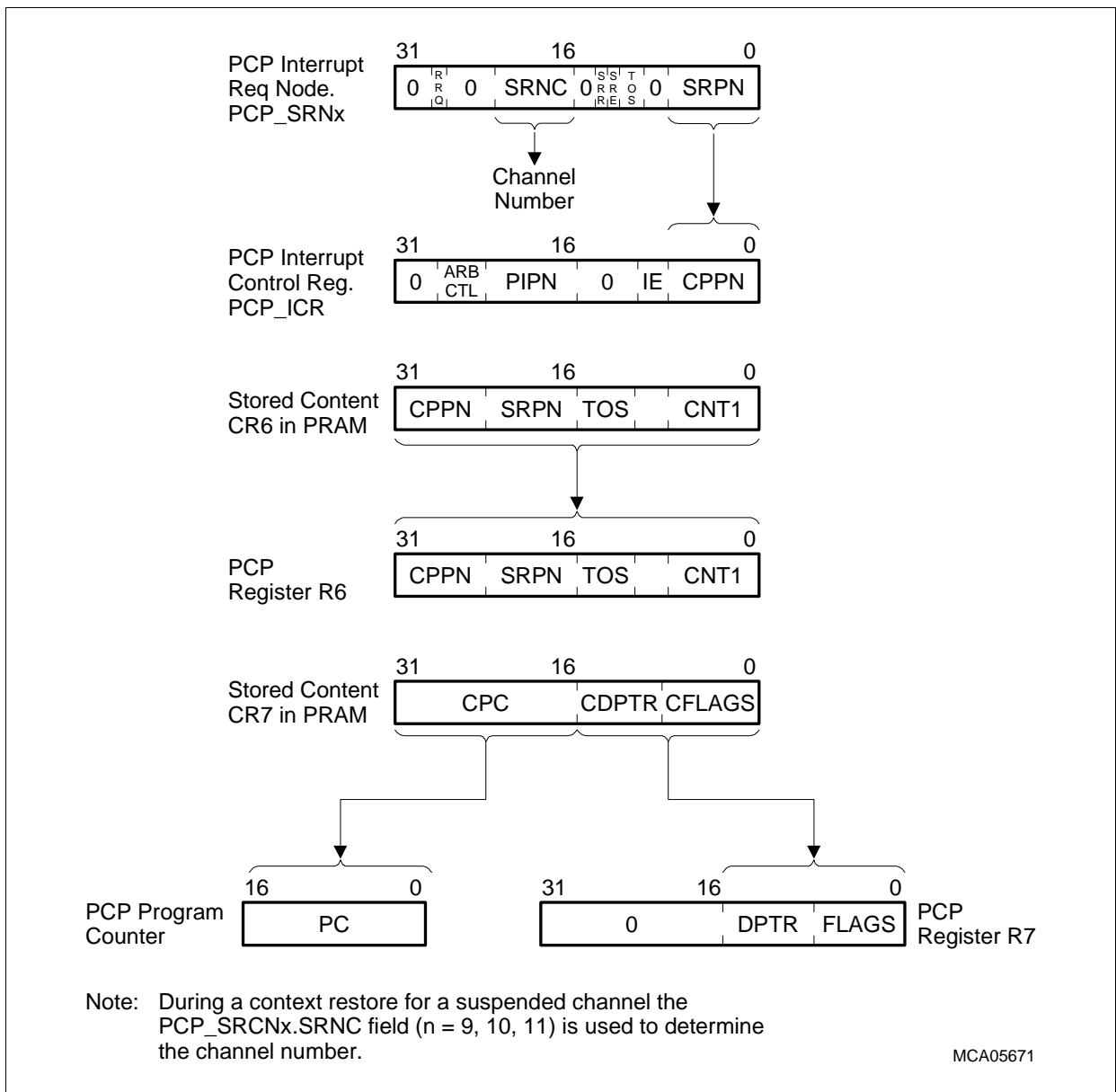


Figure 11-6 Context Restore: Suspended Channel Restart

11.3.2.4 Context Save Operation for CR6 and CR7

The operation of R6 and R7 context save varies according to whether the save operation is the result of a channel exit condition, or whether the channel is being suspended in favor of a higher-priority channel program.

Channel Resume Mode

Figure 11-7 illustrates the operation of a context save for a channel exit when Channel Resume Mode has been selected. The value written to CR7 is created by concatenating the 16-bit PC value with the lower 16 bits of R7. CR6 is written with the value taken from R6.

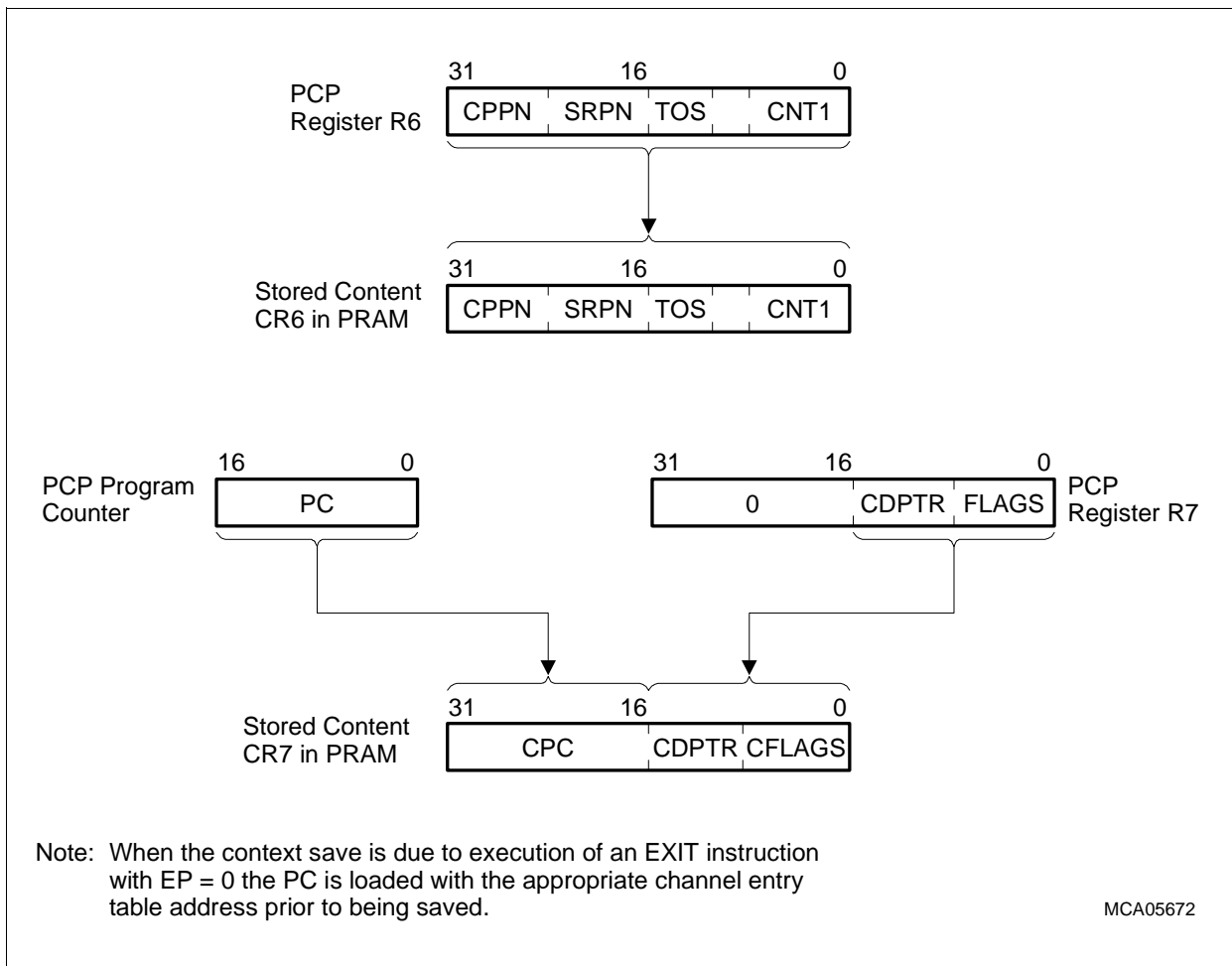


Figure 11-7 Context Save: Channel Exit in Channel Resume Mode

Channel Restart Mode

Figure 11-8 illustrates the operation of a context save for a channel exit when Channel Restart Mode has been selected. This is the same as for Channel Resume mode except that the PC value is discarded, and the appropriate Channel Entry Table address is written to CR7[31:16].

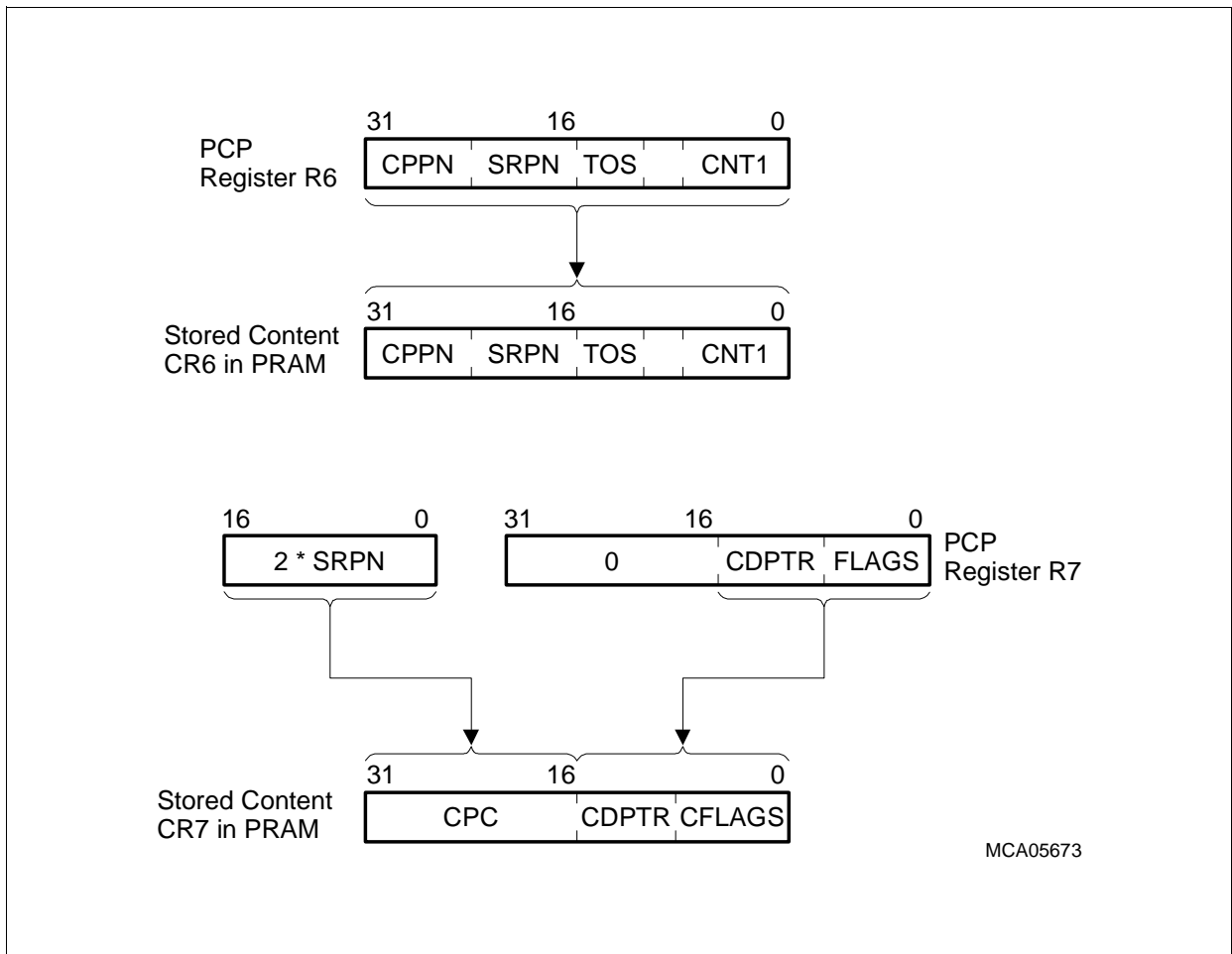


Figure 11-8 Context Save: Channel Exit in Channel Restart Mode

Channel Suspend

Figure 11-9 illustrates the operation of a context save for a channel that is being suspended. This is the same as for Channel Resume mode except that an interrupt request is created to allow the channel to be restarted at a later time. This restore operation utilizes one of three specially extended SRNs (see **Page 11-70**) to store the interrupt request. The information stored as part of the interrupt request is the channel number (SRPN), and the operating priority (CPPN) with which the channel was operating prior to being suspended. This operation in conjunction with the suspended channel restore operation shown in **Figure 11-6** allows the temporary suspension of a channel in favor of a higher-priority channel.

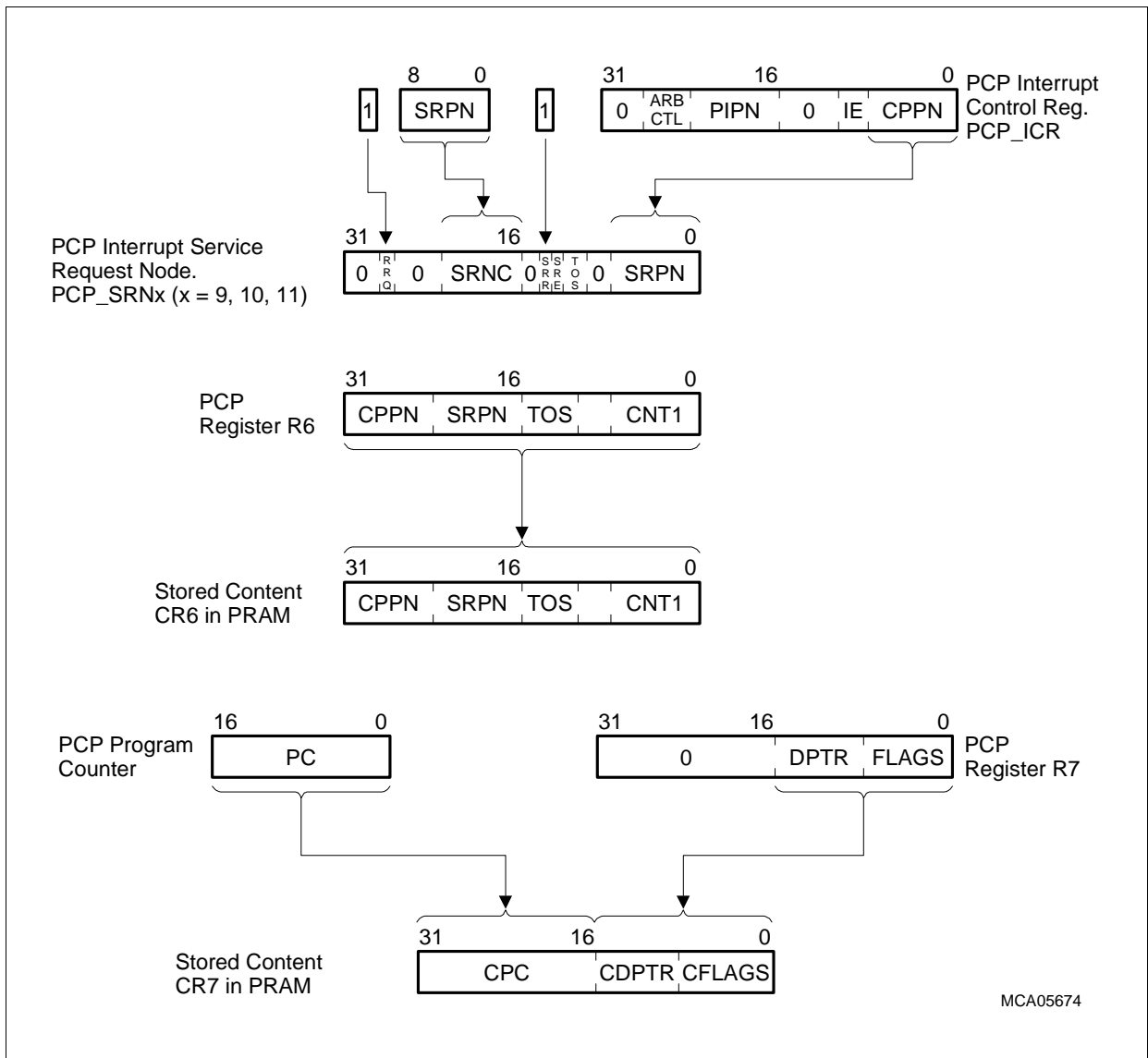


Figure 11-9 Context Save: Channel Suspend

11.3.2.5 Initialization of the Contexts

The programmer is responsible for configuring each channel program's context before commencing operation. Because this must be done by writing to the PCP across the FPI Bus, it is important to understand exactly where each channel program's context is from the FPI Bus perspective (see [Page 11-48](#) for details).

11.3.2.6 Context Save Optimization

The PCP has an optimized context-switching strategy consisting of optimization of both context load and store. During a context load in which the channel that is starting is also the last channel that the PCP was running then the PCP GPRs already contain the values appropriate to the channel. In this case there is no need to reload the context (i.e. the PCP can immediately continue operation at the appropriate point in the code without having to perform a context load). During a Context Store (i.e. the PCP exits a channel as a result of EXIT or DEBUG instructions or exits in response to a higher-priority channel interrupt), only those registers that have been updated (i.e. written to) since the context was loaded are saved to the CSA.

11.3.3 Channel Programs

The PCP CMEM is used to store the instruction sequences, the channel programs, for each of the PCP channels. The individual channel programs for the individual PCP service requests can usually be viewed as independent and separate programs. There is no background program defined and running for the PCP in TC1796 as there would be with traditional processors.

When the PCP receives a service request for a specific channel program, it needs to determine exactly which channel program to activate and where to start its execution. To accommodate different application needs, the PCP architecture allows the selection of two different entry methods into the channel programs:

- Channel Restart Mode
- Channel Resume Mode

Channel Restart Mode forces the PCP to begin each channel program from a known fixed point in the CMEM that is related to the interrupt number. At the entry point related to the interrupt number in question, there will typically be a jump instruction which vectors the PCP to the main body of the channel program. This is identical to the traditional interrupt vector jump table. In Channel Restart Mode, channel code execution will always start at the same address in the interrupt entry table each time the channel is requested.

Channel Resume Mode allows the PCP to begin execution at the PC address restored as part of the channel program context. This mode allows for code to be contiguous and start at any arbitrary address. It also allows for the implementation of interrupt-driven state machines, and even the sharing of code across multiple programs with different context.

The selection of one of the two modes is a global PCP setting, that is, it applies to all channels. Selection is made via the PCP_CS.RCB bit in the PCP configuration register PCP_CS (see [Page 11-56](#)).

11.3.3.1 Channel Restart Mode

Channel Restart Mode is selected with PCP_CS.RCB = 1. In this mode, the PCP views the CMEM as being partitioned into an interrupt entry table at the beginning of the CMEM, and a general code storage area above this table.

The interrupt entry table consists of two instruction slots (2×16 -bit) for each channel. When a PCP service request is received, the PCP calculates the start PC for the requested channel by a simple equation based on the SRPN of that request ($PC = 2 \times SRPN$). It then executes the instruction found on that address. If more than two instructions are required for the operation of the channel program, then one of the instructions within the interrupt entry table must be a jump to the remainder of the channel's code. The PCP executes the channel's code until an exit condition or higher-priority interrupt is detected.

Peripheral Control Processor (PCP)

It is recommended that all EXIT instructions for all channels should use the EP = 0 setting when the PCP is operated in Channel Restart Mode (see [Page 11-89](#)).

Note that when Channel Restart Mode is in use a Channel Entry Table must be provided with a valid entry for every channel being used. [Figure 11-10](#) shows an example of CMEM organization when Channel Restart Mode has been selected. Failure to provide a valid entry for all channels that are in use will lead to invalid PCP operation.

11.3.3.2 Channel Resume Mode

Channel Resume Mode is selected with PCP_CS.RCB = 0. In this mode, the user can arbitrarily determine the address at which the channel program will be started the next time it is invoked. For this purpose, the PC is saved and restored as part of the context of a PCP channel.

Additional flexibility is available when Channel Resume Mode is globally selected by configuring each EXIT instruction to determine the channel start address to be used on the next invocation of a channel (see [Page 11-89](#)). When the EP = 0 setting is used, the PC value saved in the channel's context (saved in CPC) is the address of the appropriate location in the channel entry table. This forces the channel to start at the appropriate location in the interrupt entry table at next invocation. When the EP = 1 setting is used, the PC value saved in the channel's context is the address of the instruction immediately following the EXIT instruction. The use of the EP = x setting with the EXIT instruction allows the mixture of channels that use a Channel Restart strategy with others using a Channel Resume strategy, and also allows individual channels to use either strategy as appropriate on different invocations.

Note: A valid entry within a Channel Entry Table must be provided for every channel that uses an EXIT instruction with the EP = 0 setting when Channel Resume Mode has been selected. Failure to provide a valid entry for such channels will lead to invalid PCP operation.

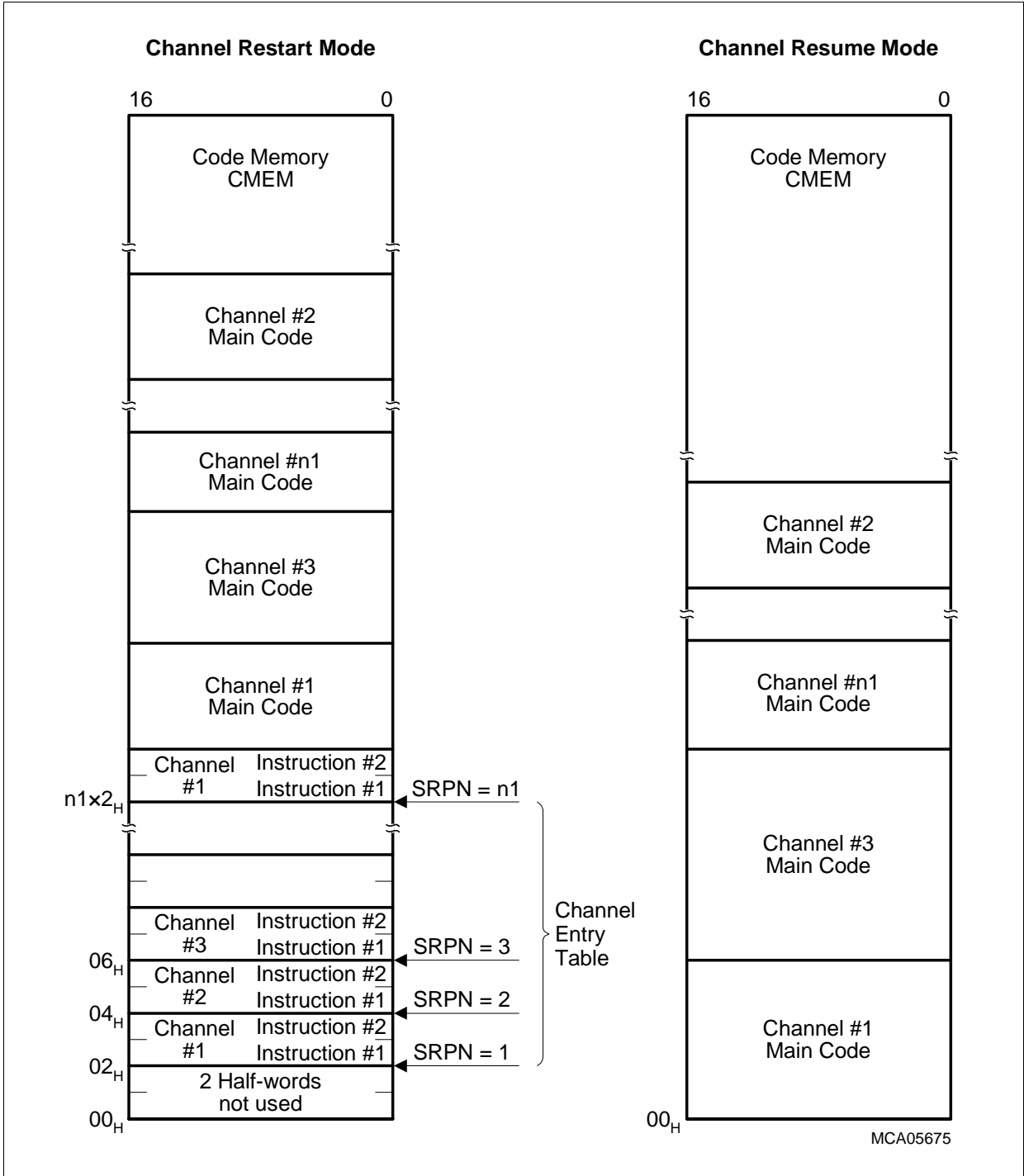


Figure 11-10 Examples of Code Memory Organization for Channel Restart and Channel Resume Modes

Note: The CMEM address offsets in the above figure are shown as PCP instruction (half-word) offsets. To obtain FPI address offsets (byte offset), multiply each offset by two.

11.4 PCP Operation

This section describes how to initialize the PCP, how to invoke a channel program, and the general operation of the PCP.

11.4.1 PCP Initialization

The PCP is placed in a quiescent state when the TC1796 is first powered-on or reset. Before a channel program can be enabled, the PCP as a whole must be initialized by some other FPI Bus master, typically the CPU. Initialization steps include:

- Configure global PCP registers
 - Initialize PCP Control and Status Register (with PCP_CS.EN = 0)
 - Configure interrupt system via PCP_ICR
- Load channel programs into the CMEM.
- Load initial context (if/as required) of channel programs in PRAM (R0-R7 for Maximum Context, R4-R7 for Small Context, R6-R7 for Minimum Context). Only those registers in each channel whose initial content is required on first invocation of the channel need to be loaded. This may need to include the initial PC, depending on the value of PCP_CS.RCB.
- Clear R7 in the context for unused channels
- Enable PCP operation PCP_CS.EN = 1.

Now, the PCP is able to begin accepting interrupts and executing channel programs.

11.4.2 Channel Invocation and Context Restore Operation

A channel program is started when one or other of the following conditions occurs:

- The current round of PCP interrupt arbitration results in a winning interrupt number (SRPN) and the PCP is currently quiescent (has exited the previous channel and stored the context for that channel).
- The current round of PCP interrupt arbitration results in a winning interrupt number (SRPN) that has to be greater than the current channel priority (SRPN > CPPN), if suitable Service Request Node space is available in the PSRN to store a suspended interrupt request and the current channel allows interrupts (R7.IEN = 1). See also [Page 11-33](#)).

When this happens the winning SRPN becomes the current interrupt, and a context restore operation occurs before the new channel program can begin operation, as follows:

- The context of the channel (= winning SRPN) is restored from PRAM into the GPRs from the appropriate address within the CSA. Depending on the value of PCP_CS.CS, a Full, Small, or Minimum Context restore is performed.
- The new priority level of the PCP is taken from R6.CPPN field and is written to PCP_ICR.CPPN. This value can be useful during debugging, as the CPPN of the currently executing or last-executed channel program can be read from PCP_ICR.

Peripheral Control Processor (PCP)

After the channel program starts, the value of R6 may be changed without altering the value of the effective CPPN, because updates to the value of R6.CPPN have no effect until the next invocation of the channel program.

- If the R7.CEN bit is clear (0), then an error has occurred because a disabled channel program has been invoked, the PCP_ES.DCR bit is set to flag the error, and the channel program performs an error exit (see [Page 11-29](#)).
- If the R7.CEN bit is set (1), then code execution begins at the value of the restored PC or at the address of the interrupt routine in the Channel Entry Table, depending on the value of PCP_CS.RCB.

Special care needs to be taken regarding the number of clock cycles required to switch context when the last state of the PCP before channel exit was execution of channel “N” (SRPN = “N”), and the current service request is also SRPN = “N”. In this case, the PCP Processor Core should not load the context (as the PCP GPRs already contain the correct content), but continue operation from the current point and state (noting that the PC value should be set to the appropriate channel entry point if PCP_CS.RCB = 1).

11.4.3 Channel Exit and Context Save Operation

The context of a channel program must be saved when it terminates. Three events can cause the termination of a channel program:

- Execution of the EXIT instruction (normal termination)
- Occurrence of an error
- Execution of the DEBUG instruction (channel termination is optional). The DEBUG instruction must be only used in DEBUG mode; otherwise an “Illegal Operation” (IOP) error will be generated.

These channel termination possibilities are described in the next sections.

11.4.3.1 Normal Exit

Under normal circumstances, a channel program finishes by executing an EXIT instruction. This instruction has several setting fields that allow the user to specify a number of optional actions to be performed during the channel exit sequence (see [Page 11-89](#)). These optional actions are:

- Decrement counter CNT1
- Set the start PC for the next channel invocation to the next instruction address (Channel Resume) or to the channel entry address (Channel Restart)
- Disable further invocations of this channel
- Generate an interrupt request to the CPU or to the PCP itself

When the EXIT instruction is executed, the following sequence occurs:

- If EC = 1 is specified Counter R6.CNT1 is decremented and the CN1Z flag is updated.

Peripheral Control Processor (PCP)

- If ST = 1 is specified bit R7.CEN (Channel Enable) is cleared (i.e. the channel is disabled).
- If EP = 0 is specified **or** PCP_CS.RCB = 1 (Channel Restart Mode has been selected), the PCP program counter to be saved to context location CR7.PC is set to the appropriate channel entry table address. If EP = 1 is specified **and** PCP_CS.RCB = 1 (Channel Resume Mode has been selected), the PCP program counter to be saved to context location CR7.PC is set to the address of the instruction immediately following the EXIT instruction.
- If INT = 1 is specified **and** the specified condition cc_B is True, then an interrupt request is raised according to the SRPN value held in R6.SRPN. The interrupt is asserted via one of the PCP_SRCx registers, where x is determined by the combination of the value of R6.TOS and the list of free entries. This allows the conditional creation of a service request to the CPU or PCP with the SRPN value indicated in register R6.SRPN.
- The channel program's context (including all register modifications caused within this EXIT sequence) is saved to the appropriate region in the PRAM Context Save Area. Depending on the value of PCP_CS.CS, either a Full, Small, or Minimum Context save is used.

Special care needs to be taken to optimize the number of clock cycles required to perform a Context Save. During a Context Save, the PCP Processor Core needs only to save those registers that have been written since the last Context Restore was performed.

Note: Particular attention must be paid to the values of R6 and R7 prior to execution of the EXIT instruction. When posting an interrupt request, the user must ensure that R6.SRPN and R6.TOS contain the correct values to generate the required interrupt request. When using the Outer Loop Counter (CNT1), the user must ensure that the value in R6.CNT1 will provide the required function. When using interrupt priority management, the user must ensure that R6.CPPN contains the interrupt priority with which the channel is to run on next invocation. If the channel is to be subsequently re-invoked, the user must ensure that the Channel Enable Bit (R7.CEN) is set.

11.4.3.2 Error Condition Channel Exit

PCP error conditions can occur for a variety of reasons (e.g. an invalid operation code was executed by a channel program, or an FPI Bus error occurred). When an error condition occurs, the PCP Error Status Register (PCP_ES) is updated to reflect the error, and the channel program is aborted. The error exit sequence is as follows:

- The channel enable bit R7.CEN is cleared. This means the channel program will be unable to restart until another FPI Bus master has re-configured the channel program's stored context to set CR7.CEN to 1 again.

Peripheral Control Processor (PCP)

- The PC of the instruction that was executing when the error occurred is stored in PCP_ES.EPC.
- The number of the channel program that was executing when the error occurred is stored in PCP_ES.EPN.
- The error type is set in the appropriate field of register PCP_ES.
- The context is saved back to the PRAM CSA. Depending on the chosen context size (PCP_ES.CS) a Full, Small, or Minimum Context save is performed.
- If the error condition was not due to an FPI Bus error or a DEBUG instruction, then an interrupt request to the CPU is generated with the priority number stored in register PCP_CS.ESR.

The repetitive posting of PCP error interrupts will not cause an overwhelming number of interrupts to the CPU. In this situation, the PCP's CPU service request queue (see [Page 11-33](#)) will quickly fill, and force the PCP to stall until the CPU can resolve the situation.

Note: An error condition (other than an FPI Bus error) will result in an interrupt being sent to the CPU. The interrupt routine that responds to this interrupt must be capable of dealing with the cause as recorded in PCP_ES, and it must be able to restore the channel program to operation. The minimum required to restart the channel program is to set the context value of CR7.CEN = 1.

11.4.3.3 Debug Exit

If the DEBUG instruction is programmed to stop the channel program execution (SDB = 1 has been specified), the PCP performs an exit sequence that is very similar to the error exit sequence, with the exception that no interrupt request to the CPU is generated. This sequence is:

- If RTA = 0, then the channel enable bit R7.CEN is cleared. This means the channel program will be unable to restart until another FPI Bus master has re-configured the channel program's stored context to set CR7.CEN to 1 again. Otherwise, the R7.CEN bit remains unchanged, and the PC is decremented (such that it points to the DEBUG instruction)
- If EDA = 1, a break-point event is generated
- If DAC = 1, then the PCP_CS.EN bit is cleared. This means that the PCP will not execute any further channel programs until the PCP_CS.EN bit is set by another FPI Bus master
- The address of the DEBUG instruction (i.e. the current PC) is stored in register PCP_ES.PC
- The current channel number is stored in register PCP_ES.PN

The execution of the current channel program is stopped at the point of the DEBUG instruction. This instruction only disables the current channel; the PCP will continue to operate, accepting service requests for other channels as they arise.

Peripheral Control Processor (PCP)

Note: The *DEBUG* instruction must be only used in *DEBUG* mode; otherwise an “Illegal Operation” (IOP) error will be generated.

11.5 PCP Interrupt Operation

The PICU and the PSRNs (PCP_SRC[11:0]) are similar to the CPU’s ICU and all other SRNs in the system. They do, however, have some special characteristics, which are described in the following sections. **Figure 11-11** shows an overview of the PCP interrupt scheme.

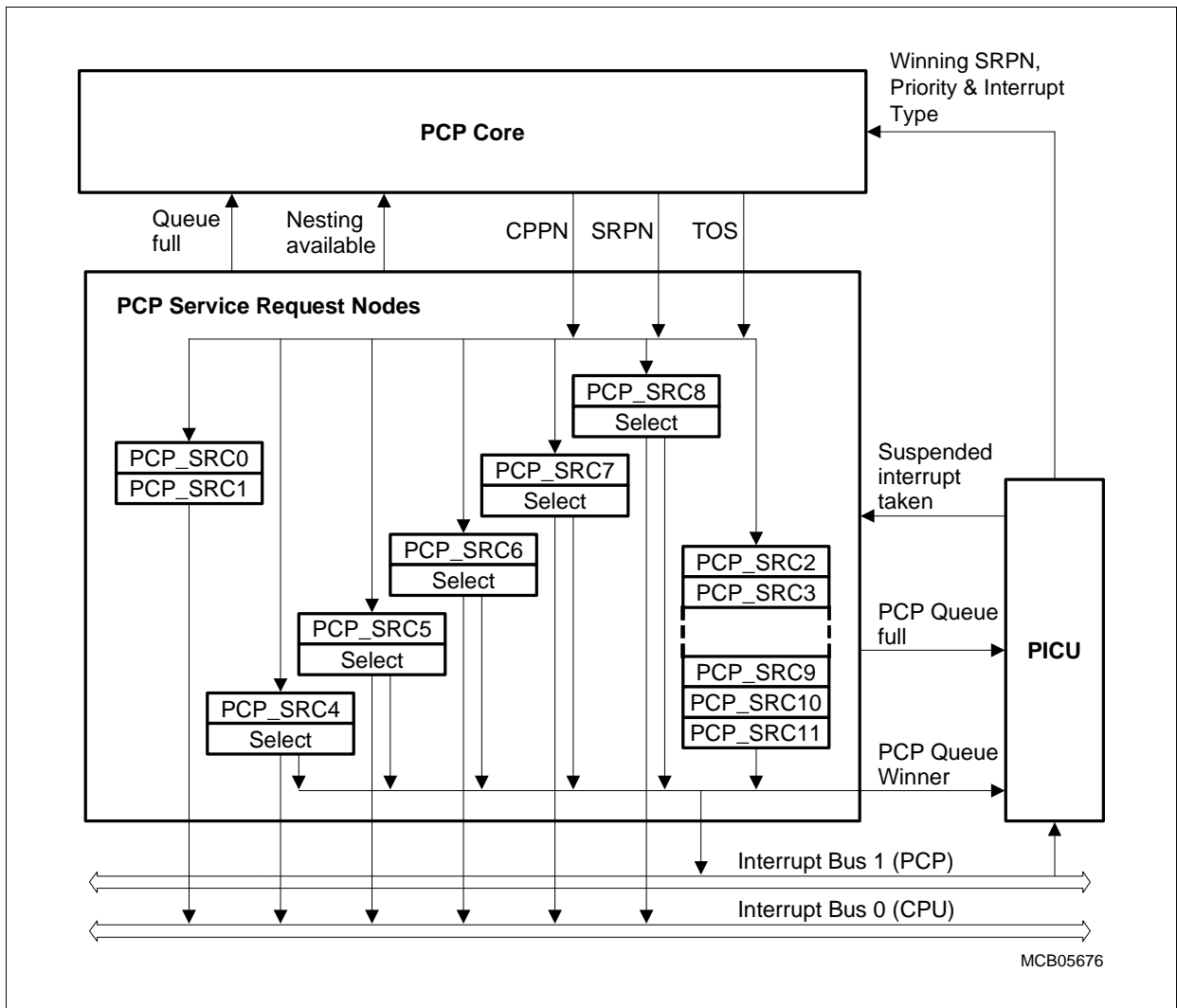


Figure 11-11 PCP Interrupt Block Diagram

11.5.1 Issuing Service Requests to CPU or PCP

The PCP may use one of two mechanisms to raise an interrupt request to the CPU or itself. The first, and most inefficient, method for a PCP channel program is to issue

Peripheral Control Processor (PCP)

service requests by performing an FPI Bus write operation to an external service request node (SRN). Alternately, the PCP can raise a service request using one of its own internal SRN's. An interrupt can only be generated by the PCP via an internal SRN when executing an EXIT instruction, or when an error condition occurs. In the following descriptions, PCP service requests triggered through an EXIT instruction or the occurrence of an error are called “implicit” PCP service requests to distinguish them from the “explicit” way of generating a service request through an FPI Bus write to a service request node external to the PCP.

11.5.2 PCP Interrupt Control Unit

The PICU operates in a similar manner to the ICU of the CPU. The PICU manages the PCP service request arbitration bus, and handles the communication of service requests and priority numbers to and from the PCP kernel. The PCP_ICR register is provided to control and monitor the arbitration process.

When one or more service requests to the PCP are activated, the PICU performs an arbitration round to determine the request with the highest priority. It then places the priority number of this “winning” service request into the PIPN field of register PCP_ICR, and generates a service request to the PCP kernel.

If the PCP kernel is currently busy processing a channel program, the new request is left pending until the current channel program has finished.

When the PCP kernel is ready to accept a new service request, it calculates the context start address from the Pending Interrupt Priority Number, PIPN, stored in register ICR, and begins with the context restore. It notifies the PICU of the acceptance of this request, and in turn the PICU acknowledges the winner of the last arbitration round. This service request node then clears its Service Request Flag, SRR.

There is one special condition in which the PICU operates differently from the way that the CPU Interrupt Control Unit operates. This special operation is described at [Page 11-36](#).

The PCP interrupt arbitration can be adapted to the application's needs and characteristics through controls in register PCP_ICR. Bit field PCP_ICR.PARBCYC controls the number of arbitration cycles per arbitration round (one through four cycles), while bit PCP_ICR.PONECYC controls whether one arbitration cycle equals one or two system (FPI Bus) clock cycles.

11.5.3 PCP Service Request Nodes

The PCP contains twelve service request nodes, including twelve service request control registers, PCP_SRC0 to PCP_SRC11, which are provided for implicit PCP service requests. The service request control registers differ from standard SRC registers in that they are fully controlled by the PCP kernel; they are read-only registers during PCP operation. The user cannot generate interrupts by writing to them.

Peripheral Control Processor (PCP)

The twelve service request nodes are split into four groups of two nodes each.

- The first group, containing registers PCP_SRC0 and PCP_SRC1, handles implicit PCP service requests targeted to the CPU. The Type-of-Service control fields, TOS, of these registers are hard-wired to 00_B, directing the requests to the CPU.
- The second group, registers PCP_SRC2 and PCP_SRC3, handles the service requests targeted to the PCP itself. The respective TOS field of these registers are hard-wired to 01_B, directing the requests to the PCP.
- The third group, containing registers PCP_SRC4 to PCPSRC8, have programmable TOS fields which allow these registers to be assigned (at configuration time) to any of the available interrupt buses.
- The fourth group, containing registers PCP_SRC9, PCP_SRC10 and PCP_SRC11, are an extended version of a standard Service Request Node. These handle service requests targeted to the PCP itself, including service requests representing a suspended interrupt. The respective TOS field of these registers are hard-wired to 01_B, directing the requests to the PCP.

The service request enable bits, SRE, of the PCP_SRCx registers are hard-wired to 1, meaning these service requests are always enabled.

Note: The number of interrupt buses is device-dependent. Programming a PCP_SRCx register (x = 4 to 8) with a TOS value representing a non-available interrupt bus (10_B or 11_B in the TC1796) will disable Service Request Node x.

The actual service request flag and the service request priority number of the PCP_SRCx registers are updated by the PCP when it generates an implicit service request. The way this is performed is described in the following section.

The service request nodes in each of the groups described above are implemented as queues with the appropriate number of entries. When the PCP generates an implicit service request, it places the request into the next available free entry of the appropriate queue rather than writing it into a specific register. Queue management logic automatically ensures proper handling of the queue. If all entries of a queue are filled with pending service requests, the queue management reports this condition to the PCP kernel via a “queue full” signal.

In the following descriptions, the terms “CPU Queue” and “PCP Queue” are used to refer to the queues in the two groups of PCP service request nodes.

11.5.4 Issuing PCP Service Requests

The PCP can issue implicit service requests on the execution of an EXIT instruction, when suspending a channel, or when an error occurs during a channel program execution. While the service request generation for the EXIT instruction is optional, a service request is always generated when a channel is suspended or an error occurs. Further differences between these three mechanisms are detailed in the following sections.

11.5.4.1 Service Request on EXIT Instruction

An implicit PCP service request is issued when the INT field of the EXIT instruction is set to 1 and the specified condition code, cc_B, of this instruction is true. Such a service request can be issued to any of the available interrupt buses, depending on the programmed value in the TOS field of register R6. The PCP examines the TOS field in register R6 and issues a service request to the appropriate queue of the service request nodes. Along with this request, it passes the service request priority number stored in the SRPN field of register R6 to the queue. If the queue has a free entry left, the service request flag, SRR, of the associated service request register, PCP_SRCx, will be set, and the service request priority number will be written to the SRPN field of the SRC register. Please see [Page 11-36](#) for the case there is no free entry in the queue.

Because the desired service request is programmed through the TOS and SRPN fields in register R6, each channel program can issue its individual service request. Note that this register needs to be programmed properly if a service request is to be generated by the EXIT instruction.

11.5.4.2 Service Request on Suspension of Interrupt

An implicit PCP service request is issued when the PCP suspends execution of the ongoing channel program in favor of a service request with a higher priority. Such a service request is always issued to the PCP's own interrupt bus and is stored in one of the three extended Service Request Nodes (PCP_SCR9, PCP_SRC10, PCPSRC11). Along with this request, it passes the current channel operating priority (CPPN) as a SRPN and also the channel number (the original SRPN). The service request flag, SRR, and the Restart Request flag, RRQ, of the associated service request register, PCP_SRCx, will be set, the Operating Priority will be written to the SRPN field, and the channel number will be written to the SRNC field of the SRC register.

Use of the Operating Priority as the SRPN for resumption of the channel program ensures that during subsequent arbitration rounds the PCP will resume execution of the suspended channel program at the appropriate time.

The PCP treats an interrupt request with the RRQ bit set in a special fashion. In this case the PCP clears the interrupt request bit in the appropriate internal Service Request Node but does not issue an interrupt acknowledge to any external nodes. This prevents the unwanted clearing of external service requests with an SRPN that matches the priority of a suspended channel.

Note: The PCP will only suspend channel operation when there are two or more free service Request Nodes with the appropriate TOS value for the PCP, and one of the free Service Request Nodes is an Extended Service Request Node. This allows for the posting of an interrupt request to the PCP on exit from the new channel program.

11.5.4.3 Service Request on Error

While a service request triggered through an EXIT instruction is optional and can be issued either to the CPU or to the PCP itself, a service request due to an error condition will always be automatically issued and will always be directed to the CPU. The PCP issues a service request to the CPU queue of the service request nodes. Along with this request, it passes the SRPN stored in the ESR field of register PCP_CS to the queue. If the queue has a free entry left, the SRR flag of the associated service request register, PCP_SRCx, will be set, and the SRPN will be written to the SRPN field of the SRC register. See next section for the case when there is no free entry in the queue (queue full).

Due to the fact that the priority number is stored in the global control register PCP_CS, all channel programs share the same service request routine in case of an error. The exact cause of the error and the channel number of the program that was executed when the error occurred can be determined through examination of the contents of the Error/Debug Status Register, PCP_ES.

11.5.4.4 Queue Full Operation

Queuing the implicit service requests typically allows the PCP to continue with the next service request without stalling. The depth of the queue and the number of channel programs using them determines the stall rate. Depending on the selected service provider (via R6.TOS in case of an EXIT interrupt or always to the CPU in case of an error interrupt), the request is routed to a free entry in the appropriate queue.

If no free entries are available in a queue at the time the PCP wants to post a request to that queue, the PCP is forced to stall until an entry becomes clear. This ensures that the PCP does not lose any interrupts. An entry in a queue becomes free when its SRR flag is cleared through an acknowledge from the PICU (that is, the CPU or PCP, as appropriate, has started to service this request).

One special case needs to be resolved for the PCP-related queue through special operation of the PICU. Consider the case in which the PCP queue is full, meaning registers PCP_SRC2, PCP_SRC3 and PCP_SRC9 to PCP_SRC11 are already loaded with pending service requests to the PCP. If the PCP kernel now needed to post an additional service request into that queue, a deadlock situation would be generated: The PCP would stall, since there is not a free entry in the PCP queue in which to place the request. In turn, as the PCP is stalled, it cannot accept new service requests and so the PCP service request queue cannot be emptied. This would result in a deadlock of the PCP.

To avoid such a deadlock, the PICU performs a special arbitration round as soon as the PCP queue becomes full. In this arbitration round, only the service request nodes assigned to the PCP queue are allowed to participate; all service requests from nodes external to the PCP are excluded, regardless of whether their priorities are higher or

Peripheral Control Processor (PCP)

lower than those of the PCP queue. In this way, it is guaranteed that one entry in the PCP queue gets serviced, freeing one slot in the queue.

The PCP programmer needs to carefully consider this special operation. It ensures that deadlocks are avoided, but it implies that if too many PCP channel programs post service requests to the PCP (self-interrupt), the PCP will have to service these rather than outside interrupt sources. Depending on the priority given to these requests, this could undermine an otherwise appropriate use of the interrupt priority scheme. It is recommended that the system be designed such that in most cases, high-priority numbers can be assigned to these self-interrupts so that they can win normal arbitration rounds, avoiding the situation where the PCP queue becomes full.

Note: If the CPU queue is full, the PCP can continue to operate until it needs to post another service request to the CPU queue.

11.6 PCP Error Handling

The PCP contains a number of fail-safe mechanisms to ensure that error conditions are handled gracefully and predictably. In addition to providing an extra level of system robustness suitable for high integrity and safety-critical systems, these mechanisms can often ease the task of finding programming errors during the development process. Whenever an error is detected, the channel program that was executing exits and the PCP_ES register is updated with information to allow determination of the error that occurred, the instruction address, and the channel program that was executing when the error occurred (see [Page 11-30](#)).

11.6.1 Enforced PRAM Partitioning

As previously discussed, the PRAM can be considered as being split into two distinct areas. The lower of these two areas is the CSA (see [Page 11-14](#)) used for storing context information for each active channel while the channel program is not actually executing. The remainder of PRAM is available for general use and is typically used to hold variables and global data.

The default configuration of the PCP allows the PCP to use PRAM as a single area. While this default configuration allows complete flexibility regarding the use of PRAM, this flexibility also introduces the possibility of invalid PCP operation as a result of the following issues:

- Any channel program is allowed to write to any PRAM location (including any location in the CSA). This means that a channel program may be inadvertently programmed to corrupt the context save region belonging to another channel, causing invalid operation of the corrupted channel when it next executes.
- Generation of an interrupt request to the PCP with a priority number that would cause loading of a context from outside the CSA will cause the spurious execution of a channel program with an invalid context loaded from outside the CSA.

To avoid spurious PCP operation as a result of either of these programming errors, the PCP can be optionally configured via the global PCP control and status register (PCP_CS) to enforce strict partitioning of PRAM. PRAM partitioning is selected by programming PCP_CS.PPE = 1 and the size of the CSA in use is selected via the PCP_CS.PPS bit field (see [Page 11-56](#)). When PRAM partitioning has been enabled, a PCP error will be generated on either of the following events:

- A channel program executes a PRAM write instruction with a target area within the CSA. This prevents a channel from corrupting the context save region of any other channel.
- An incoming interrupt request causes the PCP to attempt to load a context from outside the CSA. This prevents the PCP from running an invalid channel program as a result of an invalid interrupt request.

Peripheral Control Processor (PCP)

Note: Enabling PRAM partitioning (PCP_CS.PPE = 1) with a CSA size of zero (PCP_CS.PPS = 0) is an invalid setting and will cause a PCP error event whenever any interrupt request is received by the PCP.

11.6.2 Channel Watchdog

The Channel Watchdog is a PCP internal watchdog that optionally allows the user to ensure that the PCP will not become locked into executing a single channel due to an endless loop or unexpected software sequence. As each channel executes, the PCP maintains an internal count of the number of instructions that have been read from CMEM since the channel started. If the watchdog function is enabled (by programming PCP_CS.CWE = 1) and the internal instruction fetch counter reaches the threshold programmed by the user (programmed via PCP_CS.CWT), a PCP error is generated. The threshold setting (PCP_CS.CWT) is global to all channels. From this it follows that the threshold must be selected to be greater than the maximum number of instructions that can be fetched by any channel program, taking all channels into consideration. It should be noted that the instruction width of the PCP is 16 bits and that therefore execution of an instruction that is encoded into 32 bits (e.g. LDL.IL) will generate two CMEM instruction reads. That will therefore cause the internal watchdog counter to be incremented twice.

Note: Enabling the Channel Watchdog function (PCP_CS.CWE = 1) with a threshold of zero (PCP_CS.CWT = 0) is an invalid setting and will cause a PCP error event whenever any interrupt request is received by the PCP.

11.6.3 Invalid Opcode

The PCP includes the Invalid Opcode mechanism to check that each instruction fetched from CMEM is a legal instruction. If the PCP attempts to execute an illegal instruction, then a PCP error is generated.

Note: The DEBUG instruction must be only used in DEBUG mode otherwise it will be considered to be an illegal operation and will generate an IOP error.

11.6.4 Instruction Address Error

An Instruction Address Error is generated if the PCP attempts to execute an instruction from an illegal address. An address is considered to be illegal if:

- The address is outside the available CMEM area (see [Page 11-125](#) for the CMEM size implemented in this derivative)

and/or

- The specified address could not be contained in the 16-bit PC (i.e. an address calculation yielded a 16-bit unsigned overflow).

Peripheral Control Processor (PCP)

The second of these cases can result from an address calculation either from the execution of a PC relative jump instruction (either a JC, JC.I, or JL instruction), or the PC being incremented following execution of the previous instruction.

11.7 Instruction Set Overview

The following sections present an overview of the instruction set and the available addressing modes of the PCP in the TC1796.

11.7.1 DMA Primitives

Table 11-3 DMA Transfer Instructions

DMA Transfer	BCOPY	Move block of data value from FPI Bus source address location to FPI Bus destination address location. Optionally increment or decrement source and destination pointer registers. Optionally repeat instruction until counter CNT1 reaches 0.
	COPY	Move value from FPI Bus source address location to FPI Bus destination address location. Optionally increment or decrement source and destination pointer registers. Optionally repeat instruction until counter CNT1 reaches 0.

11.7.2 Load and Store

Note: If a conditional instruction's condition code is false, the operation will be treated as a "No Operation". Register values will not be changed and the flags will not be updated.

Table 11-4 Load and Store Instructions

Load	LD.F	Load value from FPI Bus address location into register (FPI Bus address = register content)
	LD.I	Load immediate value into register
	LD.IF	Load value from FPI Bus address location into register (FPI Bus address = register content + immediate offset)
	LD.P	Load value from PRAM address location into register (PRAM address = DPTR + register offset)
	LD.PI	Load value from PRAM address location into register (PRAM address = DPTR + immediate offset)
	LDL.IL	Load 16-bit immediate value into lower bits [15:0] of register
	LDL.IU	Load 16-bit immediate value into upper bits [31:16] of register
Store	ST.F	Store register value to FPI Bus address location (FPI Bus address = register content)
	ST.IF	Store register value to FPI Bus address location (FPI Bus address = register content + immediate offset)
	ST.P	Store register value to PRAM address location (PRAM address = DPTR + register offset)
	ST.PI	Store register value to PRAM address location (PRAM address = DPTR + immediate offset)
Move	MOV	Conditionally move register value to register

11.7.3 Arithmetic and Logical Instructions

Arithmetic instructions that are fully register-based execute conditionally depending on the specified Condition Code A (see [Page 11-73](#)). All other arithmetic instructions such as PRAM (.PI), indirect (.I), and FPI (.F and .IF) execute unconditionally.

Note: If a conditional instruction's condition code is false, the operation will be treated as a "No Operation". Register values will not be changed and the flags will not be updated.

Table 11-5 Arithmetic Instructions

Add	ADD	Add register to register (conditionally)
	ADD.I	Add immediate value to register
	ADD.F	Add content of FPI Bus address location to register (byte, half-word or word)
	ADD.PI	Add content of PRAM address location to register
Subtract	SUB	Subtract register from register (conditionally)
	SUB.I	Subtract immediate value from register
	SUB.F	Subtract content of FPI Bus address location from register (byte, half-word or word)
	SUB.PI	Subtract content of PRAM address location from register
Compare	COMP	Compare register to register (conditionally)
	COMP.I	Compare immediate value to register
	COMP.F	Compare content of FPI Bus address location to register (byte, half-word or word)
	COMP.PI	Compare content of PRAM address location to register
Negate	NEG	Negate register (2's complement, conditionally)

Logical instructions that are fully register-based execute conditionally as determined by the specified Condition Code A. All other logical instructions, such as PRAM (.PI), indirect (.I), and FPI (.F and .IF) execute unconditionally.

Table 11-6 Logical Instructions

Logical And	AND	Register AND register (conditionally)
	AND.F	Content of FPI Bus address location AND register (byte, half-word or word)
	AND.PI	Content of PRAM address location AND register
	MCLR.PI	Clear specified bits within a PRAM location
Logical Or	OR	Register OR register (conditionally)
	OR.F	Content of FPI Bus address location OR register (byte, half-word or word)
	OR.PI	Content of PRAM address location OR register
	MSET.PI	Set specified bits within a PRAM location
Logical Exclusive-Or	XOR	Register XOR register (conditionally)
	XOR.F	Content of FPI Bus address location XOR register (byte, half-word or word)
	XOR.PI	Content of PRAM address location XOR register
Logical Not	NOT	Invert register (1's complement, conditionally)
Shift	SHL	Shift left register
	SHR	Shift right register
Rotate	RL	Rotate left register
	RR	Rotate right register
Prioritize	PRI	Calculate position of first set bit (1-bit) in register, from left

Note: If a conditional instruction's condition code is false, the operation will be treated as a "No Operation". Register values will not be changed and the flags will not be updated.

11.7.4 Bit Manipulation

All bit manipulation instructions except INB are executed unconditionally. If conditional bit handling is required, INB should be used.

Table 11-7 Bit Manipulation Instructions

Set Bit	SET	Set bit in register
	SET.F	Set bit in FPI Bus address location
Clear Bit	CLR	Clear bit in register
	CLR.F	Clear bit in FPI Bus address location
Insert Bit	INB	Insert carry flag into register (position given by content of a register)
	INB.I	Insert carry flag into register (position given by immediate value)
Check Bit	CHKB	Set carry flag depending on value of specified register bit

11.7.5 Flow Control

Table 11-8 describes flow control instructions of the PCP in the TC1796.

Table 11-8 Flow Control Instructions

Jump	JC	Jump conditionally to PC + short immediate offset address
	JC.A	Jump conditionally to immediate absolute address
	JC.I	Jump conditionally to PC + register offset address
	JC.IA	Jump conditionally to register absolute address
	JL	Jump unconditionally to PC + long immediate offset address
Exit Channel	EXIT	Unconditionally exit channel program execution (optionally generate interrupt request and/or inhibit channel)
No Operation	NOP	Low-power No-Operation
Debug	DEBUG	Conditionally generate debug event (optionally stop channel program execution)

11.7.6 Addressing Modes

The PCP needs to address locations in memory in different ways, as determined by the type of memory being accessed and the type of action being performed on that location.

11.7.6.1 FPI Bus Addressing

All FPI Bus accesses from the PCP are indirect to some extent. The main indirect addressing on the FPI Bus uses a 32-bit absolute address located in the GPR indicated in the instruction. This address must be properly aligned for the type of data access – byte, half-word or word. If it is not aligned, the results are undefined.

- Effective Target Address [31:0] = $\langle R[a] \rangle$

where a is the number of the register, for instance, R2. Instructions using this address mode are indicated through the “.F” suffix.

For indirect-plus-immediate addressing on the FPI Bus, the 32-bit absolute address located in the GPR indicated in the instruction is added to the immediate 5-bit offset value encoded in the instruction. This address must be properly aligned for the type of data access (byte, half-word or word). If it is not aligned, the results are undefined.

- Effective Target Address [31:0] = $\langle R[a] \rangle + \#offset5$

where a is the number of the register and $\#offset5$ is a 5-bit immediate offset value. Instructions using this addressing mode are indicated through the “.IF” suffix (only available for load and store, LD.IF and ST.IF).

This addressing mode is particularly useful for managing peripherals, where the peripheral base address is held in $R[a]$, and the offset can index directly into a specific control register.

The BCOPY and COPY instructions use the indirect absolute addressing with predefined PCP registers. Register R4 is used as the source address pointer, while R5 represents the destination address pointer.

- Effective Source Address [31:0] = $\langle R4 \rangle$
- Effective Destination Address [31:0] = $\langle R5 \rangle$

Note: All FPI Bus accesses by the PCP are performed in Supervisor mode.

Note: The PCP is not allowed to access its own registers via instructions executed in the PCP.

11.7.6.2 PRAM Addressing

The PRAM is always addressed indirectly by the PCP. The normal address used is the value of the R7.DPTR field (8 bits) concatenated with an immediate 6-bit offset value encoded in the instruction, yielding a 14-bit word address. This enables access to 16 Kwords (64 Kbytes). Because R7.DPTR is part of a channel program's context, a channel program may alter the DPTR value at any time.

- Effective PRAM Address[13:0] = <R7.DPTR> << 6 + #offset6

Instructions using this addressing mode are indicated through the “.PI” suffix.

To provide effective indexing into large tables or stores of data, an alternate form of indirect addressing can also be used on load and store operations to PRAM. The value of the DPTR field (8 bits) is concatenated with the least significant 6 bits of R[a], again yielding a 14-bit word address. The most significant bits [31:6] of R[a] are ignored.

- Effective PRAM Address[13:0] = <R7.DPTR> << 6 + <R[a][5:0]>

Instructions using this addressing mode are indicated through the “.P” suffix (load and store only, LD.P and ST.P).

11.7.6.3 Bit Addressing

Single bits can be addressed in the PCP GPRs or in FPI Bus address locations. A 5-bit value indicates the location of a bit in the register specified in the instruction. This bit location is either given through an immediate value in the instruction or through the lower five bits of a second register (indirect addressing).

- Effective Bit Position[31:0] = #imm5
- Effective Bit Position[31:0] = <R[a][5:0]>

The immediate bit addressing is used by instructions SET and CLR and their variants as well as by INB.I and CHKB. Indirect bit addressing is used by the INB instruction only.

11.7.6.4 Flow Control Destination Addressing

The jump instructions are split into two groups: PC-relative jumps, and jumps to an absolute address.

For PC-relative jumps, the destination address is a positive or negative offset from the PC of the next instruction. The offset is either contained in the lower 16 bits of a register (the upper 16 bits are ignored), or is given as immediate value of the instruction. The immediate values are sign-extended to 16 bits. If the effective jump address is outside the available CMEM area (or the jump address calculation caused an overflow), then a PCP error condition has occurred.

- Effective JUMP Address[15:0] = NextPC + Signed(R[a][15:0]); +/- 32K instructions
- Effective JUMP Address[15:0] = NextPC + Sign-Extend(#offset10);
+/- 512 instructions

Peripheral Control Processor (PCP)

- Effective JUMP Address[15:0] = NextPC + Sign-Extend(#offset6);
+/- 32 instructions

The function NextPC indicates the instruction that would be fetched next by the program counter. Instructions using this addressing are JL, JC and JC.I.

For absolute addressing, the actual address in CMEM where program flow is to resume is either an immediate value #imm16 in the CMEM location immediately following the jump instruction, or it is contained in the lower 16 bits of a register. If the value is greater than the PC size implemented, an error condition has occurred.

- Effective JUMP Address[15:0] = #imm16
- Effective JUMP Address[15:0] = <R[a]>

Instructions using these addressing modes are JC.A (immediate absolute address) and JC.IA (indirect absolute address).

11.8 Accessing PCP Resources from the FPI Bus

Any FPI Bus master (on the TC1796's System Peripheral Bus) can access the three distinct PCP address ranges from the FPI Bus side. Normally, the CPU initializes the control registers via FPI Bus access. Thereafter, the PCP should not access its control registers itself through PCP instructions. Apart from the access via FPI Bus, there is no direct way to the PCP control registers.

Accesses to the PCP control and status register, the PRAM, and the CMEM are detailed in the following sections.

11.8.1 Access to the PCP Control Registers

FPI Bus accesses to the PCP control registers must always be performed in Supervisor Mode with word accesses; byte or half-word accesses will result in a bus error.

All PCP control registers can be read at any time. Write operations are only possible to the PCP_CS register, all other register are read-only. Register PCP_CS can be optionally Endinit-protected via bit PCP_CS.EIE (see [Page 11-56](#)).

11.8.2 Access to the PRAM

FPI Bus accesses to the PRAM must always be performed with word accesses; byte or half-word accesses will result in a bus error.

Attention needs to be paid when accessing the CSAs and data sections of the PCP channel programs. The location of a specific channel's CSA is dependent on the chosen Context Model, Full, Small or Minimum context. [Table 11-9](#) shows these addresses.

Table 11-9 FPI Bus Access to CSAs

Channel	Full Context	Small Context	Minimum Context
0 (see note)	PRAM Base Address + 00 _H	PRAM Base Address + 00 _H	PRAM Base Address + 00 _H
1	PRAM Base Address + 20 _H	PRAM Base Address + 10 _H	PRAM Base Address + 08 _H
2	PRAM Base Address + 40 _H	PRAM Base Address + 20 _H	PRAM Base Address + 10 _H
3	PRAM Base Address + 60 _H	PRAM Base Address + 30 _H	PRAM Base Address + 18 _H
n	PRAM Base Address + n × 20 _H	PRAM Base Address + n × 10 _H	PRAM Base Address + n × 08 _H

Peripheral Control Processor (PCP)

Note: Since channel 0 is not defined (no service request with SRPN = 0), the first area is not an actual CSA. It is recommended that this area should not be used by PCP channel programs.

The FPI Bus address of a word location pointed to by the Data Pointer R7_DPTR is calculated by the following formula:

- Effective FPI Bus address[31:0] = (PRAM Base Address) + (<DPTR> << 6)

11.8.3 Access to the CMEM

FPI Bus accesses to the CMEM must always be performed with word accesses; byte or half-word accesses will result in a bus error.

When using a channel entry table, the FPI Bus address of a specific channel's entry location is given by the following formula:

- Effective FPI Bus address[31:0] = (CMEM Base Address) + $04_H \times$ channel number

The FPI Bus address of an instruction pointed to by the PCP program counter, PC, is calculated by the following formula:

- Effective FPI Bus address[31:0] = (CMEM Base Address) + <PC> << 1

11.9 Debugging the PCP

For debugging the PCP, a special instruction, `DEBUG`, is provided. This instruction can only be used when the PCP is in Debug Mode. It can be placed at important locations inside the code to track and trace program execution. The execution of the instruction depends on a condition code specified with the instruction. The actions programmed for this instruction will only take place if the specified condition is true.

The following actions are performed when the `DEBUG` instruction is executed and the condition code is true:

- Store the current PC, i.e. the address of the `DEBUG` instruction, in register `PCP_ES.EPC`
- Store the current channel number in register `PCP_ES.EPN`

In addition, the following operations can be programmed through fields in the `DEBUG` instruction:

- Optionally stop the channel program execution (instruction field `SDB`)
- Optionally generate an external debug event at pin `BRKOUT` (instruction field `EDA`)
- Optionally prevent the PCP from executing any further channel programs (instruction field `DAC`)
- Optionally cause the PCP to decrement the PC prior to saving the channel context (instruction field `RTA`)

If the `DEBUG` instruction is programmed to stop the channel program execution, the action taken by the PCP depends on the value of the `RTA` instruction field:

- If `RTA = 0`, the PCP disables further invocations of the current channel through clearing bit `R7.CEN`, and then performs a context save. The execution of this channel is stopped at the point of the `DEBUG` instruction. If the `DAC` instruction field = 0, the PCP will continue to operate, accepting service requests for other channels as they arise. Since the stopped channel was disabled before saving its context, service requests for this channel will result in an error exit (see [Page 11-30](#)). When re-enabling the channel, its enable bit `CEN` in the saved context location `CR7` must be set.
- If `RTA = 1`, the PCP does not modify bit `R7.CEN` (i.e. the channel remains enabled), decrements the PC (so that it again points to the `DEBUG` instruction), and then performs a context save. The execution of this channel is stopped at the point of the `DEBUG` instruction. If the `DAC` instruction field = 0, the PCP will continue to operate, accepting service requests for channels as they arise. Since the stopped channel was not disabled before saving its context, service requests for this channel will not result in an error exit, but will simply cause re-execution of the `DEBUG` instruction and hence a repeat of the channel exit.

Note: When a channel is stopped by `DEBUG`, the context of the stopped channel will be saved to the appropriate region of the CSA before the channel terminates. Where a Small or Minimum Context model is being used, the values of the GPRs not

Peripheral Control Processor (PCP)

included in the context will not be saved, and indeed these register values may be changed by the operation of another active channel. In this case, the required registers should be explicitly saved to PRAM by store instructions prior to execution of the DEBUG instruction.

If the DEBUG instruction is programmed to stop all channel program execution, the PCP disables further invocations of any channel by clearing bit PCP_CS.EN. The execution of this channel is only stopped according to the SDB instruction field value. The PCP will only start to reaccept service requests when PCP_CS.EN is written to 1.

Note: The DEBUG instruction must be only used in DEBUG mode; otherwise it will generate an IOP error.

Note: If PCP_CS.RCB = 0 (Channel Resume Mode), then the channel program will begin executing at whatever PC is restored from the context location CR7.PC. If PCP_CS.RCB = 1 (Channel Restart Mode), then the channel program is forced to always start at its channel entry table location, no matter what the restored context value is for the PC. This means that in Channel Restart Mode, it is not possible to restart the channel program from where it was halted by the debug event. It is recommended that when using Channel Restart Mode, the user should also program all EXIT instructions with the "EP = 0" setting to allow selection of Channel Resume Mode for debugging without changing operation of the channel programs.

11.10 PCP Registers

The PCP can be viewed as being a peripheral on the FPI Bus. As with any other peripheral, there are control registers, normally set by the CPU acting as an external FPI Bus master to the PCP during initialization. Control registers select the operating modes of the PCP, and status registers provide information about the current state of the PCP to the external FPI Bus master. [Figure 11-12](#) gives an overview of the PCP registers.

PCP Register Overview

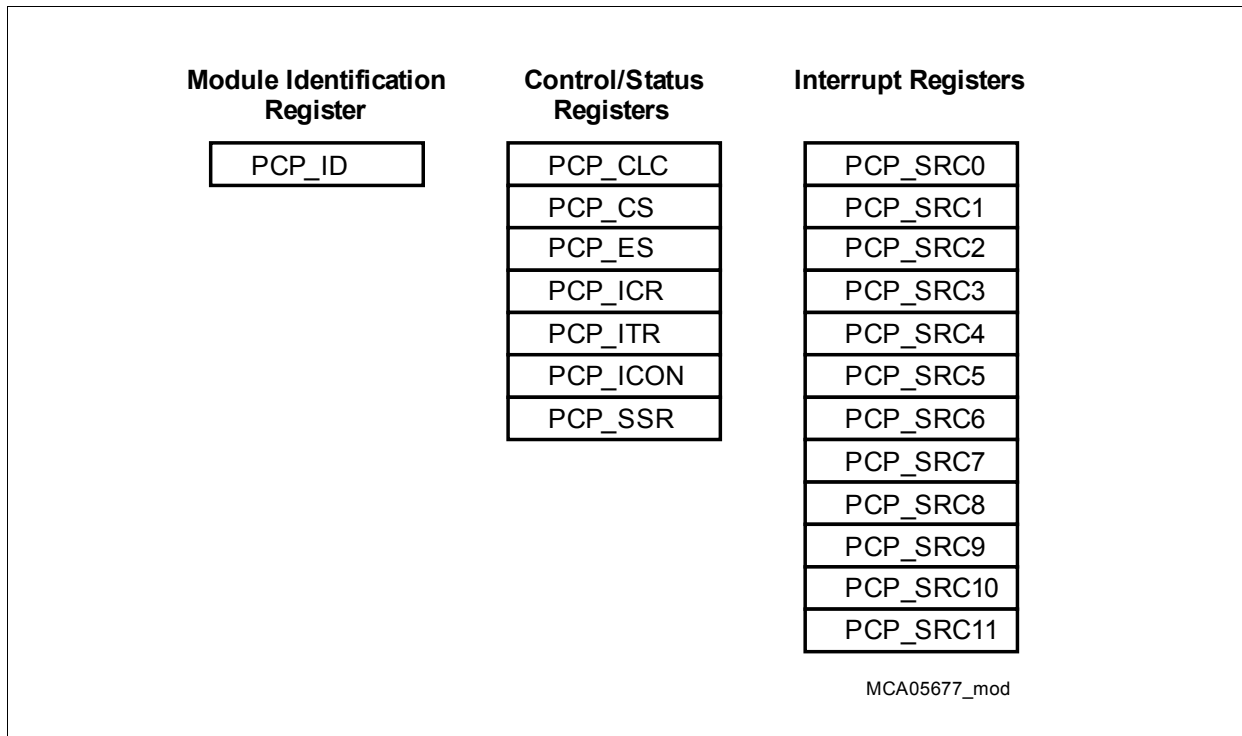


Figure 11-12 PCP Registers

The complete address map of the PCP is described in [Table 18-25](#) on [Page 18-77](#) of this TC1796 System Units (Vol. 1 of 2) User's Manual.

Table 11-10 Registers Address Space - PCP Registers

Module	Base Address	End Address	Note
PCP	F004 3F00 _H	F004 3FFF _H	-

Table 11-11 Registers Overview - PCP Registers

Register Short Name	Register Long Name	Offset Address ¹⁾	Description see
PCP_CLC	PCP Clock Control Register	0000 _H	Page 11-55
PCP_ID	PCP Module Identification Register	0008 _H	Page 11-54
PCP_CS	PCP Control/Status Register	0010 _H	Page 11-56
PCP_ES	PCP Error/Debug Status Register	0014 _H	Page 11-59
PCP_ICR	PCP Interrupt Control Register	0020 _H	Page 11-61
PCP_ITR	PCP Interrupt Threshold Control Register	0024 _H	Page 11-63
PCP_ICON	PCP Interrupt Configuration Register	0028 _H	Page 11-64
PCP_SSR	PCP Stall Status Register	002C _H	Page 11-66
PCP_SRC11	PCP Service Request Control Register 11	00D0 _H	Page 11-70
PCP_SRC10	PCP Service Request Control Register 10	00D4 _H	
PCP_SRC9	PCP Service Request Control Register 9	00D8 _H	
PCP_SRC8	PCP Service Request Control Register 8	00DC _H	
PCP_SRC7	PCP Service Request Control Register 7	00E0 _H	
PCP_SRC6	PCP Service Request Control Register 6	00E4 _H	
PCP_SRC5	PCP Service Request Control Register 5	00E8 _H	
PCP_SRC4	PCP Service Request Control Register 4	00EC _H	
PCP_SRC3	PCP Service Request Control Register 3	00F0 _H	Page 11-68
PCP_SRC2	PCP Service Request Control Register 2	00F4 _H	Page 11-67
PCP_SRC1	PCP Service Request Control Register 1	00F8 _H	
PCP_SRC0	PCP Service Request Control Register 0	00FC _H	

1) The absolute register address is calculated as follows:
Module Base Address ([Table 11-10](#)) + Offset Address (shown in this column)

Accessing of Control Registers

The control registers are accessible by any master via the FPI Bus. The control registers must be configured at initialization and then left unaltered. This is typically done by the CPU. The only setting that can be dynamically modified is the PCP_CS.EN bit. All other bits must only be modified when PCP_CS.EN = 0 and PCP_CS.RS = 0.

The PCP control and status registers are accessible only to the CPU when it is operating in Supervisor mode. PCP control and status registers must be accessed with 32-bit read and write operations only.

Peripheral Control Processor (PCP)

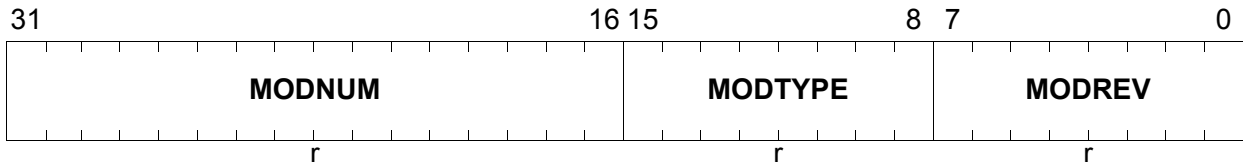
11.10.1 Module Identification Register, PCP_ID

The PCP Module Identification Register ID contains read-only information about the PCP module version.

PCP_ID

PCP Module Identification Register (008_H)

Reset Value: 0020 C0XX_H



Field	Bits	Type	Description
MODREV	[7:0]	r	Module Revision Number MODREV defines the module revision number. The value of a module revision starts with 01 _H (first revision).
MODTYPE	[15:8]	r	Module Type This bit field defines the module as a 32-bit module: C0 _H
MODNUM	[31:16]	r	Module Number Value This bit field defines the module identification number for the PCP: 0020 _H

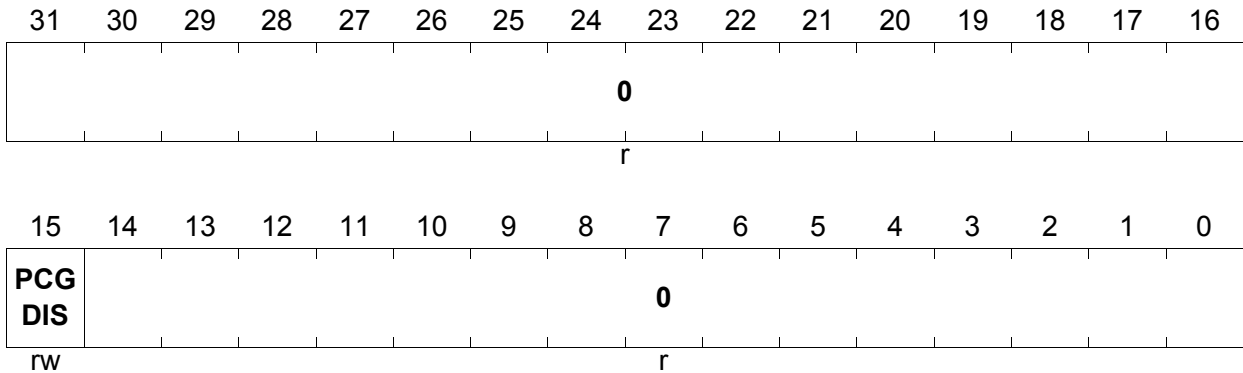
11.10.2 PCP Clock Control Register, PCP_CLC

PCP_CLC

PCP Clock Control Register

(00_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PCGDIS	15	rw	Clock Gating Disable Bit Allows clock gating to be disabled. 0 _B PCP Internal Clock stops when PCP is idle (default after reset) 1 _B PCP Internal Clock always runs
0	[14:0], [31:16]	r	Reserved Read as 0; should be written with 0.

Peripheral Control Processor (PCP)

11.10.3 PCP Control and Status Register, PCP_CS

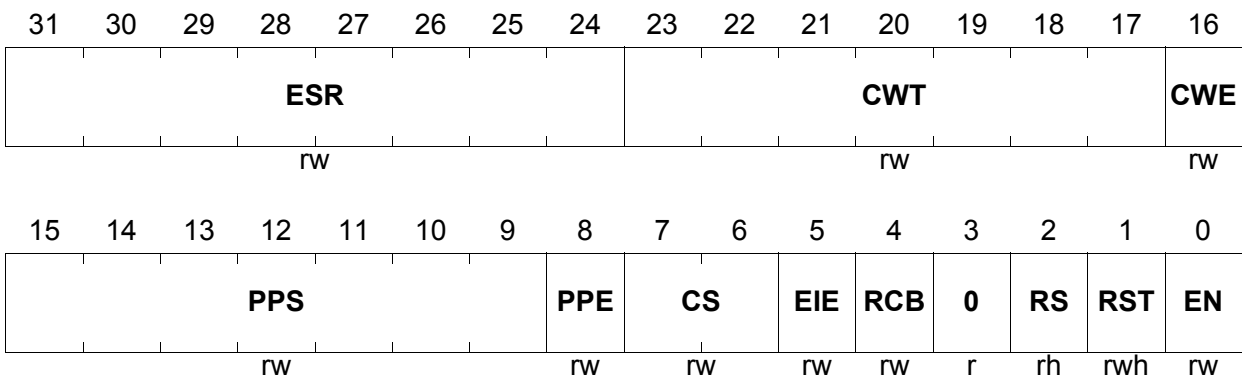
This register can be Endinit-protected via bit EIE.

PCP_CS

PCP Control/Status Register

(10_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
EN	0	rw	PCP Enable 0 _B PCP is disabled for operation (default) 1 _B PCP is enabled for operation <i>Note: This bit does not enable/disable clocks for power saving. It stops the PCP from accepting new service requests.</i>
RST	1	rwh	PCP Reset Request 0 _B No PCP software reset operation is requested 1 _B A PCP software reset is requested. Halt any operating channel. Reset all control registers to default values. Reset PCP state to default value. RST is always read as 0, but is written with 1 in order to initiate a reset. See also Page 11-126 .
RS	2	rh	PCP Run/Stop Status Flag 0 _B PCP is stopped or idle (default) 1 _B PCP is currently running
0	3	r	Reserved Read as 0; should be written with 0.

Peripheral Control Processor (PCP)

Field	Bits	Type	Description
RCB	4	rw	<p>Channel Start Mode Control</p> <p>0_B Channel resume operation mode selected; channel start PC is taken from restored context</p> <p>1_B Channel restart operation mode selected; channel start PC is derived from the requested channel number (= priority number of service request)</p> <p><i>Note: This is a global control bit and applies to all channels.</i></p>
EIE	5	rw	<p>Endinit Enable</p> <p>0_B Writes to PCP_CS are disabled if Endinit-protection is enabled.</p> <p>1_B Writes to PCP_CS are enabled if Endinit-protection is enabled.</p>
CS	[7:6]	rw	<p>Context Size Selection</p> <p>00_B Use Full Context for all channels</p> <p>01_B Use Small Context for all channels</p> <p>10_B Use Minimum Context for all channels</p> <p>11_B Reserved</p>
PPE	8	rw	<p>PRAM Partitioning Enable</p> <p>0_B PRAM is not partitioned</p> <p>1_B PRAM is partitioned</p> <p><i>Note: When partitioned, the PRAM is divided into two areas (CSA and remainder). A PCP error will be generated on an inappropriate action in either region (PCP write operation with a target address in the CSA, or context restore from outside the CSA).</i></p>

Peripheral Control Processor (PCP)

Field	Bits	Type	Description
PPS	[15:9]	rw	<p>PRAM Partition Size</p> <p>0_D Default, only allowed with PPE = 0</p> <p>1_D 000CSA contains 3 context save regions</p> <p>..._D ...</p> <p>n_D CSA contains 1 + 2 × n context save regions</p> <p><i>Note: The actual size of the CSA (in words) is given by the formula (2 × n + 1) × m, where m is the number of registers in the selected Context Model.</i></p> <p><i>If PPE = 1 and the PCP attempts to perform a data write to PRAM addresses below the CSA, an error condition has occurred.</i></p> <p><i>This setting also controls the maximum channel number (MCN) used in system. If n = 1, the maximum channel number is MCN = 2 × n. If the SRPN is greater than MCN, an error condition has occurred.</i></p> <p><i>For example, setting PPS to n = 3 will give a CSA containing 7 context save regions. As channel 0 cannot be used and MCN = 6, channels 1 to 6 are allowed.</i></p>
CWE	16	rw	<p>Channel Watchdog Enable</p> <p>0_B Disable Channel Watchdog</p> <p>1_B Enable Channel Watchdog</p> <p><i>Note: When enabled, the Channel Watchdog counts the number of instructions executed since the channel started. If this number exceeds the Channel Watchdog Threshold, a PCP error is generated.</i></p>
CWT	[23:17]	rw	<p>Channel Watchdog Threshold</p> <p>0_D Reserved</p> <p>1_D Threshold = 16 instructions</p> <p>..._D ...</p> <p>127_D Threshold = 16 × '127' instructions</p>
ESR	[31:24]	rw	<p>Error Service Request Number</p> <p>SRPN for interrupt to CPU on an error condition.</p> <p>00_H No interrupt request posted (default)</p> <p>value n_H Post n as SRNP interrupt to CPU on an error condition (n not equal 00_H)</p>

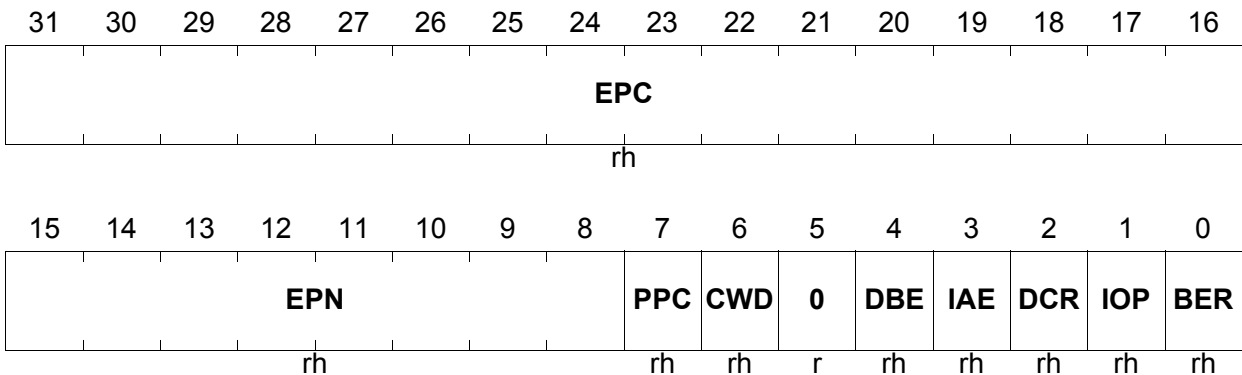
Peripheral Control Processor (PCP)

11.10.4 PCP Error/Debug Status Register, PCP_ES

This is a read-only register, providing state information about error and debug conditions.

PCP_ES

PCP Error/Debug Status Register (14_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
BER	0	rh	<p>Bus Error Flag</p> <p>Set if the last error/debug event was an error generated by an FPI Bus error or an invalid address access; otherwise, clear.</p> <p><i>Note: An FPI Bus error event does not cause the PCP to post an error interrupt to the CPU. An FPI Bus error interrupt is, however, generated by the FPI control logic.</i></p>
IOP	1	rh	<p>Invalid Opcode</p> <p>Set if the last error/debug event was an error generated by the PCP attempting to execute an Invalid Opcode (i.e. the value fetched from CMEM for execution by the PCP did not represent a valid instruction), otherwise clear.</p>
DCR	2	rh	<p>Disabled Channel Request Flag</p> <p>Set if the last error/debug event was an error generated by receipt of an interrupt request with an SRPN that attempted to start a disabled PCP channel; otherwise, clear.</p>

Peripheral Control Processor (PCP)

Field	Bits	Type	Description
IAE	3	rh	Instruction Address Error Set if the last error/debug event was an error generated by the PCP attempting to fetch an instruction from an address outside the implemented CMEM range as a result of a jump or branch instruction; otherwise, clear.
DBE	4	rh	Debug Event Flag Set if the last error/debug event was a debug event. <i>Note: A debug event does not cause the posting of an interrupt to the CPU.</i>
0	5	r	Reserved Read as 0.
CWD	6	rh	Channel Watchdog Triggered Set if the last error/debug event was an error generated by a channel program attempting to execute more instructions than allowed by PCP_CS.CWT.
PPC	7	rh	PRAM Partitioning Check Set if the last error/debug event was an error generated by a channel program attempting to perform a write to a PRAM address within the CSA, or receipt of an interrupt request that would have caused a context restore from outside the CSA.
EPN	[15:8]	rh	Error Service Request Priority Number Channel number of the channel that was operating when the last error/debug event occurred. The value stored is the SRPN which invoked this channel (= channel number), NOT the current PCP priority number stored in field CPPN in register PICR. Default = 00 _H
EPC	[31:16]	rh	Error PC PC value of the instruction that was executing when an error or debug event occurred. Default = 0000 _H .

Note: An interrupt request with the SRPN held in PCP_CS.ESR is posted to the CPU whenever a PCP error event, other than an FPI Bus error occurs. FPI Bus error interrupt generation is automatically handled by the FPI Bus control logic, rather than by the PCP. The execution of a DEBUG instruction is not classed as an error event and does not therefore generate an interrupt request to the CPU. The entire

Peripheral Control Processor (PCP)

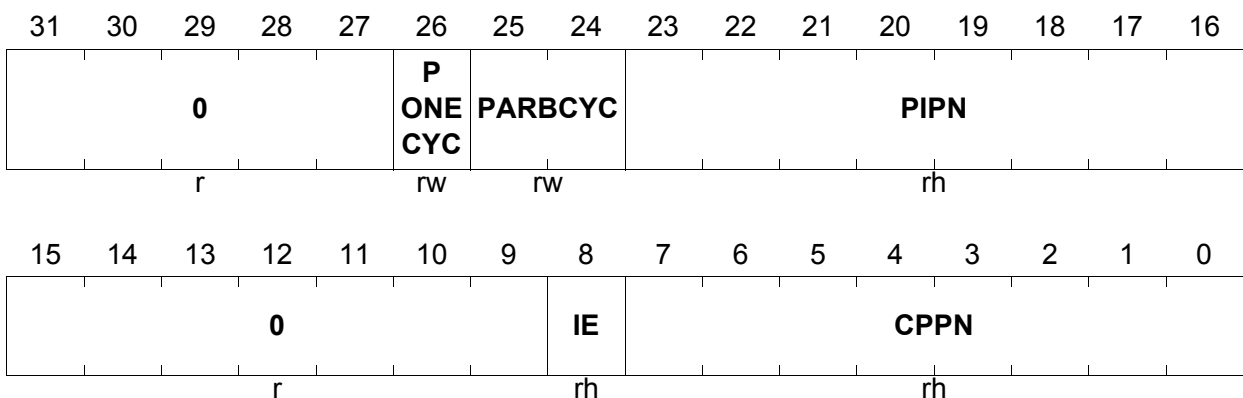
contents of the register are updated whenever there is a debug or an error event detected (i.e. all status/error bits, other than the bit representing the last PCP error/debug event, are cleared). This register therefore only provides a record of the last error/debug event encountered. The only way to clear this register is to reset the PCP.

11.10.5 PCP Interrupt Control Register, PCP_ICR

This register controls the operation of the PCP Interrupt Control Unit (PICU).

PCP_ICR

PCP Interrupt Control Register (20_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
CPPN	[7:0]	rh	Current PCP Priority Number This field indicates the current priority level of the PCP and is automatically updated by hardware on entry into an interrupt service routine.
IE	8	rh	Reserved
PIPN	[23:16]	rh	Pending Interrupt Priority Number This read-only field is updated by the PICU at the end of each arbitration process, and indicates the priority number of a pending request. PIPN is set to 00 _H when no request is pending and at the beginning of a new arbitration process.

Peripheral Control Processor (PCP)

Field	Bits	Type	Description
PARBCYC	[25:24]	rw	<p>Number of Arbitration Cycles Control</p> <p>This bit field controls the number of arbitration cycles used to determine the request with the highest priority. It follows the same coding scheme as described for the CPU interrupt arbitration.</p> <p>00_B Four arbitration cycles (default) 01_B Three arbitration cycles 10_B Two arbitration cycles 11_B One arbitration cycle</p>
PONECYC	26	rw	<p>Clocks per Arbitration Cycle Control</p> <p>This bit determines the number of clocks per arbitration cycle.</p> <p>0_B Two clocks per arbitration cycle (default) 1_B One clock per arbitration cycle</p>
0	[15:9], [31:27]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

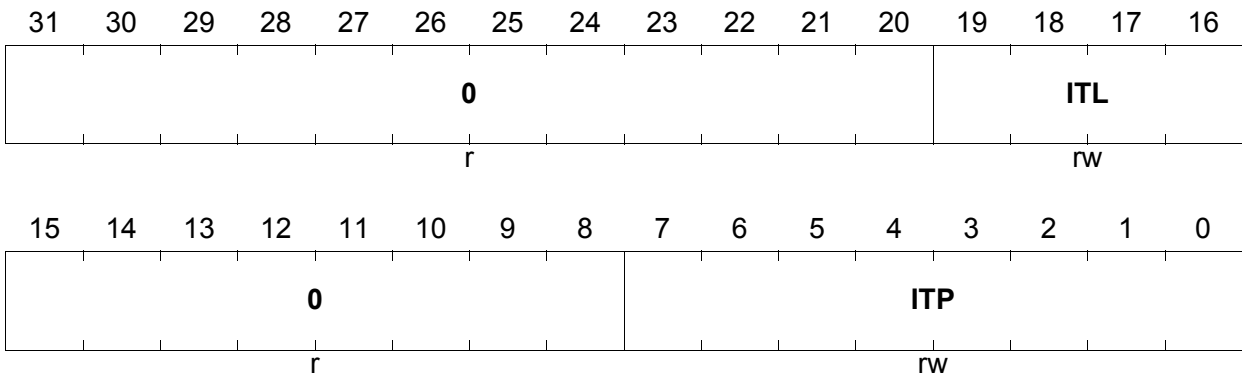
11.10.6 PCP Interrupt Threshold Register, PCP_ITR

PCP_ITR

PCP Interrupt Threshold Control Register

(24_H)

Reset Value: 0000 0000_H



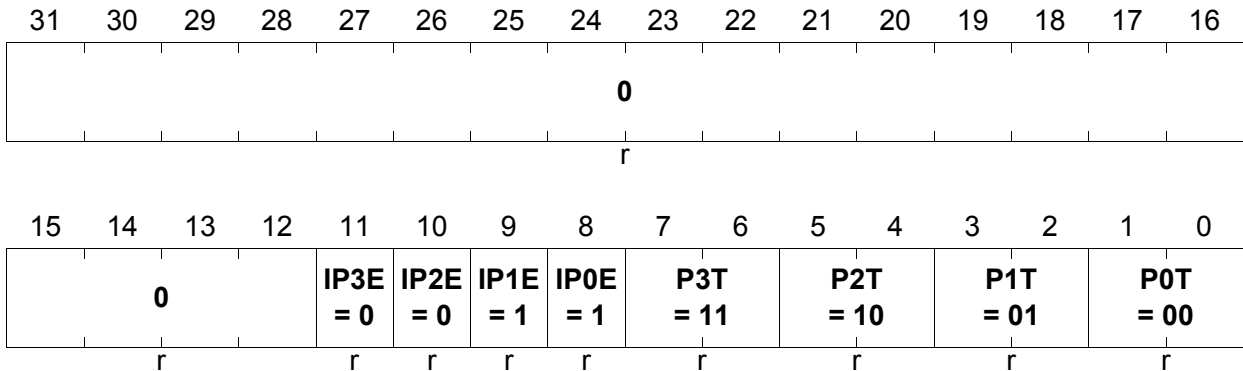
Field	Bits	Type	Description
ITP	[7:0]	rw	<p>PCP Interrupt Threshold Service Request Priority Number</p> <p>This field contains the interrupt priority that is to be posted to the interrupt queue associated with interrupt bus 0 when the threshold condition is reached (setting this value to 0 or disables the threshold detection mechanism).</p>
ITL	[19:16]	rw	<p>Interrupt Threshold Level</p> <p>This bit field specifies the number of active interrupt entries at which an warning interrupt should be issued to the interrupt queue associated with interrupt bus 0 (i.e. when the number of active port 0 interrupt requests stored in all SRCx registers reaches this value then an interrupt is posted to port 0 with the priority programmed into the ITP field). When ITL is programmed to 0 or is \geq the number of SRCx registers that can contain port 0 interrupt requests, the threshold warning mechanism is disabled).</p>
0	[15:8], [31:20]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

11.10.7 PCP Interrupt Configuration Register, PCP_ICON

PCP_ICON

PCP Interrupt Configuration Register (28_H)

Reset Value: 0000 03E4_H



Field	Bits	Type	Description
P0T	[1:0]	r	PCP Interrupt Bus 0 TOS Mapping This field reflects the TOS associated with interrupt bus 0 (CPU interrupt arbitration bus). The PCP should use this value in R6.TOS when it wishes to raise an interrupt request via interrupt bus 0.
P1T	[3:2]	r	PCP Interrupt Bus 1 TOS Mapping This field reflects the TOS associated with interrupt bus 1 (PCP interrupt arbitration bus). The PCP should use this value in R6.TOS when it wishes to raise an interrupt request to itself (the PCP is always connected to interrupt bus 1).
P2T	[5:4]	r	PCP Interrupt Bus 2 TOS Mapping This field reflects the TOS associated with interrupt bus 2. <i>Note: Interrupt bus 2 is not available in the TC1796.</i>
P3T	[7:6]	r	PCP Interrupt Bus 3 TOS Mapping This field reflects the TOS associated with interrupt bus 3. <i>Note: Interrupt bus 3 is not available in the TC1796.</i>
IP0E	8	r	PCP Interrupt Bus 0 Enable This bit reflects the status of interrupt bus 0 (CPU interrupt arbitration bus). Interrupt bus 0 is always enabled.

Peripheral Control Processor (PCP)

Field	Bits	Type	Description
IP1E	9	r	PCP Interrupt Bus 1 Enable This bit reflects the status of interrupt bus 1 (PCP interrupt arbitration bus). Interrupt bus 1 is always enabled.
IP2E	10	r	PCP Interrupt Bus 2 Enable This bit reflects the status of interrupt bus 2. Interrupt bus 2 is always disabled (not implemented in the TC1796).
IP3E	11	r	PCP Interrupt Bus 3 Enable This bit reflects the status of interrupt bus 3. Interrupt bus 3 is always disabled (not implemented in the TC1796).
0	[31:12]	r	Reserved Read as 0.

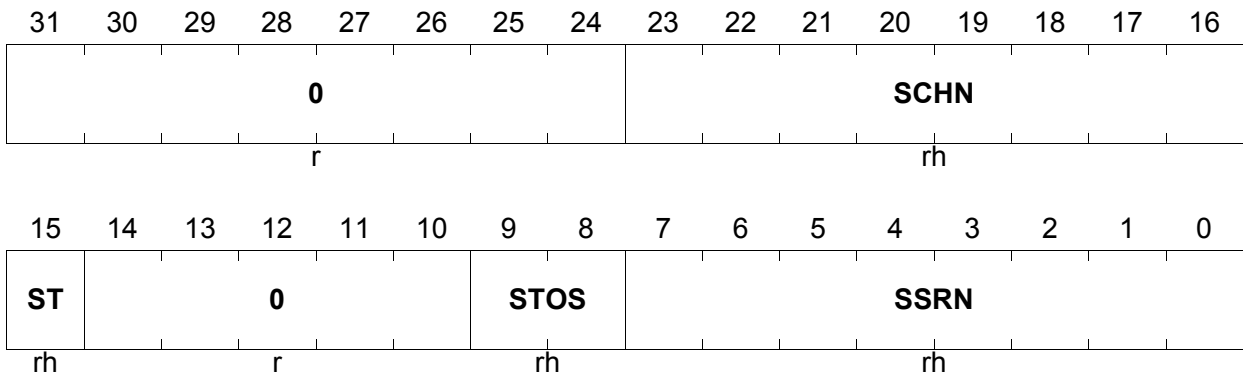
11.10.8 PCP Stall Status Register, PCP_SSR

PCP_SSR

PCP Stall Status Register

(2C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SSRN	[7:0]	rh	PCP Stalled Service Request Number This field shows the Service Request Number that was being posted when the last (or present) stall condition occurred. This bit field can only be cleared by a reset.
STOS	[9:8]	rh	PCP Stalled Type-of-Service This field shows the Type-Of-Service to which an interrupt was being posted that caused the last (or present) stall condition (i.e. the service request queue that was full when the PCP attempted to post a request to it). This bit field can only be cleared by a reset.
ST	15	rh	PCP Stalled Status This bit shows the stalled status of the PCP 0 _B PCP is not stalled 1 _B PCP is stalled
SCHN	[23:16]	rh	PCP Stalled Channel Number This field shows the channel number of the channel that was executing when the last (or present) stall condition occurred. This bit field can only be cleared by a reset.
0	[14:10], [31:24]	r	Reserved Read as 0.

Peripheral Control Processor (PCP)

11.10.9 PCP Service Request Control Registers, PCP_SRC[1:0]

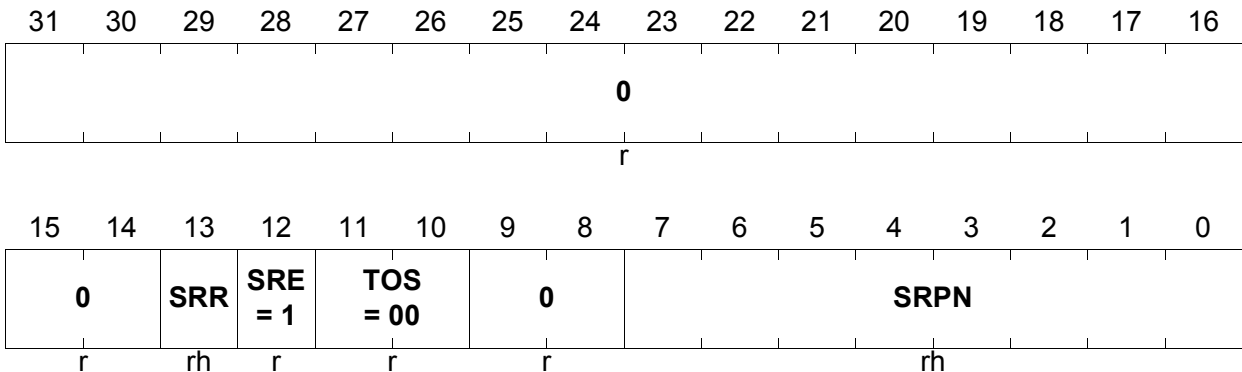
Service request nodes for interrupt bus 0 (CPU interrupt arbitration bus).

PCP_SRC_x (x = 0-1)

PCP Service Request Control Register x

(FC_{H-x*4_H})

Reset Value: 0000 1000_H



Field	Bits	Type	Description
SRPN	[7:0]	rh	PCP Node x Service Request Priority Number This number is automatically set by the PCP if it needs to place a service request on interrupt bus 0 (CPU interrupt arbitration bus). Default after reset is 00 _H .
TOS	[11:10]	r	PCP Node x Type-of-Service State Always read as 00 _B . This means TOS is associated with interrupt bus 0 (CPU interrupt arbitration bus).
SRE	12	r	PCP Node x Service Request Enable Always read as 1 (enabled).
SRR	13	rh	PCP Node x Service Request Flag 0 _B No service requested (default). 1 _B Valid active service requested.
0	[9:8], [15:14], [31:16]	r	Reserved Read as 0.

Peripheral Control Processor (PCP)

11.10.10 PCP Service Request Control Registers, PCP_SRC[3:2]

Service request nodes for interrupt bus 1 (PCP interrupt arbitration bus).

PCP_SRC2

PCP Service Request Control Register 2

(F4_H)

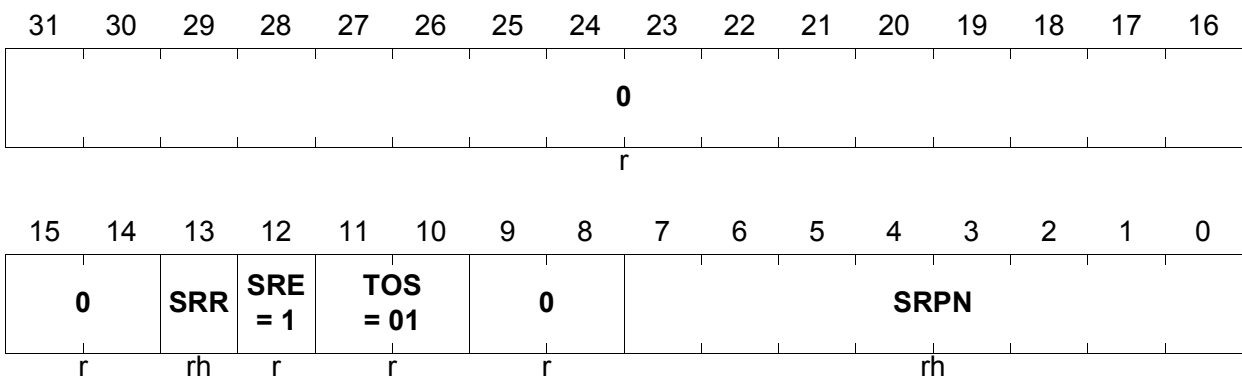
Reset Value: 0000 1400_H

PCP_SRC3

PCP Service Request Control Register 3

(F0_H)

Reset Value: 0000 1400_H



Field	Bits	Type	Description
SRPN	[7:0]	rh	PCP Node x Service Request Priority Number This number is automatically set by the PCP if it needs to place a service request on interrupt bus 1 (PCP interrupt arbitration bus). Default after reset is 00 _H .
TOS	[11:10]	r	PCP Node x Type-of-Service State Always read as 01 _B . This means TOS is associated with interrupt bus 1 (PCP interrupt arbitration bus).
SRE	12	r	PCP Node x Service Request Enable Always read as 1 (enabled).
SRR	13	rh	PCP Node x Service Request Flag 0 _B No service requested (default). 1 _B Valid active service requested.
0	[9:8], [15:14], [31:16]	r	Reserved Read as 0.

Peripheral Control Processor (PCP)

11.10.11 PCP Service Request Control Registers, PCP_SRC[8:4]

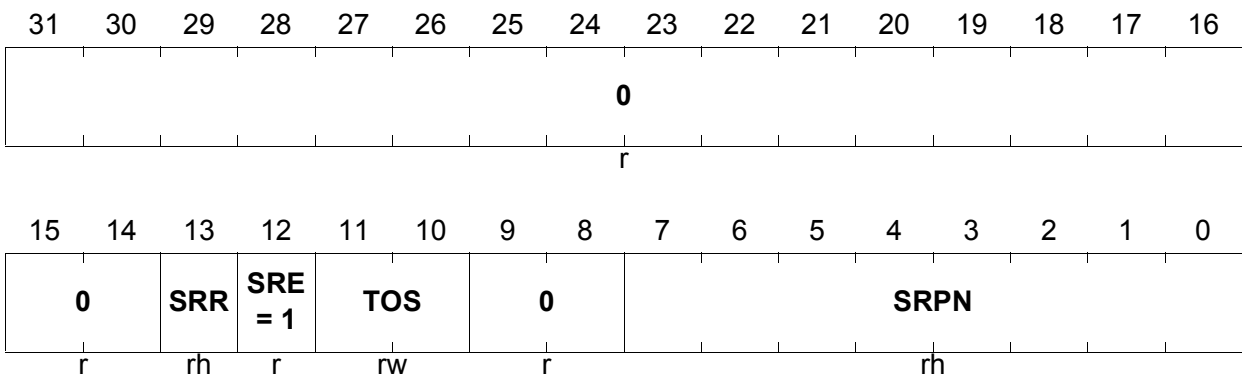
Service request nodes programmable for interrupt bus 0 (CPU interrupt arbitration bus) or 1 (PCP interrupt arbitration bus).

PCP_SRCx (x = 4-8)

PCP Service Request Control Register x

(FC_H-x*4_H)

Reset Value: 0000 1000_H



Field	Bits	Type	Description
SRPN	[7:0]	rh	PCP Node x Service Request Priority Number This number is automatically set by the PCP if it needs to place a service request on interrupt bus 0 (CPU interrupt arbitration bus) or 1 (PCP interrupt arbitration bus). Default after reset is 00 _H .
TOS	[11:10]	rw	PCP Node x Type-of-Service State/Control TOS value depends on the interrupt mapping that has been selected (see Page 11-119). This bit field must be written at PCP configuration time and should not subsequently modified while the PCP is operating.
SRE	12	r	PCP Node x Service Request Enable Always read as 1 (enabled).
SRR	13	rh	PCP Node x Service Request Flag 0 _B No service requested (default). 1 _B Valid active service requested.
0	[9:8], [15:14], [31:16]	r	Reserved Read as 0; should be written with 0.

Peripheral Control Processor (PCP)

11.10.12 PCP Service Request Control Registers, PCP_SRC[11:9]

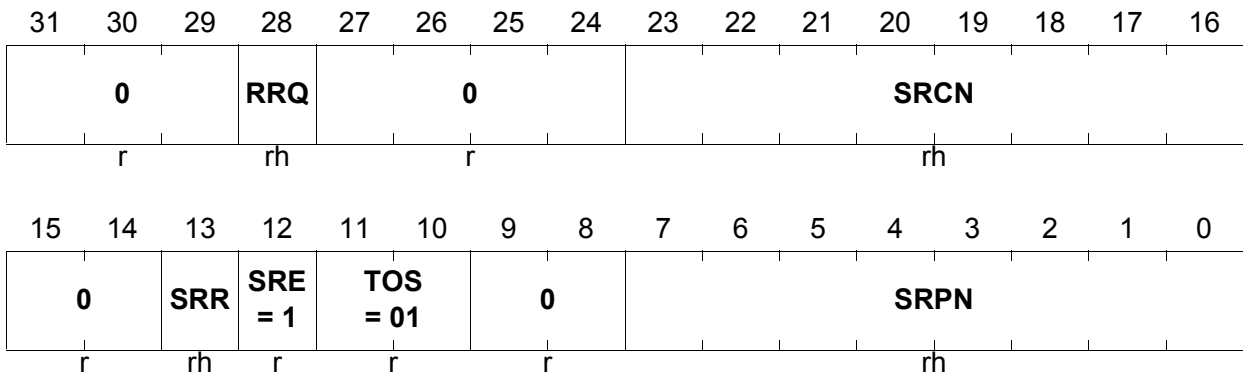
Service request nodes for interrupt bus 1 (PCP interrupt arbitration bus) with suspended interrupt capability.

PCP_SRCx (x = 9-11)

PCP Service Request Control Register x

(FC_H-x*4_H)

Reset Value: 0000 1400_H



Field	Bits	Type	Description
SRPN	[7:0]	rh	<p>PCP Node x Service Request Priority Number This number is automatically set by the PCP if it needs to place a service request on interrupt bus 0 (CPU interrupt arbitration bus) or 1 (PCP interrupt arbitration bus).</p> <p>When the PCP interrupt request contained was raised by the PCP Processor Core when executing an EXIT instruction then this bit field contains the SRPN value taken from R6 when the exit instruction was executed. When the PCP interrupt request was raised by the PCP Processor Core when suspending execution of a channel program in order to service a higher-priority interrupt then this bit field contains the CPPN value of the PCP when the interrupt was suspended.</p>
TOS	[11:10]	r	<p>PCP Node x Type-of-Service State Always read as 01_B. This means TOS is associated with interrupt bus 1 (PCP interrupt arbitration bus).</p>
SRE	12	r	<p>PCP Node x Service Request Enable Always read as 1 (enabled).</p>

Peripheral Control Processor (PCP)

Field	Bits	Type	Description
SRR	13	rh	PCP Node x Service Request Flag 0 _B No service requested (default) 1 _B Valid active service requested
SRCN	[23:16]	rh	PCP Node x Service Request Channel Number Channel Number Entry (default = 0). When the PCP interrupt request was raised by the PCP Processor Core when executing an EXIT instruction, then this bit field contains the SRPN value taken from R6 when the exit instruction was executed. When the PCP interrupt request was raised by the PCP Processor Core when suspending execution of a channel program in order to service a higher-priority interrupt then this bit field contains the channel number of the channel that was suspended.
RRQ	28	rh	PCP Node x Channel Restart Request Set when this service request register n contains an active service request that is associated with a suspended interrupt (i.e. a channel that has been interrupted by a higher-priority channel). 0 _B The interrupt is not suspended 1 _B The interrupt is suspended RRQ is always 0 when SRR is 0.
0	[9:8], [15:14], [27:24], [31:29]	r	Reserved Read as 0.

11.11 PCP Instruction Set Details

This section describes the instruction set architecture of the PCP in detail.

11.11.1 Instruction Codes and Fields

All PCP instructions use a common set of fields to describe such things as the source register, and the state of flags. Additionally, many instructions (including arithmetic and many flow control instructions), are conditionally executed.

The descriptions of the PCP instructions are based on the following conventions.

>>, <<	Shift left or right, respectively.
[]	Indirect access based on contents of brackets (de-reference).
#immNN	Immediate value encoded into an instruction with width NN.
#offsetNN	Address offset immediate value with width NN.
NextPC	The current executing instruction's address + 1. (The next instruction to be fetched.)
cc_A, cc_B	Condition Code CONDCA/CONDCB.

11.11.1.1 Conditional Codes

Many PCP instructions have the option of being executed conditionally. The condition code of an instruction is the field that specifies the condition to be tested before the instruction is executed. Depending on the type of instruction there are 8 or 16 condition codes available. The set of 8-condition codes is referred to as CONDCA, while the set of 16-condition codes is referred to as CONDCB. The condition codes are based on an equation of the Flags held in R7. See [Table 11-12](#).

Table 11-12 Condition Codes

Description	CONDCA/B	Test (Flag Bits)	Code	Mnemonic
Unconditional	A / B	–	0 _H	cc_UC
Zero/Equal	A / B	Z = 1	1 _H	cc_Z
Not Zero/Not Equal	A / B	Z = 0	2 _H	cc_NZ
Overflow	A / B	V = 1	3 _H	cc_V
Carry/Unsigned Less Than/ Check Bit True	A / B	C = 1	4 _H	cc_C, cc_ULT
Unsigned Greater Than	A / B	C OR Z = 0	5 _H	cc_UGT
Signed Less Than	A / B	N XOR V = 1	6 _H	cc_SLT
Signed Greater Than	A / B	(N XOR V) OR Z = 0	7 _H	cc_SGT
Negative	B	N = 1	8 _H	cc_N
Not Negative	B	N = 0	9 _H	cc_NN
Not Overflow	B	V = 0	A _H	cc_NV
No Carry/Unsigned Greater than or Equal	B	C = 0	B _H	cc_NC, cc_UGE
Signed Greater Than or Equal	B	N XOR V = 0	C _H	cc_SGE
Signed Less than or Equal	B	(N XOR V) OR Z = 1	D _H	cc_SLE
CNT1 Equal Zero	B	CNZ1 = 1	E _H	cc_CNZ
CNT1 Not Equal Zero	B	CNZ1 = 0	F _H	cc_CNN

11.11.1.2 Instruction Fields

Table 11-13 lists the instruction field definitions of the PCP instruction set architecture.

Note: The exact syntax for these fields may be different depending on which tool (e.g. assembler) is used. Please refer to the respective tool descriptions.

Table 11-13 Instruction Field Definitions

Symbol	Syntax	Description
CNC		Counter Control This field is used by the COPY instruction to control the number of repetitions of the data transfer. See also Figure 11-13 at Page 11-77 .
	CNC = 00 _B	Decrement CNT0 after each transfer. Continue until CNT0 = 0, and proceed to next instruction.
	CNC = 01 _B	Post Decrement CNT0 after each transfer. Continue until CNT0 = 0, then decrement CNT1 and proceed to next instruction.
	CNC = 10 _B	Post Decrement CNT0 after each transfer. Continue until CNT0 = 0, then decrement CNT1. Reload CNT0 value and continue. Continue until CNT1 = 0, then proceed to next instruction.
	CNC = 11 _B	Reserved.
CNT0		Counter Reload Value (COPY) The COPY instruction uses an implicit counter to generate multiple data transfers. The CNT0 value given in the instruction specifies how many data transfers are to be performed by the instruction. See also Figure 11-13 on Page 11-77 .
	CNT0 = 001 _B ..111 _B	Perform 1..7 data transfers
	CNT0 = 000 _B	Perform 8 data transfers
		Block Size (BCOPY) Selects the FPI block size used for a BCOPY instruction.
	CNT0 = 00 _B	Use block size of 8 words.
	CNT0 = 10 _B	Use block size of 2 words.
CNT0 = 11 _B	Use block size of 4 words.	
Others	Reserved	
cc_A, cc_B	see Table 11-12	Condition Code Specifies conditional execution of instruction according to CONDCA or CONDCB.

Peripheral Control Processor (PCP)

Table 11-13 Instruction Field Definitions (cont'd)

Symbol	Syntax	Description
DAC	DAC = 0	Stop PCP Allow the PCP to continue to execute channel programs in response to service requests.
	DAC = 1	Prevent the PCP from executing further channel programs (PCP_CS.EN = 0).
DST+-	DST (00 _B)	Destination Address Pointer Control No Change (DST)
	DST+ (01 _B)	Post Increment by Size (DST+)
	DST- (10 _B)	Post Decrement by Size (DST-)
	(11 _B)	Reserved
EC	EC = 0	Exit Count Control No action
	EC = 1	Decrement CNT1
EDA	EDA = 0	External Debug Action No External Debug Action caused
	EDA = 1	Cause an External Debug Action (breakpoint pin etc.)
EP	EP = 0	Entry Point Control Set the PC to channel program Start. EP = 0 assumes that a Channel Entry Table exists in the base of CMEM. Failure to provide such a table will cause improper operation.
	EP = 1	Set the PC to the address contained in NextPC (next instruction) address.
INT	INT = 0	Interrupt Control No Interrupt
	INT = 1	INT = 1 AND (cc_B = True) means Issue Interrupt
RTA	RTA = 0	Action on Debug Exit Stop channel program from accepting new service requests (clear R7.CEN)
	RTA = 1	Allow further channel program execution and decrement PC (so that DEBUG instruction is re-executed on next invocation) <i>Note: This field has no effect if SDB = 0 (see below)</i>
SIZE	SIZE = 00 _B	Data Size Control Byte (8-bit)
	SIZE = 01 _B	Half-word (16-bit)
	SIZE = 10 _B	Word (32-bit)
	SIZE = 11 _B	Reserved

Table 11-13 Instruction Field Definitions (cont'd)

Symbol	Syntax	Description
SRC+-	SRC (00 _B)	Source Address Pointer Control No Change (SRC)
	SRC+ (01 _B)	Post Increment by Size (SRC+)
	SRC- (10 _B)	Post Decrement by Size (SRC-)
	(11 _B)	Reserved
S/C	S/C = 0	Test Bit Control Check for Clear (0)
	S/C = 1	Check for Set (1)
SDB	SDB = 0	Stop on Debug Continue running if debug event triggered
	SDB = 1	Stop PCP if debug event triggered
ST	ST = 0	Stop Channel Continue channel execution, Leave channel program enabled
	ST = 1	Stop Channel Execution, Perform actions according to RTA setting (see above)

11.11.2 Counter Operation for COPY Instruction

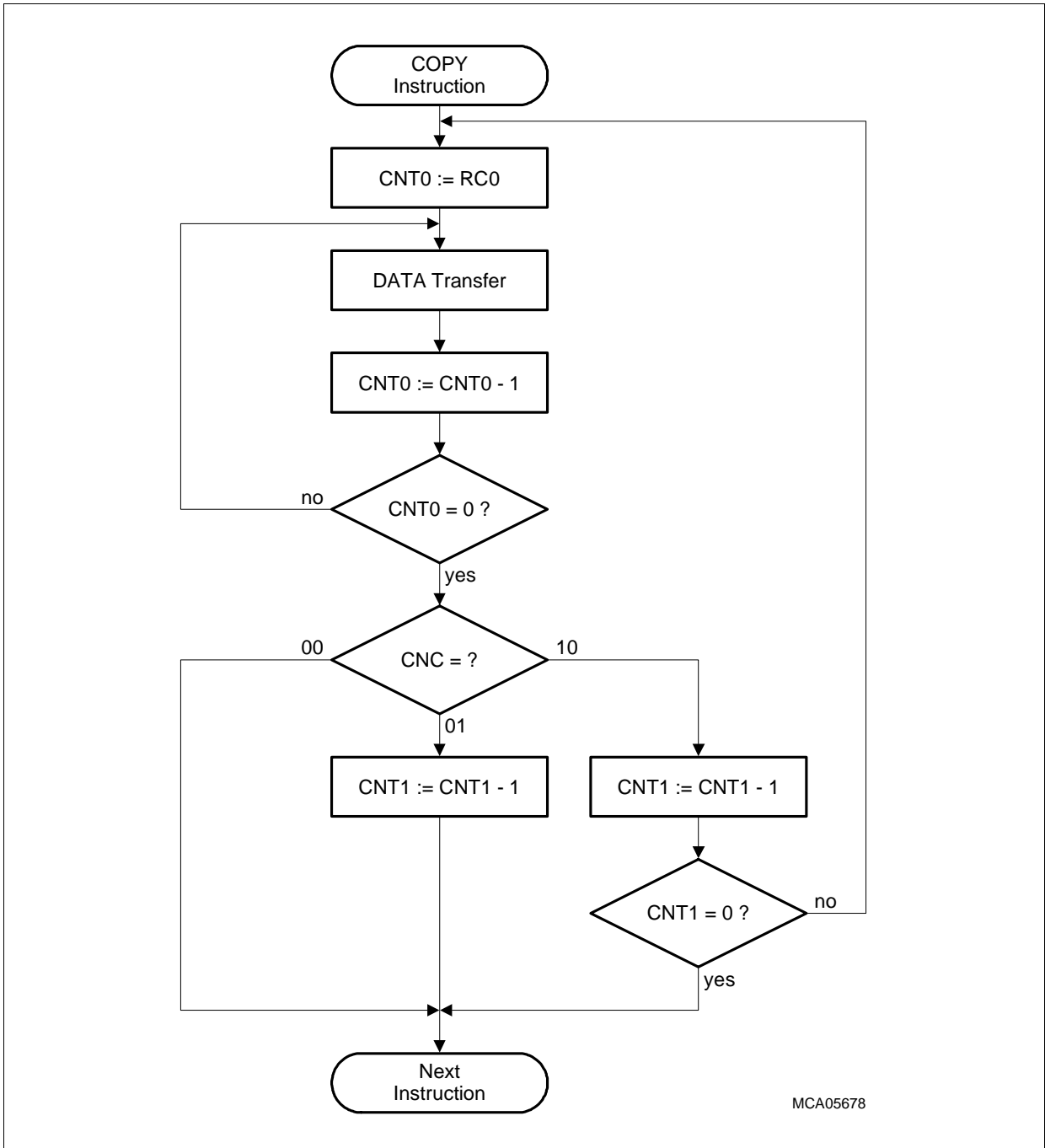


Figure 11-13 Counter Operation for COPY Instruction

11.11.3 Counter Operation for BCOPY Instruction

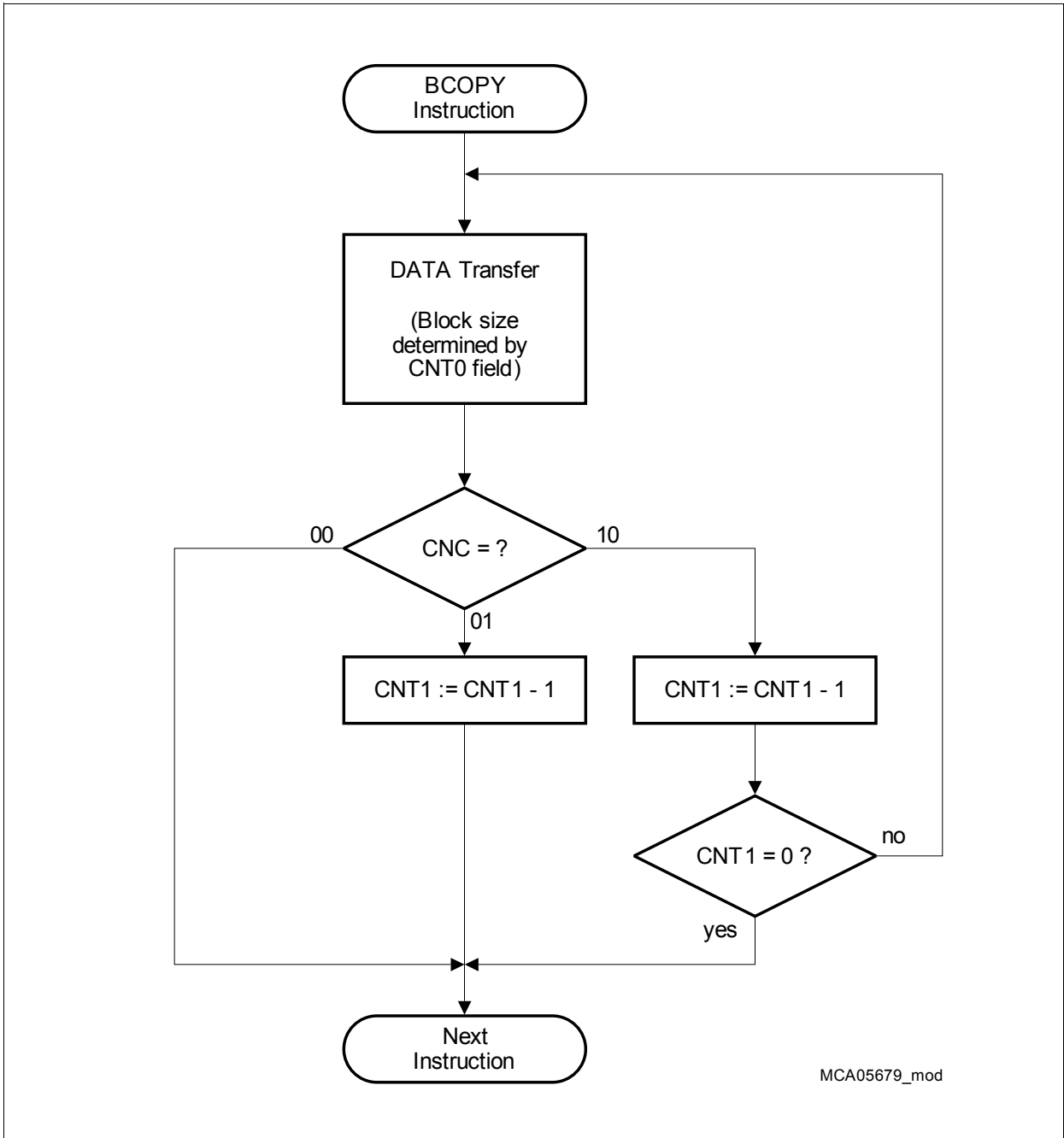


Figure 11-14 Counter Operation for BCOPY Instruction

11.11.4 Divide and Multiply Instructions

The PCP has multiply and divide capabilities (unsigned values only). All multiply and divide instructions operate on 8 bits of data (taken from the dividend for divide, from the multiplicand for multiply). This strategy allows the user to implement the appropriate number of instructions (“steps”) as required for his data format.

Each execution of a divide instruction (DSTEP) performs a division which generates 8 bits of result, and also manipulates the registers being used to allow the execution of consecutive divide (DSTEP) instructions to build divide algorithms in multiples of 8 bits (see [Page 11-122](#) for more details).

Each execution of a multiply instruction (MSTEP32 and MSTEP64) performs a multiplication on 8 bits of data (taken from the multiplicand) and also manipulates the registers to allow execution of consecutive multiply instructions to build multiply algorithms in multiples of 8 bits (see [Page 11-123](#) for more details).

The following restrictions apply to the use of Divide and Multiply instructions:

- The first instruction of any divide sequence must be the DINIT (initialization) instruction. Any additional instructions other than MINIT, MSTEP32 or MSTEP64 may also be used within the sequence as long as they do not modify any of the registers used for division (R0, Ra and Rb). All subsequent divide instructions within the sequence (DSTEP) must use the same register for dividend and the same register for divisor as used in the preceding DINIT instruction.
- The first instruction of any multiply sequence must be the MINIT (initialization) instruction. Any additional instructions other than DINIT or DSTEP may also be used within the sequence as long as they do not modify any of the registers used for multiplication (R0, Ra and Rb). All subsequent multiply instructions within the sequence (MSTEP32 and MSTEP64) must use the same register for multiplicand and the same register for multiplier as used in the preceding MINIT instruction.
- Neither of the operand registers (Ra or Rb) may be R0 (which is used implicitly within all the instructions), and the same register may not be supplied as both operand registers of an instruction (e.g. DSTEP R3, R3 is invalid).

Note: Failure to adhere to these restrictions will yield undefined results.

Note: Special care must be taken when using multiply and divide sequences when a channel program is interruptible. In this case it must be ensured that a sequence cannot be corrupted by the execution of multiply or divide instructions executed by a higher-priority channel. The R7.IEN bit can be used to ensure that a sequence is not interruptible (see [Page 11-120](#)).

In the descriptions attached to each multiply and divide instruction, a pseudo-code model is supplied to provide an unambiguous definition of the function of the instruction. The models supplied for the DSTEP and MSTEP32 instructions use 32-bit unsigned integer arithmetic, ignoring any possible overflows. The model supplied for the MSTEP64 uses

Peripheral Control Processor (PCP)

a 40-bit unsigned multiply and then shifts this result right by 8 bits (discards the least significant 8 bits of the 40-bit result).

The DSTEP instruction also has some conditions stipulated regarding input values to the instruction. Use of the pseudo-code model for the DSTEP instruction with invalid input values will yield an invalid result.

11.11.5 ADD, 32-Bit Addition

This section describes the ADD instructions of the PCP.

ADD	Syntax	ADD Rb, Ra, cc_A
	Description	If the condition CONDCA is true, then add the contents of register Ra to the contents of register Rb; place the result in Rb. If CONDCA is false, no operation is performed.
	Operation	if (CONDCA = True) then $R[b] = R[b] + R[a]$ else NOP
	Flags	N, Z, V, C
ADD.I	Syntax	ADD.I Ra, #imm6
	Description	Add the zero-extended immediate value imm6 to the contents of register Ra; place the result in Ra.
	Operation	$R[a] = R[a] + \text{zero_ext}(\text{imm6})$
	Flags	N, Z, V, C
ADD.F	Syntax	ADD.F Rb, [Ra], Size
	Description	Add the contents of the address location specified by the contents of register Ra to the contents of register Rb; place the result in Rb. <i>Note: Byte and Half-word values are zero-extended.</i>
	Operation	$R[b] = R[b] + \text{zero_ext}(\text{FPI}[R[a]])$
	Flags	N, Z, V, C
ADD.PI	Syntax	ADD.PI Ra, [#offset6]
	Description	Add the contents of the PRAM location specified by the addition of contents of the PRAM Data Pointer, shifted left by six bits, and the zero-extended 6-bit value offset6 to the contents of register Ra; place the result in Ra.
	Operation	$R[a] = R[a] + \text{PRAM}[(\text{DPTR} \ll 6) + \text{zero_ext}(\#offset6)]$
	Flags	N, Z, V, C

11.11.6 BCOPY, DMA Operation

BCOPY	Syntax	BCOPY DST+-, SRC+-, CNC, CNT0
	Description	<p>Allows the PCP to perform DMA type transfers using FPI block transfers.</p> <p>Moves the contents of FPI Bus source location to FPI Bus destination location. Source location is pointed to by the contents of register R4; destination location is pointed to by the contents of register R5. Options (see also Table 11-13 at Page 11-74) are:</p> <p>Source pointer (SRC+-): Increment, decrement or unchanged</p> <p>Destination pointer (SRC+-): Increment, decrement or unchanged</p> <p>Counter control (CNC): see Table 11-13</p> <p>Block size value (CNT0): see Table 11-13</p>
	Operation	<p>temp = zero_ext(FPI[R[4]]); value loaded and extended depending on SIZE</p> <p>FPI(R[5]) = temp</p> <p>R4 = R4 +/- n; n depending on SRC+- and CNT0</p> <p>R5 = R5 +/- n; n depending on DST+- and CNT0</p> <p>For counter operation see Figure 11-14 at Page 11-78 and Table 11-13 at Page 11-74.</p>
Flags	CN1Z	

See also [Page 11-127](#) for TC1796 specific details of the BCOPY instruction.

11.11.7 AND, 32-Bit Logical AND

This section describes the AND instructions of the PCP.

AND	Syntax	AND Rb, Ra, cc_A
	Description	If the condition CONDCA is true, then perform a bit-wise logical AND of the contents of register Ra and the contents of register Rb; place the result in Rb. If CONDCA is false, no operation is performed.
	Operation	if (CONDCA = True) then R[b] = R[b] AND R[a] else NOP
	Flags	N, Z
AND.F	Syntax	AND.F Rb, [Ra], Size
	Description	Perform a bit-wise logical AND of the contents of the address location, specified by the contents of register Ra, and the contents of register Rb; place the result in Rb.
	Operation	R[b] = R[b] AND zero_ext(FPI[R[a]])
	Flags	N, Z
AND.PI	Syntax	AND.PI Ra, [#offset6]
	Description	Perform a bit-wise logical AND of the contents of the PRAM location specified by the addition of contents of the PRAM Data Pointer, shifted left by six bits, and the zero-extended 6-bit value offset6, and the contents of register Ra; place the result in Ra.
	Operation	R[a] = R[a] AND PRAM[(DPTR<<6) + zero_ext(#offset6)]
	Flags	N, Z

11.11.8 CHKB, Check Bit

This section describes the CHKB instruction of the PCP.

CHKB	Syntax	CHKB Ra, #imm5, S/C
	Description	If bit imm5 of register Ra is equal to the specified test value S/C then set the carry flag R7.C, else clear the carry flag.
	Operation	if (R[a][imm5] = S/C) then R7_C = 1 else R7_C = 0
	Flags	C

11.11.9 CLR, Clear Bit

CLR	Syntax	CLR Ra, #imm5
	Description	Clear bit imm5 of register Ra to 0.
	Operation	R[a][imm5] = 0
	Flags	None
CLR.F	Syntax	CLR.F [Ra], #imm5, Size
	Description	Clear bit imm5 of the address location specified through the contents of register Ra to 0. This instruction is executed using a locked read-modify-write FPI Bus transaction.
	Operation	FPI[(R[a])][imm5] = 0
	Flags	None

11.11.10 COMP, 32-Bit Compare

This section describes the COMP instructions of the PCP.

COMP	Syntax	COMP Rb, Ra, cc_A
	Description	If the condition CONDCA is true, then subtract the contents of register Ra from the contents of register Rb; set the flags in register R7 according to the result of the subtraction; discard the subtraction result. If CONDCA is false, no operation is performed.
	Operation	if (CONDCA = True) then R7_FLAGS = Flags(R[b] - R[a])
	Flags	N, Z, V, C
COMP.I	Syntax	COMP.I Ra, #imm6
	Description	Subtract the sign-extended immediate value imm6 from the contents of register Ra; set the flags in register R7 according to the result of the subtraction; discard the subtraction result.
	Operation	R7_FLAGS = Flags(R[a] - sign_ext(imm6))
	Flags	N, Z, V, C
COMP.F	Syntax	COMP.F Rb, [Ra], Size
	Description	Subtract the contents of the address location specified by the contents of register Ra from the contents of register Rb; set the flags in register R7 according to the result of the subtraction; discard the subtraction result.
	Operation	R7_FLAGS = Flags(R[b] - sign_ext(FPI[R[a]]))
	Flags	N, Z, V, C
COMP.PI	Syntax	COMP.PI Ra, [#offset6]
	Description	Subtract the contents of the PRAM location specified by the addition of contents of the PRAM Data Pointer, shifted left by six bits, and the zero-extended 6-bit value offset6, from the contents of register Ra; set the flags in register R7 according to the result of the subtraction; discard the subtraction result.
	Operation	R7_FLAGS = Flags(R[a] - PRAM[(DPTR<<6) + zero_ext(#offset6)])
	Flags	N, Z, V, C

11.11.11 COPY, DMA Instruction

This section describes the COMP instruction of the PCP.

COPY	Syntax	COPY DST+-, SRC+-, CNC, RC0, SIZE
	Description	<p>Moves the contents of FPI Bus source location to FPI Bus destination location. Source location is pointed to by the contents of register R4; destination location is pointed to by the contents of register R5. Options (see also Table 11-13 at Page 11-74) are:</p> <p>Source pointer (SRC+-): Increment, decrement or unchanged</p> <p>Destination pointer (SRC+-): Increment, decrement or unchanged</p> <p>Counter control (CNC): see Table 11-13</p> <p>Counter 0 reload value (CNT0): see Table 11-13</p> <p>Data transfer width (SIZE): byte, half-word, word (pointers are incremented/decremented based upon SIZE)</p>
	Operation	<p>temp = zero_ext(FPI[R[4]]); value loaded and extended depending on SIZE FPI(R[5]) = temp R4 = R4 +/- n; n depending on SRC+- and SIZE R5 = R5 +/- n; n depending on DST+- and SIZE For counter operation see Figure 11-13 at Page 11-77 and Table 11-13 at Page 11-74.</p>
	Flags	CN1Z

11.11.12 DEBUG, Debug Instruction

This section describes the DEBUG instruction of the PCP.

DEBUG	Syntax	DEBUG EDA, DAC, RTA, SDB, cc_B
	Description	Conditionally cause a debug event if condition CONDCB is true. Optionally stop channel execution (SDB = 1) and/or generate an external debug event (EDA = 1).
	Operation	<pre> if (CONDCB = True) then if (EDA = 1) then activate BRK_OUT pin if (SDB = 1) then if (RTA = 0) then R7_CEN = 0; disable further channel invocation else PC = PC - 1 endif endif save_context idle endif if (DAC = 1) PCP_CS > EN = 0 endif set ES.DBE; indicate debug event ES.PC = NextPC ES.PN = channel_number endif </pre>
	Flags	none

Note: Execution of the DEBUG command when not in Debug Mode will cause the PCP to generate an "Illegal Operation" Error Exit.

11.11.13 DINIT, Divide Initialization Instruction

This section describes the DINIT instruction of the PCP.

DINIT	Syntax	DINIT <R0>, Rb, Ra
	Description	Initialize Divide logic ready for divide sequence (Rb / Ra) and Clear R0. If value of Ra is 0 then set V (to flag divide by 0 error); otherwise clear V. If value of Rb is 0 and value of Ra is not 0 then set Z (to flag a zero result); otherwise clear Z.
	Operation	R0 = 0
	Flags	Z, V

11.11.14 DSTEP, Divide Instruction

This section describes the DSTEP instruction of the PCP.

DSTEP	Syntax	DSTEP <R0>, Rb, Ra
	Description	Perform 1 step (eight bits) of an unsigned 32- by 32-bit divide (Rb / Ra). Shift R0 left by 8 bits, copy the most significant byte of Rb into LS byte of R0. Shift Rb left by 8 bits and add (R0 divided by Ra). Load R0 with (the remainder of R0 divided by Ra).
	Operation	$R0 = (R0 \ll 8) + (Rb \gg 24)$ $Rb = (Rb \ll 8) + R0 / Ra$ $R0 = R0 \% Ra$
	Flags	Z

Note: The value in Ra must always be greater than the value in R0 prior to execution of the DSTEP instruction. If the rules specified at [Page 11-79](#) are followed, then the above description and operation are correct. Failure to adhere to these rules will yield undefined results.

11.11.15 INB, Insert Bit

This section describes the INB instructions of the PCP.

INB	Syntax	INB Rb, Ra, cc_A
	Description	If CONDCA is true, then insert the carry flag R7.C into register Rb at the bit position specified through bits [4..0] of register Ra. If CONDCA is false, no operation is performed.
	Operation	if (CONDCA = True) then $R[b][R[a][4:0]] = R7_C$ else NOP
	Flags	None
INB.I	Syntax	INB.I Ra, #imm5
	Description	Insert the carry flag R7.C into register Ra at the bit position specified through the immediate value imm5.
	Operation	$R[a][imm5] = R7_C$
	Flags	None

11.11.16 EXIT, Exit Instruction

This section describes the EXIT instruction of the PCP.

EXIT	Syntax	EXIT EC, ST, INT, EP, cc_B
	Description	<p>Unconditionally exit channel program execution. Optionally decrement counter CNT1 (EC = 1), disable further channel invocation (ST = 1), generate an interrupt request (INT = 1) if condition CONDCB is true. Field EP is used to set the channel code entry point in Channel Resume Mode to either the address of the next instruction (EP = 1) or to the start address of the channel (EP = 0). The EXIT instruction is finished with a context save operation.</p> <p>The EP option is only in effect when Channel Resume operation is globally selected through PCP_CS.RCB = 0. If PCP_CS.RCB = 1, Channel restart mode is selected for all channels, and the EP field of the EXIT instruction is disregarded.</p>
	Operation	<pre> if (EC = 1) then CNT1 = CNT1 - 1 if (ST = 1) then R7_CEN = 0 if ((INT = 1) AND (cc_B = True)) then activate_interrupt_request if (EP = 1) then R7_PC = NextPC else R7_PC = channel_entry_point save_context </pre>
Flags	CN1Z	

11.11.17 JC, Jump Conditionally

This section describes the conditional jump instructions of the PCP.

JC	Syntax	JC offset6, cc_B
	Description	If CONDCB is true, then add the sign-extended value specified by offset6 to the contents of the PC, and jump to that address. If CONDCB is false, no operation is performed.
	Operation	if (CONDCB = True) then (PC = PC + sign_ext(offset6)) else NOP
	Flags	None
JC.A	Syntax	JC.A #address16, cc_B
	Description	If CONDCB is true, then load the value specified by address16 into the PC, and jump to that address. If CONDCB is false, no operation is performed.
	Operation	if (CONDCB = True) then (PC = address16) else NOP
	Flags	None
JC.I	Syntax	JC.I Ra, cc_B
	Description	If CONDCB is true, then add the value specified by Ra[15:0] to the contents of the PC, and jump to that address. Value Ra[15:0] is treated as a signed 16-bit number. If CONDCB is false, no operation is performed.
	Operation	if (CONDCB = True) then (PC = PC + (R[a][15:0])) else NOP
	Flags	None
JC.IA	Syntax	JC.IA Ra, cc_B
	Description	If CONDCB is true, then load the value specified by Ra[15:0] into the PC, and jump to that address. Value Ra[15:0] is treated as an unsigned 16-bit number. If CONDCB is false, no operation is performed.
	Operation	if (CONDCB = True) then (PC = (R[a][15:0])) else NOP
	Flags	None

11.11.18 JL, Jump Long Unconditional

This section describes the long jump instruction JL of the PCP.

JL	Syntax	JL offset10
	Description	Add the sign-extended value specified by offset10 to the contents of the PC, and jump to that address.
	Operation	PC = PC + sign_ext(offset10)
	Flags	None

11.11.19 LD, Load

This section describes the LD instructions of the PCP.

LD.F	Syntax	LD.F Rb, [Ra], Size
	Description	Load the zero-extended contents of the address location specified by the contents of register Ra into register Rb.
	Operation	R[b] = zero_ext(FPI[R[a]])
	Flags	N, Z
LD.I	Syntax	LD.I Ra, #imm6
	Description	Load the zero-extended value specified by imm6 into register Ra.
	Operation	R[a] = zero_ext(imm6)
	Flags	N, Z
LD.IF	Syntax	LD.IF [Ra], #offset5, Size
	Description	Load the zero-extended contents of the address location, specified by the addition of the contents of register Ra and the value specified by imm5, into register R0.
	Operation	R[0] = zero_ext(FPI[R[a] + zero_ext(imm5)])
	Flags	N, Z

Peripheral Control Processor (PCP)

LD.P	Syntax	LD.P Rb, [Ra], cc_A
	Description	If condition CONDCA is true, then load the contents of the PRAM address location, specified by the addition of contents of the PRAM Data Pointer, shifted left by six bits, and the zero-extended 6-bit value Ra[5:0] into register Rb. If condition CONDCA is false, no operation is performed.
	Operation	if (CONDCA = True) then R[b] = PRAM[(DPTR<<6) + zero_ext(R[a][5:0])] else NOP
	Flags	N, Z
LD.PI	Syntax	LD.PI Ra, [#offset6]
	Description	Load the contents of the PRAM location specified by the addition of contents of the PRAM Data Pointer, shifted left by six bits, and the zero-extended 6-bit value offset6 into register Ra.
	Operation	R[a] = PRAM[(DPTR<<6) + zero_ext(offset6)]
	Flags	N, Z

11.11.20 LDL, Load 16-bit Value

This section describes the LDL instructions of the PCP.

LDL.IL	Syntax	LDL.IL Ra, #imm16
	Description	Load the immediate value imm16 into the lower bits of register Ra (bits [15:0]). Bits [31:16] of register Ra are unaffected. Value imm16 is treated as an unsigned 16-bit number.
	Operation	R[a][15:0] = imm16
	Flags	N, Z
LDL.IU	Syntax	LDL.IU Ra, #imm16
	Description	Load the immediate value imm16 into the upper bits of register Ra (bits [31:16]). Bits [15:0] of register Ra are unaffected.
	Operation	R[a][31:16] = imm16
	Flags	N, Z

11.11.21 Multiply Initialization Instruction

This section describes the MINIT instruction of the PCP.

MINIT	Syntax	MINIT <R0>, Rb, Ra
	Description	Initialize Multiply logic ready for multiply sequence. Clear R0. If value of Ra is zero or value of Rb is zero then set Z (to flag zero result) else clear Z.
	Operation	R0 = 0
	Flags	Z

11.11.22 MOV, Move Register to Register

MOV	Syntax	MOV Rb, Ra, cc_A
	Description	If condition CONDCA is true, then move the contents of register Ra into register Rb. If CONDCA is false, no operation is performed.
	Operation	if (CONDCA = True) then R[b] = R[a] else NOP
	Flags	N, Z

11.11.23 Multiply Instructions

This section describes the multiply instructions of the PCP.

MSTEP32	Syntax	MSTEP32 <R0>, Rb, Ra
	Description	Perform an unsigned multiply step, using eight bits of data taken from Rb, keeping the least significant 32 bits of a potential 64-bit result. Left rotate Rb by 8 bits. Shift R0 left by 8 bits. Add (Ra multiplied by the least significant 8 bits of Rb) to R0. If value of R0 is zero then set Z (to signal zero result) else clear Z.
	Operation	$Rb = (Rb \ll 8) + (Rb \gg 24)$ $R0 = (R0 \ll 8) + (Rb \& 0xff) \times Ra$
	Flags	Z
MSTEP64	Syntax	MSTEP64 <R0>, Rb, Ra
	Description	Perform an unsigned multiply step, using eight bits of data taken from Rb, keeping 40 bits of a potential 64-bit result. Add (Ra multiplied by the least significant 8 bits of Rb) to R0 and retain the 40 bit result (shown as temp below). Store the most significant 32 bits of the result (temp) in R0. Shift Rb right by 8 bits. Store the least significant 8 bits of the first result (temp) in the most significant 8 bits of Rb. If value of R0 is zero then set Z (to signal zero result) else clear Z.
	Operation	$temp = R0 + Ra \times (Rb \& 0xff)$ $R0 = temp \gg 8$ $Rb = (Rb \gg 8) + ((temp \& 0xff) \ll 24)$
	Flags	Z

Note: In the case of the MSTEP64 instruction above, the “temp” variable is a 40-bit variable and all calculations are performed using 40-bit unsigned arithmetic. All other calculations use 32-bit unsigned arithmetic.

11.11.24 NEG, Negate

This section describes the NEG instruction of the PCP.

NEG	Syntax	NEG Rb, Ra, cc_A
	Description	If condition CONDCA is true, then move the 2's complement of the contents of register Ra into register Rb. If CONDCA is false, no operation is performed.
	Operation	if (CONDCA = True) then $R[b] = (-R[a])$ else NOP
	Flags	N, Z, V, C

11.11.25 NOP, No Operation

This section describes the NOP instruction of the PCP.

NOP	Syntax	NOP
	Description	No operation. The NOP instruction puts the PCP in low-power operation.
	Operation	no operation
	Flags	None

11.11.26 NOT, Logical NOT

This section describes the NOT instruction of the PCP.

NOT	Syntax	NOT Rb, Ra, cc_A
	Description	If condition CONDCA is true, then move the 1's complement of the contents of register Ra into register Rb. If CONDCA is false, no operation is performed.
	Operation	if (CONDCA = True) then $R[b] = \text{NOT}(R[a])$ else NOP
	Flags	N, Z

11.11.27 OR, Logical OR

This section describes the OR instructions of the PCP.

OR	Syntax	OR Rb, Ra, cc_A
	Description	If the condition CONDCA is true, then perform a bit-wise logical OR of the contents of register Ra and the contents of register Rb; place the result in Rb. If CONDCA is false, no operation is performed.
	Operation	if (CONDCA = True) then R[b] = R[b] OR R[a] else NOP
	Flags	N, Z
OR.F	Syntax	OR.F Rb, [Ra], Size
	Description	Perform a bit-wise logical OR of the contents of the address location, specified by the contents of register Ra, and the contents of register Rb; place the result in Rb.
	Operation	R[b] = R[b] OR zero_ext(FPI[R[a]])
	Flags	N, Z
OR.PI	Syntax	OR.PI Ra, [#offset6]
	Description	Perform a bit-wise logical OR of the contents of the PRAM location specified by the addition of contents of the PRAM Data Pointer, shifted left by six bits, and the zero-extended 6-bit value offset6, and the contents of register Ra; place the result in Ra.
	Operation	R[a] = R[a] OR PRAM[(DPTR<<6) + zero_ext(#offset6)]
	Flags	N, Z

11.11.28 PRI, Prioritize

This section describes the PRI instruction of the PCP.

PRId	Syntax	PRI Rb, Ra, cc_A
	Description	If condition CONDCA is true, then find the bit position of the most significant 1 in register Ra and put the number into register Rb. The bit location, 31..0, is encoded as a 5-bit number stored in Rb[4:0]. If the contents of Ra is zero, bit Rb[5] is set, while all other bits in Rb are cleared. If CONDCA is false, no operation is performed.
	Operation	if (CONDCA = False) then NOP else if (R[a] = 0) then R[b] = 0x20 else R[b] = bit_pos(most_significant_1(R[a]))
	Flags	N, Z

11.11.29 PRAM Bit Operations

This section describes the MCLR and MSET instructions of the PCP.

MCLR	Syntax	MCLR.PI Ra, [#offset6]
	Description	Perform an 'AND' of the contents of the specified register with the contents of the PRAM location specified by the addition of contents of the PRAM Data Pointer, shifted left by six bits, and the zero-extended 6-bit value offset. Write the result back to the PRAM location.
	Operation	$R[a] = R[a].AND.PRAM[DPTR \ll 6 + \#offset6]$ $PRAM[DPTR \ll 6 + \#offset6] = R[a]$
	Flags	N, Z
MSET	Syntax	MSET.PI Ra, [#offset6]
	Description	Perform an 'OR' of the contents of the specified register with the contents of the PRAM location specified by the addition of contents of the PRAM Data Pointer, shifted left by six bits, and the zero-extended 6-bit value offset. Write the result back to the PRAM location.
	Operation	$R[a] = R[a].OR.PRAM[DPTR \ll 6 + \#offset6]$ $PRAM[DPTR \ll 6 + \#offset6] = R[a]$
	Flags	N, Z

Note: MCLR and MSET are read/modify/write operations that cannot be interrupted by an access from another FPI master. They can be used to implement semaphore systems.

11.11.30 RL, Rotate Left

This section describes the RL instruction of the PCP.

RL	Syntax	RL Ra, #imm5
	Description	Rotate the contents of register Ra to the left by the number of bit positions specified through the 5-bit value imm5. The values defined for imm5 are 1, 2, 4 and 8. The carry flag, R7.C, is set to the last bit shifted out of bit 31 of register Ra.
	Operation	tmp = R[a] R[a] = R[a] << imm5; imm5 = 1, 2, 4, 8 R7_C = last bit shifted out of R[a] tmp = tmp >> 32 - imm5 R[a] = tmp OR R[a]
	Flags	N, Z

11.11.31 RR, Rotate Right

This section describes the RR instruction of the PCP.

RR	Syntax	RR Ra, #imm5
	Description	Rotate the contents of register Ra to the right by the number of bit positions specified through the 5-bit value imm5. The values allowed for imm5 are 1, 2, 4 and 8.
	Operation	tmp = R[a] R[a] = R[a] >> imm5; imm5 = 1, 2, 4, 8 tmp = tmp << 32 - imm5 R[a] = tmp OR R[a]
	Flags	N, Z

11.11.32 SET, Set Bit

This section describes the SET bit instruction of the PCP.

SET	Syntax	SET Ra, #imm5
	Description	Set bit imm5 of register Ra to 1.
	Operation	$R[a][imm5] = 1$
	Flags	None
SET.F	Syntax	SET.F [Ra], #imm5, Size
	Description	Set bit imm5 of the address location specified through the contents of register Ra to 1. This instruction is executed using a locked read-modify-write FPI Bus transaction.
	Operation	$FPI[(R[a])][imm5] = 1$
	Flags	None

11.11.33 SHL, Shift Left

This section describes the SHL instruction of the PCP.

SHL	Syntax	SHL Ra, #imm5
	Description	Shift the contents of register Ra to the left by the number of bit positions specified through the 5-bit value imm5. The values allowed for imm5 are 1, 2, 4 and 8. The carry flag, R7.C, is set to the last bit shifted out of bit 31 of register Ra. Zeros are shifted in from right.
	Operation	$R[a] = R[a] \ll imm5$; imm5 = 1, 2, 4, 8 R7_C = last bit shifted out of R[a]
	Flags	N, Z, C

11.11.34 SHR, Shift Right

This section describes the SHR instruction of the PCP.

SHR	Syntax	SHR Ra, #imm5
	Description	Shift the contents of register Ra to the right by the number of bit positions specified through the 5-bit value imm5. The values allowed for imm5 are 1, 2, 4 and 8. Zeros are shifted in from left.
	Operation	$R[a] = R[a] \gg \text{imm5}$; imm5 = 1, 2, 4, 8
	Flags	N, Z

11.11.35 ST, Store

This section describes the ST instructions of the PCP.

ST.F	Syntax	ST.F Rb, [Ra], Size
	Description	Store the contents of register Rb to the address location specified by the contents of register Ra. When the Size is byte or half-word, the data is stored with the internal LSB (bit 0) properly aligned to the correct FPI Bus byte or half-word lane.
	Operation	$FPI[R[a]] = R[b]$
	Flags	None
ST.IF	Syntax	ST.IF [Ra], #offset5, Size
	Description	Store the contents of R0 to the address location specified by the addition of the contents of register Ra and the value specified by imm5. When the Size is byte or half-word, the data is stored with the internal LSB (bit 0) properly aligned to the correct FPI Bus byte or half-word lane.
	Operation	$FPI[R[a] + \text{zero_ext}(\text{imm}5)] = R[0]$
	Flags	None
ST.P	Syntax	ST.P Rb, [Ra], cc_A
	Description	If condition CONDCA is true, then store the contents of Rb to the PRAM address location specified by the addition of the contents of the PRAM Data Pointer, shifted left by six bits, and the zero-extended 6-bit value Ra[5:0]. If condition CONDCA is false, no operation is performed.
	Operation	if (CONDCA = True) then $PRAM[(DPTR \ll 6) + \text{zero_ext}(R[a][5:0])] = R[b]$ else NOP
	Flags	None
ST.PI	Syntax	ST.PI Rb, [#offset6]
	Description	Store the contents of register Rb to the PRAM location specified by the addition of the contents of the PRAM Data Pointer, shifted left by six bits, and the zero-extended 6-bit value offset6.
	Operation	$PRAM[(DPTR \ll 6) + \text{zero_ext}(\text{offset}6)] = R[b]$
	Flags	None

11.11.36 SUB, 32-Bit Subtract

This section describes the SUB instructions of the PCP.

SUB	Syntax	SUB Rb, Ra, cc_A
	Description	If the condition CONDCA is true, then subtract the contents of register Ra from the contents of register Rb; place the result in Rb. If CONDCA is false, no operation is performed.
	Operation	if (CONDCA = True) then $R[b] = R[b] - R[a]$ else NOP
	Flags	N, Z, V, C
SUB.I	Syntax	SUB.I Ra, #imm6
	Description	Subtract the zero-extended immediate value imm6 from the contents of register Ra; place the result in Ra.
	Operation	$R[a] = R[a] - \text{zero_ext}(\text{imm6})$
	Flags	N, Z, V, C
SUB.F	Syntax	SUB.F Rb, [Ra], Size
	Description	Subtract the zero-extended contents of the address location specified by the contents of register Ra from the contents of register Rb; place the result in Rb.
	Operation	$R[b] = R[b] - \text{zero_ext}(\text{FPI}[R[a]])$
	Flags	N, Z, V, C
SUB.PI	Syntax	SUB.PI Ra, [#offset6]
	Description	Subtract the contents of the PRAM location specified by the addition of contents of the PRAM Data Pointer, shifted left by six bits, and the zero-extended 6-bit value offset6 from the contents of register Ra; place the result in Ra.
	Operation	$R[a] = R[a] - \text{PRAM}[(\text{DPTR} \ll 6) + \text{zero_ext}(\#\text{offset6})]$
	Flags	N, Z, V, C

11.11.37 XCH, Exchange

This section describes the XCH instructions of the PCP.

XCH.F	Syntax	XCH.F Rb, [Ra], Size
	Description	Exchange contents of R[b] and FPI[R[a]] When Size is byte or half-word, the value is stored with the internal LSB (bit 0) properly aligned to the correct FPI byte or half-word lane. The exchange is done via a locked FPI bus transfer.
	Operation	temp = R[b] R[b] = zero_ext(FPI[R[a]]) FPI[R[a]] = temp
	Flags	N, Z
XCH.PI	Syntax	XCH.PI Ra, [#offset6]
	Description	Exchange contents of R[a] and PRAM[DPTR << 6 + #offset6]. <i>Note: The exchange is un-interruptible, and locks out external accesses; it will not be interrupted by any external FPI bus master transfer requests.</i>
	Operation	temp = R[a] R[a] = PRAM[(DPTR << 6) + zero_ext(#offset6)] PRAM[(DPTR << 6) + zero_ext(#offset6)] = temp
	Flags	N, Z

11.11.38 XOR, 32-Bit Logical Exclusive OR

This section describes the XOR instructions of the PCP.

XOR	Syntax	XOR Rb, Ra, cc_A
	Description	If the condition CONDCA is true, then perform a bit-wise logical Exclusive-OR of the contents of register Ra and the contents of register Rb; place the result in Rb. If CONDCA is false, no operation is performed.
	Operation	if (CONDCA = True) then R[b] = R[b] XOR R[a] else NOP
	Flags	N, Z
XOR.F	Syntax	XOR.F Rb, [Ra], Size
	Description	Perform a bit-wise logical Exclusive-OR of the contents of the address location, specified by the contents of register Ra, and the contents of register Rb; place the result in Rb.
	Operation	R[b] = R[b] XOR zero_ext(FPI[R[a]])
	Flags	N, Z
XOR.PI	Syntax	XOR.PI Ra, [#offset6]
	Description	Perform a bit-wise logical Exclusive-OR of the contents of the PRAM location specified by the addition of contents of the PRAM Data Pointer, shifted left by six bits, and the zero-extended 6-bit value offset6, and the contents of register Ra; place the result in Ra.
	Operation	R[a] = R[a] XOR PRAM[(DPTR<<6) + zero_ext(#offset6)]
	Flags	N, Z

11.11.39 Flag Updates of Instructions

Most instructions update the state flags in R7. In [Table 11-14](#), each instruction is shown with the flags that it updates.

Table 11-14 Flag Updates

Instruction	CN1Z	V	C	N	Z
ADD	–	yes	yes	yes	yes
AND	–	–	–	yes	yes
BCOPY	yes ¹⁾	–	–	–	–
CHKB	–	–	yes	–	–
CLR	–	–	–	–	–
COMP	–	yes	yes	yes	yes
COPY	yes ¹⁾	–	–	–	–
DEBUG	–	–	–	–	–
DINIT	–	yes	–	–	yes
DSTEP	–	–	–	–	yes
EXIT	yes ¹⁾	–	–	–	–
INB	–	–	–	–	–
JC	–	–	–	–	–
JL	–	–	–	–	–
LD	–	–	–	yes ²⁾	yes
LDL	–	–	–	yes	yes
MCLR	–	–	–	yes	yes
MSET	–	–	–	yes	yes
MINIT	–	–	–	–	yes
MOV	–	–	–	yes	yes
MSTEP32	–	–	–	–	yes
MSTEP64	–	–	–	–	yes
NEG	–	yes	yes	yes	yes
NOP	–	–	–	–	–
NOT	–	–	–	yes	yes
OR	–	–	–	yes	yes
PRI	–	–	–	yes ³⁾	yes

Table 11-14 Flag Updates (cont'd)

Instruction	CN1Z	V	C	N	Z
RR	–	–	–	yes	yes
RL	–	–	yes	yes	yes
SET	–	–	–	–	–
SHR	–	–	–	yes ³⁾	yes
SHL	–	–	yes	yes	yes
ST	–	–	–	–	–
SUB	–	yes	yes	yes	yes
XCH	–	–	–	yes	yes
XOR	–	–	–	yes	yes

- 1) CN1Z is only modified by the BCOPY, COPY or EXIT instructions if the instruction has been configured to decrement R6.CNT1 (for BCOPY/COPY CNC = 1 or CNC = 2, for EXIT EC = 1). All other instructions have no effect on the CN1Z flag.
- 2) For the LD.l type of instruction, flag N is always cleared, as bit 31 of the result is always 0.
- 3) Flag N is always cleared, as bit 31 of the result is always 0.

11.11.40 Instruction Timing

This section gives some information about the duration of PCP instructions. Please note that there are various conditions that can further affect the duration of PCP instructions (e.g. external FPI accesses from another FPI Bus master to the PCP memories stall the PCP Processor Core).

Note: The clock cycles listed in [Table 11-15](#) are PCP module clock cycles. In the TC1796, the PCP is connected to the System Peripheral Bus (which is a FPI Bus) and clocked with f_{SYS} as module clock ($f_{SYSmax} = 75$ MHz, resulting in a minimum clock cycle time of 13.3 ns).

Table 11-15 Instruction Timing

Instruction	Number of Clock Cycles	Comments	Notes
Control			
NOP	1	–	–
COPY	–	–	1)
EXIT	f = 9, s = 7, m = 6	–	2)
BCOPY	–	–	1)

Peripheral Control Processor (PCP)

Table 11-15 Instruction Timing (cont'd)

Instruction	Number of Clock Cycles	Comments	Notes
FPI Access			
ADD.F	8 min.	5 int. + 3 min. for FPI read	3)
SUB.F	8 min.	5 int. + 3 min. for FPI read	3)
COMP.F	8 min.	5 int. + 3 min. for FPI read	3)
AND.F	8 min.	5 int. + 3 min. for FPI read	3)
OR.F	8 min.	5 int. + 3 min. for FPI read	3)
XOR.F	8 min.	5 int. + 3 min. for FPI read	3)
LD.F	8 min.	5 int. + 3 min. for FPI read	3)
ST.F	5 min	2 int. + 3 min. for FPI write	3)
XCH.F	8 min	4 int. + 4 min. for FPI read and write	3)
PRAM Access			
ADD.PI	2	—	—
SUB.PI	2	—	—
COMP.PI	2	—	—
MCLR.PI	4	—	—
AND.PI	2	—	—
MSET.PI	4	—	—
OR.PI	2	—	—
XOR.PI	2	—	—
LD.PI	2	—	—
ST.PI	2	—	—
XCH.PI	2	—	—
Arithmetic (Conditional)			
ADD	1	—	—
SUB	1	—	—
COMP	1	—	—
NEG	1	—	—
NOT	1	—	—
AND	1	—	—

Peripheral Control Processor (PCP)

Table 11-15 Instruction Timing (cont'd)

Instruction	Number of Clock Cycles	Comments	Notes
OR	1	–	–
XOR	1	–	–
LD.P	2	–	–
ST.P	4	–	–
MOV	1	–	–
INB	1	–	–
PRI	1	–	–
Immediate Access			
ADD.I	1	–	–
SUB.I	1	–	–
COMP.I	1	–	–
SHR	1	–	–
SHL	1	–	–
RR	1	–	–
RL	1	–	–
LDL.IU	1	–	–
LDL.IL	1	–	–
SET	1	–	–
CLR	1	–	–
LD.I	1	–	–
INB.I	1	–	–
CHKB	1	–	–
FPI + Immediate Access			
SET.F	8 min	4 int. + 4 min. for locked FPI RMW	4)
CLR.F	8 min	4 int. + 4 min. for locked FPI RMW	4)
LD.IF	8 min	5 int. + 3 min. for FPI read	3)
ST.IF	5 min	2 int. + 3 min. for FPI write	5)
Complex Maths			
DINIT	1	–	6)
DSTEP	10	–	6)

Table 11-15 Instruction Timing (cont'd)

Instruction	Number of Clock Cycles	Comments	Notes
MINIT	1	—	7)
MSTEP.L	10	—	7)
MSTEP.U	10	—	7)
Jump			
JL	4	—	—
JC	y = 4, n = 2	—	8)
JC.A	y = 4, n = 2	—	8)
JC.I	y = 4, n = 2	—	8)
JC.IA	y = 4, n = 2	—	8)
DEBUG	sdb - 0 = 2 sdb - 1 = exit_time	—	9)

- 1) The number of clock cycles for these instructions depends on several parameters such as the amount of data to be copied, the type of memory, and the effective bus load.
- 2) f = Full Context save, s = Small Context save, m = Minimum Context save Time extended until any previous ST.F instruction has completed.
- 3) Cycles = 5 internal + 3 minimum for FPI read (with 0 wait cycles + 1 cycle for bus arbitration)
- 4) Cycles = 4 internal + 4 minimum for FPI read/modify/write (with 0 wait cycles + 1 cycle bus arbitration)
- 5) Cycles = 2 internal + 3 minimum for FPI write (with 0 wait cycles + 1 cycle for bus arbitration)
Time starts after any previous ST.F instruction has completed.
- 6) 32/32 bit divide requires instruction DINIT + JC + 4 × DSTEP = 1 + 4(2) + 4 × 10 = 45 cycles
8/32 bit divide requires instruction RR + DINIT + JC + DSTEP = 1 + 1 + 4(2) + 10 = 16 cycles
- 7) 32 × 8 bit multiply requires instructions RR + MINIT + MSTEP.L = 1 + 1 + 10 = 12 cycles
32 × 16 bit multiply requires instructions 2 × RR + MINIT + 2 × MSTEP.L = 2 × 1 + 1 + 2 × 10 = 23 cycles
32 × 32 bit multiply requires instructions MINIT + 4 × MSTEP.U = 1 + 4 × 10 = 41 cycles
- 8) y = jump taken, n = jump not taken
- 9) sdb - 0 = Stop_on_Debug bit in instruction = 0 (disabling stop)
sdb - 1 = Stop_on_Debug bit in instruction = 1 (enabling stop)
exit_time = same time as for an exit instruction

11.12 Programming of the PCP

In this section, several techniques are outlined to help design channel programs. There are also examples on configuring a channel program's context.

11.12.1 Initial PC of a channel program

A channel program can begin operation at the Channel Entry Table location corresponding to the priority of the interrupt. This is much like an interrupt vector location for that channel in a traditional processor architecture. When the channel program is started, the PC is set to two-times the channel number (SRPN). Since the base of the Channel Entry Table is the bottom of the CMEM address range, and since each entry in the table is two instructions long, this address computation results in the first instruction of the channel program for that SRPN being fetched from memory for execution.

Alternately, the channel program can be made to begin executing at whatever address its restored context holds in R7.PC.

If PCP_CS.RCB = 1, then the channel program is forced to always start at its Channel Entry Table location regardless of the PC value stored in the CSA. If PCP_CS.RCB = 0, then the channel program will simply begin executing at whatever PC value is restored in the context R7.PC.

It is important to be aware of the implications of these two approaches on how CMEM should be configured, and what the initial value of the PC should be in the channel program's context that is loaded in the PRAM CSA at boot time.

11.12.1.1 Channel Entry Table

When PCP_CS.RCB = 1, the program counter of the PCP is vectored to the appropriate channel entry table each time a channel program is invoked by the receipt of an interrupt. The PCP is forced to start executing from its channel entry table location, regardless of its previous context or PC state.

If the EXIT instruction is executed with EP = 0, the PC saved during the context save operation will be the channel entry table location for that channel. That means that the next time the channel program is started, it will begin operation at the appropriate location in the Channel Entry Table.

Note: If EP = 0 is set in any channel program, or if PCP_CS.RCB = 1, a Channel Entry Table must be provided at the base of CMEM. Otherwise this table is not needed.

11.12.1.2 Channel Resume

When PCP_CS.RCB = 0, the program counter of the PCP is vectored to the address that is restored from the channel program's context. This means that before exiting, a channel program must itself arrange for where it will resume execution by configuring the value of its PC in its saved context so that it restarts at the desired location.

In this way, arbitrarily complex interrupt-driven state machines can be created as individual channel programs. Channel programs can be constructed that always start at their beginning, pick up where they left off, or pick up elsewhere, or have a mix of these approaches.

An example of a restarting channel program is shown below. Before exiting, the channel branches back to the address of the START label minus 1 (note that START - 1 = CH16) and then exits. This will leave the next value of the PC in the channel program's context as the address of the START label.

```
CH16:                                ;channel program 16
    EXIT EC=1 ST=0 INT=0 EP=1 cc_UC
                                        ;exit, no intr., leave PC @ next
START:                                ;nominal channel start address
    ST.IFbase #0x8 SIZE=32             ;output note from R0
    JC    CH16, cc_UC                  ;loop back before exit
```

Note that when the channel program is originally configured by the programmer, the PC field in the R7 context of this channel program should also be set to the address of the START label.

Similarly, an interrupt-driven state machine can be created by exiting with the next PC value pointing to the start of the next state in a state machine implemented by the channel program. The next example below shows a program starting at the address to the STATE0 label. It proceeds after the first interrupt to STATE1 - 1, where the channel program is left ready for the next state, STATE1 in the state machine. After the next interrupt, it executes to address STATE2 - 1 and the channel program is left ready for the next state, STATE2. After another interrupt, it proceeds through STATE2. The channel program jumps back to START, which is STATE0 - 1. The state machine has gone through one cycle and it is ready to restart in STATE0.

```
;This program is intended to test the sequence of exit/operate
;just as if you were implementing an interrupt driven state
;machine.
```

```
;It requires a periodic sequence of interrupts.
```

```
START:
    EXIT EC=1,ST=0,INT=0,EP=1,cc_UC    ;begin exit
STATE0:
    COMP.I    R5,#0x0                  ;compare to interrupt number it
                                        ;should be
```

Peripheral Control Processor (PCP)

```

JC          ERROR,cc_NZ      ;jump to error routine if not
                               ;correct
ADD.I      R5,#0x1          ;increment state number
EXIT EC=1,ST=0,INT=0,EP=1,cc_UC      ;begin exit
STATE1:
COMP.I     R5,#0x1          ;compare to interrupt number it
                               ;should be
JC         ERROR,cc_NZ      ;jump to error routine if not
                               ;correct
ADD.I      R5,#0x1          ;increment state number
EXIT EC=1,ST=0,INT=0,EP=1,cc_UC      ;begin exit
STATE2:
COMP.I     R5,#0x2          ;compare to interrupt number it
                               ;should be
JC         ERROR,cc_NZ      ;jump to error routine if not
                               ;correct
LD.I       R5,#0x0          ;reset state number
JC         START,cc_UC      ;jump back to start of state
                               ;machine

```

The last state could just as easily have ended with an EXIT that resets the PC to the Channel Entry Table (EP = 0) rather than jumping back to START.

11.12.2 Channel Management for Small and Minimum Contexts

If Small or Minimum Contexts are being used, only some of the registers are saved and restored. The integrity of the GPRs that are not included in the context must be handled explicitly by channel programs, since these are not saved and restored with the context of the interrupted channel program.

Channel programs may still use all registers reliably. Channel programs can be so designed that they either ignore the values in unsaved registers, or use those registers to store constants that no channel program changes. Hence they never need to be saved and restored. Alternately, channel programs can use these unused GPRs as temporary variables as long as the values of such registers cannot be corrupted by the interrupt of the channel program by a higher-priority channel (see [Page 11-119](#)).

11.12.3 Unused Registers as Globals or Constants

Registers R0 through R3 (for the Small Context Model), or R0 through R5 (for the Minimum Context Model) can be used to store constants such as addresses that are available to all channel programs. Hence, these registers hold global data, and no channel program is allowed to change them.

Since the GPRs of the PCP are not directly accessible from the FPI Bus, there does need to be an initial channel program that sets these values at or near boot time. There are

Peripheral Control Processor (PCP)

two choices here. A boot-time interrupt channel program can be invoked once to perform initialization, or there can be a program that routinely loads these values as a matter of course, and is invoked at boot time or as upon receipt of the very first interrupt.

11.12.4 Dispatch of Low Priority Tasks

A higher-priority channel program may wish to start a low-priority background task, or periodically pause and re-start itself later when there is no other action required at the moment. This can be accomplished in several ways, as follows.

- Post an SRPN to a free SRN on the FPI Bus, then EXIT.
- Perform an EXIT, posting the interrupt to the PCP, and indicating the channel number to be started.
- Use a single channel program as a list-driven or state-driven task dispatcher.

The first approach is straightforward to program, but uses a system SRN resource. Its advantage is that it allows continuous channel operation without using the interrupt queue, or risk blocking other uses of the PCP.

The second approach can be implemented by having a looping channel program continue operation in the background. It will also always be superseded by any higher-priority tasks.

The third approach uses a channel program to dispatch other non-interrupt-driven channel programs in an arbitrary order determined by the channel program dispatcher. In this way, multiple tasks could be continuously operated without over-using the PCP service-request queue. This approach would be useful when the aim is to poll for service requests in the peripheral SRN's rather than having them started by PCP hardware.

11.12.5 Code Reuse Across Channels (Call and Return)

A special jump instruction is included in the PCP instruction set to allow subroutines to be called from multiple channel programs. A routine may be jumped to directly, and then returned from using the JC.IA instruction. JC.IA allows a calling channel program to set aside a register for its return address, which will typically be the value of the next PC. The called subprogram can then execute a JC.IA, to the address stored in the register specified, causing a return-from-subroutine operation. The programmer must adopt and enforce a calling convention to determine which register holds the return address. Register R2 is normally used for this purpose.

For example:

Main Routine:		Subroutine:	
	LD.IL R2, #RETURN	SUB:	MOV...
	JC.A #SUB		ADD...
RETURN:	MOV
	...		JC.IA R2

11.12.6 Case-like Code Switches (Computed Go-To)

The JC.I instruction can be used to implement a multi-way branch for branch-on-bit or branch-on-state conditional branches. This instruction allows a conditional relative jump based on an index held in a register. If this instruction is combined with a table of jump addresses, a switch-type statement can be implemented. The default case, that is when the condition code = False, is the next instruction, as is the jump with register index = 0. The table can be any arbitrary length. The index register should be checked for range before the jump into the table is performed.

For Example:

```

                COMP R3,#5          ;compare R3 to #5 - the number of
                                ;entries in the table
                JC.I R3,cc_ULE
DEFAULT:       JL   #case_0        ;destination if R3 = 0 or
                                ;condition = false
                JL   #case_1        ;destination if R3 = 1
                JL   #case_2        ;destination if R3 = 2
                JL   #case_3        ;destination if R3 = 3
                JL   #case_4        ;destination if R3 = 4
                JL   #case_5        ;destination if R3 = 5
    
```

11.12.7 Simple DMA Operation

A simple interrupt-driven DMA requires at least the Small Context Model to operate properly. Its operation is comprised of three stages:

- The device interrupts the PCP to indicate it can receive or provide data.
- The PCP moves the amount of data it is programmed to move.
- The PCP eventually finishes and interrupts the CPU to notify it that the DMA is complete.

There are two options for implementing a simple DMA operation, copy and burst copy.

11.12.7.1 COPY Instruction

A simple DMA channel program can consist of just two instructions. In the example below, a device interrupts the PCP to notify it that it has data in its output buffer, which is 4 words deep. The COPY instruction copies 4 words to memory at a time. It decrements CNT1 (which is initialized by the CPU in CR6_CNT1 context) after each 4 word transfer. The EXIT command then executes, and if CNT1 was decremented to 0, the condition code causes it to issue an interrupt with the value held R6_SRPN.

```

COPY DST+,SRC,CNC=1,BRST=4,SIZE=32 ;do peripheral -> memory DMA
EXIT EC=0,ST=0,INT=1,EP=0,cc_CNZ ;transfer done, so exit
    
```

Peripheral Control Processor (PCP)

In the example above, the COPY instruction increments the destination held in R5 (DST+), and the source address is left constant in R4 (SRC). All permutations of decrement, increment or do not modify can be applied to either pointer register (R4 and R5) by use of the SRC and DST fields (SRC-, SRC+ or SRC and DST-, DST+ or DST). Building on this basic DMA method, scatter-gather DMA channels can be created.

11.12.7.2 BCOPY Instruction (Burst Copy)

The BCOPY instruction is in principle similar to the COPY instruction except that it uses the FPI Burst mode to perform the transfers rather than performing individual reads/writes. As for the COPY instruction, the FPI Bus is locked between the burst read and burst write to ensure that a valid set of data is transferred. The BCOPY instruction allows support of all burst sizes supported by FPI Burst Mode except a burst size of 1 (i.e. 2, 4 or 8 words). The CNT0 field is used to control the burst size. Both the source and destination addresses (R4 and R5) must be correctly aligned for the burst size being used (see the FPI Bus description for details). If either address is incorrectly aligned, the PCP will generate an Illegal Operation Error Exit.

See also [Page 11-127](#) for TC1796 specific details of the BCOPY instruction.

11.13 PCP Programming Notes and Tips

This section discusses constraints on the use of the PCP and points out some non-obvious issues.

11.13.1 Notes on PCP Configuration

- Only one Context Model may be used at a time for all channels, and the PCP must remain in that Context Model once started and configured.
- In order for a specific channel program to be enabled, its context must have $R7.CEN = 1$. If $R7.CEN = 0$, then the channel program will terminate when invoked, and cause a Disabled Channel Request error.
- The Channel Context Address from the FPI Bus as viewed during channel configuration is as follows:
 - Full Context Model: $PRAM\ Base + 20_H \times n$
 - Small Context Model: $PRAM\ Base + 10_H \times n$
 - Minimum Context Model: $PRAM\ Base + 08_H \times n$where n is the channel number.
- $PCP_CS.RCB$ and context must be consistent. If RCB is configured to 0, then each channel program will start at the PC restored from its context. If the wrong address is pre-configured in the context, the channel program will not operate properly.
- The programmer of the PCP may lock PCP_CS by setting $PCP_CS.EIE = 1$. When the global $ENDINIT$ bit is set, the PCP_CS register will no longer be writable, and attempting to do so will cause an FPI Bus error.
- An error condition will result in an interrupt being sent to the local FPI Bus master. The targeted interrupt service routine must be capable of dealing with the cause as recorded in PCP_ES , and, if required, it must be able to return the halted channel program to operation. The minimum required to do that is to set the context value of $R7.CEN = 1$.
- The only PCP Register bit that can be dynamically modified during PCP operation is the $PCP_CS.EN$ bit. When writing to any other PCP Register bits, the user must ensure that the PCP is disabled ($PCP_CS.EN = 0$) and that the PCP is quiescent ($PCP_CS.RS = 0$).

11.13.2 General Purpose Register Use

- The most significant 16 bits of R7 may not be written, and will always read back as 0. However, no error will occur if a write to the most significant 16 bits occurs.
- Care must be taken with the use of R6 as a general-use register to ensure that R6 contains the correct value prior to execution of the EXIT command. As R6 contains the CNT1 (counter used in COPY and optionally in EXIT instructions), SRPN and TOS (service request number to use during optional interrupt at channel program EXIT) fields, R6 should not be used to pass values from one invocation of a channel program to the next invocation.
- If PRAM is to be accessed programmatically, then R7.DPTR must be configured properly as a pointer into the PRAM. This points to the 64-word segment that may be addressed by the xx.P instructions and the xx.PI instructions. It is not recommended to set R7.DTPR to point into the CSA. Special care must be taken that the Context PRAM is not overwritten.
- The programmer must be careful not to inadvertently clear R7.CEN when updating R7.DTPR (or any other field in R7). This would cause the channel program to generate a disabled channel interrupt to the CPU when the next interrupt request to the channel occurs.
- Any update to the Flags that is caused by an instruction (e.g. MOV R7, R0 which updates Z and N) takes precedence over any explicit bits that are moved to R7. See [Page 11-9](#).
- The interrupt system assumes SRPN 0 is not a request. Full Context packing leaves the least significant 8×32 -bit entries where channel 0 would normally be un-used. That is, PRAM Base \rightarrow PRAM Base + 1 channel. In addition, for Small Context, the least significant 4×32 -bit entries are un-used, and for Minimum Context the least significant 2×32 -bit entries are un-used. These “un-used” entries should not be used by channel programs.
- If EP = 0 is used, or if PCP_CS.RCB = 1, a Channel Entry Table must be provided at the base of CMEM.
- If there is a plan to use the Small or Minimum Context model, and the lower registers are to hold global values, then there needs to be an initial channel program that sets these values at or near boot time. There are at least two choices for implementing this. For instance, a boot interrupt channel program can be invoked once to perform initialization, or there can be a program that routinely loads these values as a matter of course, and it is invoked at boot time, or at the very first interrupt. See [Page 11-114](#).
- When using Small or Minimum Context models and allowing a channel to be interrupted, care must be taken to ensure that the value of any registers that are not included in the context but are being used by a channel are not corrupted by interruption of the channel and subsequent operation of a higher-priority channel. Particular care must be taken when using instructions that use R0 implicitly. If

Peripheral Control Processor (PCP)

necessary, critical instruction sequences should be protected by use of the R7.IEN bit (see [Page 11-120](#)).

11.13.3 Use of Channel Interruption

- When a channel program consists of only a few instructions, it is best to configure the channel to be non-interruptible. This increases overall efficiency by removing the context save/restore overhead that would be incurred if the channel were to be interruptible.

11.13.3.1 Dynamic Interrupt Masking

A channel program can dynamically control whether it can be interrupted by use of the R7.IEN bit. When masking interrupts (by clearing R7.IEN), it must be noted that there is a delay of one instruction before the mask becomes effective. As a result the instruction that clears R7.IEN must be placed at least one instruction before the instruction sequence that is to be un-interruptible. As an example, consider the following sequence:-

```

CLR    R7,IEN    ;Clear the R7.IEN bit
                    ;<< Interrupt can occur here

NOP

                    ;<< Interrupt can occur here
                    ;First instruction of non-interruptible code
                    ;sequence

```

11.13.3.2 Control of Channel Priority (CPPN)

The PCP has three extended Service Request Nodes (PCP_SRC9, PCP_SRC10 and PCP_SRC11) that allow storage of suspended channel interrupt requests. This allows interrupt nesting to a depth of four. This limit on the nesting depth carries the danger that a high-priority service request will not be serviced because the PCP's interrupt nesting depth has been exceeded.

It is recommended that a four-level "grouping" scheme should be adopted to avoid this problem. All PCP interrupt sources should be listed in order of their SRPNs. This list should then be subdivided into four contiguous groups, Group 0 being the lowest priority and Group 3 the highest. The CSA for each channel program should be configured such that CR6.CPPN contains the SRPN value of the highest channel program within the group to which the channel belongs. As each channel starts, the Operating Priority (CPPN) of the channel is loaded from the context. Using the scheme recommended above, any channel program will run with the priority of the highest SRPN within the group. As a result, the channel can only be interrupted by a service request from a higher-priority group (e.g. a Group 0 channel program can be interrupted by a new

Peripheral Control Processor (PCP)

service request for a channel in any group from 1 to 3, a group 2 channel program can only be interrupted by a new service request for a channel in group 3).

Note: When using this scheme, each channel program must ensure prior to channel exit that the R6.CPPN field contains the appropriate value, so that when the channel is next invoked, it will run at the correct priority.

11.13.4 Implementing Divide Algorithms

As discussed in [Section 11.11.4](#), a divide algorithm **must** always start with a DINIT instruction followed by a number of DSTEP instructions (up to four depending on the data width that is required). Prior to execution of any DSTEP instruction, R0 always contains either 0 (if this is the first DSTEP instruction in a divide sequence R0 contains 0 due to the preceding DINIT instruction), or the remainder from the previous DSTEP instruction). The dividend to be used in this step is generated in R0 by taking $256 \times$ the remainder of the last DSTEP instruction ($R0 \ll 8$) and adding the most significant byte of Rb ($Rb \gg 24$) as the LSB of the new dividend.

Since the remainder of the last DSTEP instruction is by definition always less than the divisor (Ra), it can be guaranteed that the result of the division of the dividend (calculated as above) by the divisor (Ra) can always be contained within an 8-bit result. The description given at [Page 11-88](#) only holds true under this condition. If the restrictions on the use of the DSTEP instruction (specified at [Page 11-79](#)) are adhered to, the above condition will always be met and this description of the instruction is correct. Failure to adhere to these conditions will lead to invalid results, which are outside the scope of this document.

During execution of a divide sequence, Rb is used both to compile the final divide result and to hold the remnants of the original dividend. For example, in a 32-/32-bit divide sequence (which consists of 4 DSTEP instructions - see below), Rb will have the following content:

- After the 1st DSTEP instruction:
The least significant 3 bytes (24 bits) of the original 32-bit dividend (held in the most significant 3 bytes of Rb) and the most significant byte of the final result (held in the least significant byte of Rb).
- After the 2nd DSTEP instruction:
The least significant 2 bytes (16 bits) of the original 32-bit dividend (held in the most significant 2 bytes of Rb) and the most significant 2 bytes of the final result (held in the least significant 2 bytes of Rb).
- After the 3rd DSTEP instruction:
The least significant byte of the original 32-bit dividend (held in the most significant byte of Rb) and the most significant 3 bytes of the final result (held in the least significant 3 bytes of Rb).
- After the final DSTEP instruction:
The 32 bit final result.

Note that the DSTEP instruction **always** uses the divisor as a 32 bit value. In any divide sequence, the dividend can be 8, 16, 24 or 32 bits (according to the number of DSTEP instructions in the sequence) but the divisor is **always** 32 bits. Prior to the DINIT instruction, the dividend must always occupy the appropriate most significant bits within the 32-bit dividend register (Rb).

Divide Examples

Example of a 32/32 bit divide (R5 / R3):

```

DINIT      R5, R3          ;Initialize ready for the divide
JC         HANDLE_DIVIDE_BY_ZERO, cc_V ;V flag was set so
                                                ;jump to divide
                                                ;by zero error handler
DSTEP      R5, R3          ;4 DSTEP instructions
                                                ;(4 * 8 = 32 bit
DSTEP      R5, R3          ; divide)
DSTEP      R5, R3
DSTEP      R5, R3
    
```

After this sequence, R5 holds the result, R0 the remainder and R3 is unchanged.

Example of a 8/32 bit divide (R4 / R2):

```

RR         R4, 8           ;Rotate R4 right by 8 to move
                                                ;least significant byte into
                                                ;most significant byte
DINIT      R4, R2          ;Initialize ready for the divide
JC         HANDLE_DIVIDE_BY_ZERO, cc_V ;V flag was set so
                                                ;jump to divide
                                                ;by zero error handler
DSTEP      R4, R2          ;DSTEP instruction
                                                ;(1 * 8 = 8 bit divide)
    
```

After this sequence, R4 holds the result, R0 the remainder and R2 is unchanged.

Note that the above example is specified as being a 8/32 bit divide rather than an 8/8 bit divide (see comments above).

11.13.5 Implementing Multiply Algorithms

As discussed in [Section 11.11.4](#), a multiply algorithm **must** always start with a MINIT instruction, followed by a number of MSTEP32 or MSTEP64 instructions. The MSTEP32 instruction is used to compile a multiplication result contained in 32 bits, discarding any overflows. The MSTEP64 instruction is used to compile a 64-bit multiplication result with the least significant 32 bits of the result contained in Rb and the most significant 32 bits of the result contained in R0.

Multiply Examples

Example of a 32 × 8 bit multiply (R4 × R1) yielding a 32 bit result (R4 = 32 bit, R1 = 8 bit):

```

RR         R1, 8           ;Rotate least significant byte of R1 to
                                                ;most significant byte
MINIT      R1, R4          ;Initialize ready for multiply
    
```


Peripheral Control Processor (PCP)

```
MSTEP32    R1, R4    ;Perform one MSTEP32 instruction
                ;(8 bit multiply)
```

After this sequence, R0 holds the result, R1 is left unchanged (right rotated by RR instruction then left rotated by MSTEP32 instruction), and R4 is unchanged. The result is only valid if there is no overflow (i.e. the product of the 8-bit number in R1 multiplied by the 32-bit number in R4 can be contained within 32 bits). It is the user's responsibility to ensure that this is the case. The overflow condition cannot be detected after execution of the multiply sequence.

Example of a 32×16 bit multiply ($R3 \times R2$) yielding a 32 bit result
(R3 = 32 bit, R2 = 16 bit):

```
RR          R2, 8    ;Perform two 8 bit rotations
                ;(RR instructions) to get original
                ;least significant 16 bits into
                ;most significant 16 bits

RR          R2, 8
MINIT       R2, R3   ;Initialize ready for multiply
MSTEP32     R2, R3   ;Perform two MSTEP32 instructions
                ;(16 bit multiply)

MSTEP32     R2, R3
```

After this sequence, R0 holds the result, R2 is left unchanged (right rotated by two RR instructions, then left rotated by two MSTEP32 instructions), R3 is unchanged. The comment above regarding overflow also applies to this sequence.

Example of a 32×32 bit multiply ($R5 \times R2$) yielding a 64 bit result
(R5 = 32 bit, R2 = 32 bit):

```
MINIT       R2, R5   ;Initialize ready for multiply
MSTEP64     R2, R5   ;Perform 4 MSTEP64 instructions
                ;(64-bit multiply)

MSTEP64     R2, R5
MSTEP64     R2, R5
MSTEP64     R2, R5
```

After this sequence R0 and R2 hold the result (most significant word in R0, least significant word in R2), R5 is unchanged. There is no possibility of overflow as the result of 32×32 bits can always be contained in 64 bits.

11.14 Implementation of the PCP in the TC1796

The addresses of the PCP registers and memories in the TC1796 are given in the following subsections.

11.14.1 PCP Memories

In the TC1796, the location of the registers and the memories sizes of the PRAM and the CMEM are given in [Table 11-16](#).

Table 11-16 General Block Address Map

Unit		Address Range	Access Mode		Size
			Read	Write	
PCP	Reserved	F004 0000 _H - F004 3EFF _H	BE	BE	–
	PCP Registers	F004 3F00 _H - F004 3FFF _H	see Page 11-52		256 bytes
	Reserved	F004 4000 _H - F004 FFFF _H	BE	BE	–
	PCP Data Memory (PRAM)	F005 0000 _H - F005 3FFF _H	nE, 32	nE, 32	16 Kbyte
	Reserved	F005 4000 _H - F005 FFFF _H	BE	BE	–
	PCP Code Memory (CMEM)	F006 0000 _H - F006 7FFF _H	nE, 32	nE, 32	32 Kbyte

Note: “BE” means that in case of an access to this address region, a bus error is generated.

Note: The complete address map of the PCP is described in [Table 18-25](#) on [Page 18-77](#) of this TC1796 System Units (Vol. 1 of 2) User’s Manual.

11.14.2 Parity Protection for PCP Memories

In the TC1796, the PRAM and CMEM memory blocks of the PCP are both equipped with a parity error detection logic that makes it possible to detect parity errors separately for PRAM or CMEM. In case of a parity error a NMI is generated.

Note that before using parity protection for PRAM and CMEM the first time after a power-on reset operation (before setting the corresponding parity error enable bits), the corresponding PCP memories must be completely initialized by a user program that writes every memory location of it once.

More details about the parity control for on-chip memories are described in [Section 5.5](#) on [Page 5-37](#).

11.14.3 PCP Reset Operation

The PCP module can be reset by two ways:

- By a system hardware signal (hard reset)
- By programming of a PCP register bit (soft reset)

PCP Hard Reset

A PCP hard reset is always triggered if at least one of these TC1796 reset sources becomes active:

- Watchdog Timer Reset
- Software Reset
- Hardware Reset
- Power-on Reset

Each of these reset sources forces a hardware reset of the functional blocks within the PCP module. The effect of hard reset within the PCP is to:

- Halt any operating channel
- Reset all control registers to their reset values
- Reset the PCP Processor Core to its default state
- Reset the FPI Bus interface

PCP Soft Reset

A PCP soft reset is generated by writing a 1 to bit PCP_CS.RST (see also [Page 11-56](#)). A PCP soft reset differs from the PCP hard reset in that the interrupt system is not reset. Specifically, all PCP interrupt nodes with their service request control registers are not completely reset, and the PICU is also not affected.

The PCP soft reset is adopted to prevent loss of synchronization between any interrupt node and the Arbitration Unit (ICU) to which it is connected due to one, but not the other unit is reset. The effect of PCP soft reset is to:

- Halt any operating channel
- Reset all control registers (except PCP_SRCx registers) to their reset values
- PCP_SRCx registers: clear all SRR and RRQ bits and leave all other bits unchanged
- Reset the PCP Processor Core to its default state
- Reset the FPI Bus interface
- Clear all SRR and RRQ bits in all PCP_SRCx registers (all other bits are unchanged)

Note: The PCP soft reset is a useful hardware support feature in a PCP debugging environment. It has been included for backward compatibility with PCP version 1. However, generation of a PCP soft reset may be critical because it may cause a

Peripheral Control Processor (PCP)

*FPI Bus error as the PCP's interface to the FPI Bus becomes reset.
Therefore, it is generally not recommended to use the PCP soft reset capability in a PCP application program. A PCP hard reset should be generated instead.*

11.14.4 BCOPY Instruction

In the TC1796, the BCOPY instruction can be used to perform burst transfers (2, 4, or 8 words) with DMI memories (Local data RAM and Dual-port RAM) and the PCP memories. Other internal and external memories can be accessed using a burst size of 2 words only (CNT0 = 10_B).

12 Direct Memory Access Controller

This chapter describes the Direct Memory Access (DMA) Controller and the Memory Checker Module (MCHK) of the TC1796. It contains the following sections:

- Functional description of the DMA controller kernel (see [Page 12-2](#))
- DMA controller kernel register description (see [Page 12-42](#))
- TC1796 implementation-specific details of the DMA controller (interrupt control, address decoding, clock control, see [Page 12-91](#))
- Functional description of the MCHK module (see [Page 12-110](#))

Direct Memory Access Controller

12.1 DMA Controller Kernel Description

The DMA Controller of the TC1796 transfers data from data source locations to data destination locations without intervention of the CPU or other on-chip devices. One data move operation is controlled by one DMA channel. Sixteen DMA channels are provided in two independent DMA Sub-Blocks with eight DMA channels each. The Bus Switch provides the connection of two DMA Sub-Blocks to the two FPI Bus interfaces and an MLI bus interface. In the TC1796, the FPI Bus interfaces are connected to the System Peripheral Bus and the Remote Peripheral Bus. The third specific bus interface provides a connection to Micro Link Interface modules (two MLI modules in the TC1796) and other DMA-related devices (Memory Checker module in the TC1796). Clock control, address decoding, DMA request wiring, and DMA interrupt service request control are implementation-specific and managed outside the DMA controller kernel.

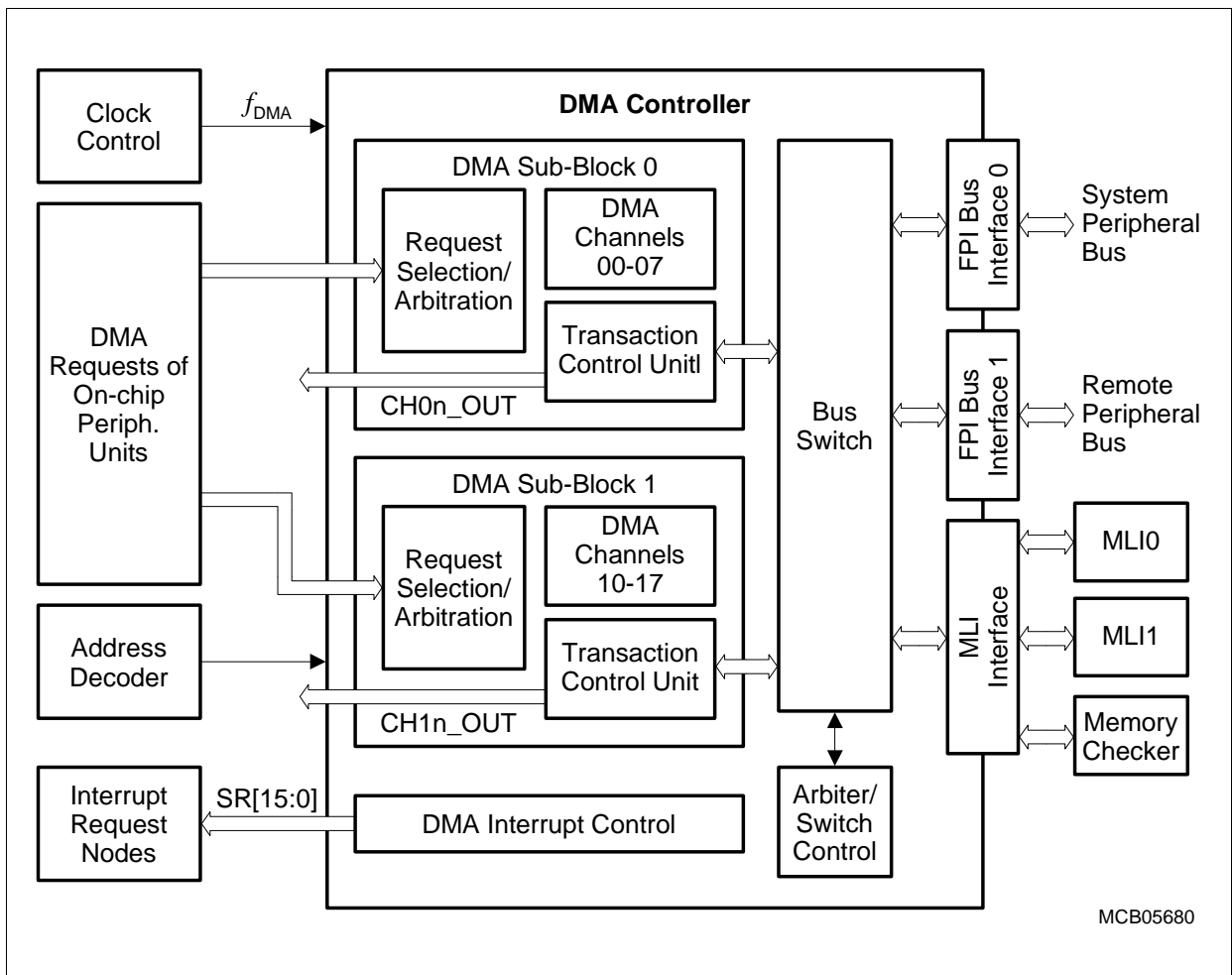


Figure 12-1 DMA Block Diagram

12.1.1 Features

- 16 independent DMA channels
 - 8 DMA channels in each DMA Sub-Block
 - Up to 8 selectable request inputs per DMA channel
 - 2-level programmable priority of DMA channels within a DMA Sub-Block
 - Software and hardware DMA request
 - Hardware requests by selected on-chip peripherals and external inputs
- Programmable priority of the DMA Sub-Blocks on the bus interfaces
- Buffer capability for move actions on the buses (at least 1 move per bus is buffered)
- Individually programmable operation modes for each DMA channel
 - Single Mode: stops and disables DMA channel after a predefined number of DMA transfers
 - Continuous Mode: DMA channel remains enabled after a predefined number of DMA transfers; DMA transaction can be repeated
 - Programmable address modification
- Full 32-bit addressing capability of each DMA channel
 - 4 Gbyte address range
 - Support of circular buffer addressing mode
- Programmable data width of DMA transfer/transaction: 8-bit, 16-bit, or 32-bit
- Micro Link bus interface support
- Register set for each DMA channel
 - Source and destination address register
 - Channel control and status register
 - Transfer count register
- Flexible interrupt generation (the service request node logic for the MLI channels is also implemented in the DMA module)
- All buses connected to the DMA module must work at the same frequency
- Read/write requests of the System Bus side to the Remote Bus peripherals are bridged to the Remote Peripheral Bus (only the DMA is master on the RPB)

12.1.2 Definition of Terms

DMA Move

A DMA move is an operation that always consists of two parts:

1. A read move that loads data from a data source into the DMA controller
2. A write move that puts data from the DMA controller to a data destination

Within a DMA move, data is always moved from the data source via the DMA controller to the data destination. Data is temporarily stored in the DMA controller. The data width of read move and write move are always identical (8-bit, 16-bit or 32-bit). Data assembly or disassembly is not supported.

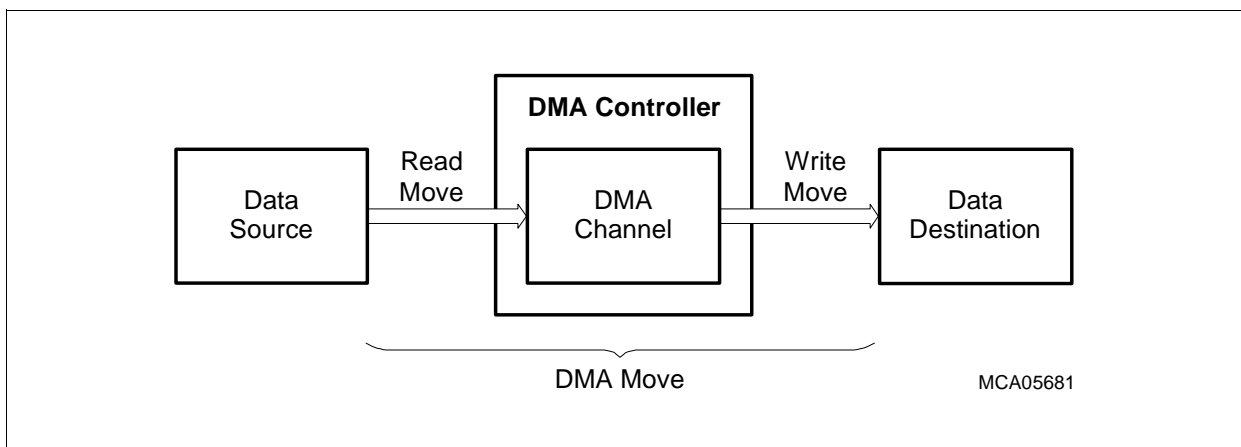


Figure 12-2 DMA Definition of Terms

DMA Transfer

A DMA transfer can be composed of 1, 2, 4, 8 or 16 DMA moves.

DMA Transaction

A DMA transaction is composed of several (at least one) DMA transfers. The Transfer Count determines the number of DMA transfers within one DMA transaction.

Example:

1024 word (32-bit wide) transactions can be composed of 256 transfers of four DMA word moves, or 128 transfers of eight DMA word moves.

12.1.3 DMA Principles

The DMA controller supports DMA moves from one address location to another one. DMA moves can be requested either by hardware or by software. DMA hardware requests are triggered by specific request lines from the peripheral modules or from other DMA channels (see [Figure 12-3](#)). The number of available DMA request lines from a peripheral module varies depending on the module functionality. Typically, the occurrence of a receive or transmit data interrupts in a peripheral module are able to generate a DMA request in parallel to an interrupt request. Therefore, the interrupt control unit and the DMA controller can react independently to interrupt and DMA requests that have been generated by one source.

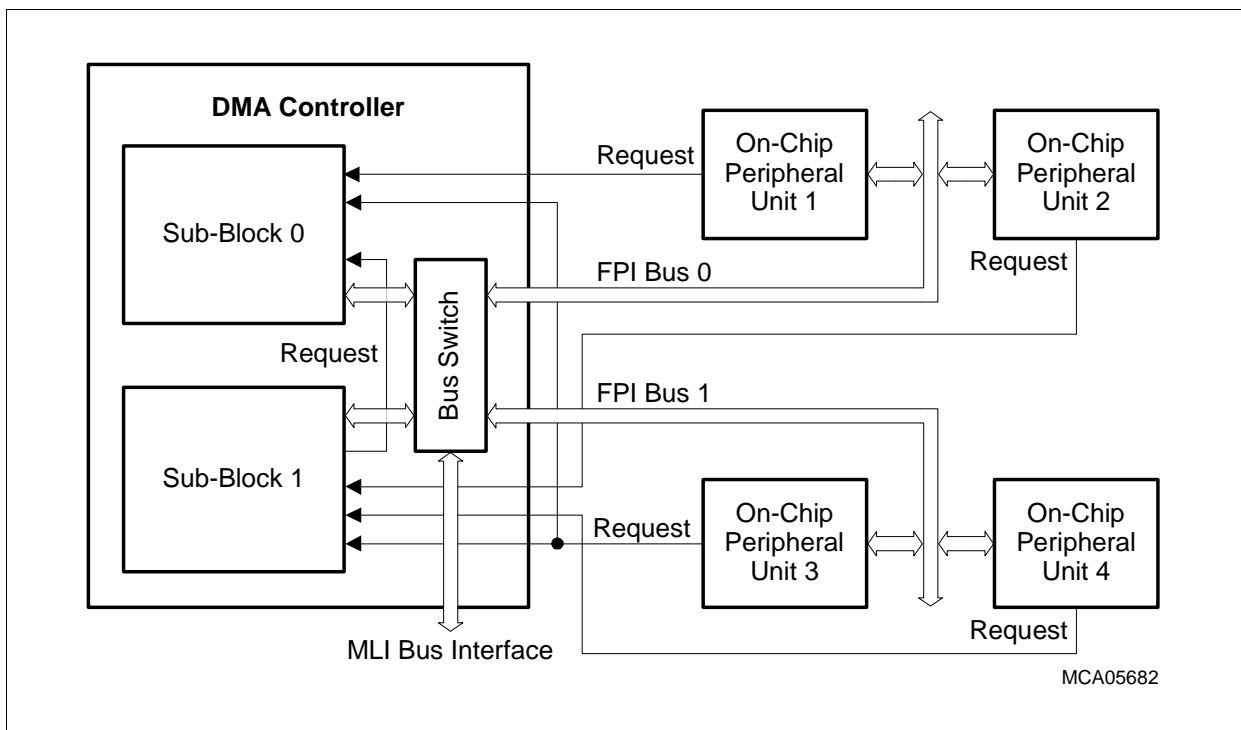


Figure 12-3 DMA Principle

The DMA controller mainly consists of two DMA Sub-Blocks and a Bus Switch. Once configured, each DMA Sub-Block is able to act as a master on the two FPI Buses.

12.1.4 DMA Channel Functionality

Each of the 2×8 DMA channels has one associated register set containing seven 32-bit registers. These registers are numbered by two indexes to indicate the related DMA channel: Index “m” refers to the DMA Sub-Block number ($m = 0-1$) and index “n” refers to the channel number ($n = 0-7$) within a DMA Sub-Block.

Example: CHCR14 is the Control Register of DMA channel 4 in Sub-Block 1.

The register set of a DMA channel register contains the following registers:

- Channel mn Control Register CHCRmn (for details, see [Page 12-76](#))
- Channel mn Status Register CHSRmn (for details, see [Page 12-80](#))
- Channel mn Interrupt Control Register CHICRmn (for details, see [Page 12-81](#))
- Channel mn Address Control Register ADRCRmn (for details, see [Page 12-83](#))
- Channel mn Source Address Register SADRmn (for details, see [Page 12-88](#))
- Channel mn Destination Address Register DADRmn (for details, see [Page 12-89](#))
- Channel mn Shadow Address Register SHADRmn (for details, see [Page 12-90](#))

12.1.4.1 Shadowed Source or Destination Address

As a typical application, an ASC module that receives data (fixed source address) has to deliver it to a memory buffer using a DMA transaction (variable destination address). After a certain amount of data has been transferred, a new DMA transaction should be initiated to deliver further ASC data into another memory buffer. While the destination address register is updated during a running DMA transaction with the actual destination address, a shadow mechanism allows programming of a new destination address without disturbing the content of the destination address register. In this case, the new destination address is written into a buffer registers, i.e. the shadow address register. At the start of the next DMA transaction, the new address is transferred from this shadow address register to the destination address register without CPU intervention. This shadow mechanism avoids the CPU having to check for the end of a DMA transaction before reprogramming address registers.

The shadow address register can be used also to store a source address. However, it cannot store source and destination address at the same time. This means that the shadow mechanism makes it possible to automatically update either a new source address, or a new destination address at the start of a DMA transaction. If both address registers (for source and destination address) have to be updated for the next DMA transaction, a running DMA transaction for this channel must be finished. After that, source and destination address registers can be written before the next DMA transaction is started.

Figure 12-4 shows the actions that take place when a source address register is updated. The update of a destination register happens in an equivalent manner.

When writing a new address to the (address of) the source or destination address register and no DMA transaction is running, the new address value is directly written into

Direct Memory Access Controller

the source or destination address register. In this case, no buffering of the address is required. When writing a new address to the (address of) the source or destination address register and a DMA transaction is running, no transfer to an address register can take place and SHADRmn holds the new address value that was written. For this operation, bit field ADRCRmn.SHCT must be set either to 01_B (address is a source address) or 10_B (new address is a destination address). At the start of the next DMA transaction, the shadow transfer takes place and the content of SHADRmn is written either into SADRmn or DADRmn (ADRCRmn.SHCT must be set accordingly). After the shadow transfer, SHADRmn is set to 0000 0000_H. Therefore, the software can check by reading the shadow address register whether or not the shadow transfer has already taken place.

Only one address register can be shadowed while a transaction is running, because the shadow register can only be assigned either to the source or to the destination address register. Note that the shadow address register transfer has the same behavior in Single or Continuous Mode. When the shadow mechanism is disabled (ADRCRmn.SHCT = 00_B), SHADRmn is always read as 0000 0000_H.

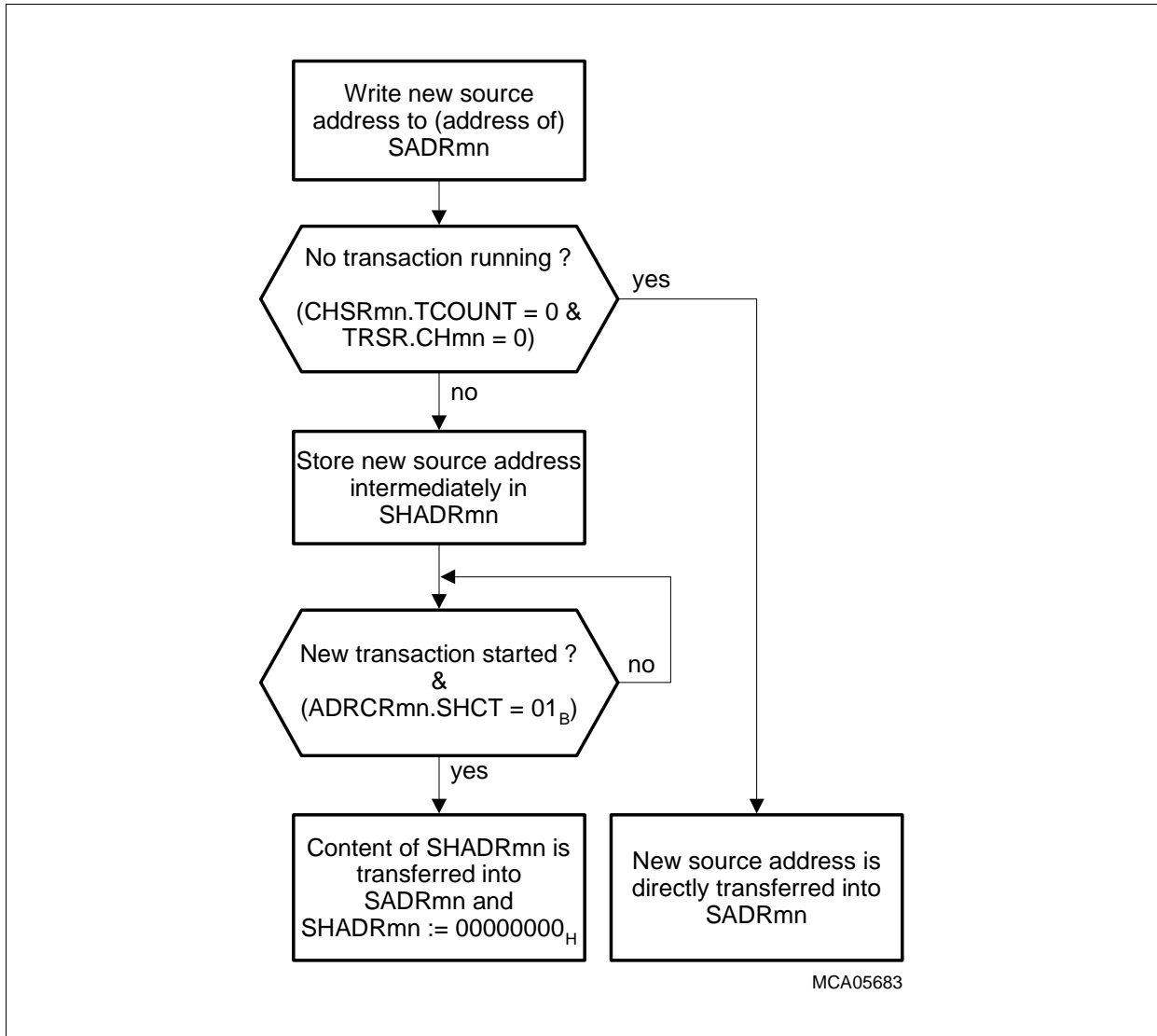


Figure 12-4 Source Address Update

The transfer count of a DMA transaction, stored in bit field CHCRmn.TREL, can also be programmed if the DMA transaction is running. At the start of a DMA transaction, TREL is transferred to bit field CHSRmn.TCOUNT, which is then updated during the DMA transaction.

No reload of address or counter will be done if TCOUNT is not equal to 0.

The reprogramming of channel specific values (except for the selected address shadow register) should be avoided while a DMA channel is active.

Direct Memory Access Controller

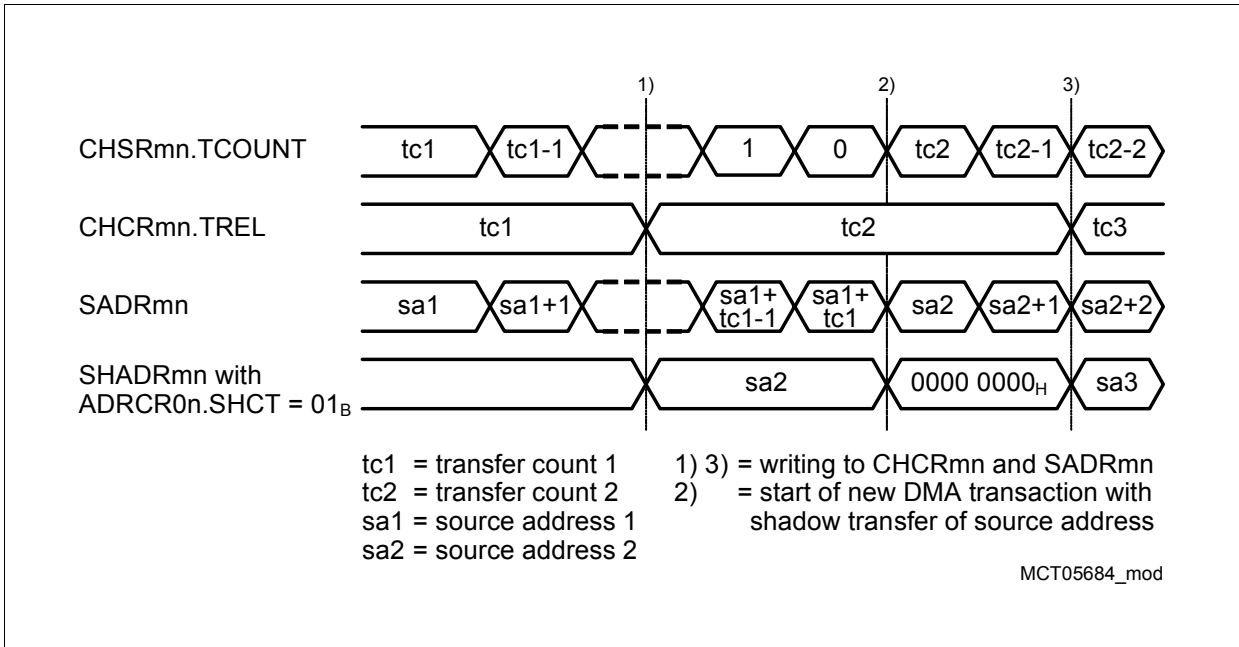


Figure 12-5 Shadow Source Address and Transfer Count Update

Figure 12-5 shows how the contents of the source address register SADRmn and the transfer count CHSRmn.TCOUNT are updated during two DMA transactions with a shadowed source address and transfer count update.

At reference point 2) the DMA transaction 1 is finished and DMA transaction 2 is started. At 1) the DMA channel is reprogrammed with two new parameters for the next DMA transaction: Transfer count tc2 and source address sa2. Source address sa2 is buffered in SADRmn and transferred to SADRmn when the new DMA transaction is started at 2). At this time, transfer count tc2 is also transferred to CHSRmn.TCOUNT.

12.1.4.2 DMA Channel Request Control

Figure 12-6 shows the control logic for DMA requests that is implemented for each DMA channel.

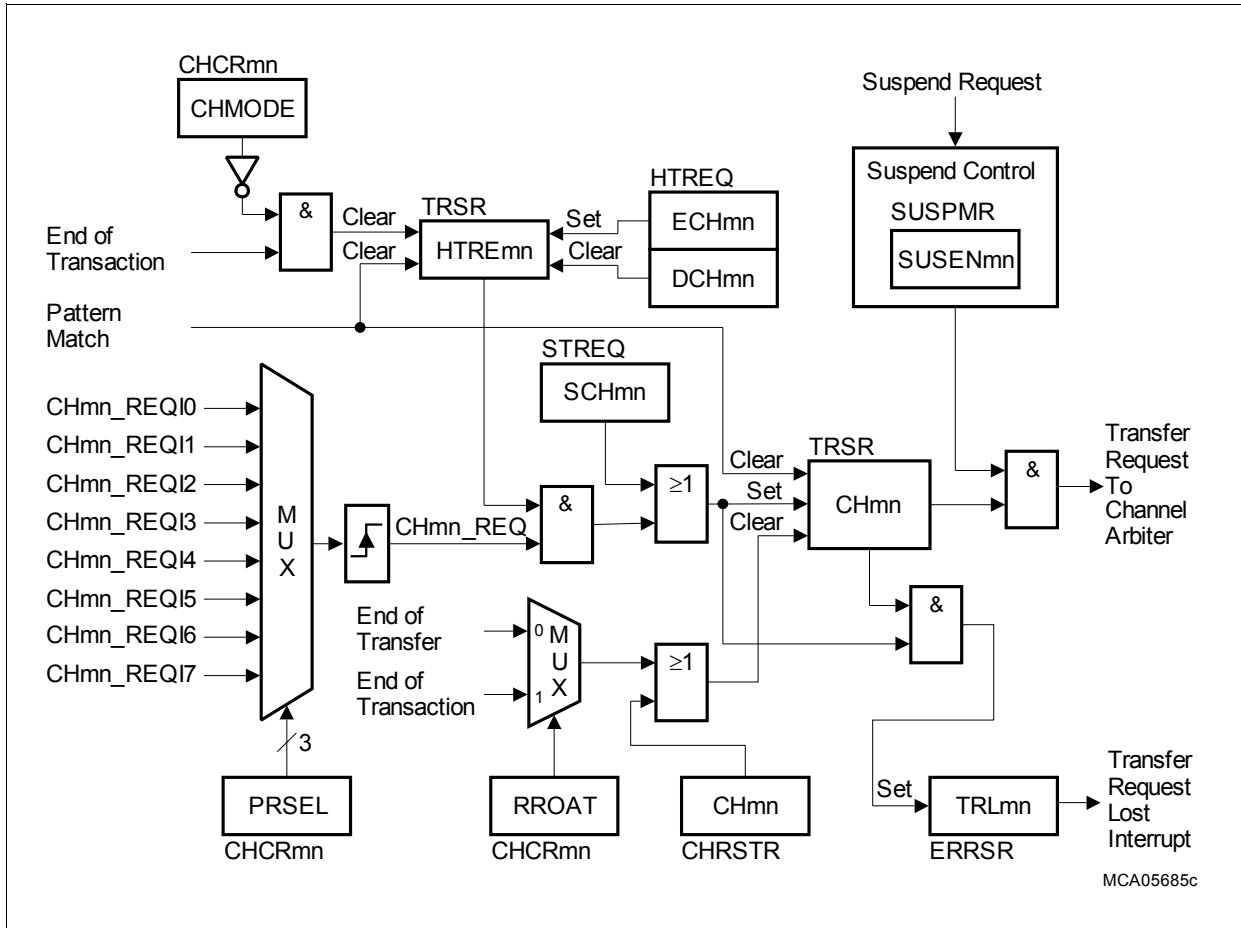


Figure 12-6 Channel Request Control

Two different types of DMA requests are possible:

- Hardware DMA requests
- Software DMA requests

The hardware request CHmn_REQ can be connected to one of eight possible hardware request input lines as selected by bit field CHCRmn.PRSEL. Hardware requests are enabled/disabled by status bit TRSR.HTREmn. HTREmn can be set/cleared by software or by hardware in Single Mode at the end of a DMA transaction. A software request can be generated by setting bit STREQ.SCH0n.

Status flag TRSR.CHmn indicates whether or not a software or hardware generated DMA request for DMA channel mn is pending. TRSR.CHmn can be cleared by software or by hardware at the end of a DMA transfer (RROAT = 0) or at the end of a DMA transaction (RROAT = 1).

Direct Memory Access Controller

If a software or a hardware DMA request is detected for channel *mn* while `TRSR.CHmn` is set, a request lost event occurs. This error event indicates that the DMA is already processing a transfer and that another transfer has been requested before the end of the previous one. In this case, bit `ERRSR.TRLmn` will be set and a transfer lost interrupt can be generated.

12.1.4.3 DMA Channel Operation Modes

The operation mode of a DMA channel is individually programmable for each DMA channel *mn*. Basically, a DMA channel can operate in the following modes:

- Software-controlled mode
- Hardware-controlled mode, in Single or Continuous Mode

In software-controlled mode, a DMA channel request is generated by setting a control bit. In hardware-controlled mode, a DMA channel request is generated by request signals typically generated by on-chip peripheral units.

In hardware-controlled Single Mode, a DMA channel *mn* becomes disabled by hardware after the last DMA transfer of its DMA transaction. In hardware-controlled Continuous Mode, a DMA channel *mn* remains enabled after the last DMA transfer of its DMA transaction.

In hardware- and software-controlled mode, a DMA request signal can be configured to trigger a complete DMA transaction or one single transfer.

Software-controlled Modes

In software-controlled mode one software request starts one complete DMA transaction or one single DMA transfer. Software-controlled modes are selected by writing `HTREQ.DCHmn = 1`. This forces status flag `TRSR.HTREmn = 0` (hardware request of DMA channel *mn* is disabled).

The software-controlled mode that initiates one complete DMA transaction to be executed is selected for DMA channel *mn* by the following write operations:

- `CHCRmn.RROAT = 1`
- `STREQ.SCHmn = 1`

Setting `STREQ.SCHmn` to 1 (this is the software request) causes the DMA transaction of DMA channel *mn* to be started and `TRSR.CHmn` to be set. At the start of the DMA transaction, the value of `CHCRmn.TREL` is loaded into `CHSRmn.TCOUNT` (transfer count or *tc*) and the DMA transfers are executed. After each DMA transfer, `TCOUNT` becomes decremented and next source and destination addresses are calculated. When `TCOUNT` reaches the 0, DMA channel *mn* becomes disabled and status flag `TRSR.CHmn` is cleared. Setting `STREQ.SCHmn` again starts a new DMA transaction of DMA channel *mn* with the parameters as actually defined in the channel register set.

Direct Memory Access Controller

The software-controlled mode that initiates a single DMA transfer to be executed is selected for DMA channel mn by the following write operations:

- CHCRmn.RROAT = 0
- STREQ.SCHmn = 1, repeated for each DMA transfer

When CHCRmn.RROAT = 0, TRSR.CHmn becomes cleared after each DMA transfer of the DMA transaction and a new software request (writing STREQ.SCHmn = 1) must be generated for starting the next DMA transfer.

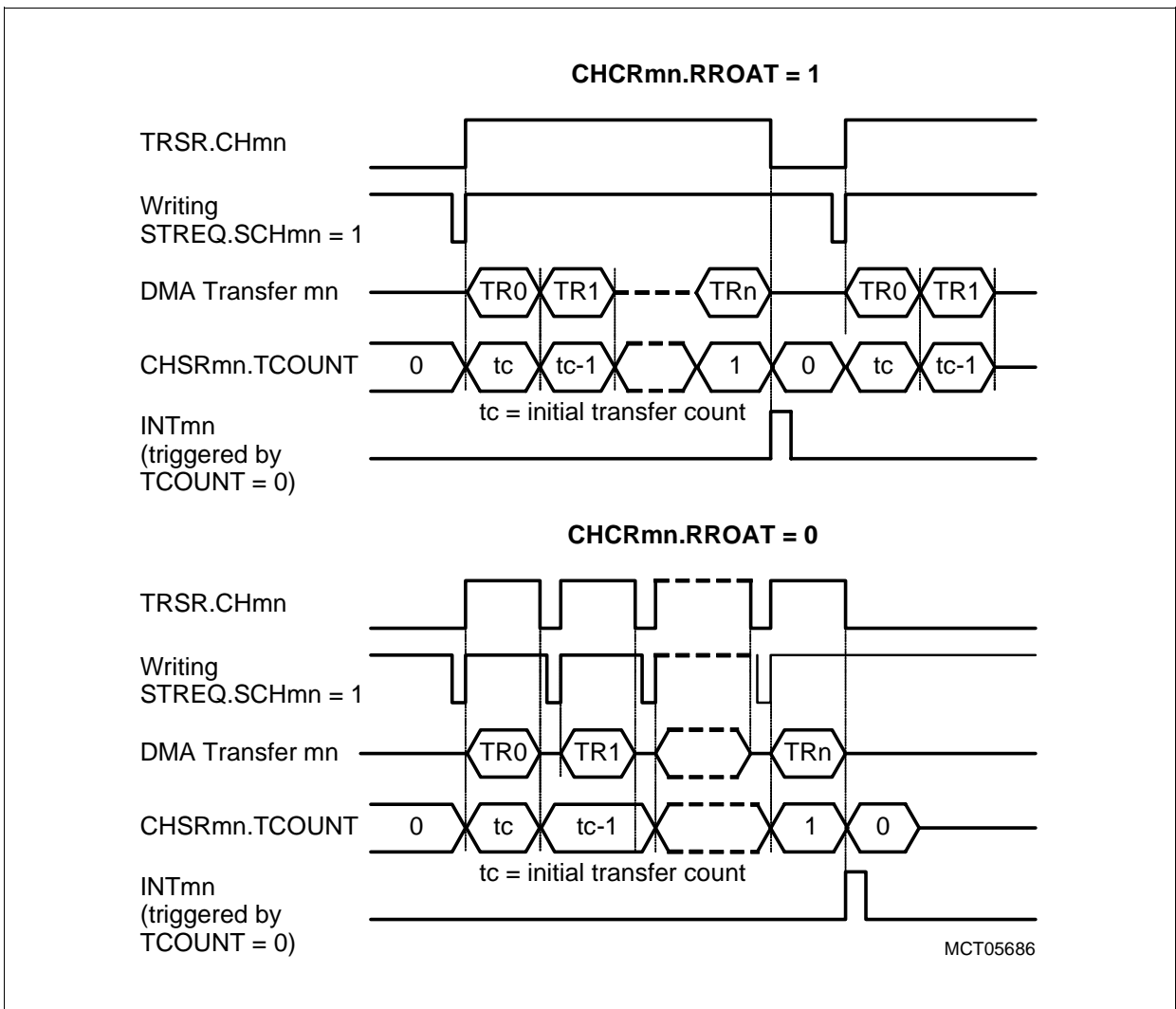


Figure 12-7 Software-controlled Mode Operation

Hardware-controlled Modes

In hardware-controlled modes a hardware request signal starts a DMA transaction or a single DMA transfer. There are two hardware-controlled modes available:

- **Single Mode:**
Hardware requests are disabled by hardware after a DMA transaction
- **Continuous Mode:**
Hardware requests are **not** disabled by hardware after a DMA transaction

Hardware-controlled Single Mode

In hardware-controlled Single Modes one hardware request starts one complete DMA transaction or one single DMA transfer. The hardware-controlled Single Mode that initiates one complete DMA transaction to be executed for DMA channel mn is selected by the following operations:

- CHCRmn.CHMODE = 0
- CHCRmn.RROAT = 1
- Selecting one of the eight hardware request inputs via CHCRmn.PRSEL
- HTREQ.ECHmn = 1

Setting HTREQ.ECHmn to 1 causes that the hardware request CHmn_REQ of channel mn to be enabled (TRSR.HTREmn = 1). Once the hardware request CHmn_REQ becomes active, the value of CHCRmn.TREL is loaded into CHSRmn.TCOUNT and the DMA transaction is started by executing its first DMA transfer. After each DMA transfer, TCOUNT becomes decremented and next source and destination addresses are calculated. When TCOUNT reaches the 0, DMA channel mn becomes disabled and status flags TRSR.CHmn and TRSR.HTREmn are cleared. In order to start a new hardware-controlled DMA transaction, hardware requests must be enabled again by setting TRSR.HTREmn through HTREQ.ECHmn = 1. The hardware request disable function in Single Mode is typically needed when a reprogramming of the DMA channel register set (addresses, transfer count) is required before the next hardware triggered DMA transaction is started.

The hardware-controlled Single Mode in which each single DMA transfer has to be requested by a hardware request signal is selected as described above, with one difference:

- CHCRmn.RROAT = 0

In this operation mode, TRSR.CHmn becomes cleared after each DMA transfer of the DMA transaction, and a new hardware request at CHmn_REQ must be generated for starting the next DMA transfer.

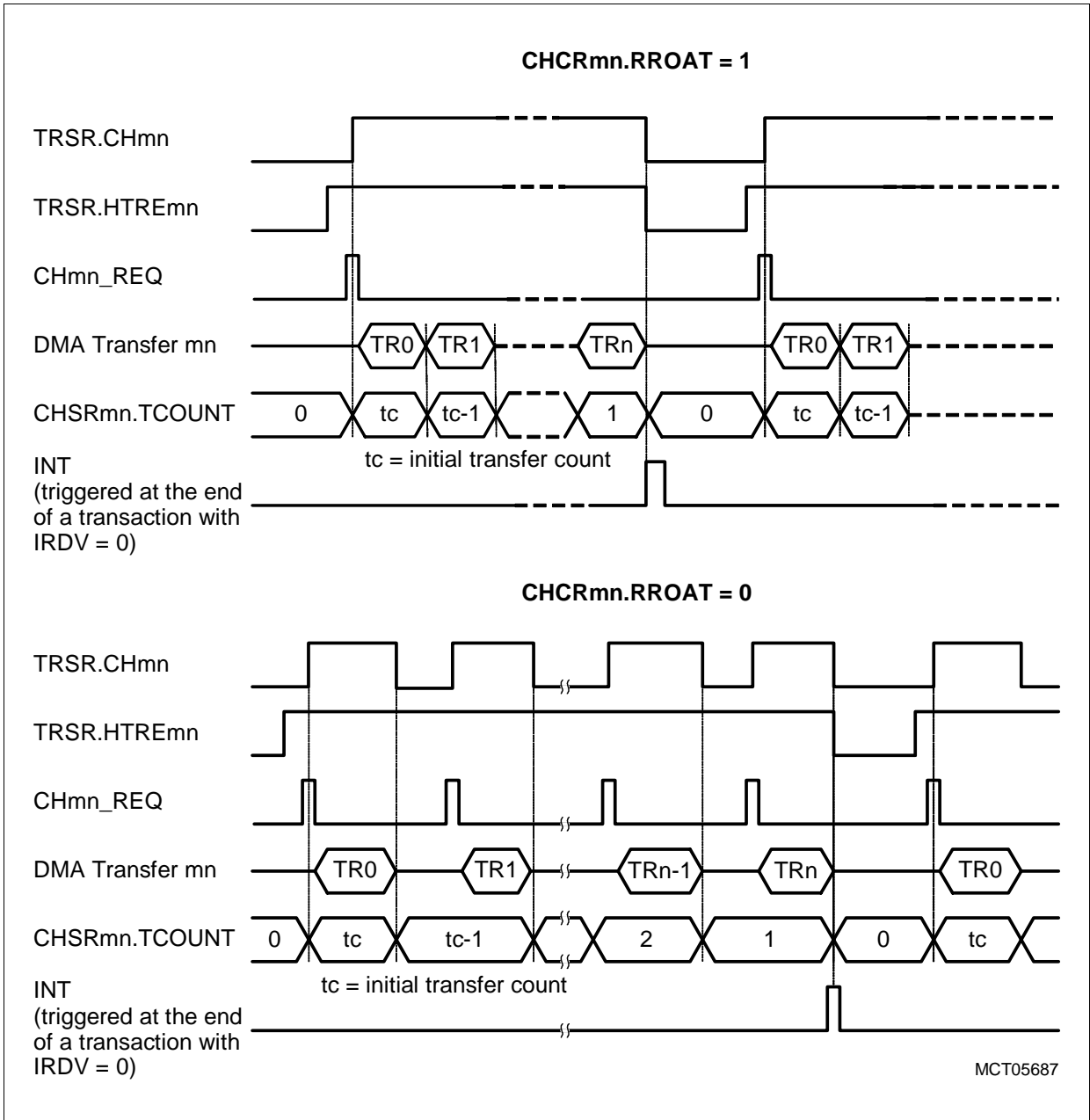


Figure 12-8 Hardware-controlled Single Mode Operation

Hardware-controlled Continuous Mode

In hardware-controlled Continuous Mode (CHCRmn.CHMODE = 1), the hardware transaction request enable bit HTREmn is not cleared at the end of a DMA transaction. A new transaction of DMA channel mn with the parameters actually stored in the channel register set of DMA channel mn is started each time when CHSRmn.TCOUNT reaches 000_H. No software re-enable for a hardware request at CHmn_REQ is required.

Direct Memory Access Controller

Combined Software/Hardware-controlled Mode

Figure 12-9 shows how software- and hardware-controlled modes can be combined. In the example, the first DMA transfer is triggered by software when setting STREQ.SCHmn. Hardware requests are still disabled. After hardware requests have been enabled by setting HTREQ.ECHmn, subsequent DMA transfers are triggered now by hardware request coming from the CHmn_REQ line.

In the example, DMA channel mn operates in Single Mode (CHCRmn.CHMODE = 0). In this mode, TRSR.HTREmn becomes cleared by hardware when CHSRmn.TCOUNT reaches 0 at the end of the DMA transaction.

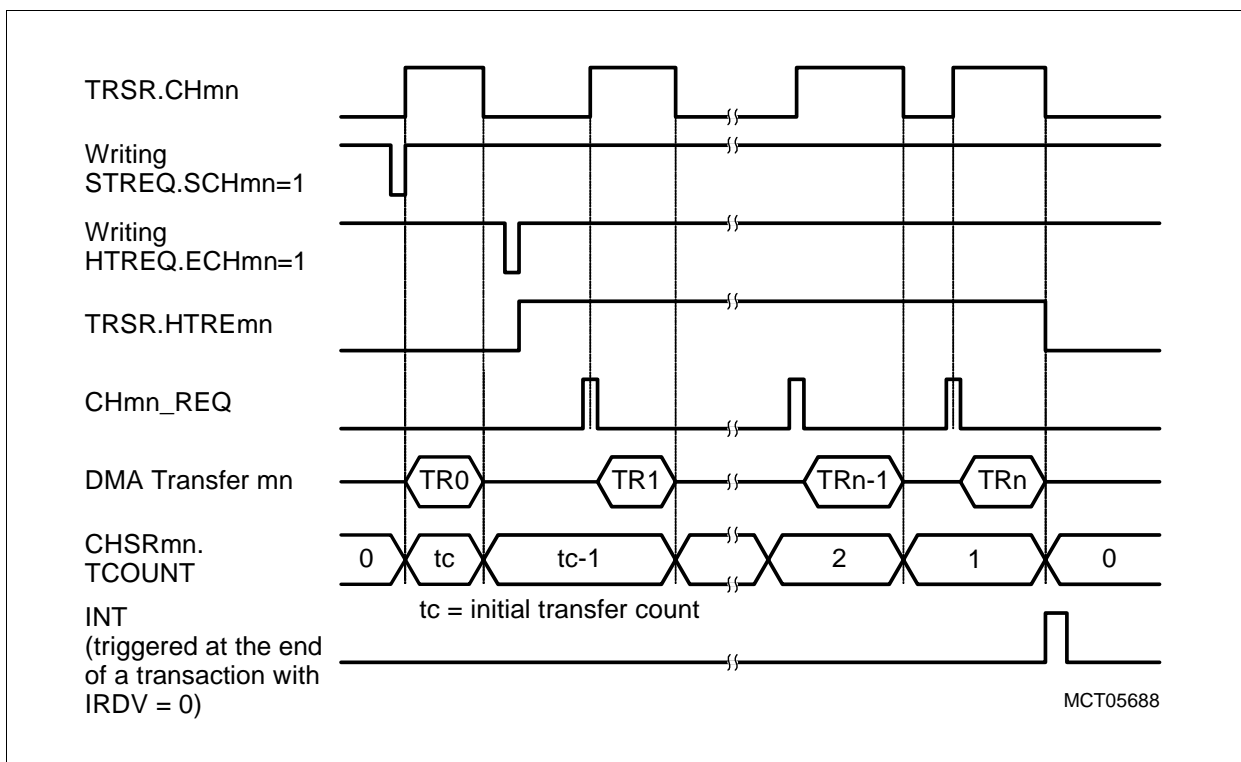


Figure 12-9 Transaction Start by Software, Continuation by Hardware

12.1.4.4 Error Conditions

The source move error flag ERRSR.FPI0SER indicates an FPI Bus error on bus 0 (SPB) that occurred during a source move (read) of a DMA transaction. The destination move error flag ERRSR.FPI0DER indicates an FPI Bus error on bus 0 (SPB) that occurred during a destination move (write) of a DMA transaction.

The transaction lost error flag ERRSR.TRLmn indicates if a DMA request for a DMA channel mn has been lost.

In the case of a read error, the write action is not executed, but the destination address is updated.

12.1.4.5 Channel Reset Operation

A DMA transaction of DMA channel mn can be stopped (channel is reset) by setting bit CHRSTR.CHmn. When a read or write FPI Bus access of DMA channel mn is executed at the time when CHRSTR.CHmn is set, this FPI Bus access is finished normally. This behavior guarantees data consistency.

When CHRST.CHmn is set to 1:

- Bits TRSR.HTREmn, TRSR.CHmn, ERRSR.TRLmn, INTSR.ICHmn, INTSR.IPMmn, WRPSR.WRPDmn, WRPSR.WRPSmn, CHSRmn.LXO, and bit field CHSRmn.TCOUNT are cleared.
- Source and destination address register will be set to the wrap boundary. SHADRmn will be cleared.
- All automatic functions are stopped for channel mn.

A user program must execute the following steps for resetting a DMA channel:

1. If hardware requests are enabled for the DMA channel mn, disable the DMA channel mn hardware requests by setting HTREQ.ECHmn = 0.
2. Writing a 1 to CHRST.CHmn.
3. Waiting (polling) until CHRST.CHmn = 0.

A user program should execute the following steps for restarting a DMA channel after it was reset:

1. Optionally (re-)configuring the address and other channel registers.
2. Restarting the DMA channel mn by setting HTREQ.ECHmn = 1 for hardware requests or STREQ.SCHmn = 1 for software requests.

The value of CHCRmn.TREL is copied to CHSRmn.TCOUNT when a new DMA transaction is requested and shadow address register contents is not equal 00000000_H.

12.1.4.6 Transfer Count and Move Count

The move count determines the number of moves (consisting of one read and one write each) to be done in each transfer. The move count allows the user to indicate to the DMA the number of moves to be done after one request. The number of moves per transfer is selected by the block mode settings (CHCRmn.BLKM).

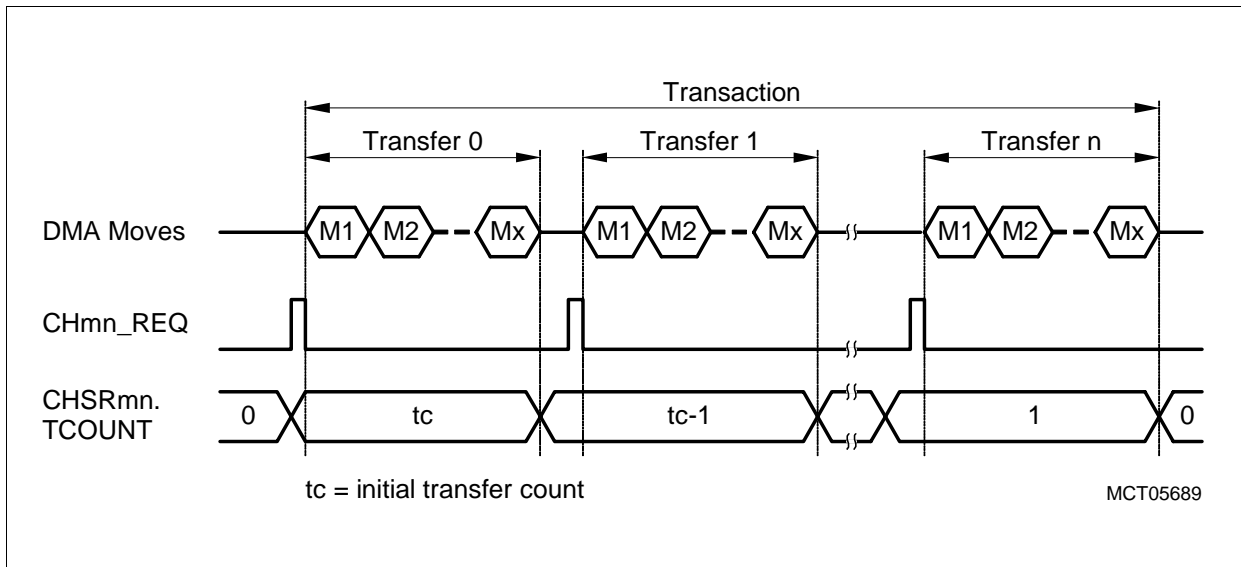


Figure 12-10 Transfer and Move Count

After a DMA move, the next source and destination addresses are calculated. Source and destination addresses are calculated independently from each other. The following address calculation parameters can be selected:

- The address offset, which is a multiple of the selected data width
- The offset direction: addition, subtraction, or none (unchanged address)

Control bits in address control register ADRCRmn determine how the addresses are incremented/decremented. Further, the data width as defined in CHCRmn.CHDW is taken into account for the address calculation.

Figure 12-11 and **Figure 12-12** show two examples of address calculation. In both examples, a data width of 16-bit (CHCRmn.CHDW = 01_B) is assumed.

Direct Memory Access Controller

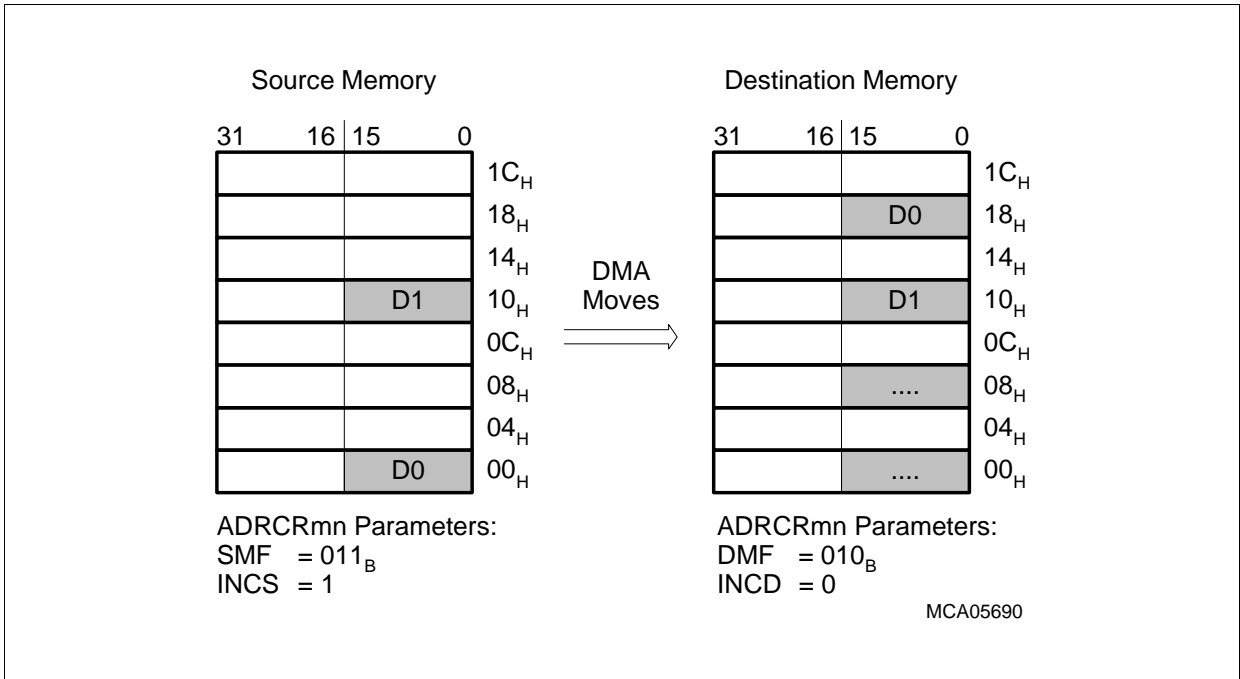


Figure 12-11 Programmable Address Modification - Example 1

In **Figure 12-11**, 16-bit half-words are transferred from a source memory with an incrementing source address offset of 10_H to a destination memory with decrementing destination addresses offset of 08_H.

In **Figure 12-12**, 16-bit half-words are transferred from a source memory with an incrementing source address offset of 02_H to a destination memory with incrementing destination addresses offset of 04_H.

Direct Memory Access Controller

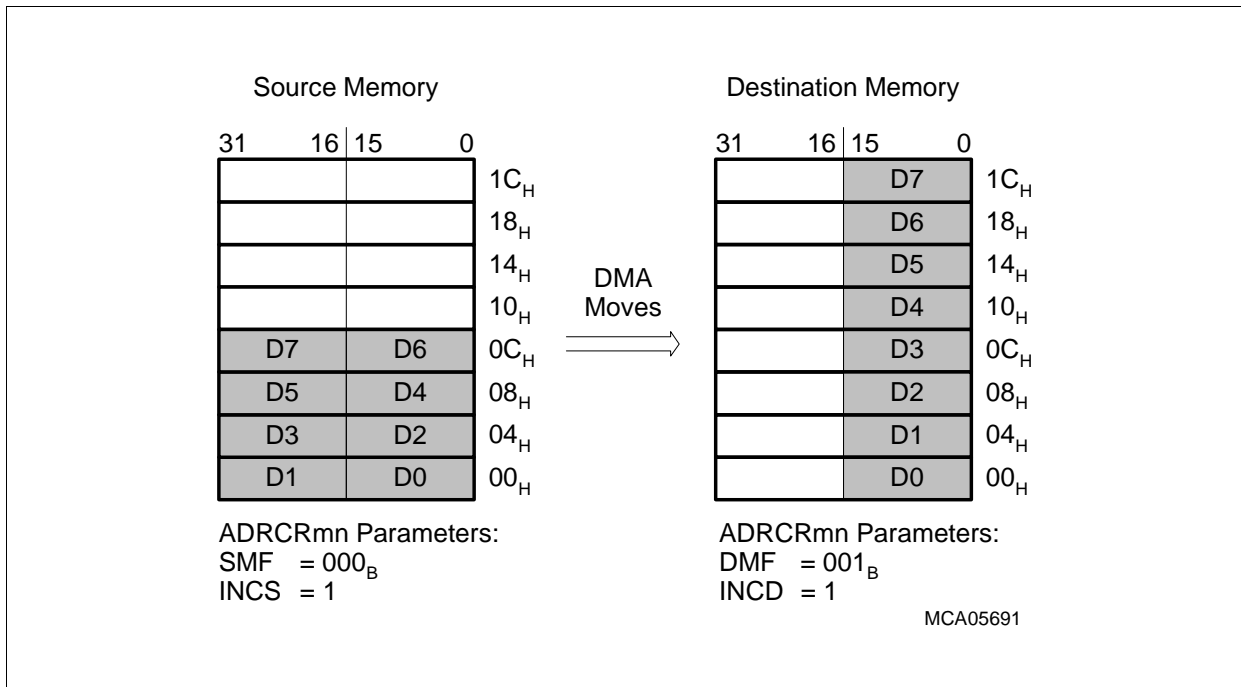


Figure 12-12 Programmable Address Modification - Example 2

12.1.4.7 Circular Buffer

Destination and source address can be configured to build a circular buffer separately for source and destination data. Within this circular buffer, addresses are updated as defined in [Figure 12-11](#) and [Figure 12-12](#) with a wrap-around at the buffer limits. The circular buffer length is determined by bit fields ADRCRmn.CBLS (for the source buffer) and ADRCRmn.CBLD (for the destination buffer). These 4-bit wide bit fields determine which bits of the 32-bit address remain unchanged at an address update. Possible buffer sizes of the circular buffers can be 2^{CBLS} or 2^{CBLD} bytes (= 1, 2, 4, 8, 16, ... up to 32k bytes).

When source or destination addresses are updated (incremented or decremented) after a DMA move, all upper bits [31:CBLS] of source address and [31:CBLD] of destination address are frozen and remain unchanged, even if a wrap-around from the lower address bits [CBLS:0] or [CBLD:0] occurred. This address-freezing mechanism always causes the circular buffers to be aligned to a multiple integer value of its size.

If the circular buffer size is less or equal than the selected address offset (see [Table 12-8](#)), the same circular buffer address will always be accessed.

12.1.5 Transaction Control Engine

The Transaction Control Unit in each DMA Sub-Block, as shown in the DMA Controller block diagram in [Figure 12-1](#), contains a Channel Arbiter and a Move Engine.

The Channel Arbiter arbitrates the transfer requests of the DMA channels, and submits the transfers parameters of the DMA channel with the highest channel priority that are needed for a DMA transfer to the Move Engine. DMA channels within a DMA Sub-Block have a two-level programmable channel priority as defined by bit CHCRmn.CHPRIO. When two transfer requests of two different DMA channels with identical channel priority become active at the same time, the DMA channel with the lowest channel number (n) is serviced first.

The Move Engine handles the execution of a DMA transfer that has been detected by the Channel Arbiter to be the next one. The Move Engine requests the required buses and loads or stores data according to the parameters of a DMA transfer. It is able to wait if a targeted bus is not available. In the Move Engine, a DMA transfer of a DMA transaction cannot be interrupted and always gets finished. This means that a DMA transfer, which can be also composed of several data moves (read move and write move), cannot be interrupted by a transfer of another DMA channel.

After a DMA transfer is finished, the Move Engine will send back the actualized address register information to the related DMA channel. Possible error conditions are also reported.

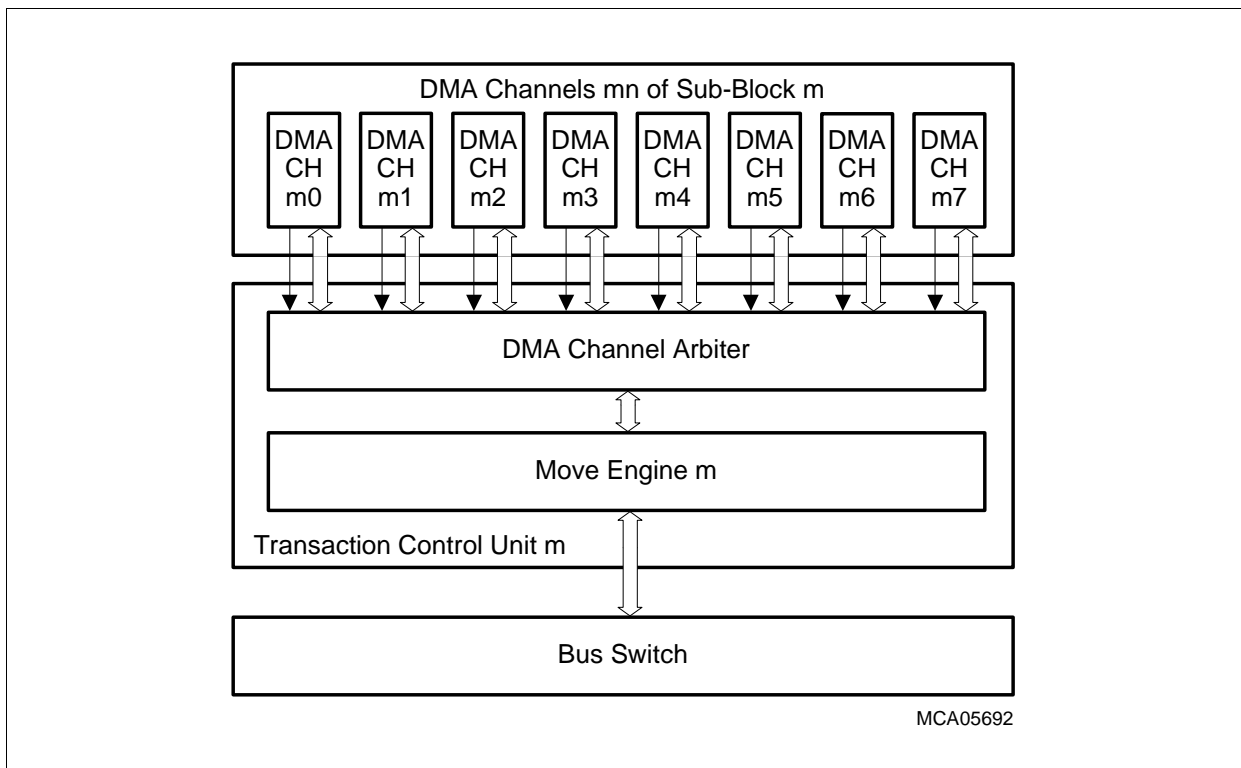


Figure 12-13 Transaction Control Engine

12.1.6 Bus Switch

The Bus Switch of the DMA controller provides the connection from the two DMA Sub-Blocks to the two FPI Bus interfaces (connected to System Peripheral Bus and Remote Peripheral Bus) and the MLI bus interface (see [Figure 12-14](#)).

A very simple bus bridge is implemented in the Bus Switch. It behaves as a feed-through for the slave interface on the FPI Bus interface 0 to the other bus interfaces (FPI Bus interface 1, MLI interface) because the working frequencies are identical on all these buses. The bridge is the only slave interface in the DMA because all the other interfaces are master on their buses. The slave interface also provides the access to the DMA and MLI registers.

The bridge functionality is covered independently from the DMA Move Engines. An access requesting from the FPI Bus interface 0 (SPB) to locations on the FPI Bus interface 1 (RPB) has priority over DMA actions. The DMA Move Engines are not involved in bridge transfers.

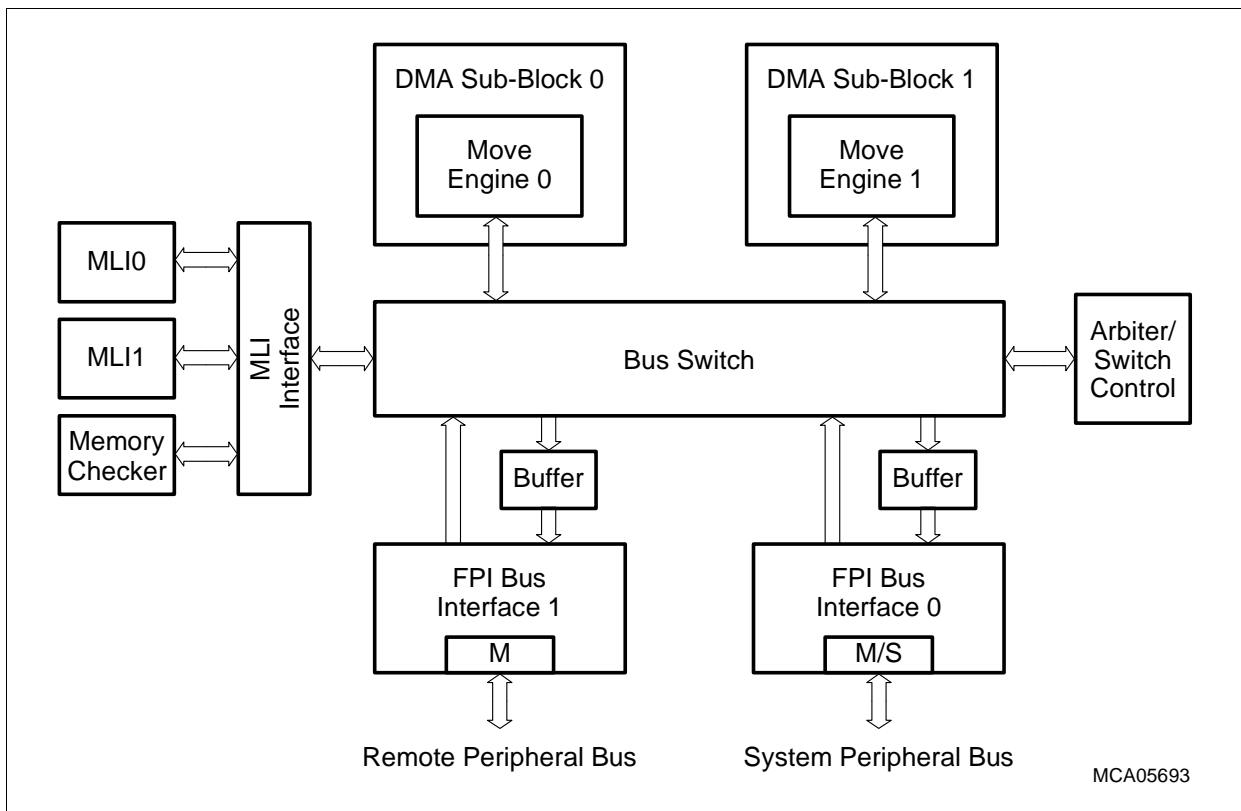


Figure 12-14 Bus Switch

One access can be buffered in the bus interfaces.

Note: The accesses of the DMA Move Engine's bus interfaces to FPI Bus interface 0 and FPI Bus interface 1 are always done in supervisor mode.

Direct Memory Access Controller

The arbiter/switch control unit arbitrates the bus requests for the switch and grants the buses connected to the switch for data transfers. [Table 12-1](#) defines the Bus Switch priorities.

Table 12-1 Bus Switch Priorities

Priority	Agent Requests	Comment
Highest	FPI Bus 0 to FPI Bus 1 FPI Bus 0 to MLI interface	The FPI Bus bridge functionality has higher priority than the DMA bus requests. Reason: minimizing wait states on the FPI Bus 0.
	DMA Move Engine write move	Priority is defined by CHCRmn.DMAPRIO of DMA channel mn that is requesting the switch.
	DMA Move Engine read move	
Lowest	MLI	–

CHCRmn.DMAPRIO determines the priority that is used when a move operation related to this channel is targeting the FPI Bus 0 (SPB). In the TC1796, the DMA controller has two priorities on FPI Bus 0 (DMA0 and DMA1), where it competes against the other bus masters in the system to access the bus. The DMAPRIO bit field also determines which priority is used by a DMA Move Engine to arbitrate the FPI Bus 0 access. DMIPRIO has no effect in the channel prioritization but is also used for Bus Switch arbitration of the two Move Engines.

Access modes (U, SV) of an access from FPI Bus 0 to FPI Bus 1 are identically transferred to FPI Bus 1. All accesses triggered by the DMA Move Engines or the MLI modules are always done in SV mode.

The DMA bridge functionality from the FPI Bus 0 to FPI Bus 1, to the MLI modules or to the memory checker does not support read/modify/write instructions.

DMA Bus Bandwidth Allocation

The Move Engines in the two DMA Sub-Blocks use a programmable priority scheme in the switch. The transfers currently requesting the master interface(s) of the DMA are internally arbitrated according to their programmable DMAPRIO value. If both Move Engines are requesting the bus interface simultaneously with identical DMAPRIO value, Move Engine 0 will get the bus first. This arbitration is done for each move operation. This arbitration scheme is valid if two concurrent write or two concurrent read requests compete for the same bus interface. A write always wins the arbitration against a read request.

12.1.7 On-Chip Debug Capabilities

The DMA controller in the TC1796 provides some debugging capabilities. These debug features support:

- Hard-suspend Mode of the DMA controller (for test purposes only)
- Soft-suspend Mode of DMA channels
- Break signal generation
- Trace signal generation

In suspend modes, the operations of DMA channels or the complete DMA module are stopped. Under certain condition conditions also a break signal is generated for the on-chip debug support logic. Further, DMA trace information can be output.

12.1.7.1 Hard-suspend Mode

In Hard-suspend Mode, the DMA controller module clock f_{DMA} is switched off and all further module actions are disabled. Module registers cannot be written anymore in Hard-suspend Mode, but combinatorial read operations of the module registers are still possible. Hard-suspend Mode of the DMA controller can be entered either by an active module suspend request signal (generated by hardware) or by writing the clock control register DMA_CLC register with an appropriate value.

When switching the DMA controller module clock f_{DMA} off, the communication of the DMA controller via its Bus Switch is blocked. Hard-suspend Mode can only be left by a reset operation.

Attention: The Hard-suspend Mode is mainly applicable for test purposes only. It can only be exit by a reset operation, and should not be used during normal operation of the DMA controller.

12.1.7.2 Soft-suspend Mode

The TC1796 on-chip debug control unit is able to generate a Soft-suspend Mode request (SUSREQ) for the DMA controller. When this soft suspend request becomes active, the state of a DMA channel becomes frozen, DMA requests are no longer forwarded, and the state of the DMA channel can be analyzed by reading the register contents.

Soft-suspend Mode of DMA channel mn is entered if its suspend enable bit SUSENmn in the Suspend Mode Register SUSPMR is set. When SUSREQ becomes active, the operation of all DMA channels mn that are enabled for Soft-suspend Mode is stopped automatically after its current DMA transfers have been finished in the transaction control unit. Afterwards, the suspend active status flag SUSPMR.SUSACmn is set, indicating that DMA channel mn is in Soft-suspend Mode. DMA channels that are disabled for Suspend Mode (SUSENmn = 0) continue with their normal operation.

In Soft-suspend Mode, register contents can be modified. These modifications are taken into account for further DMA transactions or DMA transfers of the related DMA channel

Direct Memory Access Controller

after Suspend Mode has been left again. Suspend Mode of DMA channel mn is left and its normal operation continues if either the SUSREQ signal becomes inactive, or if the enable bit SUSENmn is cleared by software.

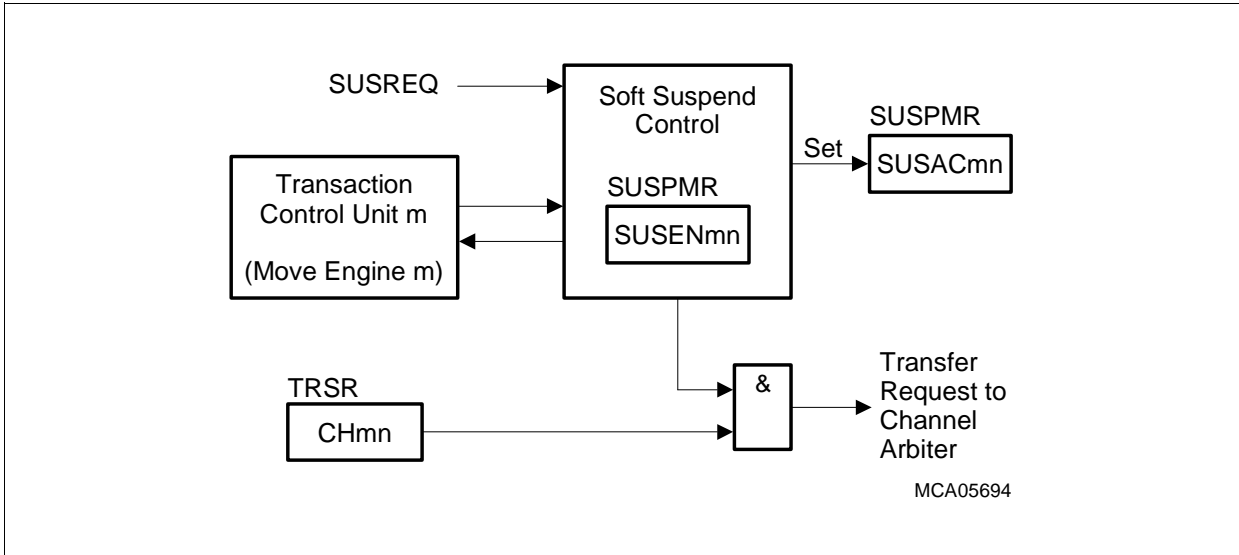


Figure 12-15 Soft-suspend Mode Control

12.1.7.3 Break Signal Generation

The DMA controller provides one BREAK output signal that is generated for the on-chip debug support logic (see [Figure 12-16](#)). Each of the DMA Sub-Blocks is able to detect two break conditions:

- Transaction lost interrupt has occurred
- DMA request transitions, indicated by bits TRSR.CHmn

The output lines of the two break conditions in each DMA Sub-Block are OR-ed together to the BREAK output signal.

A transaction lost break condition occurs in DMA Sub-Block m whenever at least one of its eight transaction lost interrupts becomes active, and when enable bit OCDSR.BRLm is set. The transaction lost interrupts do not generate a break condition if OCDSR.BRLm = 0. Transaction interrupt control is described in [Section 12.1.8.2](#).

The second break condition of DMA Sub-Block m becomes active when the transaction request bit TRSR.CHmn of one of its eight DMA channels n (as selected by OCDSR.BCHSn) indicates a transition of its state. The CHmn transition type (set, cleared, or set and cleared) is selected by bit field OCDSR.BTCRn.

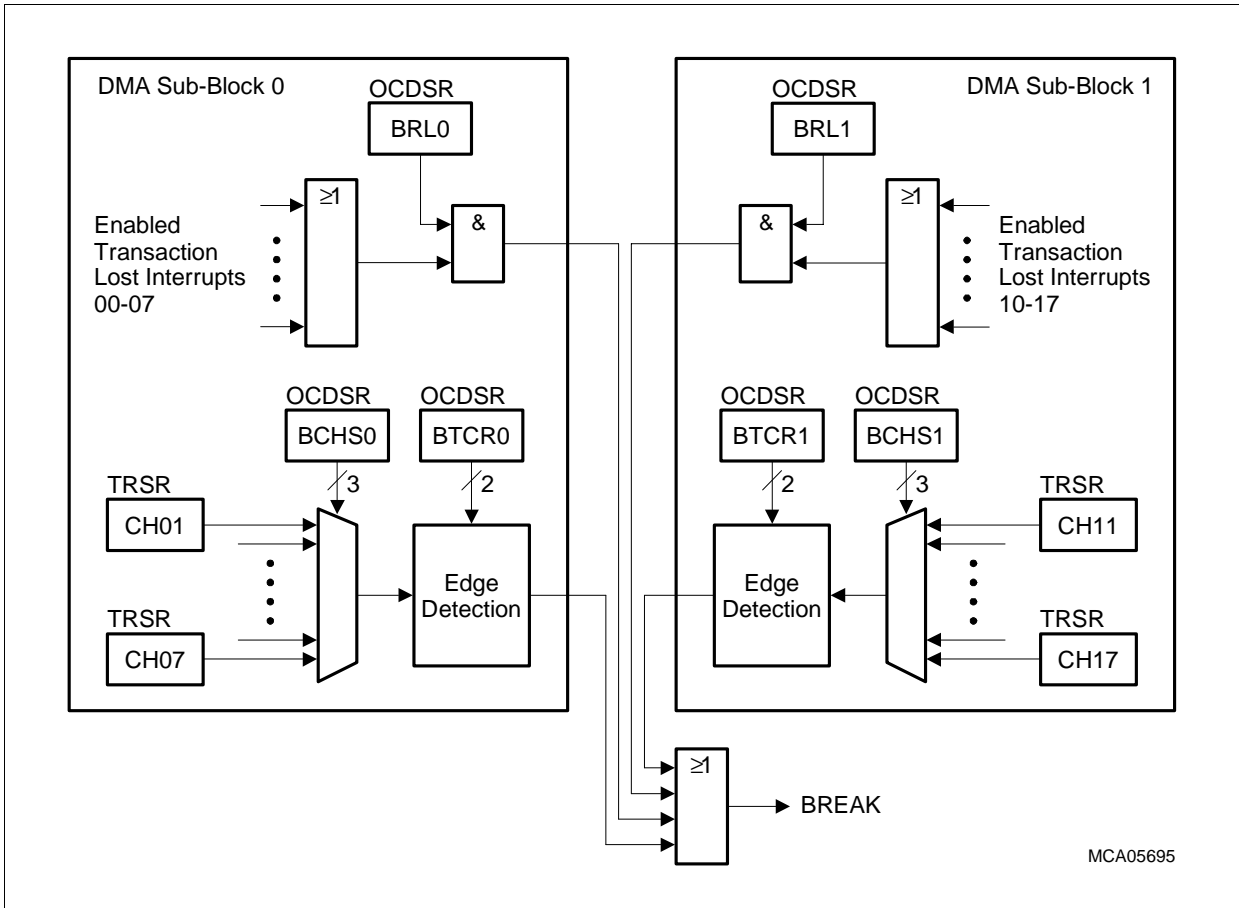


Figure 12-16 DMA Break Event Generation

12.1.7.4 Trace Signal Generation

The TC1796 provides sixteen OCDS Level 2 debug output lines OCDSL2[15:0]. These 16 output lines can be selected to output trace data of the DMA controller. Two trace data types are possible:

- Channel trace for monitoring the transaction request flags CHmn of register DMA_TRSR
- Move Engine trace for monitoring the status flags and bit fields of register DMA_MESR

Channel trace and Move Engine trace outputs are selected by bit TRCDS in the OSCU Configuration and Control Register OCNTRL. The selected DMA trace is further enabled to the trace output lines TR[15:0] by bit OCNTRL.TRCDEN.

The information that is output on the TR[15:0] trace port lines is shown in the following diagram.

Direct Memory Access Controller

TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
0								CH 07	CH 06	CH 05	CH 04	CH 03	CH 02	CH 01	CH 00

Figure 12-17 16-bit DMA Channel Transaction Request Trace

TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
RBT1		0						RBT0		ME0 WS	CH0			ME0 RS	

Figure 12-18 16-bit Move Engine Status Information Trace

12.1.8 Interrupts

The interrupt structure of the DMA controller is a very flexible control logic that allows an interrupt coming from an interrupt source within four interrupt source types to be connected to each of the sixteen interrupt outputs. This permits, for example, DMA channels that generate interrupts very rarely to share one interrupt node. The remaining interrupt nodes can be assigned to dedicated DMA channels to reduce the interrupt overhead for these channels. The four interrupt source types are:

- Channel interrupts
- Transaction lost interrupt
- Move Engine interrupts
- Wrap buffer interrupts

Some of the interrupt functions are common to all of the four interrupt source types. An interrupt event, internally generated as a request pulse, is always stored in an interrupt status flag. This interrupt status flag can be cleared by software. Further, the interrupt event can be enabled or disabled. When an interrupt event is enabled, a 4-bit Interrupt Node Pointer determines which of the sixteen interrupt outputs will be activated.

The following sections describe each of the four interrupt source types in more detail.

12.1.8.1 Channel Interrupts

Each DMA channel mn has one associated channel interrupt. It can always be activated after a DMA transfer, or when $CHSRmn.TCOUNT$ matches with the value of bit field $CHSRmn.IRDV$ after it has been decremented after a DMA transfer. The pattern detection interrupts that are combined with the channel interrupts (one common Interrupt Node Pointer $CHICRmn.INTP$), are activated when the pattern detection interrupt of DMA channel mn becomes active (when enabled by $CHCRmn.PATSEL$ not equal 00_B).

A channel interrupt of DMA channel mn is indicated when status flag $INTSR.ICHmn$ is set. The status flags $ICHmn$ and $IPMmn$ can be cleared together by software when setting bit $INTCR.CICHmn$ (or $CHRSTR.CHmn$). The channel interrupt of DMA channel mn is enabled when bit $CHICRmn.INTCT[1]$ is set. The channel interrupt pointer $CHICRmn.INTP$ determines which of the interrupt outputs $SR[15:0]$ ¹⁾ will be activated on an active channel interrupt or pattern detection interrupt. Note that the signal that is the set signal for the $ICHmn$ flag is available as $CHmn_OUT$ signal at the DMA module boundary.

Bit $CHICRmn.INTCT[0]$ selects two types of interrupt sources. For the compare operation, bit field $IRDV$ (4-bit) is zero-extended to 9-bit and then compared with the 9-bit $TCOUNT$ value. This means that a $TCOUNT$ match interrupt can be generated after one of the last 16 DMA transfers of a DMA transaction. Note that with $IRDV = 0000_B$, the

1) In the TC1796, only $SR[7:0]$ are connected to interrupt nodes.

Direct Memory Access Controller

match interrupt is generated at the end of a DMA transaction (after the last DMA transfer).

The pattern detection interrupt is indicated when status flag INTSR.IPMmn is set. The status flags IPMmn and ICHmn can be cleared together by software when setting bit INTCR.CICHmn (or CHRSTR.CHmn). The pattern detection interrupt of DMA channel mn is enabled when bit CHICRmn.PATSEL is set to a value not equal to 00_B. The channel interrupt pointer CHICRmn.INTP determines which of the interrupt outputs SR[15:0]¹⁾ will be activated on a pattern detection or channel interrupt.

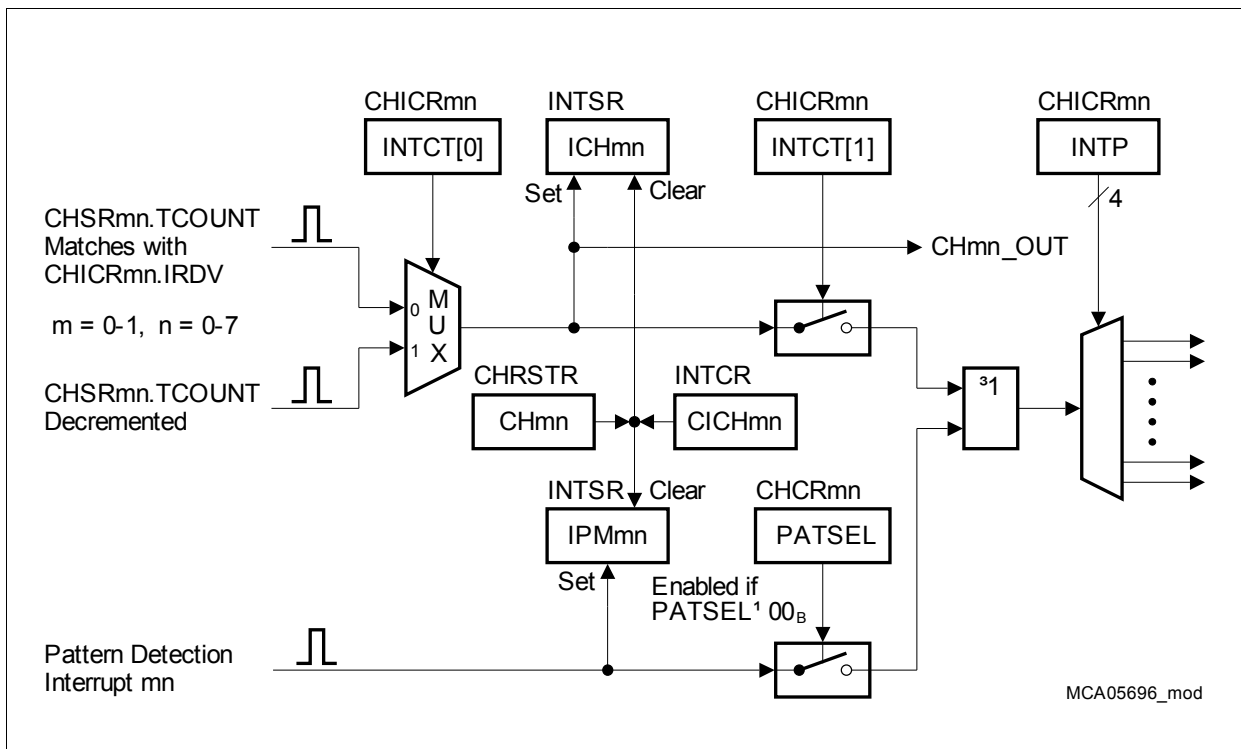


Figure 12-19 Channel Interrupts

1) In the TC1796, only SR[7:0] are connected to interrupt nodes.

12.1.8.2 Transaction Lost Interrupt

Each DMA channel mn is able to detect a transaction request lost condition. This condition becomes true when a new hardware or software DMA request occurs while the previous transaction or transfer on DMA channel mn is not finished, indicated by $TRSR.CHmn$ still set. If such a transaction request lost condition occurs, bit $ERRSR.TRLmn$ is set. The transaction lost interrupts of all DMA channels are OR-ed together to one common transaction lost interrupt that can be directed to one of the interrupt outputs $SR[15:0]$ ¹⁾ by setting the transaction lost interrupt pointer $EER.TRLINP$ with a corresponding value.

A transaction request lost condition of DMA channel mn is indicated by status flag $ERRSR.TRLmn$, which can be cleared by setting bit $CLRE.CTLmn$ or $CHRSTR.CHmn$. The transaction lost interrupt for DMA channel mn is enabled when bit $EER.ETRLmn$ is set.

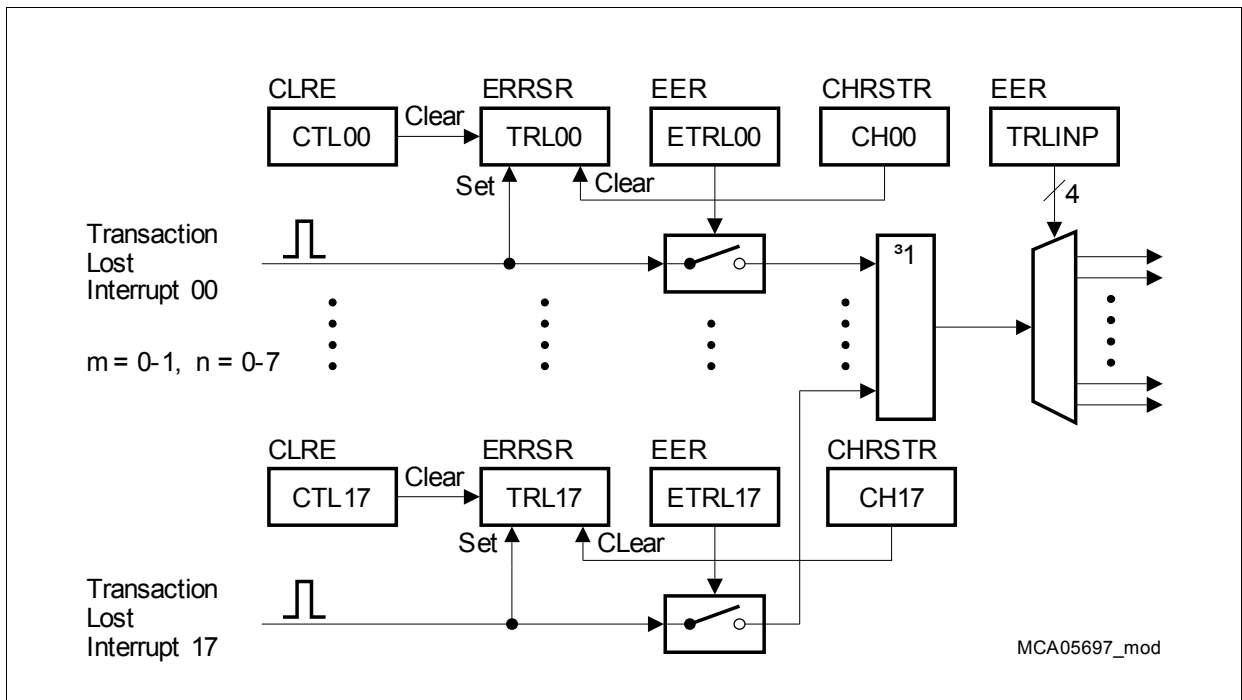


Figure 12-20 Transaction Lost Interrupt

1) In the TC1796, only $SR[7:0]$ are connected to interrupt nodes.

12.1.8.3 Move Engine Interrupts

Each of the two Move Engines is able to detect error conditions that occur during accesses to the two FPI Bus interfaces of the Bus Switch (see [Figure 12-14](#)). Two error conditions can be detected:

- Source error
- Destination error

A source error indicates an FPI Bus error that occurred during a read move from the data source. A destination error indicates an FPI Bus error that occurred during a write move to the data destination.

A source error of Move Engine m is indicated by the status flag `ERRSR.MEmSER`. Status flag `MEmSER` can be cleared by software when setting bit `CLRE.CMEmSER`. The source error interrupt of Move Engine m is enabled when bit `EER.MEmSER` is set. Separate clear, status, and enable bits are also available in each of the Move Engines for the destination error condition. Each of the two Move Engine interrupts can be directed to one of the interrupt outputs `SR[15:0]`¹⁾ by setting the Move Engine interrupt pointer `EER.MEmINP` with a corresponding value.

Note that in case of a read move error, the write move is not executed but the destination address is updated.

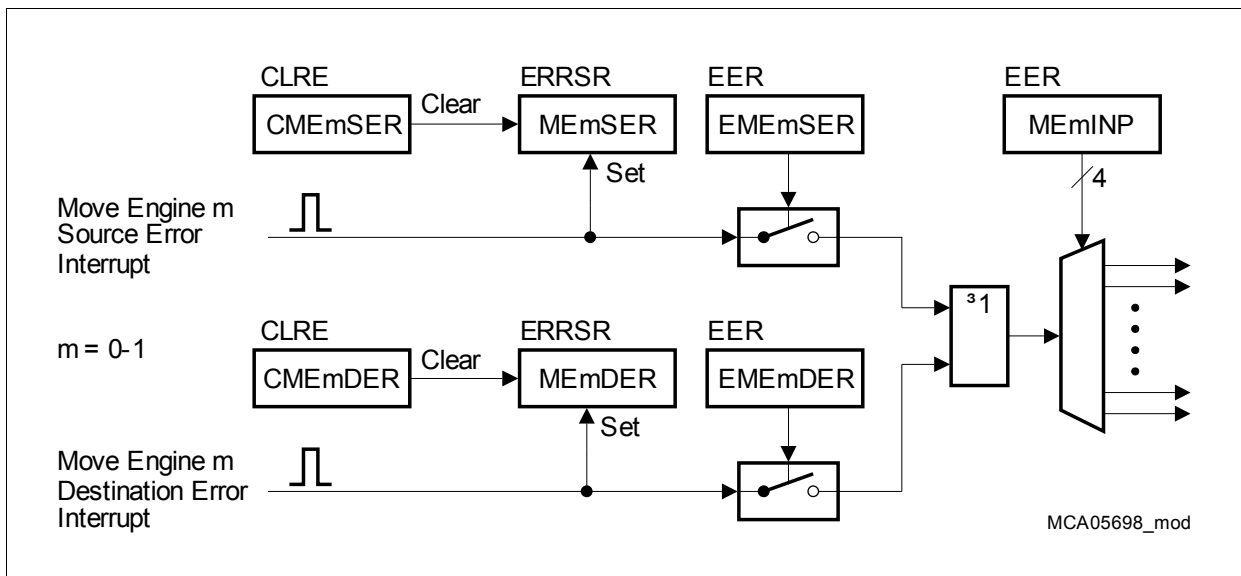


Figure 12-21 Move Engine Interrupts

1) In the TC1796, only `SR[7:0]` are connected to interrupt nodes.

Direct Memory Access Controller

When a Move Engine m source or destination error occurs, additional status bits and bit fields are provided in the error status register ERRSR to indicate the following two status conditions:

- At which FPI Bus interface a Move Engine m error occurred (FPI0ER and FPI1ER)
- For which DMA channel a Move Engine m read or write move error was reported (LECME0 and LECME1)

These error status bits and bit fields are required by error handler software to detect in detail at which FPI Bus interface and at which DMA channel the Move Engine error was generated. ERRSR.FPI0ER or ERRSR.FPI1ER are cleared when bits CLRE.CFPI0ER or CLRE.CFPI1ER, respectively, become set.

12.1.8.4 Wrap Buffer Interrupts

Each DMA channel mn is able to generate a wrap buffer interrupt for source buffer or destination buffer overflow. Further details on the pattern detection are described in [Section 12.1.9](#).

A wrap source buffer interrupt of DMA channel nm is indicated by status flag $WRPSR.WRPSmn$. A wrap destination buffer interrupt of DMA channel nm is indicated by the status flag $WRPSR.WRPDmn$. Both interrupt status flags can be cleared by software when bit $INTCR.CWRPmn$ (or $CHRSTR.CHmn$ becomes set. The wrap source buffer interrupt is enabled when bit $CHICRmn.WRPSE$ is set. The wrap destination buffer interrupt is enabled when bit $CHICRmn.WRPDE$ is set. The two interrupts for wrap source buffer and wrap destination buffer are OR-ed together to one common wrap buffer interrupt of DMA channel mn that can be directed to one of the interrupt outputs $SR[15:0]$ ¹⁾ by setting the wrap buffer interrupt pointer $CHICRmn.WRPP$ with a corresponding value. Note that the pattern match should not be enabled while a wrap buffer interrupt is enabled for the same channel.

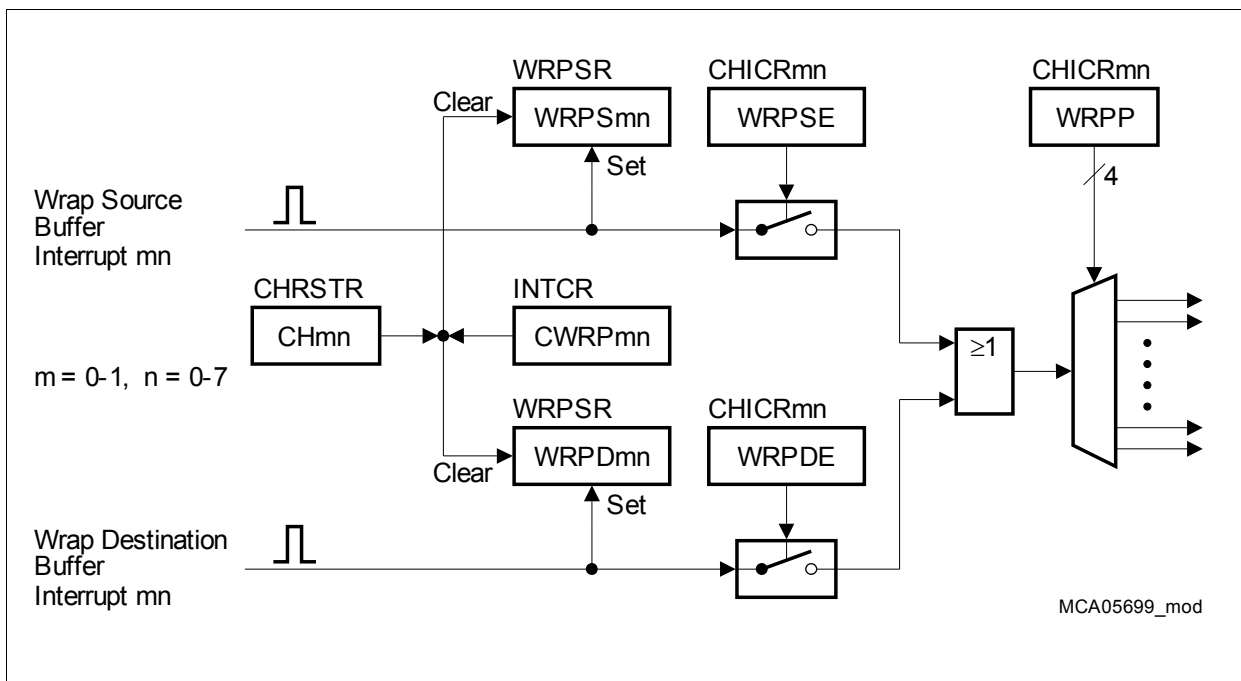


Figure 12-22 DMA Wrap Buffer Interrupts

1) In the TC1796, only $SR[7:0]$ are connected to interrupt nodes.

12.1.8.5 Interrupt Request Compressor

The interrupt control logic of the DMA controller uses an interrupt compressing scheme that allows high flexibility in interrupt processing. The request compressor logic shown in **Figure 12-23** condenses the 16 + 1 + 2 + 16 = 34 interrupt sources to the sixteen interrupt outputs. Each internal interrupt source can be directed to one of the sixteen interrupt outputs SR[15:0]¹⁾ by a 4-bit Interrupt Node Pointer. This allows also to connect more than one interrupt source to one interrupt output SR_x. Each interrupt output SR[15:0] can also be activated by writing a 1 to the corresponding bit GINTR.SIDMA_x.

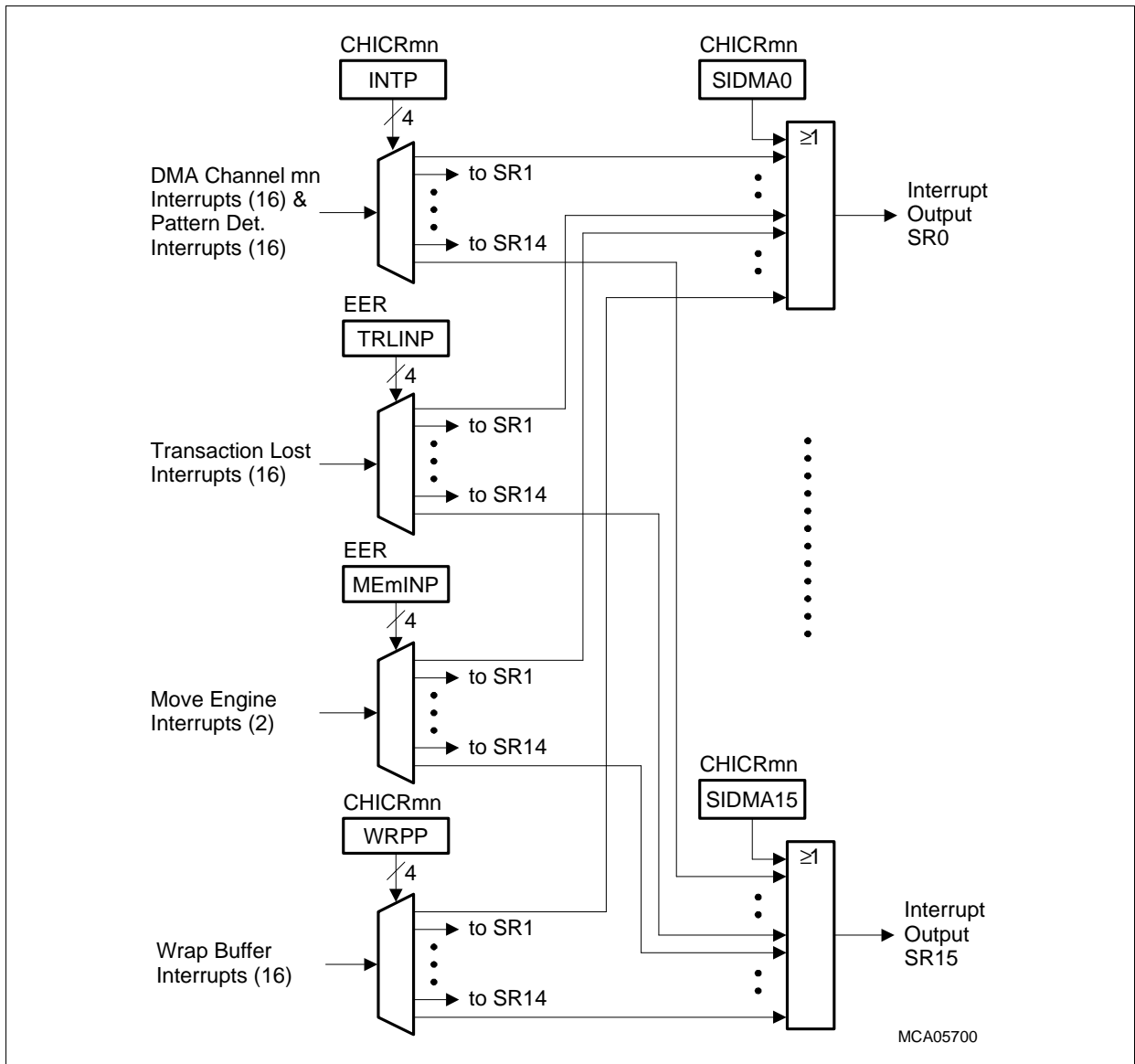


Figure 12-23 DMA Interrupt Request Compressor

1) In the TC1796, only SR[7:0] are connected to interrupt nodes.

12.1.9 Pattern Detection

Each Move Engine in a DMA Sub-Block provides a register MEmR that contains the data that was read during the last read move. Parts of this read move data can be compared after the read move to data that is stored in the Move Engine pattern register MEmPR of DMA Sub-Block m. The result of this pattern compare match is always stored in a bit (LXO) of the channel status register of the DMA channel mn that is currently executing the DMA move. Therefore, the pattern match result LXO of the previous read move can also be combined together with the pattern match result of the actual read move. MEmR is overwritten with each read move.

As the compare match patterns are stored in the Move Engine m (register MEmPR), its compare patterns are used for all DMA channels that are assigned to Move Engine m (all DMA channels of the DMA Sub-Block m).

The configuration and capabilities of the pattern detection logic further depends on the settings of CHCRmn.CHDW. CHDW determines the data width for the read and write moves individually for each DMA channel mn. Another control bit, CHCRmn.PATSEL, selects among the different operating modes for a specific value of CHDW.

Depending on CHCRmn.PATSEL and on the positive result of the comparison, two actions follow (if CHCRmn.PATSEL=00, no action will be taken when a pattern match is detected, so the wrap interrupt can be used):

- The activation of the interrupt corresponding to the current active channel mn using the Interrupt Pointer defined in CHICRmn.WRPP.
- Clear TRSR.HTREmn and TRSR.CHmn in order to stop the current transaction (Hardware and Software request enable). The value of CHSRmn.TCOUNT can be read out by the interrupt SW.

The software will have to service the interrupt and to activate again the channel.

12.1.9.1 Pattern Compare Logic

Read move data and compare match patterns are compared on a bit-wise level. The logic as shown in [Figure 12-24](#) is implemented in each COMP block of [Figure 12-25](#), [Figure 12-26](#), and [Figure 12-27](#). One COMP block controls either 8 bits or 16 bits of data and makes it possible to mask each data bit for the compare operation.

In the compare logic for one bit of the COMP block, a data bit from register MEmR is compared to the corresponding pattern bit stored in register MEmPR. If both bits are equal and a pattern mask bit stored in another part of register MEmPR is 0, the compare matched condition becomes active. When the pattern mask bit is set to 1, the compare matched condition is always active (set) for the related bit. When the compare matched conditions for each bit within a COMP block are true, the compare match output line of the COMP block becomes active.

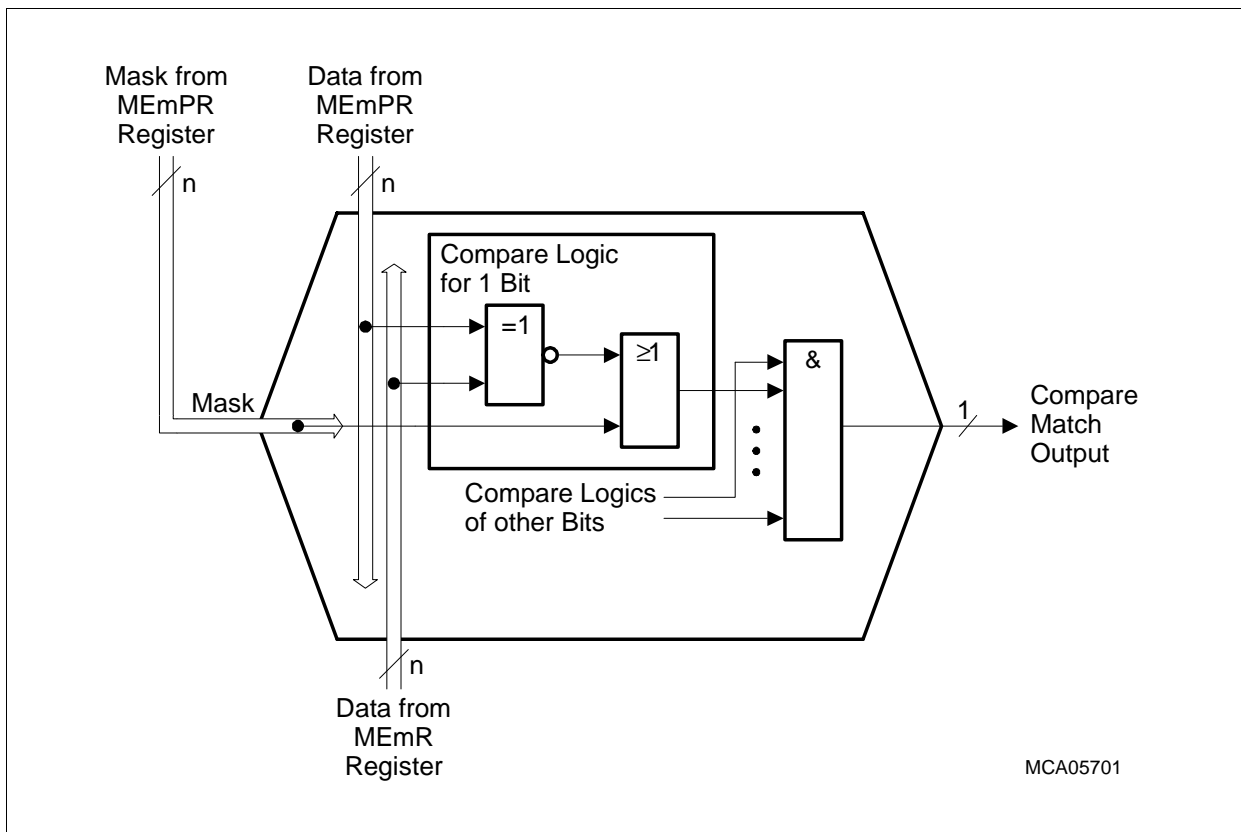


Figure 12-24 Pattern Compare Logic (COMP Block)

12.1.9.2 Pattern Detection for 8-bit Data Width

When 8-bit channel data width is selected (CHCRmn.CHDW = 00_B), the pattern detection logic is configured as shown in **Figure 12-25**. Three compare match configurations are possible.

Table 12-2 Pattern Detection for 8-bit Data Width

CHCRmn. PATSEL	Pattern Detection Operating Modes
00 _B	Pattern detection disabled
01 _B	Pattern compare RDm0 to PATm0, masked by PATm2
10 _B	Pattern compare RDm0 to PATm1, masked by PATm3
11 _B	Pattern compare RDm0 to PATm0, masked by PATm2 of the <u>actual</u> read move and Pattern compare RDm0 to PATm1, masked by PATm3 of the <u>previous</u> read move of DMA channel mn

When 8-bit channel data width is selected, the pattern detection logic makes it possible to compare the byte of one read move with two different patterns. Furthermore, after each read move the pattern match result “RDm0 with PATm1, masked by PATm3” is stored in bit CHCRmn.LXO. This operating mode allows, for example, to detect two-byte sequences in an 8-bit data stream coming from a serial peripheral unit with 8-bit data width (e.g.: recognition of carriage-return, line-feed characters). A mask operation of each compared bit is possible.

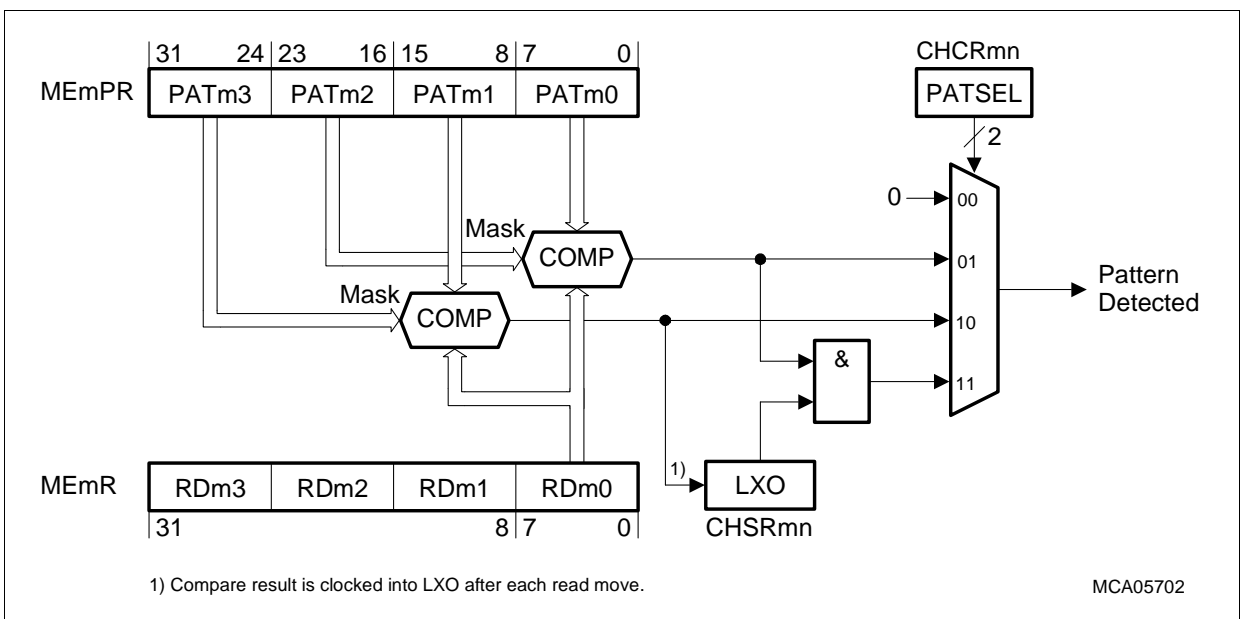


Figure 12-25 Pattern Detection for 8-bit Data Width (CHCRmn.CHDW = 00_B)

12.1.9.3 Pattern Detection for 16-bit Data Width

When 16-bit channel data width is selected (CHCRmn.CHDW = 01_B) the pattern detection logic can be configured as shown in [Figure 12-26](#). Three compare match configurations are possible.

Table 12-3 Pattern Detection for 16-bit Data Width

CHCRmn. PATSEL	ADRCRmn. INCS	Pattern Detection Operating Modes
00 _B	–	Pattern detection disabled
01 _B	–	Aligned Mode: Pattern compare RDm[1:0] to PATm[1:0], masked by PATm[3:2]
10 _B	0	Un-aligned Mode 1 (Source Address Decrement): Pattern compare RDm1 to PATm0, masked by PATm2 of the <u>actual</u> read move and Pattern compare RDm0 to PATm1, masked by PATm3 (LXO) of the <u>previous</u> read move of DMA channel mn
	1	Un-aligned Mode 2 (Source Address Increment): Pattern compare RDm0 to PATm1, masked by PATm3 of the <u>actual</u> read move and Pattern compare RDm1 to PATm0, masked by PATm2 (LXO) of the <u>previous</u> read move of DMA channel mn
11 _B	0 or 1	Combined Mode: Pattern compare for aligned mode (PATSEL = 01 _B) or un-aligned modes (PATSEL = 10 _B)

When 16-bit channel data width is selected, the pattern detection logic makes it possible to compare the complete half-word of one read move only (aligned mode) or to compare upper and lower byte of two consecutive read moves (un-aligned modes). Both modes can be combined, too (combined mode). A mask operation of each compared bit is possible.

In un-aligned mode 1 (source address decremented), the high byte (RDm1) of the current and the low byte (RDm0) of the previous 16-bit read move are compared.

In un-aligned mode 2 (source address incremented), the low byte (RDm0) of the current and the high byte (RDm1) of the previous 16-bit read move are compared.

If it is not known on which byte boundary (even or odd address) the 16-bit pattern to be detected is located, the combined mode should be used. This mode is the most flexible mode that combines the pattern search capability for aligned and un-aligned 16-bit data searches.

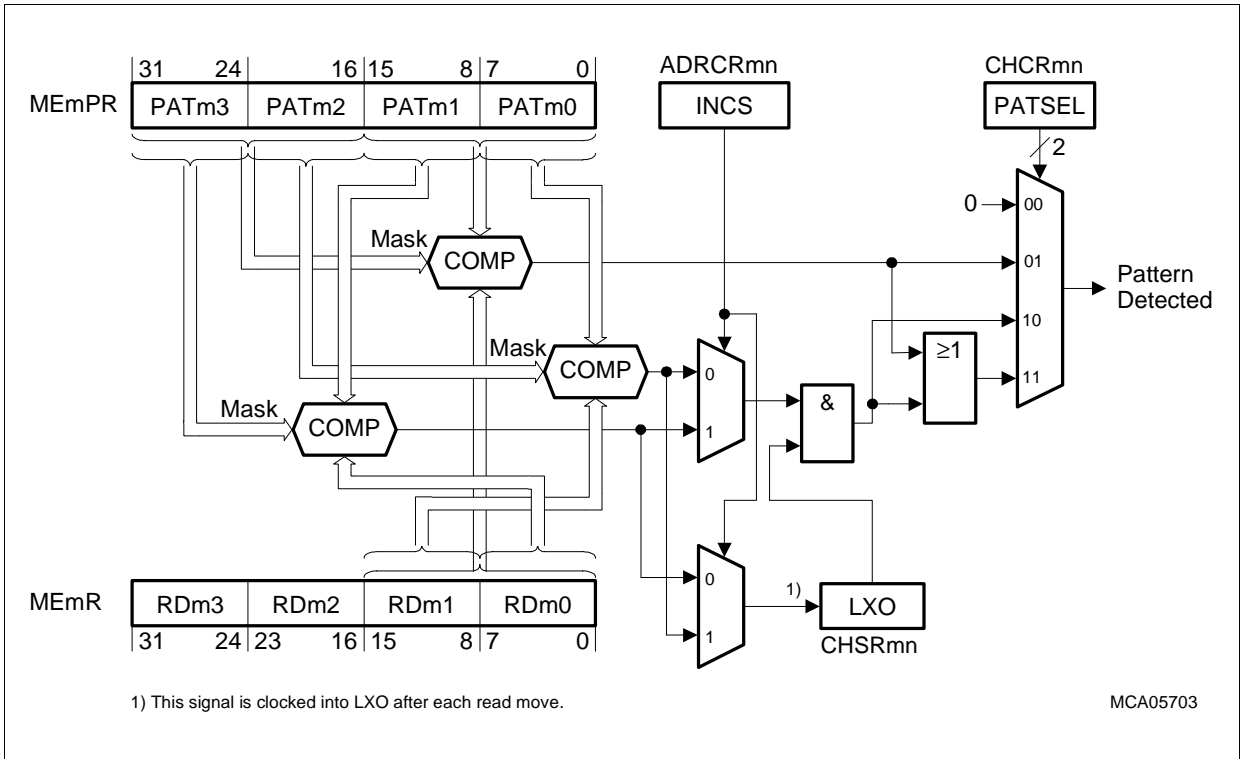


Figure 12-26 Pattern Detection for 16-bit Data Width (CHCRmn.CHDW = 01_B)

12.1.9.4 Pattern Detection for 32-bit Data Width

When 32-bit channel data width is selected (CHCRmn.CHDW = 10_B), the pattern detection logic is configured as shown in [Figure 12-27](#). Three compare match configurations are possible.

Table 12-4 Pattern Detection for 32-bit Data Width

CHCRmn. PATSEL	Pattern Detection Operating Modes
00 _B	Pattern detection disabled
01 _B	Unmasked pattern compare RDm[1:0] to PATm[1:0]
10 _B	Unmasked pattern compare RDm[3:2] to PATm[3:2]
11 _B	Unmasked pattern compare RDm[3:0] to PATm[3:0]

In 32-bit channel data width mode, the pattern detection logic makes it possible to compare the lower half-word only, the upper half-word only, or the complete 32-bit word with a pattern stored in the MEmPR register. A mask operation is not possible.

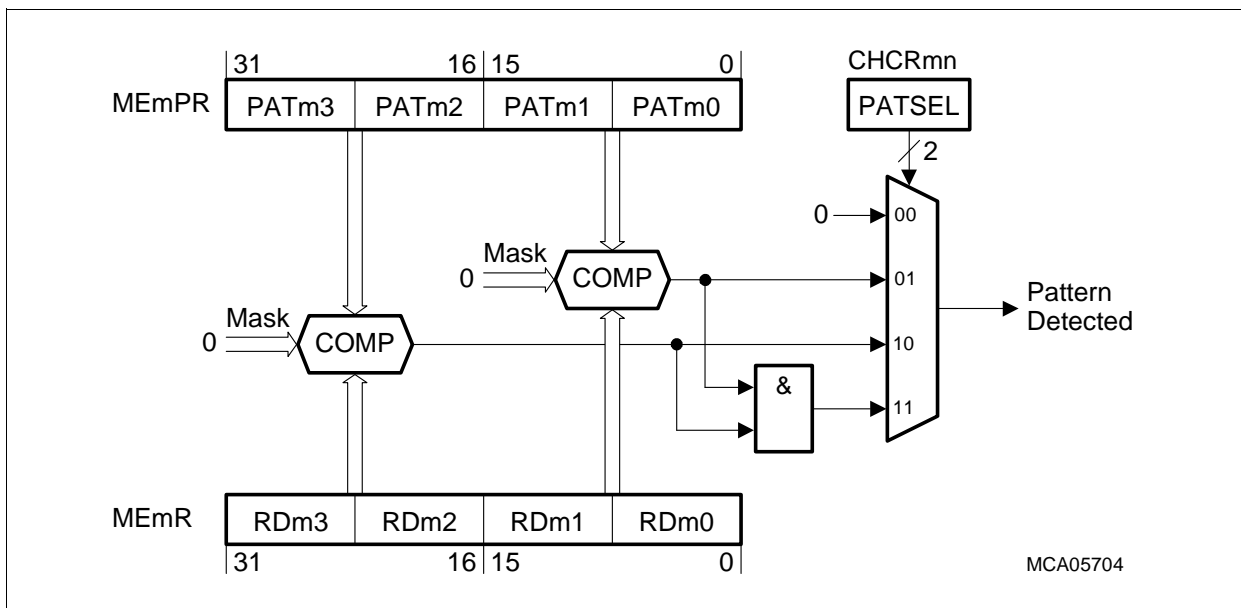


Figure 12-27 Pattern Detection for 32-bit Data Width (CHCRmn.CHDW = 10_B)

12.1.9.5 Access Protection

The DMA controller provides an access protection logic that makes it possible to disable read and write accesses of the Move Engines to specific parts of the memory map. Each address of a read move and a write move is always checked to determine if it is within an address range that is enabled for read/write access. If no address range is valid for an actual move address, a Move Engine interrupt can be generated.

The access protection logic handles two levels of address range definitions:

- Fixed address range
- Programmable address range extension

There are 32 fixed address ranges available that can be individually enabled/disabled in each Move Engine by the address range enable bits AEN_x (x = 0-31): These bits are located in the Move Engine m access enable register MEmAENR. If bit AEN_x is set, read/write accesses to the associated address range x are allowed. If bit AEN_x is cleared (default after reset), read/write accesses to the associated address range x are not executed and a Move Engine interrupt for source or destination move is generated (see also [Page 12-30](#)).

There are four programmable address range extensions available for each Move Engine. These programmable address range extensions make it possible to determine sub-ranges within four of the fixed address ranges. The parameters for the four sub-ranges are stored in the Move Engine m access range register MEmARR. The programmable address range extension is a feature that is applicable for memory access protection of memory blocks. In such an application, several memory sections are defined as sub-ranges of a complete memory block.

Figure 12-28 shows the two levels of address range definitions with the resulting address sub-ranges of the programmable address range extension. In a fixed address range, the width of fixed and variable address bits is constant. Number 'a' determines the lowest bit position of the fixed address, and is fixed individually and product-specific for each of the 32 fixed address ranges. With the programmable address range extension, the variable address part of the fixed address range definition (as defined by AEN_x) is reduced by the definition of a programmable number (up to 32) of sub-ranges. Bit field MEmARR.SIZE determines the sub-range size and bit field MEmARR.SLICE determines which of the sub-ranges is currently selected for access protection control. The two parameters (SIZE, SLICE) of the four address range extensions are numbered by index 'n' (n = 0-3).

Two sub-range examples (see [Figure 12-28](#)):

- $2^3 = 8$ sub-ranges are available with SIZE = 100_B. SLICE[2:0] selects one out of the eight sub-ranges. SLICE[4:3] is "don't care".
- $2^7 = 128$ sub-ranges are basically available with SIZE_n = 000_B. SLICE_n[4:0] selects one out of the lowest 32 sub-ranges. The upper $3 \times 32 = 96$ sub-ranges are not selectable (fixed address bits a-1 and a-2).

Direct Memory Access Controller

Note: The definition of the fixed address ranges x and the assignment of each sub-range to one of the fixed address ranges is product-specific. The definitions of the address ranges for the DMA controller as implemented in the TC1796 are defined on [Page 12-98](#).

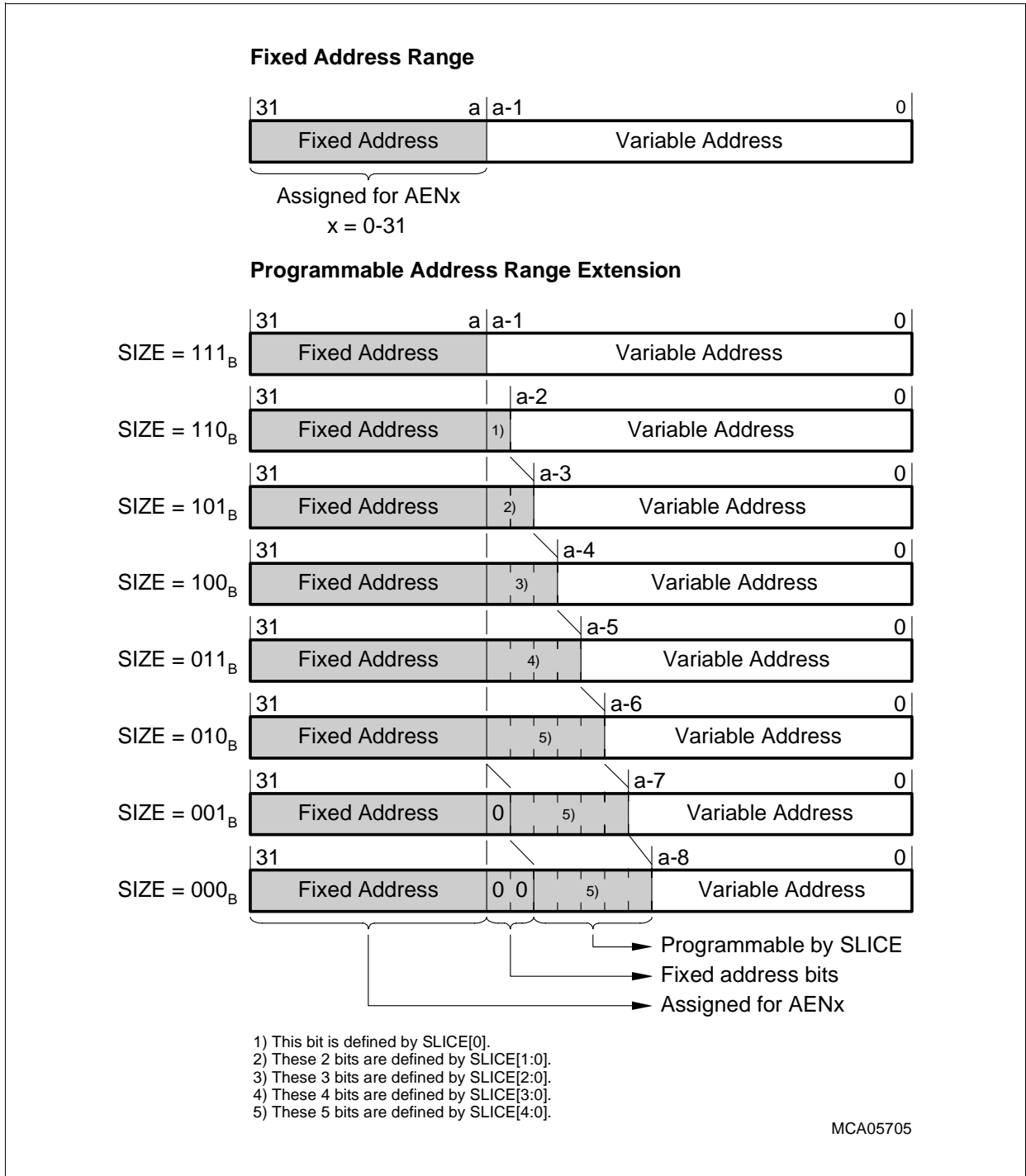


Figure 12-28 Access Protection Address Range Definitions

Direct Memory Access Controller

12.2 DMA Module Kernel Registers

Figure 12-29 and Table 12-6 show all registers associated with the DMA Controller Kernel.

DMA Registers Overview

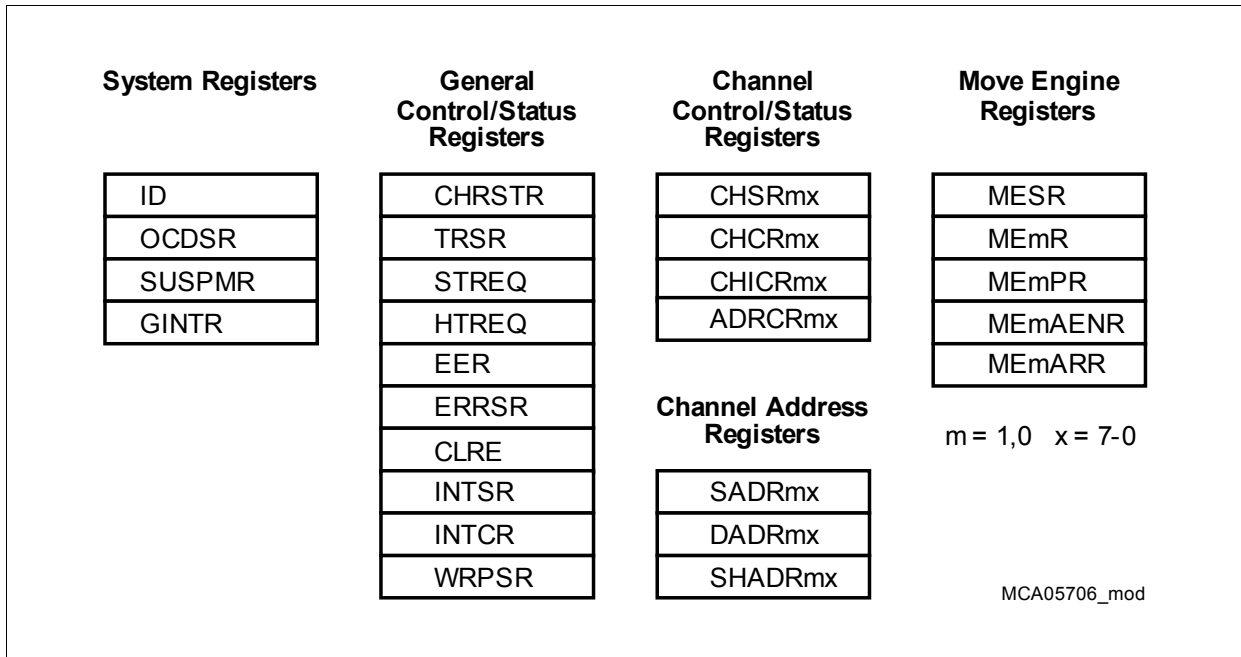


Figure 12-29 DMA Kernel Registers

The complete and detailed address map of the DMA module is described in Table 18-23 on Page 18-53 of the TC1796 User's Manual System Units part (Volume 1).

Note: For documentation automation purposes, the DMA channel numbering index "x" is used in the DMA module kernel register descriptions. This index is represented by "n" for DMA channel numbering in the functional description. For example, an index reference "mx" is equivalent to index "mn". Index "m" (m = 0-1) refers to the DMA Sub-Blocks and indexes "n" and "x" (n = x = 0-7) refer to the DMA channels.

Table 12-5 Registers Address Space - DMA Kernel Registers

Module	Base Address	End Address	Note
DMA	F000 3C00 _H	F000 3EFF _H	—

Direct Memory Access Controller

Table 12-6 Registers Overview - DMA Kernel Registers

Register Short Name	Register Long Name	Offset Address ¹⁾	Description see
DMA_ID	DMA Module Identification Register	008 _H	Page 12-45
DMA_CHRSTR	DMA Channel Reset Request Register	010 _H	Page 12-51
DMA_TRSR	DMA Transaction Request State Register	014 _H	Page 12-52
DMA_STREQ	DMA Software Transaction Request Register	018 _H	Page 12-54
DMA_HTREQ	DMA Hardware Transaction Request Register	01C _H	Page 12-55
DMA_EER	DMA Enable Error Register	020 _H	Page 12-56
DMA_ERRSR	DMA Error Status Register	024 _H	Page 12-59
DMA_CLRE	DMA Clear Error Register	028 _H	Page 12-62
DMA_GINTR	DMA Global Interrupt Set Register	02C _H	Page 12-50
DMA_MESR	DMA Move Engine Status Register	030 _H	Page 12-69
DMA_MEmR	Move Engine m Read Register (m = 0, 1)	34 _H + m × 4	Page 12-71
DMA_MEmPR	Move Engine m Pattern Register (m = 0,1)	3C _H + m × 4	Page 12-72
DMA_MEmAENR	Move Engine m Access Enable Register (m = 0, 1)	44 _H + m × 8	Page 12-73
DMA_MEmARR	Move Engine m Access Range Register (m = 0, 1)	48 _H + m × 8	Page 12-74
DMA_INTSR	DMA Interrupt Status Register	054 _H	Page 12-64
DMA_INTCR	DMA Interrupt Clear Register	058 _H	Page 12-67
DMA_WRPSR	DMA Wrap Status Register	05C _H	Page 12-66
DMA_OCDSR	DMA OCDS Register	064 _H	Page 12-46
DMA_SUSPMR	DMA Suspend Mode Register	068 _H	Page 12-48
DMA_CHSRmx	DMA Channel mx Status Register (m = 0-1, x = 0-7)	(m × 8 + x) × 20 _H + 80 _H	Page 12-80
DMA_CHCRmx	DMA Channel mx Control Register (m = 0-1, x = 0-7)	(m × 8 + x) × 20 _H + 84 _H	Page 12-76

Direct Memory Access Controller

Table 12-6 Registers Overview - DMA Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Address¹⁾	Description see
DMA_CHICRmx	DMA Channel mx Interrupt Control Register (m = 0-1, x = 0-7)	$(m \times 8 + x) \times 20_H + 88_H$	Page 12-81
DMA_ADRCRmx	DMA Channel mx Address Control Register (m = 0-1, x = 0-7)	$(m \times 8 + x) \times 20_H + 8C_H$	Page 12-83
DMA_SADRMx	DMA Channel mx Source Address Register (m = 0-1, x = 0-7)	$(m \times 8 + x) \times 20_H + 90_H$	Page 12-88
DMA_DADRMx	DMA Channel mx Destination Address Register (m = 0-1, x = 0-7)	$(m \times 8 + x) \times 20_H + 94_H$	Page 12-89
DMA_SHADRMx	DMA Channel mx Shadow Address Register (m = 0-1, x = 0-7)	$(m \times 8 + x) \times 20_H + 98_H$	Page 12-90

- 1) The absolute register address is calculated as follows:
 Module Base Address ([Table 12-5](#)) + Offset Address (shown in this column)
 Further, the following ranges for parameters m and x are valid: m = 0-1, x = 0-7

Note: Register bits marked with “w” are virtual registers and do not contain flip-flops. They are always read as 0.

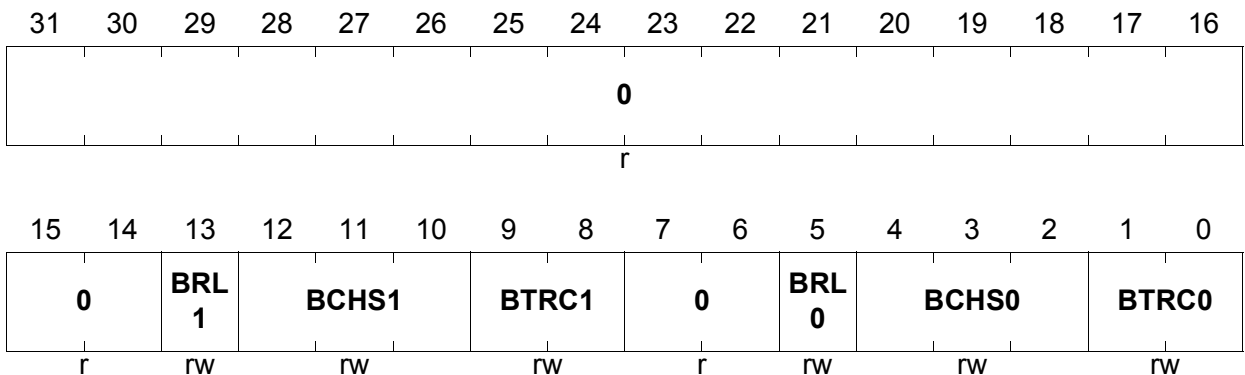
Note: The DMA registers can only be written in supervisor mode. Some of them are also Endinit-protected.

Direct Memory Access Controller

The OCDS Register describes the break capability of the DMA module. OCDSR is only reset with the OCDS Reset.

DMA_OCDSR

DMA OCDS Register (064_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
BTRC0	[1:0]	rw	<p>Break Trigger Condition In Sub-Block 0</p> <p>This bit field determines the transition type for the transaction request bit TRSR.CH0x that leads to a break condition in DMA Sub-Block 0.</p> <p>00_B No break condition is generated.</p> <p>01_B A break condition is generated when TRSR.CH0x changes from 0 to 1.</p> <p>10_B A break condition is generated when TRSR.CH0x changes from 1 to 0.</p> <p>11_B A break condition is generated when TRSR.CH0x changes its state.</p>
BCHS0	[4:2]	rw	<p>Break Channel Select In Sub-Block 0</p> <p>This bit field determines the DMA channel n of DMA Sub-Block 0 whose transaction request bit TRSR.CH0x is observed for signal transitions as defined by BTRC0.</p> <p>000_B DMA channel 00 selected</p> <p>001_B DMA channel 01 selected</p> <p>..._B ...</p> <p>110_B DMA channel 06 selected</p> <p>111_B DMA channel 07 selected</p>

Direct Memory Access Controller

Field	Bits	Type	Description
BRL0	5	rw	<p>Break On Request Lost in Sub-Block 0</p> <p>This bit field determines whether a BREAK signal is generated for DMA Sub-Block 0 when at least one of its eight transaction lost interrupts becomes active.</p> <p>0_B No break condition is generated. 1_B A break condition is generated for DMA Sub-Block 0 when at least one of its eight transaction lost interrupts becomes active.</p>
BTRC1	[9:8]	rw	<p>Break Trigger Condition In Sub-Block 1</p> <p>This bit field determines the transition type for the transaction request bit TRSR.CH1x that leads to a break condition in DMA Sub-Block 1.</p> <p>00_B No break condition is generated. 01_B A break condition is generated when TRSR.CH1x changes from 0 to 1. 10_B A break condition is generated when TRSR.CH1x changes from 1 to 0. 11_B A break condition is generated when TRSR.CH1x changes its state.</p>
BCHS1	[12:10]	rw	<p>Break Channel Select In Sub-Block 1</p> <p>This bit field determines the DMA channel n of DMA Sub-Block 1 whose transaction request bit TRSR.CH1x is observed for signal transitions as defined by BTRC1.</p> <p>000_B DMA channel 10 selected 001_B DMA channel 11 selected ..._B ... 110_B DMA channel 16 selected 111_B DMA channel 17 selected</p>
BRL0	5	rw	<p>Break On Request Lost in Sub-Block 1</p> <p>This bit field determines whether a BREAK signal is generated for DMA Sub-Block 1 when at least one of its eight transaction lost interrupts becomes active.</p> <p>0_B No break condition is generated. 1_B A break condition is generated for DMA Sub-Block 0 when at least one of its eight transaction lost interrupts becomes active.</p>
0	[7:6], [31:14]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Direct Memory Access Controller

The Suspend Mode Register contains bits for each DMA channel that make it possible to enable/disable its Soft-suspend Mode capability and that indicate its suspend status.

DMA_SUSPMR

DMA Suspend Mode Register

(068_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SUS AC 17	SUS AC 16	SUS AC 15	SUS AC 14	SUS AC 13	SUS AC 12	SUS AC 11	SUS AC 10	SUS AC 07	SUS AC 06	SUS AC 05	SUS AC 04	SUS AC 03	SUS AC 02	SUS AC 01	SUS AC 00
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SUS EN 17	SUS EN 16	SUS EN 15	SUS EN 14	SUS EN 13	SUS EN 12	SUS EN 11	SUS EN 10	SUS EN 07	SUS EN 06	SUS EN 05	SUS EN 04	SUS EN 03	SUS EN 02	SUS EN 01	SUS EN 00
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
SUSEN0x (x = 0-7)	x	rw	<p>Suspend Enable for DMA Channel 0x This bit enables the soft suspend capability individually for each DMA channel 0x.</p> <p>0_B DMA channel 0x is disabled for Soft-suspend Mode. The DMA channel 0x does not react on an active suspend request signal SUSREQ.</p> <p>1_B DMA channel 0x is enabled for Soft-suspend Mode. If the suspend request signal SUSREQ becomes active, a DMA transaction of DMA channel 0x is stopped after the current DMA transfer has been finished.</p> <p>Soft-suspend Mode can be terminated when SUSEN0x is written with 0.</p>

Direct Memory Access Controller

Field	Bits	Type	Description
SUSEN1x (x = 0-7)	8+x	rw	<p>Suspend Enable for DMA Channel 1x This bit enables the soft suspend capability individually for each DMA channel 1x.</p> <p>0_B DMA channel 1x is disabled for Soft-suspend Mode. The DMA channel 1x does not react on an active suspend request signal SUSREQ.</p> <p>1_B DMA channel 1x is enabled for Soft-suspend Mode. If the suspend request signal SUSREQ becomes active, a DMA transaction of DMA channel 1x is stopped after the current DMA transfer has been finished.</p> <p>Soft-suspend Mode can be terminated when SUSEN1x is written with 0.</p>
SUSAC0x (x = 0-7)	16+x	rh	<p>Suspend Active for DMA Channel 0x This status bit indicates whether DMA channel 0x is in Soft-suspend Mode or not.</p> <p>0_B DMA channel 0x is not in Soft-suspend Mode or internal actions are not yet finished after the Soft-suspend Mode was requested.</p> <p>1_B DMA channel 0x is in Soft-suspend Mode.</p>
SUSAC1x (x = 0-7)	24+x	rh	<p>Suspend Active for DMA Channel 1x This status bit indicates whether DMA channel 1x is in Soft-suspend Mode or not.</p> <p>0_B DMA channel 1x is not in Soft-suspend Mode or internal actions are not yet finished after the Soft-suspend Mode was requested.</p> <p>1_B DMA channel 1x is in Soft-suspend Mode.</p>

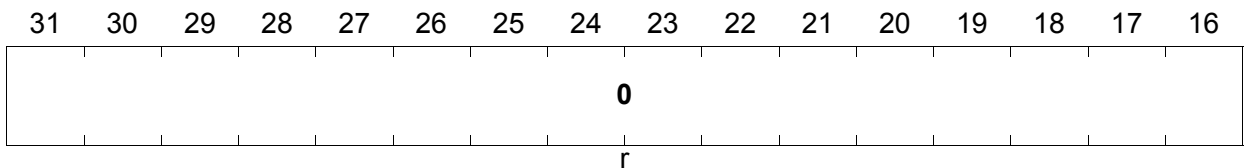
Note: This register is only reset by the OCDS reset.

Direct Memory Access Controller

The Global Interrupt Set Register makes it possible to activate the interrupt output lines of the DMA by software.

DMA_GINTR

DMA Global Interrupt Set Register (02C_H) **Reset Value: 0000 0000_H**



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SI	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI
DMA	DMA	DMA	DMA	DMA	DMA	DMA	DMA	DMA	DMA	DMA	DMA	DMA	DMA	DMA	DMA
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
SIDMA_x (x = 0-15)	x	w	Set DMA Interrupt Output Line x 0 _B No action 1 _B DMA interrupt output line SR _x will be activated. Reading this bit returns a 0.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

Note: In the TC1796, only interrupt output lines SR[7:0] are available. Therefore, only bits SIDMA[7:0] are effective.

Direct Memory Access Controller

12.2.2 General Control/Status Registers

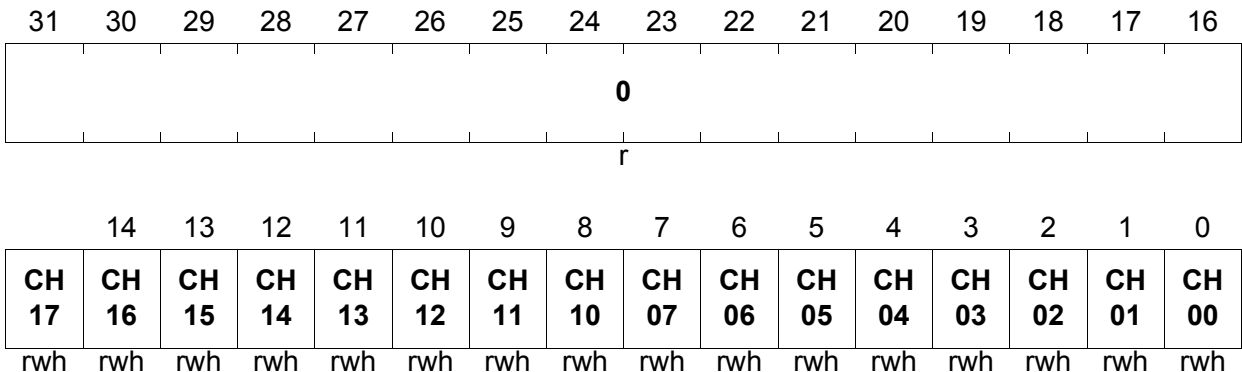
The bits in the Channel Reset Request Register are used to reset DMA channel mx.

DMA_CHRSTR

DMA Channel Reset Request Register

(010_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CH0x (x = 0-7)	x	rwh	Channel 0x Reset These bits force the DMA channel 0x to stop its current DMA transaction. Once set by software, this bit will be automatically cleared when the channel has been reset. Writing a 0 to CH0x has no effect. 0 _B No action (write) or the requested channel reset has been reset (read). 1 _B DMA channel 0x is stopped. More details see Page 12-16 .
CH1x (x = 0-7)	8+x	rwh	Channel 1x Reset These bits force the DMA channel 1x to stop its current DMA transaction. Once set by software, this bit will be automatically cleared when the channel has been reset. Writing a 0 to CH1x has no effect. 0 _B No action (write) or the requested channel reset has been reset (read). 1 _B DMA channel 1x is stopped. More details see Page 12-16 .
0	[31:16]	r	Reserved Read as 0; should be written with 0.

Direct Memory Access Controller

The bits in the Transaction Request State Register indicates which DMA channel is processing a request, and which DMA channel has hardware transaction requests enabled.

DMA_TRSR

DMA Transaction Request State Register

(014_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HT	HT	HT	HT	HT	HT	HT	HT	HT	HT	HT	HT	HT	HT	HT	HT
RE	RE	RE	RE	RE	RE	RE	RE	RE	RE	RE	RE	RE	RE	RE	RE
17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH	CH	CH	CH	CH	CH	CH	CH	CH	CH	CH	CH	CH	CH	CH	CH
17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
CH0x (x = 0-7)	x	rh	Transaction Request State of DMA Channel 0x 0 _B No DMA request is pending for channel 0x. 1 _B A DMA request is pending for channel 0x. CH0x is cleared when a pattern match is detected.
CH1x (x = 0-7)	8+x	rh	Transaction Request State of DMA Channel 1x 0 _B No DMA request is pending for channel 1x. 1 _B A DMA request is pending for channel 1x. CH1x is cleared when a pattern match is detected.

Direct Memory Access Controller

Field	Bits	Type	Description
HTRE0x (x = 0-7)	16+x	rh	<p>Hardware Transaction Request Enable State of DMA Channel 0x</p> <p>0_B Hardware transaction request for DMA Channel 0x is disabled. An input DMA request will not trigger the channel 0x.</p> <p>1_B Hardware transaction request for DMA Channel 0x is enabled. The transfers of a DMA transaction are controlled by the corresponding channel request line of the DMA requesting source.</p> <p>HTRE0x is set to 0 when CHSR0x.TCOUNT is decremented and CHSR0x.TCOUNT = 0. HTRE0x can be enabled and disabled with HTREQ.ECH0x or HTREQ.DCH0x.</p> <p>HTRE0x is cleared when a pattern match is detected.</p>
HTRE1x (x = 0-7)	24+x	rh	<p>Hardware Transaction Request Enable State of DMA Channel 1x</p> <p>0_B Hardware transaction request for DMA Channel 1x is disabled. An input DMA request will not trigger the channel 1x.</p> <p>1_B Hardware transaction request for DMA Channel 1x is enabled. The transfers of a DMA transaction are controlled by the corresponding channel request line of the DMA requesting source.</p> <p>HTRE1x is set to 0 when CHSR1x.TCOUNT is decremented and CHSR1x.TCOUNT = 0. HTRE1x can be enabled and disabled with HTREQ.ECH1x or HTREQ.DCH1x.</p> <p>HTRE1x is cleared when a pattern match is detected,</p>

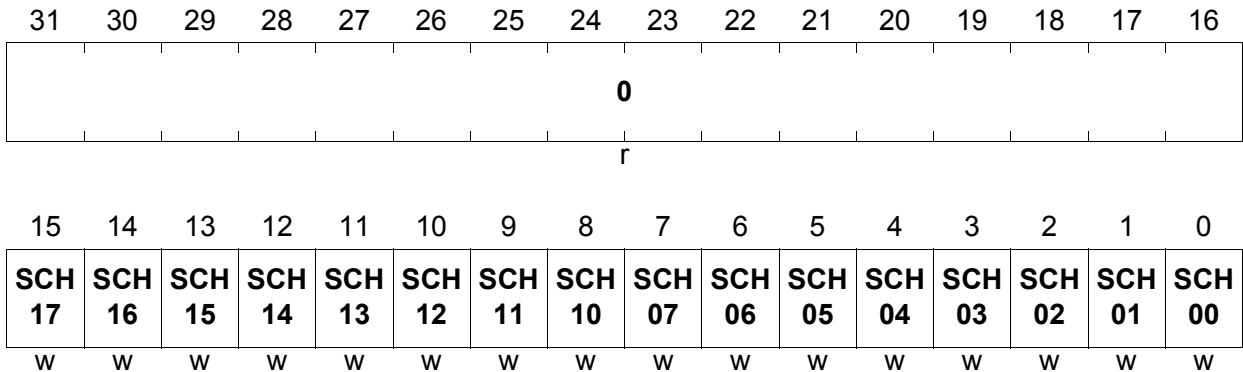
Direct Memory Access Controller

The bits in the Software Transaction Request Register are used to generate a DMA transaction request by software.

DMA_STREQ

**DMA Software Transaction Request Register
(018_H)**

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SCH0x (x = 0-7)	x	w	Set Transaction Request for DMA Channel 0x 0 _B No action. 1 _B A transaction for DMA channel 0x is requested. When setting SCH0x, TRSR.CH0x becomes set to indicate that a DMA request is pending for DMA channel 0x.
SCH1x (x = 0-7)	8+x	w	Set Transaction Request for DMA Channel 1x 0 _B No action. 1 _B A transaction for DMA channel 1x is requested. When setting SCH1x, TRSR.CH1x becomes set to indicate that a DMA request is pending for DMA channel 1x.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

Note: Register bits marked with “w” are virtual and are not stored in flip-flops. Reading STREQ returns 0 when read.

Direct Memory Access Controller

The bits in the Hardware Transaction Request Register enable or disable DMA hardware requests.

DMA_HTREQ

**DMA Hardware Transaction Request Register
(01C_H)**

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DCH	DCH	DCH	DCH	DCH	DCH	DCH	DCH	DCH	DCH	DCH	DCH	DCH	DCH	DCH	DCH
17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECH	ECH	ECH	ECH	ECH	ECH	ECH	ECH	ECH	ECH	ECH	ECH	ECH	ECH	ECH	ECH
17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Field	Bits	Type	Description
ECH0x (x = 0-7)	x	w	Enable Hardware Transfer Request for DMA Channel 0x
ECH1x (x = 0-7)	8+x	w	Enable Hardware Transfer Request for DMA Channel 1x
DCH0x (x = 0-7)	16+x	w	Disable Hardware Transfer Request for DMA Channel 0x
DCH1x (x = 0-7)	24+x	w	Disable Hardware Transfer Request for DMA Channel 1x

Table 12-7 Conditions to Set/Clear the Bits TRSR.HTREmx

HTREQ.ECHmx	HTREQ.DCHmx	Transaction Finishes ¹⁾ for Channel mx	Modification of TRSR.HTREmx
0 _B	0 _B	0 _B	Unchanged
1 _B	0 _B	0 _B	Set
X _B	1 _B	X _B	Clear
X _B	X _B	1 _B	Clear

1) In Single Mode only. In Continuous Mode, the end of a transaction has no impact.

Direct Memory Access Controller

The Enable Error Register describes how the DMA controller reacts to errors. It enables the interrupts for the loss of a transaction request or Move Engine errors.

DMA_EER

DMA Enable Error Register

(020_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TRLINP				ME1INP				ME0INP				E ME1 DER	E ME1 SER	E ME0 DER	E ME0 SER
rw				rw				rw				rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E TRL 17	E TRL 16	E TRL 15	E TRL 14	E TRL 13	E TRL 12	E TRL 11	E TRL 10	E TRL 07	E TRL 06	E TRL 05	E TRL 04	E TRL 03	E TRL 02	E TRL 01	E TRL 00
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ETRL0x (x = 0-7)	x	rw	Enable Transaction Request Lost for DMA Channel 0x This bit enables the generation of an interrupt when the set condition for ERRSR.TRL0x is detected. 0 _B The interrupt generation for a request lost event for channel 0x is disabled. 1 _B The interrupt generation for a request lost event for channel 0x is enabled.
ETRL1x (x = 0-7)	8+x	rw	Enable Transaction Request Lost for DMA Channel 1x This bit enables the generation of an interrupt when the set condition for ERRSR.TRL1x is detected. 0 _B The interrupt generation for a request lost event for channel 1x is disabled. 1 _B The interrupt generation for a request lost event for channel 1x is enabled.
EME0SER	16	rw	Enable Move Engine 0 Source Error This bit enables the generation of a Move Engine 0 source error interrupt. 0 _B Move Engine 0 source error interrupt is disabled. 1 _B Move Engine 0 source error interrupt is enabled.

Direct Memory Access Controller

Field	Bits	Type	Description
EME0DER	17	rw	<p>Enable Move Engine 0 Destination Error This bit enables the generation of a Move Engine 0 destination error interrupt.</p> <p>0_B Move Engine 0 destination error interrupt is disabled.</p> <p>1_B Move Engine 0 destination error interrupt is enabled.</p>
EME1SER	18	rw	<p>Enable Move Engine 1 Source Error This bit enables the generation of a Move Engine 1 source error interrupt.</p> <p>0_B Move Engine 1 source error interrupt is disabled.</p> <p>1_B Move Engine 1 source error interrupt is enabled.</p>
EME1DER	19	rw	<p>Enable Move Engine 0 Destination Error This bit enables the generation of a Move Engine 0 destination error interrupt.</p> <p>0_B Move Engine 1 destination error interrupt is disabled.</p> <p>1_B Move Engine 1 destination error interrupt is enabled.</p>
ME0INP	[23:20]	rw	<p>Move Engine 0 Error Interrupt Node Pointer ME0INP determines the number n (n = 0-15) of the service request output SRn that becomes active on a Move Engine 0 source or destination interrupt.</p> <p>0000_B SR0 selected for Move Engine 0 interrupt</p> <p>0001_B SR1 selected for Move Engine 0 interrupt</p> <p>..._B ...</p> <p>1110_B SR14 selected for Move Engine 0 interrupt</p> <p>1111_B SR15 selected for Move Engine 0 interrupt</p> <p><i>Note: In the TC1796 only SR[7:0] are connected to interrupt nodes.</i></p>

Direct Memory Access Controller

Field	Bits	Type	Description
ME1INP	[27:24]	rw	<p>Move Engine 1 Error Interrupt Node Pointer</p> <p>ME1INP determines the number n (n = 0-15) of the service request output SRn that becomes active on a Move Engine 1 source or destination interrupt.</p> <p>0000_B SR0 selected for Move Engine 1 interrupt 0001_B SR1 selected for Move Engine 1 interrupt ..._B ... 1110_B SR14 selected for Move Engine 1 interrupt 1111_B SR15 selected for Move Engine 1 interrupt</p> <p><i>Note: In the TC1796 only SR[7:0] are connected to interrupt nodes.</i></p>
TRLINP	[31:28]	rw	<p>Transaction Lost Interrupt Node Pointer</p> <p>TRLINP determines the number n (n = 0-15) of the service request output SRn that becomes active on a transaction lost interrupt.</p> <p>0000_B SR0 selected for transaction lost interrupt 0001_B SR1 selected for transaction lost interrupt ..._B ... 1110_B SR14 selected for transaction lost interrupt 1111_B SR15 selected for transaction lost interrupt</p> <p><i>Note: In the TC1796, only SR[7:0] are connected to interrupt nodes.</i></p>

Direct Memory Access Controller

The Error Status Register indicates if the DMA controller couldn't answer to a request because the previous request was not terminated (see [Section 12.1.4.4](#)). It indicates also the FPI Bus accesses that have been terminated with errors.

DMA_ERRSR

DMA Error Status Register

(024_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MLI 1		LEC ME1		MLI 0		LEC ME0		0		FPI1 ER	FPI0 ER	ME1 DER	ME1 SER	ME0 DER	ME0 SER
rh		rh		rh		rh		r		rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRL 17	TRL 16	TRL 15	TRL 14	TRL 13	TRL 12	TRL 11	TRL 10	TRL 07	TRL 06	TRL 05	TRL 04	TRL 03	TRL 02	TRL 01	TRL 00
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
TRL0x (x = 0-7)	x	rh	Transaction/Transfer Request Lost of DMA Channel 0x 0 _B No request lost event has been detected for DMA channel 0x. 1 _B A new DMA request was detected while TRSR.CH0x=1 (request lost event). This bit is cleared by software when writing a 1 to CLRE.CTL0x, or by a DMA channel reset (writing CHRSTR.CH0x = 1).
TRL1x (x = 0-7)	8+x	rh	Transaction/Transfer Request Lost of DMA Channel 1x 0 _B No request lost event has been detected for DMA channel 1x. 1 _B A new DMA request was detected while TRSR.CH1x=1 (request lost event). This bit is cleared by software when writing a 1 to CLRE.CTL1x, or by a DMA channel reset (writing CHRSTR.CH1x = 1).

Direct Memory Access Controller

Field	Bits	Type	Description
ME0SER	16	rh	<p>Move Engine 0 Source Error This bit is set whenever a Move Engine 0 error occurred during a source (read) move of a DMA transfer, or a request could not be serviced due to the access protection.</p> <p>0_B No Move Engine 0 source error has occurred. 1_B A Move Engine 0 source error has occurred.</p>
ME0DER	17	rh	<p>Move Engine 0 Destination Error This bit is set whenever a Move Engine 0 error occurred during a destination (write) move of a DMA transfer, or a request could not be serviced due to the access protection.</p> <p>0_B No Move Engine 0 destination error has occurred. 1_B A Move Engine 0 destination error has occurred.</p>
ME1SER	18	rh	<p>Move Engine 1 Source Error This bit is set whenever a Move Engine 1 error occurred during a source (read) move of a DMA transfer, or a request could not be serviced due to the access protection.</p> <p>0_B No Move Engine 1 source error has occurred. 1_B A Move Engine 1 source error has occurred.</p>
ME1DER	19	rh	<p>Move Engine 1 Destination Error This bit is set whenever a Move Engine 1 error occurred during a destination (write) move of a DMA transfer, or a request could not be serviced due to the access protection.</p> <p>0_B No Move Engine 1 destination error has occurred. 1_B A Move Engine 1 destination error has occurred.</p>
FPI0ER	20	rh	<p>SPB Error This bit is set whenever a move that has been started by the DMA/MLI master interface on FPI Bus interface 0 leads to an error.</p> <p>0_B No error occurred. 1_B An error occurred on FPI Bus interface 0.</p>

Direct Memory Access Controller

Field	Bits	Type	Description
FPI1ER	21	rh	<p>RPB Error</p> <p>This bit is set whenever a move that has been started by the DMA/MLI master interface on FPI Bus interface 1 leads to an error.</p> <p>0_B No error occurred. 1_B An error occurred on FPI Bus interface 1.</p>
LECME0	[26:24]	rh	<p>Last Error Channel Move Engine 0</p> <p>This bit field indicates the channel number of the last channel of Move Engine 0 leading to an FPI Bus error that has occurred.</p>
MLI0	27	rh	<p>MLI0 Error Source</p> <p>This bit is set whenever an FPI Bus error occurred due to an action of MLI0.</p> <p>0_B No FPI error occurred due to MLI0. 1_B An FPI error occurred due to MLI0.</p>
LECME1	[30:28]	rh	<p>Last Error Channel Move Engine 1</p> <p>This bit field indicates the channel number of the last channel of Move Engine 1 leading to an FPI Bus error that has occurred.</p>
MLI1	31	rh	<p>MLI1 Error Source</p> <p>This bit is set whenever an FPI Bus error occurred due to an action of MLI1.</p> <p>0_B No FPI error occurred due to MLI1. 1_B An FPI error occurred due to MLI1.</p>
0	[23:22]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Direct Memory Access Controller

The Clear Error Register contains bits that make it possible to clear the Transaction Request Lost flags or the Move Engine error flags.

DMA_CLRE

DMA Clear Error Register

(028_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLR MLI1	0			CLR MLI0	0					C FPI1 ER	C FPI0 ER	C ME1 DER	C ME1 SER	C ME0 DER	C ME0 SER
w	r			w	r					w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTL 17	CTL 16	CTL 15	CTL 14	CTL 13	CTL 12	CTL 11	CTL 10	CTL 07	CTL 06	CTL 05	CTL 04	CTL 03	CTL 02	CTL 01	CTL 00
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
CTL0x (x = 0-7)	x	w	Clear Transaction Request Lost for DMA Channel 0x 0 _B No action 1 _B Clear DMA channel 0x transaction request lost flag ERRSR.TRL0x
CTL1x (x = 0-7)	8+x	w	Clear Transaction Request Lost for DMA Channel 1x 0 _B No action 1 _B Clear DMA channel 1x transaction request lost flag ERRSR.TRL1x
CME0SER	16	w	Clear Move Engine 0 Source Error 0 _B No action 1 _B Clear source error flag ERRSR.ME0SER.
CME0DER	17	w	Clear Move Engine 0 Destination Error 0 _B No action 1 _B Clear destination error flag ERRSR.ME0DER.
CME1SER	18	w	Clear Move Engine 1 Source Error 0 _B No action 1 _B Clear source error flag ERRSR.ME1SER.
CME1DER	19	w	Clear Move Engine 1 Destination Error 0 _B No action 1 _B Clear destination error flag ERRSR.ME1DER.

Direct Memory Access Controller

Field	Bits	Type	Description
CFPI0ER	20	w	Clear SPB Error 0 _B No action 1 _B Clear error flag ERRSR.FPI0ER.
CFPI1ER	21	w	Clear RPB Error 0 _B No action 1 _B Clear error flag ERRSR.FPI1ER.
CLRMLI0	27	w	Clear MLI0 Error 0 _B No action 1 _B Clear error flag ERRSR.MLI0.
CLRMLI1	31	w	Clear MLI1 Error 0 _B No action 1 _B Clear error flag ERRSR.MLI1.
0	[26:22], [30:28]	r	Reserved Read as 0; should be written with 0.

Direct Memory Access Controller

The Interrupt Status Register indicates if CHSRmx.TCOUNT matches with CHCRmx.IRDV, or if CHSRmx.TCOUNT has been decremented (depending on CHICRmx.INTCT[0]), or if a pattern has been detected. These conditions can also generate an interrupt if enabled (see [Figure 12-19](#) on [Page 12-28](#)).

DMA_INTSR

DMA Interrupt Status Register (054_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPM	IPM	IPM	IPM	IPM	IPM	IPM	IPM	IPM	IPM	IPM	IPM	IPM	IPM	IPM	IPM
17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ICH	ICH	ICH	ICH	ICH	ICH	ICH	ICH	ICH	ICH	ICH	ICH	ICH	ICH	ICH	ICH
17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
ICH0x (x = 0-7)	x	rh	<p>Interrupt from Channel 0x</p> <p>This bit indicates that channel 0x has raised an interrupt for TCOUNT = IRDV or if TCOUNT has been decremented (depending on CHICR.INTCT[0]). This bit (and IP0x) is cleared by software when writing a 1 to INTCR.CICH0x or by a channel reset (writing CHRSTR.CH0x = 1).</p> <p>0_B A channel interrupt has not been detected. 1_B A channel interrupt has been detected.</p>
ICH1x (x = 0-7)	8+x	rh	<p>Interrupt from Channel 1x</p> <p>This bit indicates that channel 1x has raised an interrupt for TCOUNT = IRDV or if TCOUNT has been decremented (depending on CHICR.INTCT[0]). This bit (and IP1x) is cleared by software when writing a 1 to INTCR.CICH1x or by a channel reset (writing CHRSTR.CH1x = 1).</p> <p>0_B A channel interrupt has not been detected. 1_B A channel interrupt has been detected.</p>

Direct Memory Access Controller

Field	Bits	Type	Description
IPM0x (x = 0-7)	16+x	rh	<p>Pattern Detection from Channel 0x</p> <p>This bit indicates that a pattern has been detected for channel 0x while the pattern detection has been enabled. This bit (and ICH0x) is cleared by software when writing a 1 to INTCR.CICH0x or by a channel reset (writing CHRSTR.CH0x = 1).</p> <p>0_B A pattern has not been detected. 1_B A pattern has been detected.</p>
IPM1x (x = 0-7)	24+x	rh	<p>Pattern Detection from Channel 1x</p> <p>This bit indicates that a pattern has been detected for channel 1x while the pattern detection has been enabled. This bit (and ICH1x) is cleared by software when writing a 1 to INTCR.CICH1x or by a channel reset (writing CHRSTR.CH1x = 1).</p> <p>0_B A pattern has not been detected. 1_B A pattern has been detected.</p>

Direct Memory Access Controller

The Wrap Status Register gives information about the channels that did a wrap-around on their source or destination buffer(s). This condition can also lead to an interrupt if it is enabled.

DMA_WRPSR

DMA Wrap Status Register

(05C_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRP D17	WRP D16	WRP D15	WRP D14	WRP D13	WRP D12	WRP D11	WRP D10	WRP D07	WRP D06	WRP D05	WRP D04	WRP D03	WRP D02	WRP D01	WRP D00
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRP S17	WRP S16	WRP S15	WRP S14	WRP S13	WRP S12	WRP S11	WRP S10	WRP S07	WRP S06	WRP S05	WRP S04	WRP S03	WRP S02	WRP S01	WRP S00
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
WRPS0x (x = 0-7)	x	rh	Wrap Source Buffer for Channel 0x These bits indicate which channels have done a wrap-around of their source buffer(s). 0 _B No wrap-around occurred for channel 0x. 1 _B A wrap-around occurred for channel 0x. This bit is cleared by software by writing a 1 to INTCR.CWRP0x or CHRSTR.CH0x.
WRPS1x (x = 0-7)	8+x	rh	Wrap Source Buffer for Channel 1x These bits indicate which channels have done a wrap-around of their source buffer(s). 0 _B No wrap-around occurred for channel 1x. 1 _B A wrap-around occurred for channel 1x. This bit is cleared by software by writing a 1 to INTCR.CWRP1x or CHRSTR.CH1x.
WRPD0x (x = 0-7)	16+x	rh	Wrap Destination Buffer for Channel 0x These bits indicate which channels have done a wrap-around of their destination buffer(s). 0 _B No wrap-around occurred for channel 0x. 1 _B A wrap-around occurred for channel 0x. This bit is cleared by software by writing a 1 to INTCR.CWRP0x or CHRSTR.CH0x.

Direct Memory Access Controller

Field	Bits	Type	Description
WRPD1x (x = 0-7)	24+x	rh	Wrap Destination Buffer for Channel 1x These bits indicate which channels have done a wrap-around of their destination buffer(s). 0 _B No wrap-around occurred for channel 1x. 1 _B A wrap-around occurred for channel 1x. This bit is cleared by software by writing a 1 to INTCR.CWRP1x or CHRSTR.CH1x.

The bits in the Interrupt Clear Register make it possible to clear the channel interrupt flags and the wrap buffer interrupt flags for DMA Channels mx.

DMA_INTCR

DMA Interrupt Clear Register

(058_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C
WRP	WRP	WRP	WRP	WRP	WRP	WRP	WRP	WRP	WRP	WRP	WRP	WRP	WRP	WRP	WRP
17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C
ICH	ICH	ICH	ICH	ICH	ICH	ICH	ICH	ICH	ICH	ICH	ICH	ICH	ICH	ICH	ICH
17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Field	Bits	Type	Description
CICH0x (x = 0-7)	x	w	Clear Interrupt for DMA Channel 0x These bits make it possible to clear the channel interrupt flags INTSR.ICH0x and INTSR.IPM0x of DMA channel 0x by software. 0 _B No action. 1 _B Bits INTSR.ICH0x and INTSR.IPM0x are cleared.

Direct Memory Access Controller

Field	Bits	Type	Description
CICH1x (x = 0-7)	8+x	w	<p>Clear Interrupt for DMA Channel 1x These bits make it possible to clear the channel interrupt flags INTSR.ICH1x and INTSR.IPM1x of DMA channel 1x by software.</p> <p>0_B No action. 1_B Bits INTSR.ICH1x and INTSR.IPM1x are cleared.</p>
CWRP0x (x = 0-7)	16+x	w	<p>Clear Wrap Buffer Interrupt for DMA Channel 0x These bits make it possible to clear the wrap source buffer interrupt flag WRPSR.WRPS0x and the wrap destination buffer interrupt flag WRPSR.WRPD0x (both together) of DMA channel 0x by software.</p> <p>0_B No action. 1_B Bits WRPSR.WRPS0x and WRPSR.WRPD0x are cleared.</p>
CWRP1x (x = 0-7)	24+x	w	<p>Clear Wrap Buffer Interrupt for DMA Channel 1x These bits make it possible to clear the wrap source buffer interrupt flag WRPSR.WRPS1x and the wrap destination buffer interrupt flag WRPSR.WRPD1x (both together) of DMA channel 1x by software.</p> <p>0_B No action. 1_B Bits WRPSR.WRPS1x and WRPSR.WRPD1x are cleared.</p>

Direct Memory Access Controller

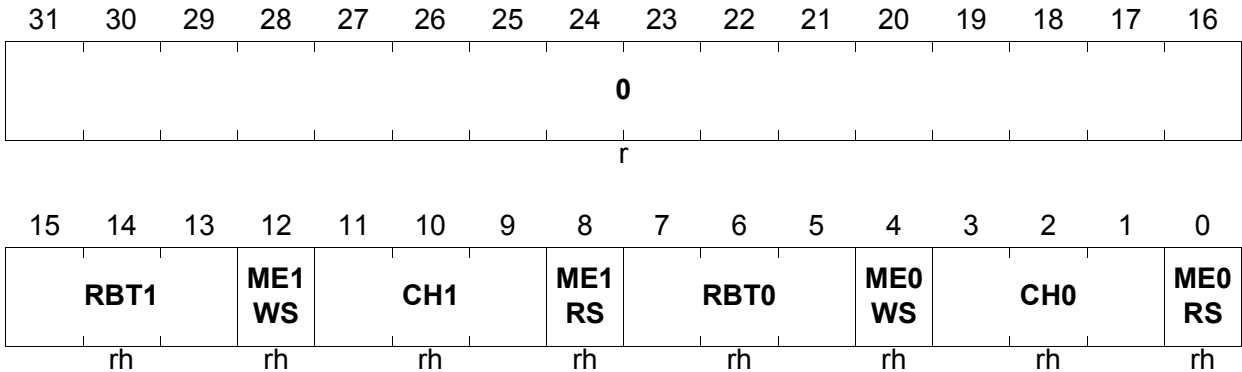
12.2.3 Move Engine Registers

The Move Engine Status Register is a read-only register that holds status information about the transaction handled by the Move Engines.

DMA_MESR

DMA Move Engine Status Register (030_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
ME0RS	0	rh	Move Engine 0 Read Status 0 _B Move Engine 0 is not performing a read. 1 _B Move Engine 0 is performing a read.
CH0	[3:1]	rh	Reading Channel in Move Engine 0 This bit field indicates which channel number is currently being processed by the Move Engine 0.
ME0WS	4	rh	Move Engine 0 Write Status 0 _B Move Engine 0 is not performing a write. 1 _B Move Engine 0 is performing a write.

Direct Memory Access Controller

Field	Bits	Type	Description
RBT0	[7:5]	rh	<p>Read Buffer Trace for FPI Bus Interface 0</p> <p>This bit field contains trace information from the buffer in the FPI Bus Interface 0. In the TC1796 it indicates the source of a bus access to the Remote Peripheral Bus.</p> <p>000_B Reset value or bridge from FPI0 to FPI1. 001_B DMA Move Engine 0 010_B DMA Move Engine 1 011_B MLI0 100_B MLI1</p> <p>Other bit combinations are reserved. RBT0 is useful for emulation purposes. It is not recommended to evaluate this bit field during normal operation of the TC1796.</p>
ME1RS	8	rh	<p>Move Engine 1 Read Status</p> <p>0_B Move Engine 1 is not performing a read. 1_B Move Engine 1 is performing a read.</p>
CH1	[11:9]	rh	<p>Reading Channel in Move Engine 1</p> <p>These bit field indicates which channel number is currently being processed by the Move Engine 1.</p>
ME1WS	12	rh	<p>Move Engine 1 Write Status</p> <p>0_B Move Engine 1 is not performing a write. 1_B Move Engine 1 is performing a write.</p>
RBT1	[15:13]	rh	<p>Read Buffer Trace for FPI Bus Interface 1</p> <p>This bit field contains trace information from the buffer in the FPI Bus Interface 1. In the TC1796, it indicates the source of a bus access to the System Peripheral Bus.</p> <p>000_B Reset value or bridge from FPI0 to FPI1. 001_B DMA Move Engine 0 010_B DMA Move Engine 1 011_B MLI0 100_B MLI1</p> <p>Other bit combinations are reserved. RBT1 is useful for emulation purposes. It is not recommended to evaluate this bit field during normal operation of the TC1796.</p>
0	[31:16]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Direct Memory Access Controller

The Move Engine m ($m = 0,1$) Read Register indicates the value that has just been read by Move Engine m . The value in this register is compared to the bits in register MEMPR according to the bit fields CHCR0x.PATSEL.

DMA_ME0R

DMA Move Engine 0 Read Register

(034_H)

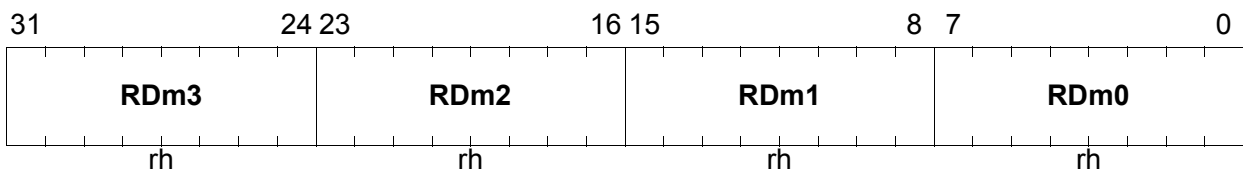
Reset Value: 0000 0000_H

DMA_ME1R

DMA Move Engine 1 Read Register

(038_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
RDm0, RDm1, RDm2, RDm3	[7:0], [15:8], [23:16], [31:24]	rh	Read Value for Move Engine m Contains the 32-bit read data (four bytes RDm[3:0]) that is stored in the Move Engine m after each read move. The content of MEMR is overwritten after each read move of a DMA channel belonging to DMA Sub-block m .

Direct Memory Access Controller

The Move Engine *m* (*m* = 0,1) Pattern Register contains the patterns (mask and/or compare bits) to be processed by the pattern detection logic in Move Engine *m*.

DMA_ME0PR

DMA Move Engine 0 Pattern Register

(03C_H)

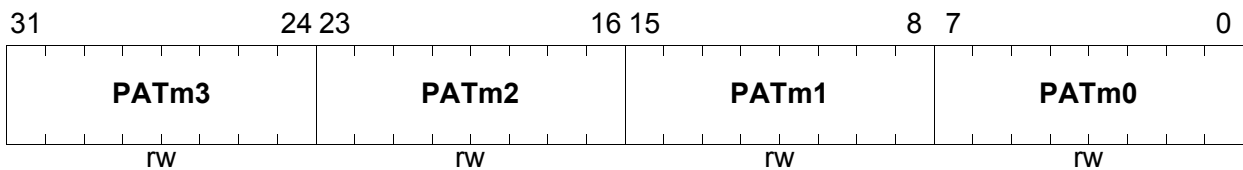
Reset Value: 0000 0000_H

DMA_ME1PR

DMA Move Engine 1 Pattern Register

(040_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PATm0, PATm1, PATm2, PATm3	[7:0], [15:8], [23:16], [31:24]	rw	Pattern for Move Engine <i>m</i> Determines up to four 8-bit compare patterns/mask patterns to be processed by the pattern detection logic in Move Engine <i>m</i> . Depending on the pattern detection configuration (CHCRmx.PATSEL) and channel data width (CHCRmx.CHDW), the patterns are processed as bytes or half-words.

Direct Memory Access Controller

The DMA Move Engine m (m = 0,1) Access Enable Register controls the access protection. It enables/disables the address protection ranges n (n = 0-31) for Move Engine m.

DMA_ME0AENR

DMA Move Engine 0 Access Enable Register
(044_H)

Reset Value: 0000 0000_H

DMA_ME1AENR

DMA Move Engine 1 Access Enable Register
(04C_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Field	Bits	Type	Description
AENn (n = 0-31)	n	rW	<p>Address Range x Enable</p> <p>This bit enables the read and write capability of the DMA Move Engines for address range n (n = 0-31).</p> <p>0_B DMA read and write moves to address range n are disabled.</p> <p>1_B DMA read and write moves to address range n are enabled.</p> <p>If AENn = 0 for a read/write move to address range n, the read/write move is not executed and a source/destination Move Engine interrupt is generated</p>

Note: See [Table 12-11](#) on [Page 12-98](#) for the TC1796 specific address range definition.

Direct Memory Access Controller

The DMA Move Engine m (m = 0,1) Access Range Register determines number and size of the sub-ranges for address range extension n (n = 0-3). See also [Figure 12-28](#) for bit field definitions.

DMA_ME0ARR

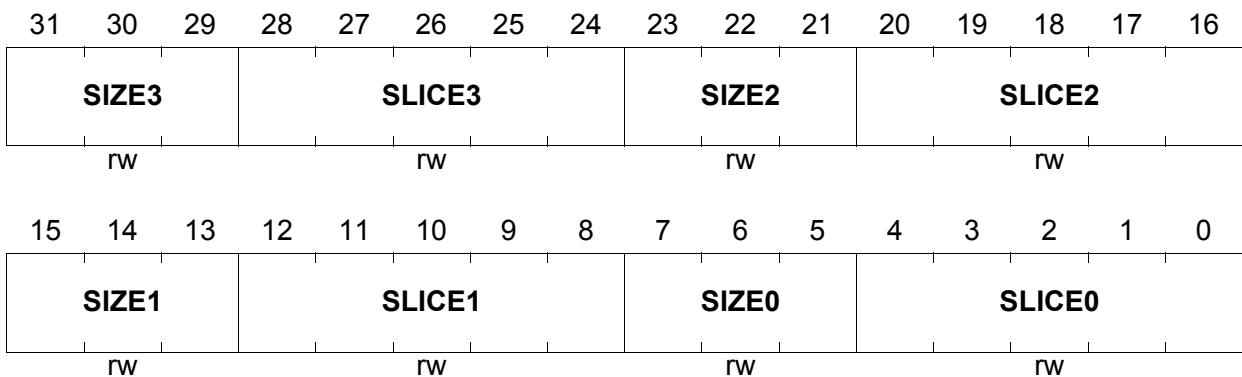
DMA Move Engine 0 Access Range Register
(048_H)

Reset Value: 0000 0000_H

DMA_ME1ARR

DMA Move Engine 1 Access Range Register
(050_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SLICE0	[4:0]	rw	Address Slice 0 SLICE0 selects a specific sub-range within address range extension 0.
SIZE0	[7:5]	rw	Address Size 0 SIZE0 determines the sub-range size within address range extension 0.
SLICE1	[12:8]	rw	Address Slice 1 SLICE1 selects a specific sub-range within address range extension 1.
SIZE1	[15:13]	rw	Address Size 1 SIZE1 determines the sub-range size within address range extension 1.
SLICE2	[20:16]	rw	Address Slice 2 SLICE2 selects a specific sub-range within address range extension 2.

Direct Memory Access Controller

Field	Bits	Type	Description
SIZE2	[23:21]	rw	Address Size 2 SIZE2 determines the sub-range size within address range extension 2.
SLICE3	[28:24]	rw	Address Slice 3 SLICE3 selects a specific sub-range within address range extension 3.
SIZE3	[31:29]	rw	Address Size 3 SIZE3 determines the sub-range size within address range extension 3.

Note: See [Section 12.3.2](#) on [Page 12-98](#) for the TC1796 specific address range and address range extension definitions.

Direct Memory Access Controller

12.2.4 Channel Control/Status Registers

The Channel Control Register for DMA channel mx contains its configuration and its control bits and bit fields.

DMA_CHCR0x (x = 0-7)

DMA Channel 0x Control Register

($x*20_H+84_H$)

Reset Value: 0000 0000_H

DMA_CHCR1x (x = 0-7)

DMA Channel 1x Control Register

($x*20_H+184_H$)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	DMA PRIO	0	CH PRIO	0	PATSEL		0	CHDW		CH MO DE	RRO AT	BLKM			
r	rw	r	rw	r	rw		r	rw		rw	rw	rw			
PRSEL				0				TREL							
rw				r				rw							

Field	Bits	Type	Description
TREL	[8:0]	rw	<p>Transfer Reload Value</p> <p>This bit field contains the number of DMA transfers for s DMA transaction of DMA channel mx. This 9-bit transfer count value is loaded into CHSRmx.TCOUNT at the start of a DMA transaction (when TRSR.CHmx becomes set and CHSRmx.TCOUNT = 0). TREL can be written during a running DMA transaction because TCOUNT will be updated (decremented) during the DMA transaction.</p> <p>If TREL = 0 or if TREL = 1, TCOUNT will be loaded with 1 when a new transaction is started (at least one DMA transfer must be executed per DMA transaction).</p>

Direct Memory Access Controller

Field	Bits	Type	Description
PRSEL	[15:13]	rw	<p>Peripheral Request Select</p> <p>This bit field controls the hardware request input multiplexer of DMA channel mx (m=0,1; see Figure 12-6 on Page 12-10).</p> <p>000_B Input CHmx_REQI0 selected 001_B Input CHmx_REQI1 selected 010_B Input CHmx_REQI2 selected 011_B Input CHmx_REQI3 selected 101_B Input CHmx_REQI4 selected 110_B Input CHmx_REQI5 selected 110_B Input CHmx_REQI6 selected 111_B Input CHmx_REQI7 selected</p>
BLKM	[18:16]	rw	<p>Block Mode</p> <p>BLKM determines the number of DMA moves executed during one DMA transfer.</p> <p>000_B One DMA transfer has 1 DMA moves 001_B One DMA transfer has 2 DMA moves 010_B One DMA transfer has 4 DMA moves 011_B One DMA transfer has 8 DMA moves 100_B One DMA transfer has 16 DMA moves others Reserved; do not use these combinations. See also Figure 12-10 on Page 12-17.</p>
RROAT	19	rw	<p>Reset Request Only After Transaction</p> <p>RROAT determines whether or not the TRSR.CHmx transfer request state flag is cleared after each transfer.</p> <p>0_B TRSR.CHmx is cleared after each transfer. A transfer request is required for each transfer. 1_B TRSR.CHmx is cleared when TCOUNT = 0 after a transfer. One transfer request starts a complete DMA transaction.</p>

Direct Memory Access Controller

Field	Bits	Type	Description
CHMODE	20	rw	<p>Channel Operation Mode CHMODE determines the clear condition for control bit TRSR.HTREmx of DMA channel mx.</p> <p>0_B Single Mode operation is selected for DMA channel mx. After a transaction, DMA channel mx is disabled for further hardware requests (TRSR.HTREmx is cleared by hardware). TRSR.HTREmx must be set again by software for starting a new transaction.</p> <p>1_B Continuous Mode operation is selected for DMA channel mx. After a transaction, bit TRSR.HTREmx remains set.</p>
CHDW	[22:21]	rw	<p>Channel Data Width CHDW determines the data width for the read and write moves of DMA channel mx.</p> <p>00_B 8-bit (byte) data width for moves selected 01_B 16-bit (half-word) data width for moves selected 10_B 32-bit (word) data width for moves selected 11_B Reserved</p>
PATSEL	[25:24]	rw	<p>Pattern Select This bit field selects the mode of the pattern detection logic. Depending on the channel data width, PATSEL selects different pattern detection configurations. If pattern detection is enabled (PATSEL not equal 00_B), the pattern detection interrupt line will be activated on the selected pattern match.</p> <p>8-Bit channel data width (CHDW = 00_B): Selected pattern detection configuration; see Table 12-2 on Page 12-4.</p> <p>16-Bit channel data width (CHDW = 01_B): Selected pattern detection configuration; see Table 12-3 on Page 12-5.</p> <p>32-Bit channel data width (CHDW = 10_B): Selected pattern detection configuration; see Table 12-4 on Page 12-8.</p>
CHPRIO	28	rw	<p>Channel Priority CHPRIO determines the priority of DMA channel mx for the channel arbitration of Move Engine m.</p> <p>0_B DMA channel mx has a low channel priority. 1_B DMA channel mx has a high channel priority.</p>

Direct Memory Access Controller

Field	Bits	Type	Description
DMAPRIO	30	rw	<p>DMA Priority</p> <p>This bit determines the bus request priority that is used when a move operation related to channel mx is requesting FPI Bus 0. The value of DMAPRIO is also used for the DMA-internal arbitration between the Move Engines of the DMA.</p> <p>This bit has no effect in the channel prioritization and in the prioritization between the Move Engines.</p> <p>0_B Low priority selected 1_B High priority selected</p>
0	[12:9], 23, [27:26], 29, 31	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Direct Memory Access Controller

The Channel Status Register contains the current transfer count and a pattern detection compare result.

DMA_CHSR0x (x = 0-7)

DMA Channel 0x Status Register

($x \cdot 20_H + 80_H$)

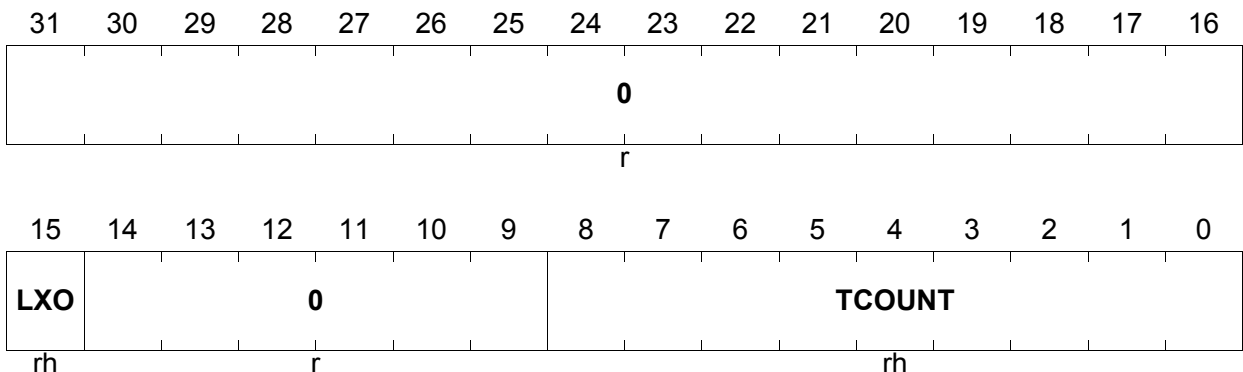
Reset Value: 0000 0000_H

DMA_CHSR1x (x = 0-7)

DMA Channel 1x Status Register

($x \cdot 20_H + 180_H$)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
TCOUNT	[8:0]	rh	<p>Transfer Count Status</p> <p>TCOUNT holds the actual value of the DMA transfer count for DMA channel mx. TCOUNT is loaded with the value of CHCRmx.TREL when TRSR.CHmx becomes set (and TCOUNT = 0). After each DMA transfer, TCOUNT is decremented by 1.</p>
LXO	15	rh	<p>Old Value of Pattern Detection</p> <p>This bit contains the compare result of a pattern compare operation when 8-bit or 16-bit data width is selected.</p> <p>8-bit data width: see Table 12-2 and Figure 12-25</p> <p>16-bit data width: see Table 12-3 and Figure 12-26</p> <p>0_B The corresponding pattern compare operation didn't find a pattern match at the last move.</p> <p>1_B The corresponding pattern compare operation found a pattern match at the last move.</p>
0	[14:9], [31:16]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Direct Memory Access Controller

The Channel Interrupt Control Register control the interrupts generation.

DMA_CHICR0x (x = 0-7)

DMA Channel 0x Interrupt Control Register

($x \cdot 20_H + 88_H$)

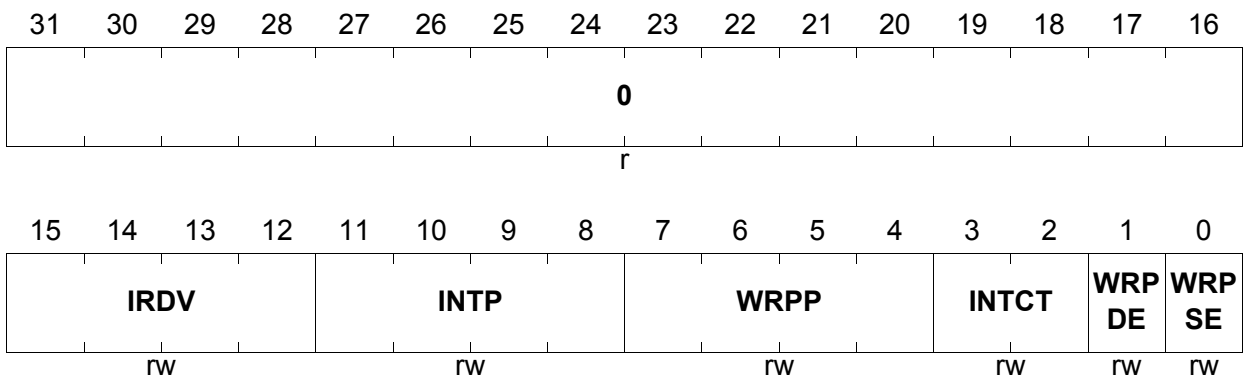
Reset Value: 0000 0000_H

DMA_CHICR1x (x = 0-7)

DMA Channel 1x Interrupt Control Register

($x \cdot 20_H + 188_H$)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
WRPSE	0	rw	Wrap Source Enable 0 _B Wrap source buffer interrupt disabled 1 _B Wrap source buffer interrupt enabled
WRPDE	1	rw	Wrap Destination Enable 0 _B Wrap destination buffer interrupt disabled 1 _B Wrap destination buffer interrupt enabled
INTCT	[3:2]	rw	Interrupt Control 00 _B No interrupt will be generated on changing the TCOUNT value. The bit INTSR.ICHmx is set when TCOUNT equals IRDV. 01 _B No interrupt will be generated on changing the TCOUNT value. The bit INTSR.ICHmx is set when TCOUNT is decremented. 10 _B An interrupt is generated and bit INTSR.ICHmx is set each time TCOUNT equals IRDV. 11 _B Interrupt is generated and bit INTSR.ICHmx is set each time TCOUNT is decremented.

Direct Memory Access Controller

Field	Bits	Type	Description
WRPP	[7:4]	rw	<p>Wrap Pointer WRPP determines the number n (n = 0-15) of the service request output SRn that becomes active on a wrap buffer interrupt.</p> <p>0000_B SR0 selected for channel mx wrap buffer interrupt</p> <p>0001_B SR1 selected for channel mx wrap buffer interrupt</p> <p>..._B ...</p> <p>1111_B SR15 selected for channel mx wrap buffer interrupt</p> <p><i>Note: In the TC1796, only SR[7:0] are connected to interrupt nodes.</i></p>
INTP	[11:8]	rw	<p>Interrupt Pointer INTP determines the number n (n = 0-15) of the service request output SRn that becomes active on a channel interrupt.</p> <p>0000_B SR0 selected for channel mx interrupt</p> <p>0001_B SR1 selected for channel mx interrupt</p> <p>..._B ...</p> <p>1111_B SR15 selected for channel mx interrupt</p> <p><i>Note: In the TC1796, only SR[7:0] are connected to interrupt nodes.</i></p>
IRDV	[15:12]	rw	<p>Interrupt Raise Detect Value These bits specify the value of CHSRmx.TCOUNT for which the Interrupt Threshold Limit should be raised (m = 0-1).</p>
0	[31:16] [7:4]	r	<p>Reserved Read as 0; should be written with 0.</p>

Note: The interrupt node of the wrap-around interrupts is shared with the pattern match interrupt. In order to support interrupt generation in case of a pattern match, the wrap-around interrupt should be disabled. If the wrap-around interrupts are used, the pattern match interrupt should not be used. The settings are independent for each DMA channel.

Direct Memory Access Controller

The Address Control Register controls how source and destination addresses are updated after a DMA move. Furthermore, it determines whether or not a source or destination address register update is shadowed.

DMA_ADRCR0x (x = 0-7)

DMA Channel 0x Address Control Register

($x \cdot 20_H + 8C_H$)

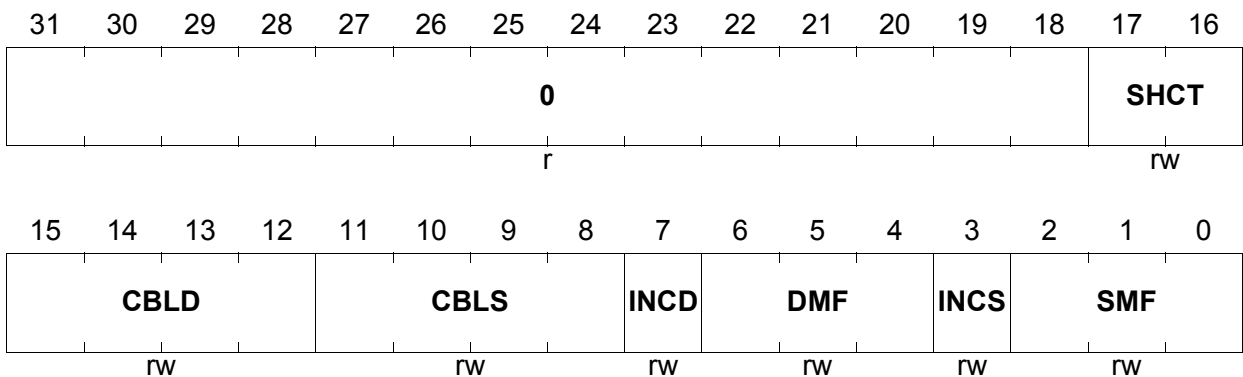
Reset Value: 0000 0000_H

DMA_ADRCR1x (x = 0-7)

DMA Channel 1x Address Control Register

($x \cdot 20_H + 18C_H$)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SMF	[2:0]	rw	<p>Source Address Modification Factor This bit field and the data width as defined in CHCRmx.CHDW determine an address offset value by which the source address is modified after each DMA move. See also Table 12-8.</p> <p>000_B Address offset is 1 × CHDW. 001_B Address offset is 2 × CHDW. 010_B Address offset is 4 × CHDW. 011_B Address offset is 8 × CHDW. 100_B Address offset is 16 × CHDW. 101_B Address offset is 32 × CHDW. 110_B Address offset is 64 × CHDW. 111_B Address offset is 128 × CHDW.</p>

Direct Memory Access Controller

Field	Bits	Type	Description
INCS	3	rw	<p>Increment of Source Address</p> <p>This bit determines whether the address offset as selected by SMF will be added to or subtracted from the source address after each DMA move. The source address is not modified if CBL5 = 0000_B.</p> <p>0_B Address offset will be subtracted. 1_B Address offset will be added.</p>
DMF	[6:4]	rw	<p>Destination Address Modification Factor</p> <p>This bit field and the data width as defined in CHCRmx.CHDW determine an address offset value by which the destination address is modified after each DMA move. The destination address is not modified if CBLD = 0000_B. See also Table 12-8.</p> <p>000_B Address offset is 1 × CHDW. 001_B Address offset is 2 × CHDW. 010_B Address offset is 4 × CHDW. 011_B Address offset is 8 × CHDW. 100_B Address offset is 16 × CHDW. 101_B Address offset is 32 × CHDW. 110_B Address offset is 64 × CHDW. 111_B Address offset is 128 × CHDW.</p>
INCD	7	rw	<p>Increment of Destination Address</p> <p>This bit determines whether the address offset as selected by DMF will be added to or subtracted from the destination address after each DMA move. The destination address is not modified if CBLD = 0000_B.</p> <p>0_B Address offset will be subtracted. 1_B Address offset will be added.</p>

Direct Memory Access Controller

Field	Bits	Type	Description
CBLS	[11:8]	rw	<p>Circular Buffer Length Source</p> <p>This bit field determines which part of the 32-bit source address register remains unchanged and is not updated after a DMA move operation (see also Page 12-19). Therefore, CBLS also determines the size of the circular source buffer.</p> <p>0000_B Source address SADR[31:0] is not updated. 0001_B Source address SADR[31:1] is not updated. 0010_B Source address SADR[31:2] is not updated. 0011_B Source address SADR [31:3] is not updated. ..._B ... 1110_B Source address SADR[31:14] is not updated. 1111_B Source address SADR[31:15] is not updated.</p>
CBLD	[15:12]	rw	<p>Circular Buffer Length Destination</p> <p>This bit field determines which part of the 32-bit destination address register remains unchanged and is not updated after a DMA move operation (see also Page 12-19). Therefore, CBLD also determines the size of the circular destination buffer.</p> <p>0000_B Destination address DADR[31:0] is not updated. 0001_B Destination address DADR[31:1] is not updated. 0010_B Destination address DADR[31:2] is not updated. 0011_B Destination address DADR [31:3] is not updated. ..._B ... 1110_B Destination address DADR[31:14] is not updated. 1111_B Destination address DADR[31:15] is not updated.</p>

Direct Memory Access Controller

Field	Bits	Type	Description
SHCT	[17:16]	rw	<p>Shadow Control This bit field determines whether an address is transferred into the shadow address register when writing to source or destination address register.</p> <p>00_B Shadow address register not used. Source and destination address register are written directly.</p> <p>01_B Shadow address register used for source address buffering. When writing to SADRmx, the address is buffered in SHADRmx and transferred to SADRmx with the start of the next DMA transaction.</p> <p>10_B Shadow address register used for destination address buffering. When writing to DADRmx, the address is buffered in SHADRmx and transferred to DADRmx with the start of the next DMA transaction.</p> <p>11_B Reserved In case of SHCT = 01_B or 10_B, SHCT must not be changed until the next DMA transaction has been started.</p>
0	[31:18]	r	<p>Reserved Read as 0; should be written with 0.</p>

Direct Memory Access Controller

Table 12-8 shows the offset values that are added or subtracted to/from a source or destination address register after a DMA move. Bit field SMF and bit INCS determine the offset value for the source address. Bit field DMF and bit INCD determine the offset value for the destination address.

Table 12-8 Address Offset Calculation Table

CHCRmx.CHDW = 00 _B (8-bit Data Width)			CHCRmx.CHDW = 01 _B (16-bit Data Width)			CHCRmx.CHDW = 10 _B (32-bit Data Width)		
SMF DMF	INCS INCD	Address Offset	SMF DMF	INCS INCD	Address Offset	SMF DMF	INCS INCD	Address Offset
000 _B	0	-1	000 _B	0	-2	000 _B	0	-4
	1	+1		1	+2		1	+4
001 _B	0	-2	001 _B	0	-4	001 _B	0	-8
	1	+2		1	+4		1	+8
010 _B	0	-4	010 _B	0	-8	010 _B	0	-16
	1	+4		1	+8		1	+16
011 _B	0	-8	011 _B	0	-16	011 _B	0	-32
	1	+8		1	+16		1	+32
100 _B	0	-16	100 _B	0	-32	100 _B	0	-64
	1	+16		1	+32		1	+64
101 _B	0	-32	101 _B	0	-64	101 _B	0	-128
	1	+32		1	+64		1	+128
110 _B	0	-64	110 _B	0	-128	110 _B	0	-256
	1	+64		1	+128		1	+256
111 _B	0	-128	111 _B	0	-256	111 _B	0	-512
	1	+128		1	+256		1	+512

Note: CHCRmx.CHDW = 11_B is reserved and should not be used.

Direct Memory Access Controller

12.2.5 Channel Address Registers

The Source Address Register contains the 32-bit source address. If a DMA channel mx is active, SADRmx is updated continuously (if programmed) and shows the actual source address that is used for read moves within DMA transfers.

DMA_SADR0x (x = 0-7)

DMA Channel 0x Source Address Register

($x \cdot 20_H + 90_H$)

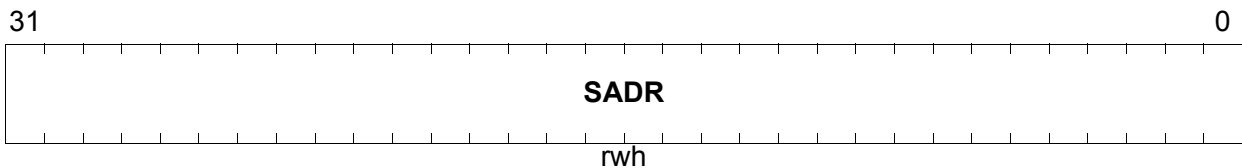
Reset Value: 0000 0000_H

DMA_SADR1x (x = 0-7)

DMA Channel 1x Source Address Register

($x \cdot 20_H + 190_H$)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SADR	[31:0]	rwh	Source Start Address This bit field holds the actual 32-bit source address of DMA channel mx that is used for read moves.

A write to SADRmx is executed directly only when the DMA channel mx is inactive (CHSRmx.TCOUNT = 0 and TRSR.CHmx = 0). If DMA channel mx is active when writing to SADRmx, the source address will not be written into SADRmx directly but will be buffered in the shadow register SHADRmx until the start of the next DMA transaction. During this shadowed address register operation, bit field ADRCRmx.SHCT must be set to 01_B.

Direct Memory Access Controller

The Destination Address Register contains the 32-bit destination address. If a DMA channel is active, DADR_{mx} is updated continuously (if programmed) and shows the actual destination address that is used for write moves within DMA transfers.

DMA_DADR0x (x = 0-7)

DMA Channel 0x Destination Address Register

(x*20_H+94_H)

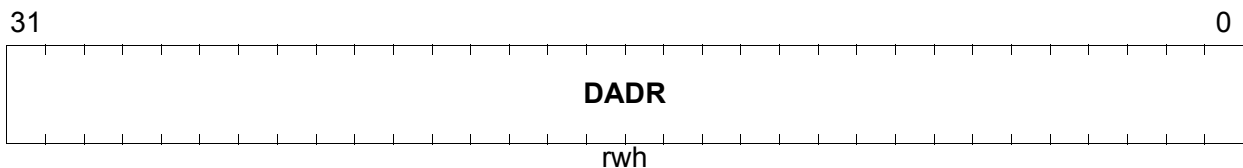
Reset Value: 0000 0000_H

DMA_DADR1x (x = 0-7)

DMA Channel 1x Destination Address Register

(x*20_H+194_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
DADR	[31:0]	rwh	Destination Address This bit field holds the actual 32-bit destination address of DMA channel mx that is used for write moves.

A write to DADR_{mx} is executed directly only when the DMA channel mx is inactive (CHSR_{mx}.TCOUNT = 0 and TRSR.CH_{mx} = 0). If DMA channel mx is active when writing to DADR_{mx}, the source address will not be written into DADR_{mx} directly but will be buffered in the shadow register SHADR_{mx} until the start of the next DMA transaction. During this shadowed address register operation, bit field ADRCR_{mx}.SHCT must be set to 10_B.

Direct Memory Access Controller

The Shadow Address Register holds the shadowed source or destination address before it is written into the source or destination address register. SHADRmx can be read only.

DMA_SHADR0x (x = 0-7)

DMA Channel 0x Shadow Address Register

($x*20_H+98_H$)

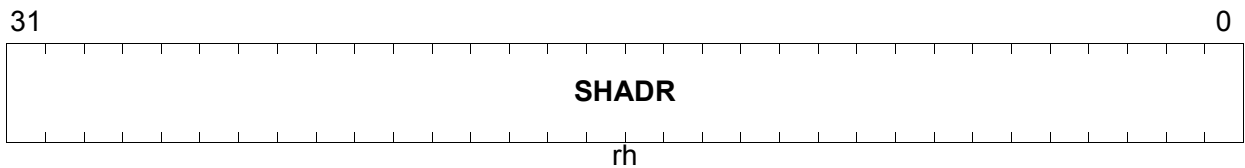
Reset Value: 0000 0000_H

DMA_SHADR1x (x = 0-7)

DMA Channel 1x Shadow Address Register

($x*20_H+198_H$)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SHADR	[31:0]	rh	Shadowed Address This bit field holds the shadowed 32-bit source or destination address of DMA channel mx.

SHADRmx is written when source or destination address buffering is selected (ADRCRmx.SHCT = 01_B or ADRCRmx.SHCT = 10_B) and a transaction is running. While the shadow mechanism is disabled, SHADR is set to 0000 0000_H.

The value stored in the SHADR is automatically set to 0000 0000_H when the shadow transfer takes place. The user can read the shadow register in order to detect if the shadow transfer has already taken place. If the value in SHADR is 0000 0000_H, no shadow transfer can take place and the corresponding address register is modified according to the circular buffer rules.

12.3 DMA Module Implementation

This section describes the TC1796 DMA module interfaces with the clock control, interrupt control, and address decoding.

Figure 12-30 shows the TC1796-specific implementation details and interconnections of the DMA module. The DMA module is supplied by a separate clock control, address decoding, interrupt control, and the request input wiring matrix.

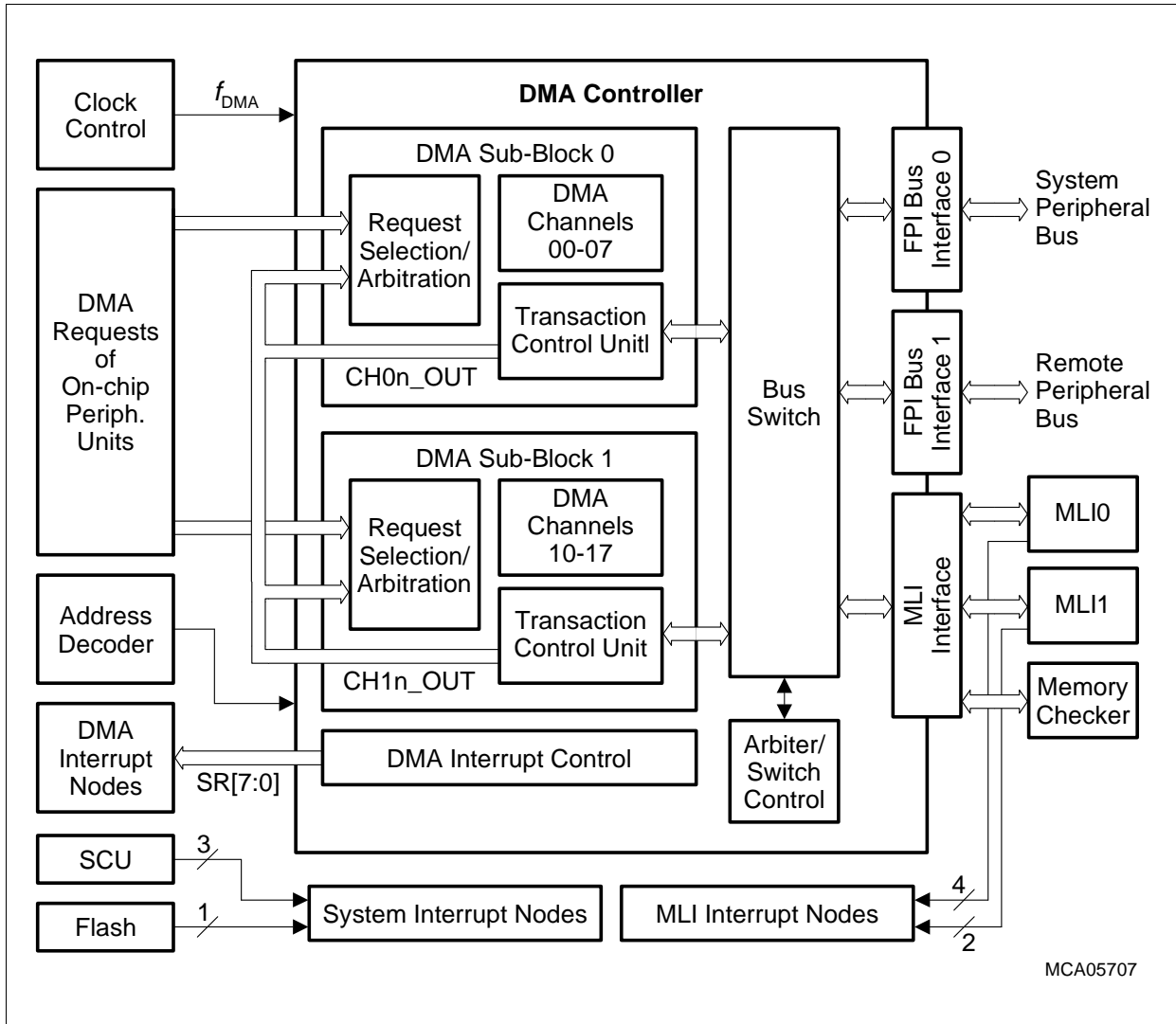


Figure 12-30 DMA Module Implementation and Interconnections

The request sources of the peripheral modules (ADC0/1, MSC0/1, MLI0/1, FADC, MultiCAN, and SCU) are associated with Interrupt Node Pointers and individual interrupt enable bits. As a result, each of the internal requests of a module can be routed independently to any of the interrupt output lines (INT_Ox) of the module. The modules ASC0/1 and SSC0/1 have interrupt output lines that are directly connected to a specific request source.

Direct Memory Access Controller

12.3.1 DMA Request Wiring Matrix

The DMA request input lines of each DMA channel within DMA Sub-Block 0 and 1 are connected to request output lines from the peripheral modules according [Table 12-9](#) and [Table 12-10](#).

Table 12-9 DMA Request Assignment for DMA Sub-Block 0

DMA Channel	DMA Request Line	DMA Requesting Unit	Selected by
00	CH17_OUT	DMA channel 17	CHCR00.PRSEL = 000 _B
	SCU_REQ0	Ext. Request Unit ¹⁾	CHCR00.PRSEL = 001 _B
	TIR	SSC0	CHCR00.PRSEL = 010 _B
	RIR	SSC0	CHCR00.PRSEL = 011 _B
	TIR	SSC1	CHCR00.PRSEL = 100 _B
	SR4	MLI0	CHCR00.PRSEL = 101 _B
	SR4	MLI1	CHCR00.PRSEL = 110 _B
	not connected (0)		CHCR00.PRSEL = 111 _B
01	CH00_OUT	DMA channel 00	CHCR01.PRSEL = 000 _B
	SCU_REQ1	Ext. Request Unit ¹⁾	CHCR01.PRSEL = 001 _B
	TIR	SSC1	CHCR01.PRSEL = 010 _B
	RIR	SSC1	CHCR01.PRSEL = 011 _B
	TIR	SSC0	CHCR01.PRSEL = 100 _B
	SR5	MLI0	CHCR01.PRSEL = 101 _B
	SR5	MLI1	CHCR01.PRSEL = 110 _B
	not connected (0)		CHCR01.PRSEL = 111 _B
02	CH01_OUT	DMA channel 01	CHCR02.PRSEL = 000 _B
	SCU_REQ2	Ext. Request Unit ¹⁾	CHCR02.PRSEL = 001 _B
	SR4	ADC0	CHCR02.PRSEL = 010 _B
	SR4	ADC1	CHCR02.PRSEL = 011 _B
	SR4	MLI0	CHCR02.PRSEL = 100 _B
	SR4	MLI1	CHCR02.PRSEL = 101 _B
	TIR	SSC0	CHCR02.PRSEL = 110 _B
	not connected (0)		CHCR02.PRSEL = 111 _B

Direct Memory Access Controller

Table 12-9 DMA Request Assignment for DMA Sub-Block 0 (cont'd)

DMA Channel	DMA Request Line	DMA Requesting Unit	Selected by
03	CH02_OUT	DMA channel 02	CHCR03.PRSEL = 000 _B
	SCU_REQ3	Ext. Request Unit ¹⁾	CHCR03.PRSEL = 001 _B
	SR5	ADC0	CHCR03.PRSEL = 010 _B
	SR5	ADC1	CHCR03.PRSEL = 011 _B
	SR5	MLI0	CHCR03.PRSEL = 100 _B
	SR5	MLI1	CHCR03.PRSEL = 101 _B
	TIR	SSC1	CHCR03.PRSEL = 110 _B
	not connected (0)		
04	CH03_OUT	DMA channel 03	CHCR04.PRSEL = 000 _B
	SCU_REQ0	Ext. Request Unit ¹⁾	CHCR04.PRSEL = 001 _B
	TIR	ASC0	CHCR04.PRSEL = 010 _B
	RIR	ASC0	CHCR04.PRSEL = 011 _B
	TIR	ASC1	CHCR04.PRSEL = 100 _B
	INT_O0	MultiCAN	CHCR04.PRSEL = 101 _B
	TIR	SSC0	CHCR04.PRSEL = 110 _B
	not connected (0)		
05	CH04_OUT	DMA channel 04	CHCR05.PRSEL = 000 _B
	SCU_REQ1	Ext. Request Unit ¹⁾	CHCR05.PRSEL = 001 _B
	TIR	ASC1	CHCR05.PRSEL = 010 _B
	RIR	ASC1	CHCR05.PRSEL = 011 _B
	TIR	ASC0	CHCR05.PRSEL = 100 _B
	INT_O0	MultiCAN	CHCR05.PRSEL = 101 _B
	RIR	SSC0	CHCR05.PRSEL = 110 _B
	not connected (0)		

Direct Memory Access Controller

Table 12-9 DMA Request Assignment for DMA Sub-Block 0 (cont'd)

DMA Channel	DMA Request Line	DMA Requesting Unit	Selected by
06	CH05_OUT	DMA channel 05	CHCR06.PRSEL = 000 _B
	SCU_REQ2	Ext. Request Unit ¹⁾	CHCR06.PRSEL = 001 _B
	SR2	MSC0	CHCR06.PRSEL = 010 _B
	SR2	MSC1	CHCR06.PRSEL = 011 _B
	SR6	MLI0	CHCR06.PRSEL = 100 _B
	SR6	MLI1	CHCR06.PRSEL = 101 _B
	INT_O1	MultiCAN	CHCR06.PRSEL = 110 _B
	not connected (0)		CHCR06.PRSEL = 111 _B
07	CH06_OUT	DMA channel 06	CHCR07.PRSEL = 000 _B
	SCU_REQ3	Ext. Request Unit ¹⁾	CHCR07.PRSEL = 001 _B
	SR3	MSC0	CHCR07.PRSEL = 010 _B
	SR3	MSC1	CHCR07.PRSEL = 011 _B
	SR7	MLI0	CHCR07.PRSEL = 100 _B
	SR7	MLI1	CHCR07.PRSEL = 101 _B
	INT_O2	MultiCAN	CHCR07.PRSEL = 110 _B
	not connected (0)		CHCR07.PRSEL = 111 _B

1) The external request unit located in the System Control Unit handles DMA requests coming from GPTA and external inputs.

Direct Memory Access Controller

Table 12-10 DMA Request Assignment for DMA Sub-Block 1

DMA Channel	DMA Request Source	DMA Request Unit	Selected by
10	CH07_OUT	DMA channel 07	CHCR10.PRSEL = 000 _B
	SCU_REQ0	Ext. Request Unit ¹⁾	CHCR10.PRSEL = 001 _B
	SR0	FADC	CHCR10.PRSEL = 010 _B
	SR6	ADC0	CHCR10.PRSEL = 011 _B
	SR6	ADC1	CHCR10.PRSEL = 100 _B
	SR4	MLI0	CHCR10.PRSEL = 101 _B
	SR4	MLI1	CHCR10.PRSEL = 110 _B
	not connected (0)		
11	CH10_OUT	DMA channel 10	CHCR11.PRSEL = 000 _B
	SCU_REQ1	Ext. Request Unit ¹⁾	CHCR11.PRSEL = 001 _B
	SR1	FADC	CHCR11.PRSEL = 010 _B
	SR7	ADC0	CHCR11.PRSEL = 011 _B
	SR7	ADC1	CHCR11.PRSEL = 100 _B
	SR5	MLI0	CHCR11.PRSEL = 101 _B
	SR5	MLI1	CHCR11.PRSEL = 110 _B
	not connected (0)		
12	CH11_OUT	DMA channel 11	CHCR12.PRSEL = 000 _B
	SCU_REQ2	Ext. Request Unit ¹⁾	CHCR12.PRSEL = 001 _B
	SR2	FADC	CHCR12.PRSEL = 010 _B
	SR4	ADC0	CHCR12.PRSEL = 011 _B
	SR4	ADC1	CHCR12.PRSEL = 100 _B
	SR6	MLI0	CHCR12.PRSEL = 101 _B
	SR6	MLI1	CHCR12.PRSEL = 110 _B
	not connected (0)		

Direct Memory Access Controller

Table 12-10 DMA Request Assignment for DMA Sub-Block 1 (cont'd)

DMA Channel	DMA Request Source	DMA Request Unit	Selected by
13	CH12_OUT	DMA channel 12	CHCR13.PRSEL = 000 _B
	SCU_REQ3	Ext. Request Unit ¹⁾	CHCR13.PRSEL = 001 _B
	SR3	FADC	CHCR13.PRSEL = 010 _B
	SR5	ADC0	CHCR13.PRSEL = 011 _B
	SR5	ADC1	CHCR13.PRSEL = 100 _B
	SR7	MLI0	CHCR13.PRSEL = 101 _B
	SR7	MLI1	CHCR13.PRSEL = 110 _B
	not connected (0)		
14	CH13_OUT	DMA channel 13	CHCR14.PRSEL = 000 _B
	SCU_REQ0	Ext. Request Unit ¹⁾	CHCR14.PRSEL = 001 _B
	SR0	FADC	CHCR14.PRSEL = 010 _B
	SR6	ADC0	CHCR14.PRSEL = 011 _B
	SR6	ADC1	CHCR14.PRSEL = 100 _B
	SR3	MSC0	CHCR14.PRSEL = 101 _B
	SR3	MSC1	CHCR14.PRSEL = 110 _B
	not connected (0)		
15	CH14_OUT	DMA channel 14	CHCR15.PRSEL = 000 _B
	SCU_REQ1	Ext. Request Unit ¹⁾	CHCR15.PRSEL = 001 _B
	SR1	FADC	CHCR15.PRSEL = 010 _B
	SR7	ADC0	CHCR15.PRSEL = 011 _B
	SR7	ADC1	CHCR15.PRSEL = 100 _B
	INT_O3	MultiCAN	CHCR15.PRSEL = 101 _B
	RIR	SSC1	CHCR15.PRSEL = 110 _B
	not connected (0)		

Direct Memory Access Controller

Table 12-10 DMA Request Assignment for DMA Sub-Block 1 (cont'd)

DMA Channel	DMA Request Source	DMA Request Unit	Selected by
16	CH15_OUT	DMA channel 15	CHCR16.PRSEL = 000 _B
	SCU_REQ2	Ext. Request Unit ¹⁾	CHCR16.PRSEL = 001 _B
	SR2	FADC	CHCR16.PRSEL = 010 _B
	SR6	MLI0	CHCR16.PRSEL = 011 _B
	SR6	MLI1	CHCR16.PRSEL = 100 _B
	INT_O0	MultiCAN	CHCR16.PRSEL = 101 _B
	SR3	MSC0	CHCR16.PRSEL = 110 _B
	not connected (0)		CHCR16.PRSEL = 111 _B
17	CH16_OUT	DMA channel 16	CHCR17.PRSEL = 000 _B
	SCU_REQ3	Ext. Request Unit ¹⁾	CHCR17.PRSEL = 001 _B
	SR3	FADC	CHCR17.PRSEL = 010 _B
	SR7	MLI0	CHCR17.PRSEL = 011 _B
	SR7	MLI1	CHCR17.PRSEL = 100 _B
	INT_O1	MultiCAN	CHCR17.PRSEL = 101 _B
	SR3	MSC1	CHCR17.PRSEL = 110 _B
	not connected (0)		CHCR17.PRSEL = 111 _B

1) The external request unit located in the System Control Unit handles DMA requests coming from GPTA and external inputs.

Direct Memory Access Controller

12.3.2 Access Protection Assignment

DMA access protection as described on [Page 12-40](#) requires the assignment of 32 fixed address range. [Table 12-11](#) shows this address range assignment as implemented in the TC1796.

Table 12-11 DMA Access Protection Address Ranges

Access Protection Range			Related Module(s)
No. n	Enable Bit in MEmAENR	Selected Address Range	
0	AEN0	F000 0000 _H - F000 00FF _H F010 C200 _H - F010 C2FF _H	SCU, incl. WDT MEMCHK
1	AEN1	F000 0100 _H - F000 01FF _H F010 0000 _H - F010 00FF _H	SBCU RBCU
2	AEN2	F000 0200 _H - F000 02FF _H	STM
3	AEN3	F000 0400 _H - F000 04FF _H	OCDS
4	AEN4	F000 0800 _H to F000 08FF _H F000 0900 _H to F000 09FF _H	MSC0 MSC1
5	AEN5	F000 0A00 _H to F000 0AFF _H	ASC0
6	AEN6	F000 0B00 _H to F000 0BFF _H	ASC1
7	AEN7	F000 0C00 _H - F000 0FFF _H	P0, P1, P2, P3
8	AEN8	F000 1000 _H - F000 17FF _H	P4, P5, P6, P7, P8, P9, P10
9	AEN9	F000 1800 _H - F000 2FFF _H	GPTA0, GPTA1, LTCA2
10	AEN10	F000 3C00 _H - F000 3EFF _H	DMA
11	AEN11	F000 4000 _H - F000 5FFF _H	MultiCAN
12	AEN12	F004 0000 _H - F005 FFFF _H	PCP Registers, PCP Data Memory
13	AEN13	F006 0000 _H - F006 7FFF _H	PCP Code Memory
14	AEN14	F010 0100 _H - F010 01FF _H	SSC0
15	AEN15	F010 0200 _H - F010 02FF _H	SSC1
16	AEN16	F010 0300 _H - F010 03FF _H	FADC
17	AEN17	F010 0400 _H - F010 07FF _H	ADC0, ADC1
18	AEN18	F010 A000 _H - F010 BFFF _H	DPRAM
19	AEN19	F010 C000 _H - F010 C0FF _H F01E 0000 _H - F01E 7FFF _H F020 0000 _H - F023 FFFF _H	MLI0 Module, MLI0 Small TWs, MLI0 Large TWs

Direct Memory Access Controller

Table 12-11 DMA Access Protection Address Ranges (cont'd)

Access Protection Range			Related Module(s)
No. n	Enable Bit in MEmAENR	Selected Address Range	
20	AEN20	F010 C100 _H - F010 C1FF _H F01E 8000 _H - F01E FFFF _H F024 0000 _H - F027 FFFF _H	MLI1, MLI1 Small TWs, MLI1 Large TWs
21	AEN21	F7E0 FF00 _H - F7E0 FFFF _H F7E1 0000 _H - F7E1 FFFF _H F800 0500 _H - F87F FFFF _H	CPS, CPU SFRs & GPRs, PMU, Flash Regs, DMU, DBCUC, DMI, PMI, PBCUC, LFI
22	AEN22	F800 0000 _H - F800 03FF _H	EBU
23	AEN23	8000 0000 _H - 807F FFFF _H A000 0000 _H - A07F FFFF _H	Program Flash Space
24	AEN24	8080 0000 _H - 8FDF FFFF _H A080 0000 _H - AFDF FFFF _H	External EBU Space
25	AEN25	8FE0 0000 _H - 8FEF FFFF _H AFE0 0000 _H - AFEF FFFF _H	Data Flash Space
26	AEN26	8FF0 0000 _H - 8FFF BFFF _H AFF0 0000 _H - AFFF BFFF _H	Emulation Device Memory Space
27	AEN27	8FFF C000 _H - 8FFF FFFF _H AFFF C000 _H - AFFF FFFF _H	Boot ROM
28	AEN28	D800 0000 _H - DEFF FFFF _H E000 0000 _H - E7FF FFFF _H	External Peripherals & External Emulator
29	AEN29	E800 0000 _H - E83F FFFF _H	LMU Image (E80x translated to C00x)
30	AEN30	E840 0000 _H - E84F FFFF _H	DMI Image (E84x translated to D00x)
31	AEN31	E850 0000 _H - E85F FFFF _H	PMI Image (E85x translated to D40x)

Direct Memory Access Controller

In the TC1796, two internal memory areas are assigned for access protection using programmable address sub-ranges:

- 8-Kbyte Dual-Ported RAM (DPRAM), assigned as address range 18. The sub-ranges are controlled by bit fields MEmARR.SIZE0 and MEmARR.SLICE0.
- 64-Kbyte SRAM, assigned as address range 29. The sub-ranges are controlled by bit fields MEmARR.SIZE1 and MEmARR.SLICE1.

Bit fields SIZE0 and SLICE0 for the DPRAM sub-range access protection are coded as shown in [Table 12-12](#).

Table 12-12 DPRAM Address Protection Sub-Range Definitions

SIZE0	Sub-Ranges	SLICE0	Selected Address Range
000 _B	32 sub-ranges of 64 byte	00000 _B 00001 _B ... 11111 _B	F010 A000 _H - F010 A03F _H F010 A040 _H - F010 A07F _H ... F010 A7C0 _H - F010 A7FF _H F010 A800 _H - F010 BFFF _H is not selectable
001 _B	32 sub-ranges of 128 byte	00000 _B 00001 _B ... 11111 _B	F010 A000 _H - F010 A07F _H F010 A080 _H - F010 A0FF _H ... F010 AF80 _H - F010 AFFF _H F010 B000 _H - F010 BFFF _H is not selectable
010 _B	32 sub-ranges of 256 byte	00000 _B 00001 _B ... 11111 _B	F010 A000 _H - F010 A0FF _H F010 A100 _H - F010 A1FF _H ... F010 BF00 _H - F010 BFFF _H
011 _B	16 sub-ranges of 512 byte	X0000 _B X0001 _B ... X1111 _B	F010 A000 _H - F010 A1FF _H F010 A200 _H - F010 A3FF _H ... F010 BE00 _H - F010 BFFF _H
100 _B	8 sub-ranges of 1 Kbyte	XX000 _B XX001 _B ... XX111 _B	F010 A000 _H - F010 A3FF _H F010 A400 _H - F010 A7FF _H ... F010 BC00 _H - F010 BFFF _H
101 _B	4 sub-ranges of 2 Kbyte	XXX00 _B XXX01 _B XXX10 _B XXX11 _B	F010 A000 _H - F010 A7FF _H F010 A800 _H - F010 AFFF _H F010 B000 _H - F010 B7FF _H F010 B800 _H - F010 BFFF _H

Direct Memory Access Controller

Table 12-12 DPRAM Address Protection Sub-Range Definitions (cont'd)

SIZE0	Sub-Ranges	SLICE0	Selected Address Range
110 _B	2 sub-ranges of 4 Kbyte	XXXX0 _B XXXX1 _B	F010 A000 _H - F010 AFFF _H F010 B000 _H - F010 BFFF _H
111 _B	8 Kbyte	XXXXX _B	F010 A000 _H - F010 BFFF _H

Bit fields SIZE1 and SLICE1 for the SRAM sub-range access protection (with address translation from E800 to C000) are coded as shown in [Table 12-13](#).

Table 12-13 SRAM Address Protection Sub-Range Definitions

SIZE1	Sub-Ranges	SLICE1	Selected Address Range
000 _B	32 sub-ranges of 512 byte	00000 _B 00001 _B ... 11111 _B	E800 0000 _H - E800 01FF _H E800 0200 _H - E800 03FF _H ... E800 3E00 _H - E800 3FFF _H E800 4000 _H - E800 FFFF _H is not selectable
001 _B	32 sub-ranges of 1 Kbyte	00000 _B 00001 _B ... 11111 _B	E800 0000 _H - E800 03FF _H E800 0400 _H - E800 07FF _H ... E800 7C00 _H - E800 7FFF _H E800 8000 _H - E800 FFFF _H is not selectable
010 _B	32 sub-ranges of 2 Kbyte	00000 _B 00001 _B ... 11111 _B	E800 0000 _H - E800 07FF _H E800 0800 _H - E800 0FFF _H ... E800 F800 _H - E800 FFFF _H
011 _B	16 sub-ranges of 4 Kbyte	X0000 _B X0001 _B ... X1111 _B	E800 0000 _H - E800 0FFF _H E800 1000 _H - E800 1FFF _H ... E800 F000 _H - E800 FFFF _H
100 _B	8 sub-ranges of 8 Kbyte	XX000 _B XX001 _B ... XX111 _B	E800 0000 _H - E800 1FFF _H E800 2000 _H - E800 3FFF _H ... E800 E000 _H - E800 FFFF _H
101 _B	4 sub-ranges of 16 Kbyte	XXX00 _B XXX01 _B XXX10 _B XXX11 _B	E800 0000 _H - E800 3FFF _H E800 4000 _H - E800 7FFF _H E800 8000 _H - E800 BFFF _H E800 C000 _H - E800 FFFF _H

Direct Memory Access Controller

Table 12-13 SRAM Address Protection Sub-Range Definitions (cont'd)

SIZE1	Sub-Ranges	SLICE1	Selected Address Range
110 _B	2 sub-ranges of 32 Kbyte	XXXX0 _B XXXX1 _B	E800 0000 _H - E800 7FFF _H E800 8000 _H - E800 FFFF _H
111 _B	64 Kbyte	XXXXX _B	E800 0000 _H - E800 FFFF _H

Note: Sub-ranges 2 and 3 with bit fields MEmARR.SIZE2, MEmARR.SLICE2, MEmARR.SIZE3, and MEmARR.SLICE3 are not implemented.

Note: The 16 Kbyte stand-by RAM (SBRAM) located at addresses C03FC000_H to C03FFFFFF_H cannot be accessed by the DMA controller even if the corresponding enable bit MEmAENR.AEN29 is set.

12.3.3 Implementation-specific DMA Registers

The DMA controller as implemented in the TC1796 contains the following additional registers:

- DMA clock control register
- Service request control registers for DMA controller interrupts (DMA_SRCx)
- Service request control registers for MLI module interrupts (DMA_MLIySRC.x)
- Service request control registers for system related interrupts (DMA_SYSSRCx)

Figure 12-32 is an overview of these registers.

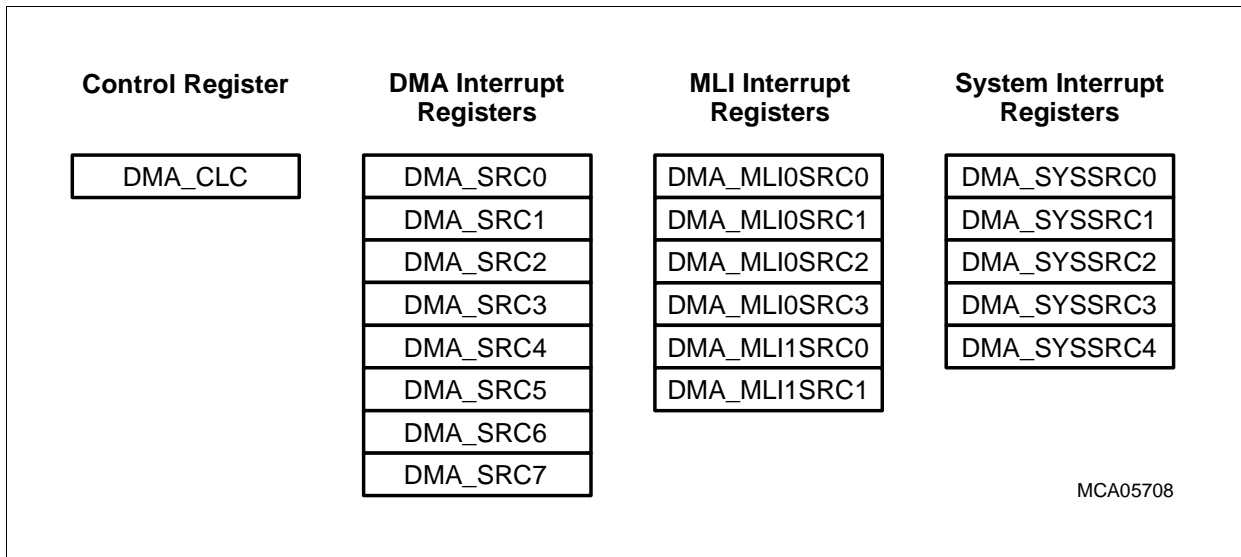


Figure 12-31 DMA Implementation-specific Registers

Note: Further details on interrupt handling and processing are described in the “Interrupt System” chapter of the TC1796 System Units User’s Manual.

The clock generation and interrupt control configuration as implemented in the DMA controller module is shown in **Figure 12-32**.

The DMA controller and the two MLI modules (MLI0 and MLI1) are supplied from a common module clock f_{DMA} that has the frequency of the system clock f_{SYS} and is controlled via the DMA_CLC clock control register. The MLI modules do not have their own clock control registers. Their input clock is derived from the DMA clock divided by separate fractional divider registers.

The control of the suspend and break features is done independently inside each module, except for the hard-suspend feature of the DMA (clock switch-off). This hard-suspend feature should only be used if the system is reset after the suspend state is left. In Hard-suspend Mode, the FPI Bus interface 1 (RPB) can not be accessed, because the DMA Bus Switch is no longer clocked.

The DMA controller contains in total 18 interrupt request nodes with its interrupt service request control registers:

Direct Memory Access Controller

- Eight interrupt requests $SR[7:0] = INT_O[7:0]$ from the DMA controller; upper eight interrupt requests of the DMA controller $INT_O[15:8]$ are not connected.
- Four interrupt requests $SR[3:0] = INT_O[3:0]$ from the MLI0 module; upper four interrupt requests of the MLI0 module $INT_O[7:4]$ are not connected.
- Two interrupt requests $SR[1:0] = INT_O[1:0]$ from the MLI1 module; upper six interrupt requests of the MLI0 module $INT_O[7:2]$ are not connected.
- Four system interrupts $SR[3:0]$ from the SCU.

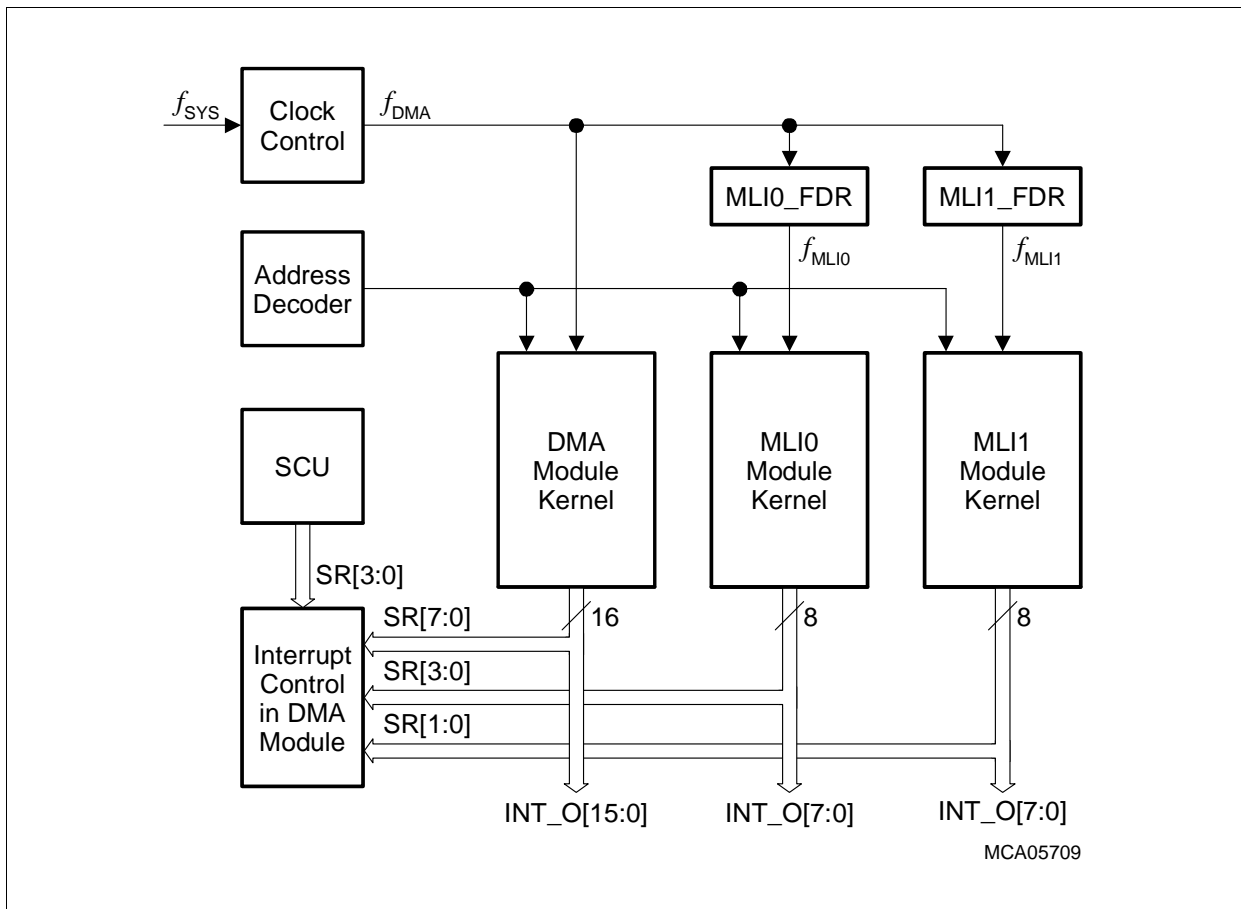


Figure 12-32 Implementation of the DMA Module and the MLI Modules

Direct Memory Access Controller

12.3.3.1 Clock Control Register

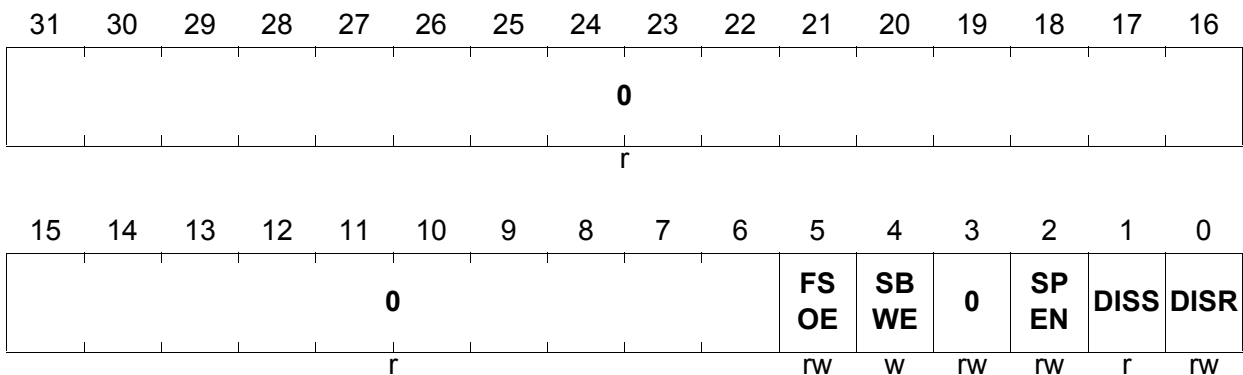
The Clock Control Register controls the f_{DMA} module clock signal. This clock is also used for the MLI modules as a common clock that can be individually divided for the MLI modules.

DMA_CLC

DMA Clock Control Register

(000_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
DISR	0	rw	Module Disable Request Bit Used for enable/disable control of the module
DISS	1	r	Module Disable Status Bit Bit indicates the current status of the module
SPEN	2	rw	Module Suspend Enable for OCDS Used to enable the suspend mode.
0	3	rw	Reserved; Returns 0 if read; <u>must</u> be written with 0.
SBWE	4	w	Module Suspend Bit Write Enable for OCDS Determines whether SPEN and FSOE are write-protected.
FSOE	5	rw	Fast Switch Off Enable Used for fast clock switch off in OCSD Suspend Mode.
0	[31:6]	r	Reserved Read as 0; should be written with 0.

Note: After a hardware reset operation, the DMA module is enabled.

Note: The Suspend Mode does not modify any of the registers.

Direct Memory Access Controller

12.3.3.2 DMA Service Request Control Registers

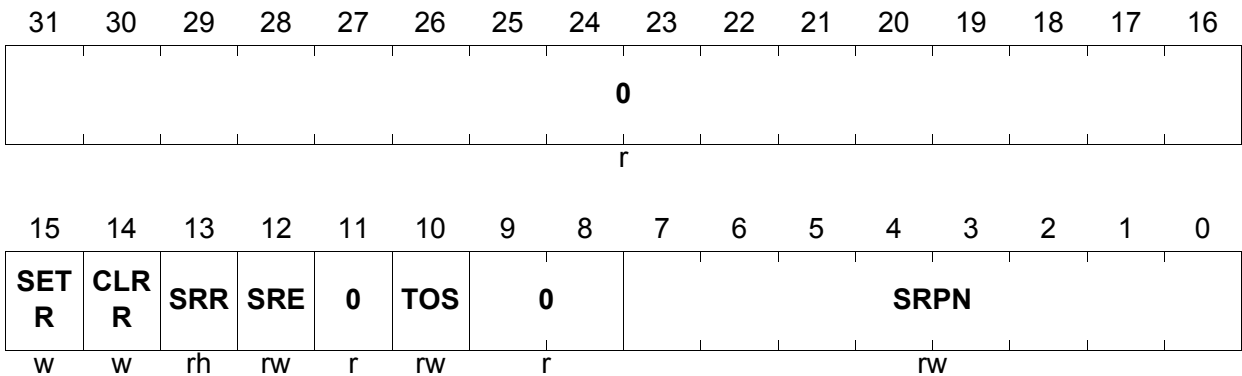
In the TC1796, only the lower eight DMA controller interrupts SR[7:0] are connected to service request control registers. The upper eight DMA controller interrupt outputs SR[15:8] are not used and not connected.

DMA_SRCx (x = 0-7)

DMA Service Request Control Register x

(2FC_H-x*4_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SRPN	[7:0]	rw	Service Request Priority Number
TOS	10	rw	Type of Service Control
SRE	12	rw	Service Request Enable
SRR	13	rh	Service Request Flag
CLRR	14	w	Request Clear Bit
SETR	15	w	Request Set Bit
0	[9:8], 11, [31:16]	r	Reserved Read as 0; should be written with 0.

Direct Memory Access Controller

12.3.3.3 MLI Service Request Control Registers

The Service Request Control Registers of the MLI modules are located inside the DMA address area, because the MLI modules do not have own FPI Bus interfaces. They share one FPI Bus interface with the DMA controller.

The MLI0 module has eight interrupt output lines. Only four of them [3:0] are controlled by the MLI0 service request registers. The MLI1 module has also eight interrupt output lines, but only two of them [1:0] are controlled by to the MLI1 service request registers.

DMA_MLI0SRCx (x = 0-3)

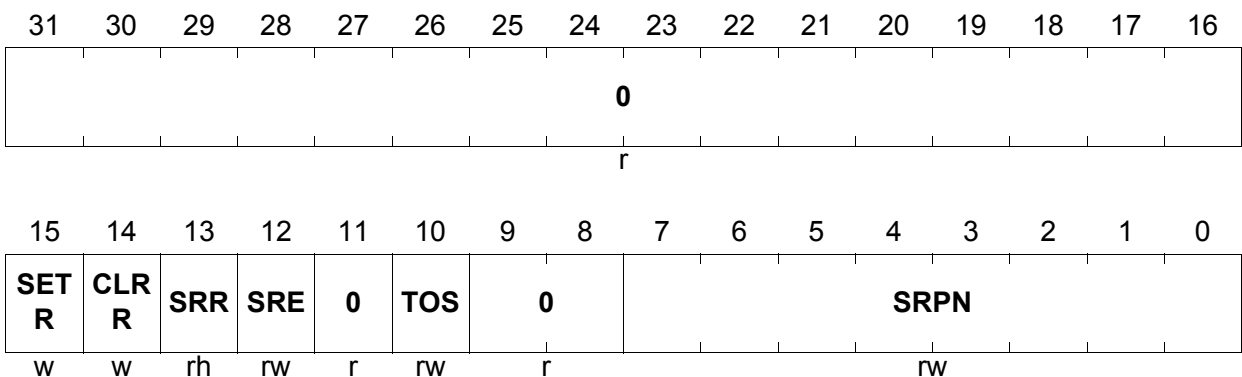
DMA MLI0 Service Request Control Register x
(2AC_H-x*4_H)

Reset Value: 0000 0000_H

DMA_MLI1SRCx (x = 0-1)

DMA MLI1 Service Request Control Register x
(2BC_H-x*4_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SRPN	[7:0]	rw	Service Request Priority Number
TOS	10	rw	Type of Service Control
SRE	12	rw	Service Request Enable
SRR	13	rh	Service Request Flag
CLRR	14	w	Request Clear Bit
SETR	15	w	Request Set Bit
0	[9:8], 11, [31:16]	r	Reserved Read as 0; should be written with 0.

Note: The bit coding of the MLI0/MLI1 service request registers is identical to that of the DMA Service Request Control Registers shown on the previous page.

Direct Memory Access Controller

12.3.3.4 System Interrupt Service Request Control Register

System interrupts of other on-chip modules are controlled by five system service request control registers which are located in the DMA address space. The five system service request registers are used according [Table 12-14](#).

Table 12-14 Assignment of System Interrupt Nodes

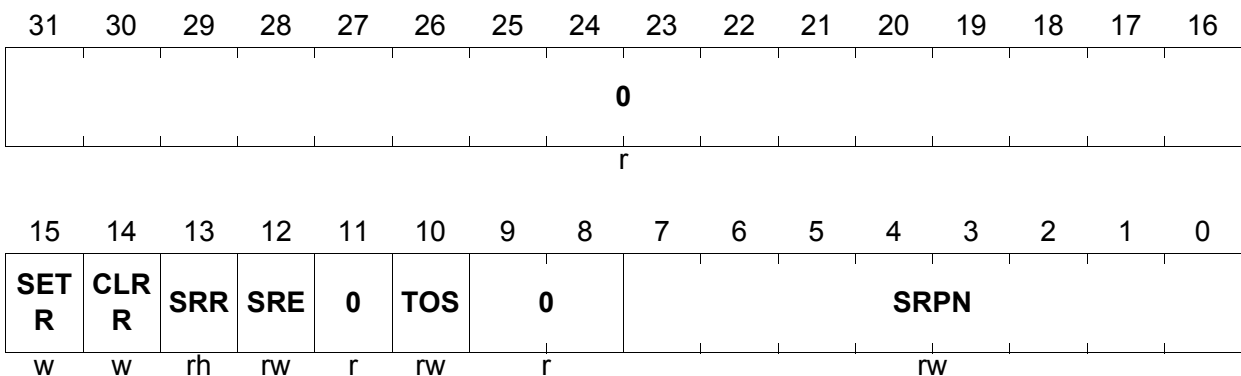
System Service Request Node	Connected to System Interrupt Source
DMA_SYSSCR0	Floating Point Unit (FPU) interrupt (see also Page 5-35)
DMA_SYSSCR1	Flash module interrupt (see also Page 7-36)
DMA_SYSSCR2	External Interrupt 0 (see also Page 5-36)
DMA_SYSSCR3	External Interrupt 1 (see also Page 5-36)
DMA_SYSSCR4	None; can be used as software interrupt

DMA_SYSSRCx (x = 0-4)

DMA System Interrupt Service Request Control Register x

(29C_H-x*4_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SRPN	[7:0]	rw	Service Request Priority Number
TOS	10	rw	Type of Service Control
SRE	12	rw	Service Request Enable
SRR	13	rh	Service Request Flag
CLRR	14	w	Request Clear Bit
SETR	15	w	Request Set Bit

Direct Memory Access Controller

Field	Bits	Type	Description
0	[9:8], 11, [31:16]	r	Reserved Read as 0; should be written with 0.

Note: The bit coding of the system interrupt service request control registers is identical to that of the DMA service request control registers shown two pages before.

12.3.4 DMA Controller Address Map

The DMA controller register block address map is shown in [Figure 12-33](#). It shows how the different register blocks are arranged, and adds some absolute address information.

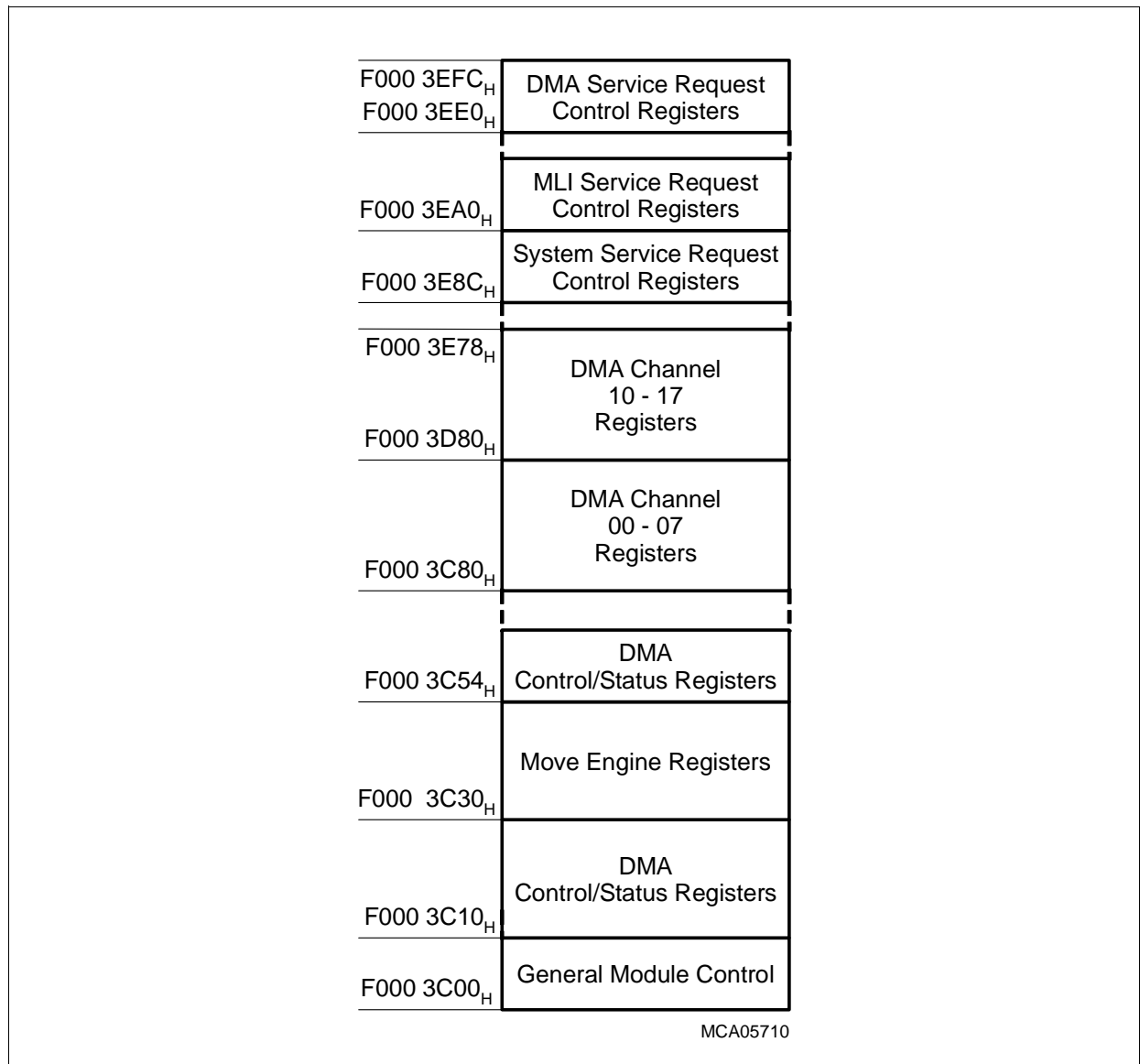


Figure 12-33 DMA Controller Register Block Address Map

12.4 Memory Checker Module

12.4.1 Functional Description

The Memory Checker Module (MCHK) makes it possible to check the data consistency of memories. Any SPB bus master may access the memory checker. Preferable the DMA controller does it as described hereafter. It uses 8-bit, 16-bit, or 32-bit DMA moves to read from the selected address area and to write the value read in a memory checker input register. With each write operation to the memory checker input register, a polynomial checksum calculation is triggered and the result of the calculation is stored in the memory checker result register.

In order to start a memory check sequence, the memory checker result register must be initialized (e.g. written with $FFFF_H$ or with a desired start value) and a DMA transaction has to be set up (start address, length, etc.). When programming the DMA channel for the memory checker with $CHCRmn.RROAT = 1$, one DMA transfer request (software or hardware triggered) starts the DMA transaction.

At the read move operations of the DMA transaction, data is always read from the memory and then written into the memory checker input register for the polynomial checksum calculation. At the end of the transaction ($CHSRmn.TCOUNT = 0$), an interrupt can be generated by the DMA channel (if $CHCRmn.RROAT = 1$), and the memory checker result register can be read out by software.

The memory checker uses the standard Ethernet polynomial, which is given by:

$$G^{32} = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \quad (12.1)$$

Note: Although the polynomial above is used for generation, the generation algorithm differs from the one that is used by the Ethernet protocol.

12.4.2 Registers

This section describes the kernel registers of the Memory Checker module.

MCHK Register Overview

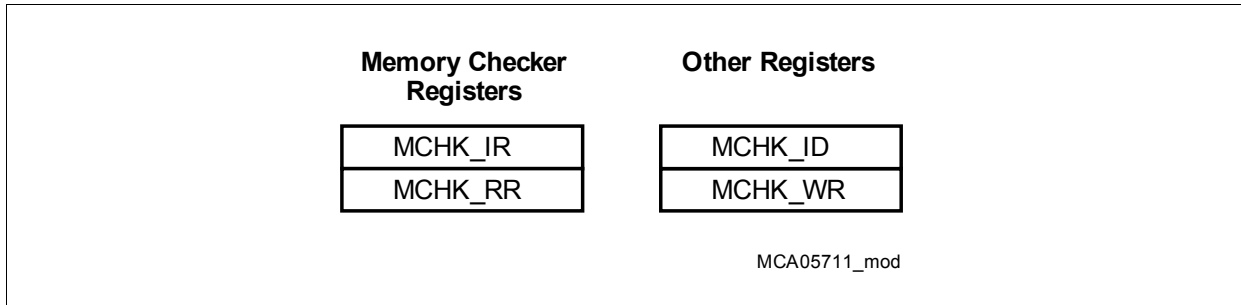


Figure 12-34 Memory Checker Registers

The complete and detailed address map of the of the Memory Checker module is described in [Table 18-31](#) on [Page 18-103](#) of the TC1796 User's Manual System Units part (Volume 1).

Table 12-15 Registers Address Space - Memory Checker

Module	Base Address	End Address	Note
MCHK	F010 C200 _H	F010 C2FF _H	–

Table 12-16 Registers Overview - Memory Checker egisters

Register Short Name	Register Long Name	Offset Address ¹⁾	Description see
MCHK_ID	Memory Checker Module Identification Reg.	0008 _H	Page 12-112
MCHK_IR	Memory Checker Input Register	0010 _H	Page 12-113
MCHK_RR	Memory Checker Result Register	0014 _H	Page 12-113
MCHK_WR	Memory Checker Write Register	0020 _H	Page 12-114

1) The absolute register address is calculated as follows:
Module Base Address ([Table 12-5](#)) + Offset Address (shown in this column)

Direct Memory Access Controller

12.4.2.1 Memory Checker Registers

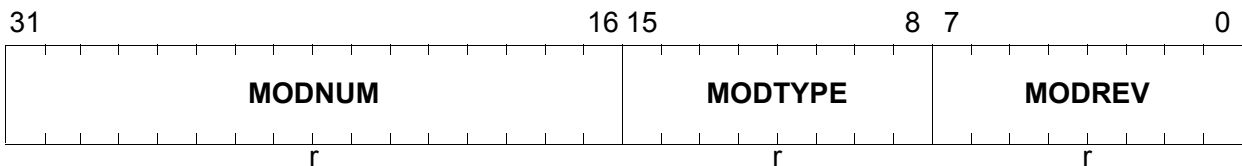
The MCHK Module Identification Register ID contains read-only information about the MCHK module version.

MCHK_ID

Memory Checker Module Identification Register

(008_H)

Reset Value: 001B C0XX_H



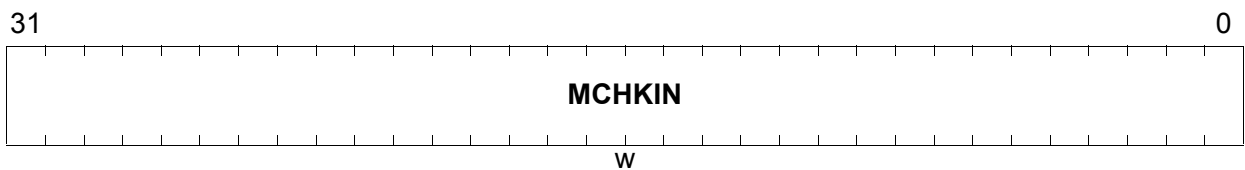
Field	Bits	Type	Description
MODREV	[7:0]	r	Module Revision Number MODREV defines the module revision number. The value of a module revision starts with 01 _H (first revision).
MODTYPE	[15:8]	r	Module Type This bit field defines the module as a 32-bit module: C0 _H
MODNUM	[31:16]	r	Module Number Value This bit field defines the module identification number for the MCHK: 001B _H

Direct Memory Access Controller

The Memory Checker Input Register is used during write moves of a memory checker related DMA transaction as data destination with its fixed register address. If the DMA moves to register MCHK_IR are 8-bit or 16-bit wide, the unused register bits of the 32-bit MCHKIN value are taken as 0s for the current result calculation.

MCHK_IR

Memory Checker Input Register (10_H) Reset Value: 0000 0000_H



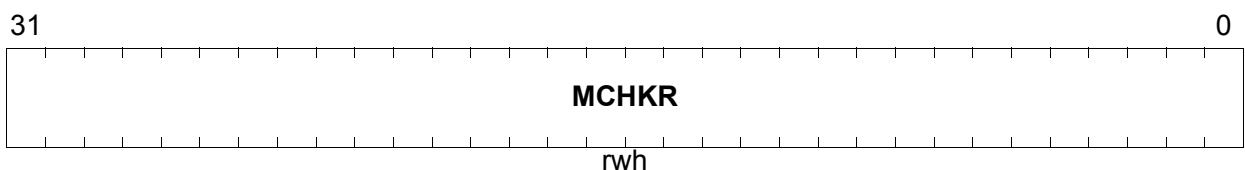
Field	Bits	Type	Description
MCHKIN	[31:0]	w	Memory Checker Input The value written to MCHKIN is used for the next checksum calculation. Any read action will deliver 0.

Note: MCHK_IR is a write-only register. Any read action will deliver 0000 0000_H.

The Memory Checker Result Register contains the result of the memory check operation. Before starting a checksum calculation operation, it should be written with an initial checksum calculation value.

MCHK_RR

Memory Checker Result Register (14_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
MCHKR	[31:0]	rwh	Memory Checker Result This bit field contains the current result of the memory checksum calculation operation.

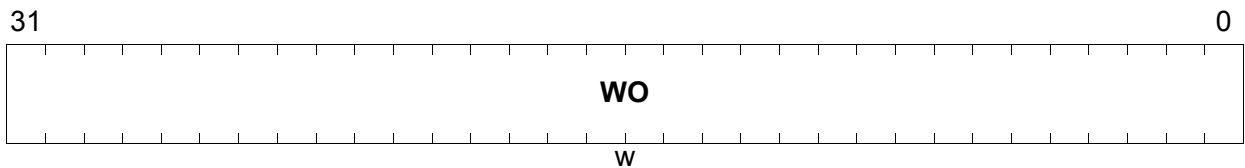
Direct Memory Access Controller

The Memory Checker Write Register is a dummy write-only register that is located within the memory checker address range. This register is near by (above) the MLI address range of the TC1796. It has no meaning for memory checker operation.

The Memory Checker Write Register can be used as dummy write register at the write back action of the DMA or MLI controller Move Engines when the pattern detection feature of the DMA controller is used. Accessing MCHK_WR with the Move Engines of the MLI or DMA controller via the Bus Switch of the DMA controller (see [Figure 12-14](#)) does not request the two FPI buses of the TC1796, SPB and RPB, because it is near the MLI modules address ranges.

MCHK_WR

Memory Checker Write Register (20_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
WO	[31:0]	w	Write-Only This write-only bit field is used to write dummy data during DMA pattern detection. The data written to WO is not taken into account for any action. Any read action of WO will deliver 0000 0000 _H .

13 LMB External Bus Unit

The External Bus Unit (EBU) of the TC1796 controls the transactions between external memories or peripheral units, and the internal memories and peripheral units. The basic interfaces of the EBU are shown in [Figure 13-1](#).

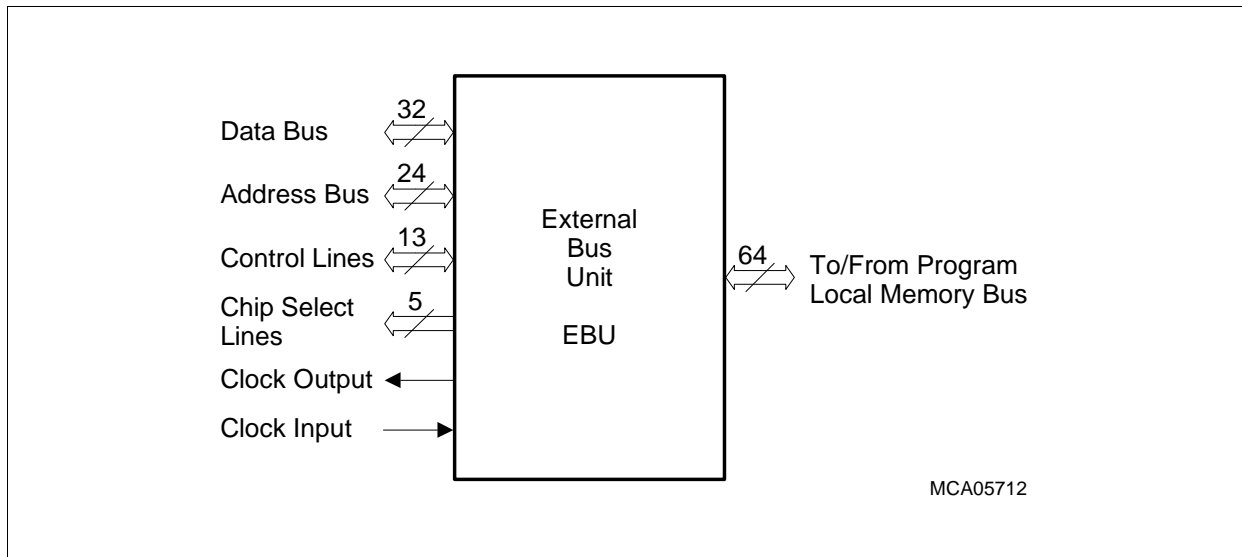


Figure 13-1 EBU Interface Diagram

The EBU is internally connected to the Program Local Memory Bus (PLMB). This configuration makes it possible to get fast access times, especially when using external Burst Flash memory devices.

The following features are supported by the EBU:

- 64-bit internal PLMB interface
- 32-bit external demultiplexed bus interface
 - Support for Intel-style and Motorola-style interface signals
 - Support for Burst Flash memory devices
 - Flexibly programmable access parameters
 - Programmable chip select lines
 - Arbitration support for external EBU bus master devices
 - Little-endian and Big-endian support.
- Scalable external bus frequency
 - Derived from PLMB frequency (f_{CPU}) divided by 2, 3, or 4
 - Maximum 75 MHz
- Data buffering supported
 - Code prefetch buffer
 - Read/write buffer

13.1 Block Diagram

Figure 13-2 shows the building blocks of the EBU.

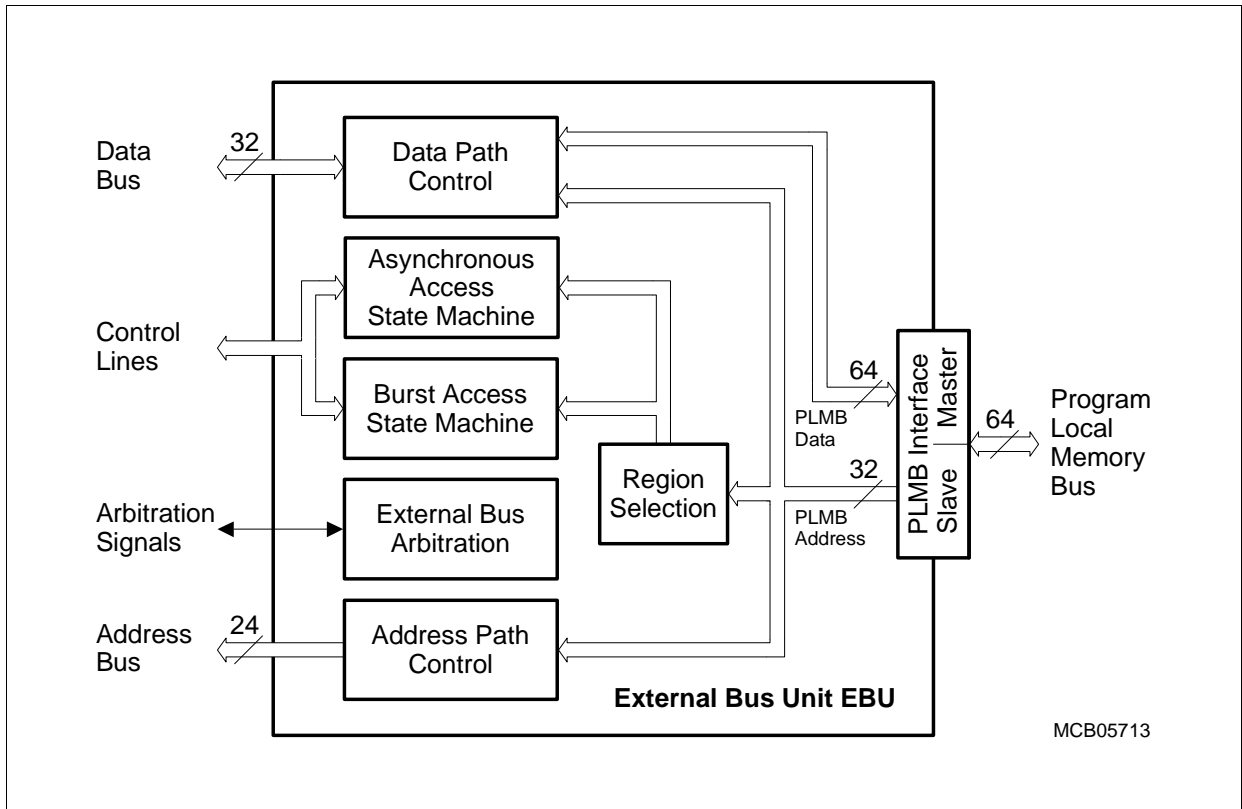


Figure 13-2 EBU Block Diagram

The EBU takes PLMB transactions and translates them into the appropriate external access(es). Two dedicated state machines control either asynchronous external accesses or Burst Mode external accesses. The EBU further contains blocks that control the external bus arbitration, the region selection, as well as the data and address paths.

13.2 EBU Interface Signals

The external EBU interface signals are listed in [Table 13-1](#) below.

Table 13-1 EBU Interface Signals

Signal/Pin	Type	Function
D[31:0]	O	Data bus lines 0-31
A[23:0]	O	Address bus lines 0-23
$\overline{\text{CS}}[3:0]$	O	Chip select 0-3
CSCOMB	O	Combined chip select for global select / emulator memory region select / emulator overlay select
BFCLKO	O	Burst Mode Flash clock output
BFCLKI	I	Burst Mode Flash clock input
$\overline{\text{RD}}$	O	Read control line
$\overline{\text{RD}}/\overline{\text{WR}}$	O	Write control line
$\overline{\text{ADV}}$	O	Address valid output
$\overline{\text{BC}}[3:0]$	O	Byte control lines 0-3
$\overline{\text{MR}}/\overline{\text{W}}$	O	Motorola style read/write control line
$\overline{\text{WAIT}}$	I	Wait input
$\overline{\text{BAA}}$	O	Burst address advance output
$\overline{\text{HOLD}}$	I	Hold request input
$\overline{\text{HLDA}}$	O	Hold acknowledge output
$\overline{\text{BREQ}}$	O	Bus request output
V_{DDEBU}	–	EBU power supply lines (2.3 - 3.3 V)

Note: After reset, all EBU pins (except BFCLKO and BFCLKI) have an internal pull-up device connected. These pull-up devices can be disconnected together by setting bit SCU_CON.EPUD.

13.2.1 Data Bus, D[31:0]

The data bus transfers data information. The width of this bus is 32 bits. External devices with 16 or 32 bits of data width can be connected to the data bus. Burst Mode instruction fetches can be performed with 16-bit or 32-bit data bus width.

The EBU adjusts the data on the data bus to the width of the external device, according to the programmed parameters in its control registers. The byte control signals $\overline{\text{BC}}[3:0]$ specify which parts of the data bus carry valid data.

Note: The TC1796 does not directly support for 8-bit data bus width. When 8-bit wide devices are used they must be arranged in pairs to implement either a 16-bit or a 32-bit wide memory region.

13.2.2 Address Bus, A[23:0]

The address bus of the EBU consists of 24 address output lines, giving a directly addressable range of 16 Mbytes. Directly addressable means that these address lines can be used to access any location within one external device, such as a memory. This external device can be selected via one of the chip select lines. Since there are four chip select lines, four such devices with up to 64 Mbytes of address range can be used in the external system.

13.2.3 Chip Selects, $\overline{\text{CS}}[3:0]$, $\overline{\text{CSCOMB}}$

The EBU provides five chip select outputs, $\overline{\text{CS}}0$, $\overline{\text{CS}}1$, $\overline{\text{CS}}2$, $\overline{\text{CS}}3$, and $\overline{\text{CSCOMB}}$. The address ranges for which these chip selects are generated are programmed separately for each chip select line in a very flexible way via the address select registers EBU_ADDRSELx. The combined chip select output $\overline{\text{CSCOMB}}$ can be typically used as a global select signal or for emulator memory selection purposes. More details are described on [Page 13-28](#).

13.2.4 Burst Flash Clock Output/Input, BFCLKO/BFCLKI

The clock output signal of the EBU is provided at pin BFCLKO. It is used for timing purposes (timing reference) during Burst Mode accesses. BFCLKO can be generated at Burst Mode accesses continuously or only during active accesses.

The clock input BFCLKI of the EBU is used to latch read data into the EBU. Normally BFCLKI is directly feedback and connected to BFCLKO. This feedback path can be configured externally to maximize the operating frequency for a given Flash device or to compensate the BFCLKO clock pad delay. More details are described on [Page 13-71](#) and [Page 13-72](#).

13.2.5 Read/Write Control Lines, RD, $\overline{\text{RD}}/\overline{\text{WR}}$ and $\overline{\text{MR}}/\overline{\text{W}}$

Two lines are provided to trigger the read ($\overline{\text{RD}}$) and write ($\overline{\text{RD}}/\overline{\text{WR}}$) operations of external devices. While some read/write devices require both signals, there are devices with only one control input. The $\overline{\text{RD}}/\overline{\text{WR}}$ line is then used for these devices. This line will go to an active-low level on a write, and will stay inactive high on a read. The external device should only evaluate this signal in conjunction with an active chip select. Thus, an active chip select in combination with a high level on the $\overline{\text{RD}}/\overline{\text{WR}}$ line indicates a read access to this device.

LMB External Bus Unit

The third read/write control line, $\overline{MR/\overline{W}}$, can be used to connect external devices directly to an asynchronous Motorola-style bus interface. $\overline{MR/\overline{W}}$ goes to a low level during a write cycle, and will stay at high during a read cycle.

13.2.6 Address Valid, \overline{ADV}

The address valid signal, \overline{ADV} , validates the address lines A[23:0]. It can be used to latch these addresses externally.

13.2.7 Byte Controls, $\overline{BC[3:0]}$

The byte control signals $\overline{BC[3:0]}$ select the appropriate byte lanes of the data bus for both read and write accesses. [Table 13-2](#) shows the activation on access to a 32-bit, 16-bit or 8-bit external device. Please note that this scheme supports little-endian devices.

Table 13-2 Byte Control Pin Usage

Width of External Device	$\overline{BC3}$	$\overline{BC2}$	$\overline{BC1}$	$\overline{BC0}$
32-bit device with byte write capability	D[31:24]	D[23:16]	D[15:8]	D[7:0]
16-bit device with byte write capability	inactive (high)	inactive (high)	D[15:8]	D[7:0]
8-bit device	inactive (high)	inactive (high)	inactive (high)	D[7:0]

The \overline{BCx} lines are not activated during Burst Mode cycles.

Signals \overline{BCx} can be programmed for different timing. The available modes cover a wide range of external devices, such as RAM with separate byte write-enable signals, and RAM with separate byte chip select signals. This allows external devices to connect without any external “glue” logic.

Table 13-3 Byte Control Signal Timing Options

Mode	EBU_BUSCONx. BCGEN	Description
Chip Select Mode	00 _B	\overline{BCx} signals have the same timing as the generated chip select \overline{CS} .
Control Mode	01 _B	\overline{BCx} signals have the same timing as the generated control signals \overline{RD} or $\overline{RD/\overline{WR}}$.
Write Enable Mode	10 _B	\overline{BCx} signals have the same timing as the generated control signal $\overline{RD/\overline{WR}}$.

13.2.8 Wait Input, $\overline{\text{WAIT}}$

This is an input signal to the EBU that is used to dynamically insert wait states into read or write data cycles controlled by the device on the external bus.

13.2.9 Burst Address Advance, $\overline{\text{BAA}}$

The Burst Address Advance output, $\overline{\text{BAA}}$, is provided for advancing the addresses during a Burst Flash access cycle.

13.2.10 Bus Arbitration Signals, $\overline{\text{HOLD}}$, $\overline{\text{HLDA}}$, and $\overline{\text{BREQ}}$

The $\overline{\text{HOLD}}$ input signal is the external bus arbitration signal that indicates to the EBU when an external bus master requests to obtain ownership of the external bus.

The $\overline{\text{HLDA}}$ output signal is the external bus arbitration signal that indicates to an external bus master that it has obtained ownership of the external bus from the EBU.

The $\overline{\text{BREQ}}$ output signal is the external bus arbitration signal that is asserted by the EBU when it requests to obtain back the ownership of the external bus.

13.2.11 EBU Power Supply

The V_{DDEBU} lines provide the power supply (2.3 - 3.3 V) for the dedicated pins of the EBU interface.

13.3 External Bus Arbitration

External bus arbitration is provided to allow the EBU to share its external bus with other master devices. This capability allows other external master devices to obtain ownership of the external bus, and to use the bus to access external devices connected to this bus. The scheme provided by the EBU is compatible with other TriCore and C166 devices and therefore allows the use of such devices as (external bus) masters together with the TC1796.

*Note: In this section, the term “external master” is used to denote a device which is located on the **external** bus and is capable of generating accesses across the external bus (i.e. is capable of driving the external bus). An external master is not able to access units that are located inside the TC1796.*

13.3.1 External Bus Modes

The EBU can operate in two bus modes on the external bus:

- Owner Mode
- Hold Mode

When in Owner Mode, the EBU operates as the master of the external bus. In other words, the EBU drives the external bus as required in order to access devices located on the external bus. While the EBU is in Owner Mode it is not possible for any other master to perform any accesses on the external bus.

During Hold Mode, the EBU tri-states the appropriate signal on the external bus in order to allow another external bus master to perform accesses on the external bus (i.e. to allow another master to drive the various external bus signals without contention with EBU).

13.3.2 Arbitration Signals and Parameters

The arbitration scheme consists of an external bus master that is responsible for controlling the allocation of the external bus. This master is referred to as the “Arbiter” within this document. The other external bus master (termed Participant within this document) requests ownership of the bus, and when necessary, from the Arbiter. The EBU can be programmed to operate either as an Arbiter or as a Participant (see [Page 13-10](#)). The following three lines are used by the EBU to arbitrate the external bus.

Table 13-4 EBU External Bus Arbitration Signals

Signal	Direction	Function
$\overline{\text{HOLD}}$	In	$\overline{\text{HOLD}}$ is asserted (low) by an external bus master when the external bus master requests to obtain ownership of the external bus from the EBU.
$\overline{\text{HLDA}}$	In/Out ¹⁾	$\overline{\text{HLDA}}$ is asserted (low) by the Arbiter to signal that the external bus is available for use by the Participant (i.e. the bus is not being used by the Arbiter). $\overline{\text{HLDA}}$ is sampled by the Participant to detect when it may use the external bus.
$\overline{\text{BREQ}}$	Out	$\overline{\text{BREQ}}$ is asserted (low) by the EBU when the EBU requests to obtain ownership of the external bus.

1) The direction of this signal depends upon the mode in which the EBU is operating (see [Page 13-10](#)).

Two components that are equipped with the EBU arbitration protocol can be directly connected together (without additional external logic) as shown below:

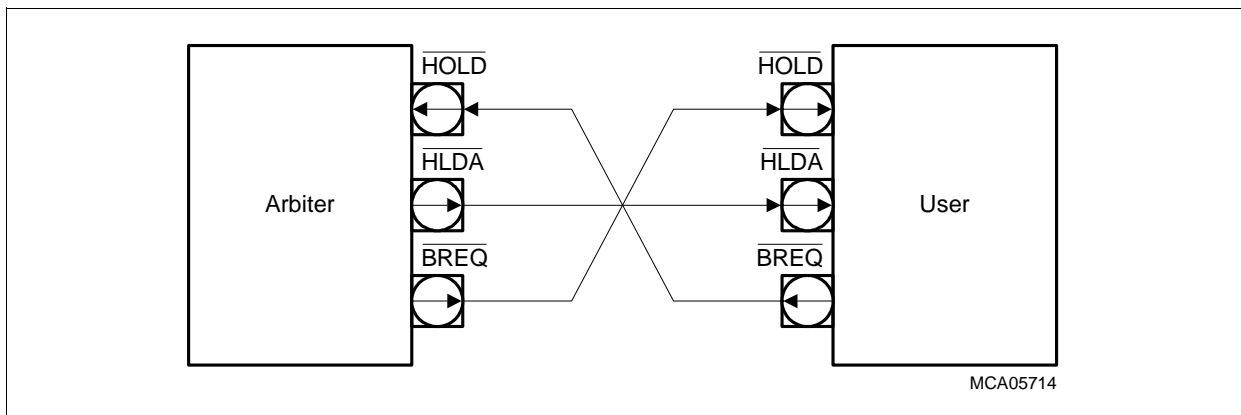


Figure 13-3 Connection of Bus Arbitration Signals

Note: In the example of [Figure 13-3](#), it is possible for the EBU to perform the function of either Arbiter or Participant (or indeed both the Arbiter and Participant may be the EBU).

[Table 13-5](#) lists the programmable parameters for the external bus arbitration.

Table 13-5 External Bus Arbitration Programmable Parameters

Parameter	Function	Description see
EBU_CON.ARBMODE	Arbitration mode selection	Page 13-83
EBU_CON.ARBSYNC	Arbitration input signal sampling control	
EBU_CON.EXTLOCK	External bus ownership locking control	
EBU_CON.TIMEOUTC	External bus time-out control	

13.3.3 Arbitration Modes

The arbitration mode of the EBU can be selected through configuration pins during reset (see Boot Selection Table on [Page 4-12](#)) or by programming the EBU_CON.ARBMODE bit field (see [Page 13-83](#)) after reset. Four different modes are available:

- No Bus Mode
- Sole Master Mode
- Arbiter Mode
- Participant Mode

13.3.3.1 No Bus Arbitration Mode

All accesses of the EBU to devices on the external bus are prohibited and will generate a PLMB bus error. The EBU operates in Hold Mode all the time. The $\overline{\text{HOLD}}$ and $\overline{\text{HLDA}}$ arbitration inputs are ignored and $\overline{\text{BREQ}}$ arbitration output remains at high (inactive) level.

No Bus Mode is selected by EBU_CON.ARBMODE = 00_B.

13.3.3.2 Sole Master Arbitration Mode

The EBU is the only master on the external bus; therefore no arbitration is necessary and the EBU has access to the external bus at any time. The EBU operates in Owner Mode all the time. The $\overline{\text{HOLD}}$ arbitration input is ignored, and the $\overline{\text{HLDA}}$ and $\overline{\text{BREQ}}$ arbitration outputs remain at high (inactive) level.

Sole Master Mode is selected by EBU_CON.ARBMODE = 11_B.

13.3.3.3 Arbiter Mode Arbitration Mode

The EBU is the default owner of the external bus (e.g. applicable when operating from external memory). Arbitration is performed if an external master (e.g. second TriCore) needs to access the external bus.

The EBU is cooperative in relinquishing ownership of the external bus while operating in Arbiter Mode. When the $\overline{\text{HOLD}}$ input is active, the EBU will generate a “retry” in response to any attempt to access the external bus from the internal PLMB. However, the EBU is aggressive in regaining ownership of the external bus while operating in Arbiter Mode. The EBU, having yielded ownership of the bus, will always request return of ownership even if there is no EBU external bus access pending.

Arbiter Mode is selected by EBU_CON.ARBMODE = 01_B.

[Table 13-6](#) and [Figure 13-4](#) show the functionality of the arbitration signals in Arbiter Mode.

Table 13-6 Function of Arbitration Pins in Arbiter Mode

Pin	Type	Pin Function in Arbiter Mode
$\overline{\text{HOLD}}$	In	In Owner Mode (EBU is the owner of the external bus), a low level at $\overline{\text{HOLD}}$ indicates a request for bus ownership from the external master. In Hold Mode (EBU is not the owner of the external bus), a high level at $\overline{\text{HOLD}}$ indicates that the external master has relinquished bus ownership, which causes the EBU to exit Hold Mode.
$\overline{\text{HLDA}}$	Out	While $\overline{\text{HLDA}}$ is high, the EBU is operating in Owner Mode. A high-to-low transition indicates that the EBU has entered Hold Mode and that the external bus is available to the external master. While $\overline{\text{HLDA}}$ is low, the EBU is operating in Hold Mode. A low-to-high transition indicates that the EBU has exited Hold Mode, and has retaken ownership of the external bus.
$\overline{\text{BREQ}}$	Out	$\overline{\text{BREQ}}$ is high during normal operation. The EBU drives $\overline{\text{BREQ}}$ low for two EBU clock cycles after entering Hold Mode (after asserting $\overline{\text{HLDA}}$ low). $\overline{\text{BREQ}}$ returns high one clock cycle after the EBU has exited Hold Mode (after driving $\overline{\text{HLDA}}$ high).

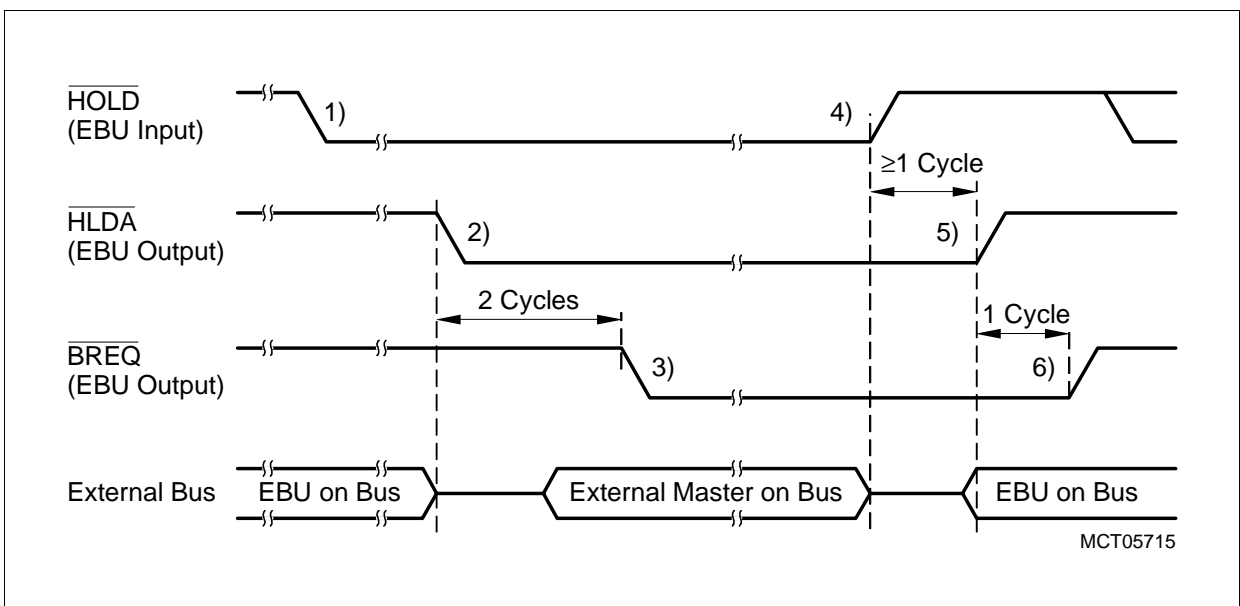


Figure 13-4 Arbitration Sequence with the EBU in Arbiter Mode

In Arbiter Mode, the arbitration sequence starts with the EBU as owner of the external bus.

LMB External Bus Unit

1. The external master wants to perform an external bus access by asserting a low signal on the HOLD input.
2. When the EBU is able to release bus ownership, it enters Hold Mode by tri-stating its bus interface lines and drives $\overline{HLDA} = 0$ to indicate that it has released the bus. At this point, the external master is allowed to drive the bus.
3. Two clock (LMBCLK) cycles after issuing \overline{HLDA} low, the EBU drives \overline{BREQ} low in order to regain bus ownership. This bus request is issued whether or not the EBU has a pending external bus access. However, the external master will ignore this signal until it has finished its bus access. This scheme assures that the external master can perform at least one complete external bus access.
4. When the external master has completed its access, it tri-states its bus interface and sets \overline{HOLD} to inactive (high) level to signal that it has released the bus back to the EBU.
5. When the EBU detects that the bus has been released, it returns \overline{HLDA} to high level and returns to Owner Mode by actively driving the bus interface lines. There is always at least one clock (LMBCLK) cycle delay from the release of the \overline{HOLD} input to the EBU driving the bus.
6. Finally, the EBU deactivates the \overline{BREQ} signal one clock (LMBCLK) cycle after deactivation of \overline{HLDA} . Now (and not earlier) the external master can generate a new hold request to the EBU.

This sequence assures that the EBU can perform at least one complete bus cycle before it re-enters Hold Mode as a result of a request from the external master.

The conditions that cause change of bus ownership when the EBU is operating in Arbiter Mode are shown in [Figure 13-5](#).

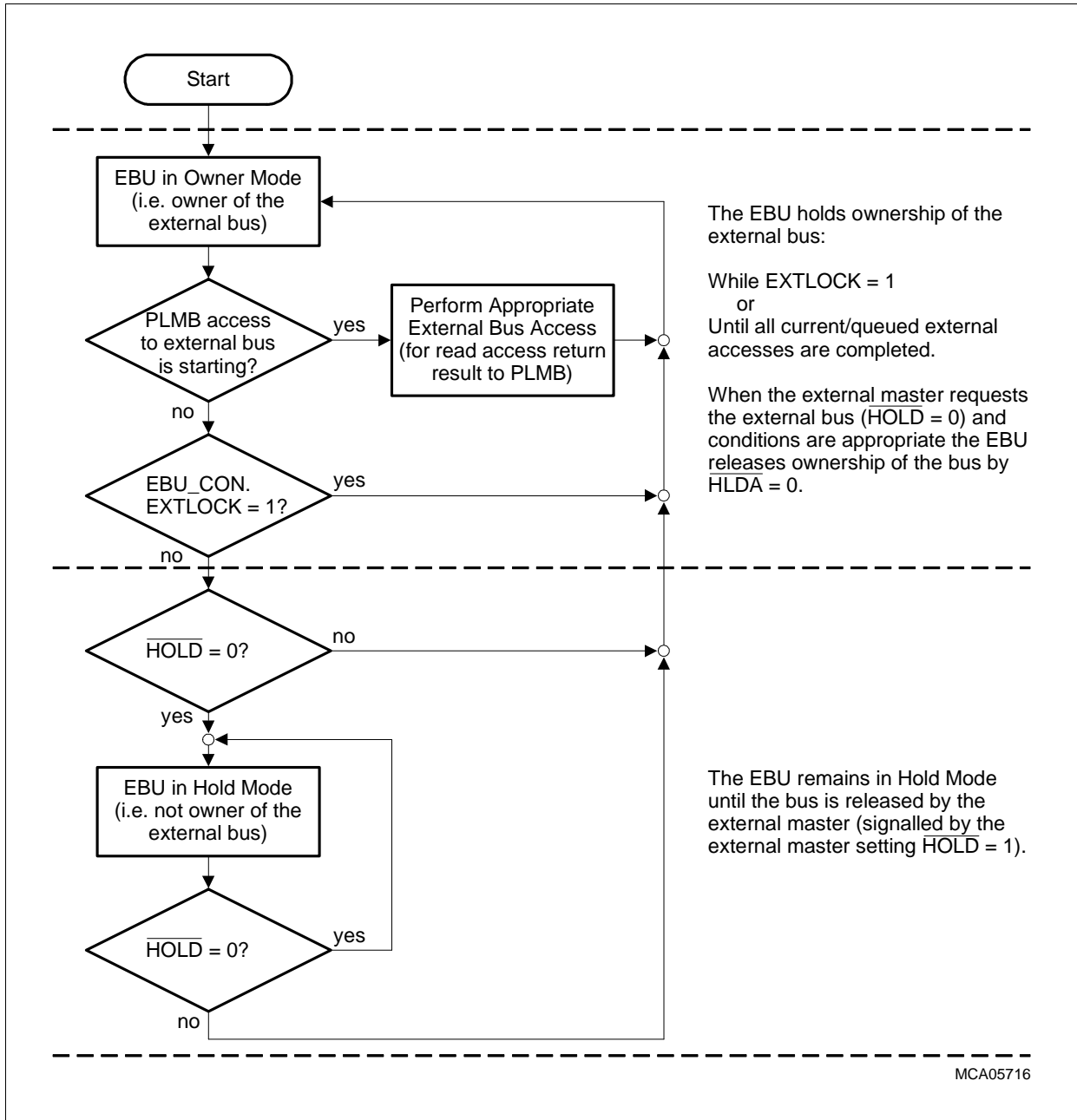


Figure 13-5 Bus Ownership Control in Arbitrator Mode

13.3.3.4 “Participant Mode” Arbitration Mode

The EBU tries to gain bus ownership only in case of pending transfers (e.g. when operating from internal memory and performing stores to external memory). While the EBU is not the owner of the external bus (default state), any PLMB access to the external bus will be issued with a retry by the EBU. Any such access will, however, cause the EBU to arbitrate for ownership of the external bus.

Once the EBU has gained ownership of the external bus, it will wait until either the occurrence of an external bus access (e.g. the repeat of the request that originally caused the arbitration to occur) or for a programmable time-out (see [Page 13-19](#)). Once the first access has been completed, the EBU will continue to accept requests from the PLMB bus until the external master asserts $\overline{HOLD} = 0$. After the external master has asserted $\overline{HOLD} = 0$, the EBU will respond to subsequent PLMB accesses to external memory with a retry, and will return ownership of the bus to the external master once any ongoing transaction is complete.

Note: Regardless of the state of the \overline{HOLD} input, the EBU will always perform at least one external bus access (assumed that there is not a time-out) before returning ownership of the bus to the external master.

The use of the arbitration signals in Participant Mode is:

Table 13-7 Function of Arbitration Pins in Participant Mode

Pin	Type	Pin Function in Participant Mode
\overline{HOLD}	In	When the EBU is not in Hold Mode ($\overline{HLDA} = 0$) and has completely taken over control of the external bus, a low level at \overline{HOLD} requests the EBU to return to Hold Mode.
\overline{HLDA}	In	When the \overline{HLDA} signal is high, the EBU is in Hold Mode. When the EBU has requested ownership of the bus by a high-to-low transition at \overline{HLDA} , the EBU is released from Hold Mode.
\overline{BREQ}	Out	\overline{BREQ} remains high as long as the EBU does not need to access the external bus. When the EBU detects that an external access is required, it sets $\overline{BREQ} = 0$ and waits for signal \overline{HLDA} to become low. When the EBU has completed the external bus access (and has re-entered Hold Mode), it will set $\overline{BREQ} = 1$ to signal that it has relinquished ownership of the external bus.

Participant Mode arbitration mode is selected by $EBU_CON.ARBMODE = 10_B$.

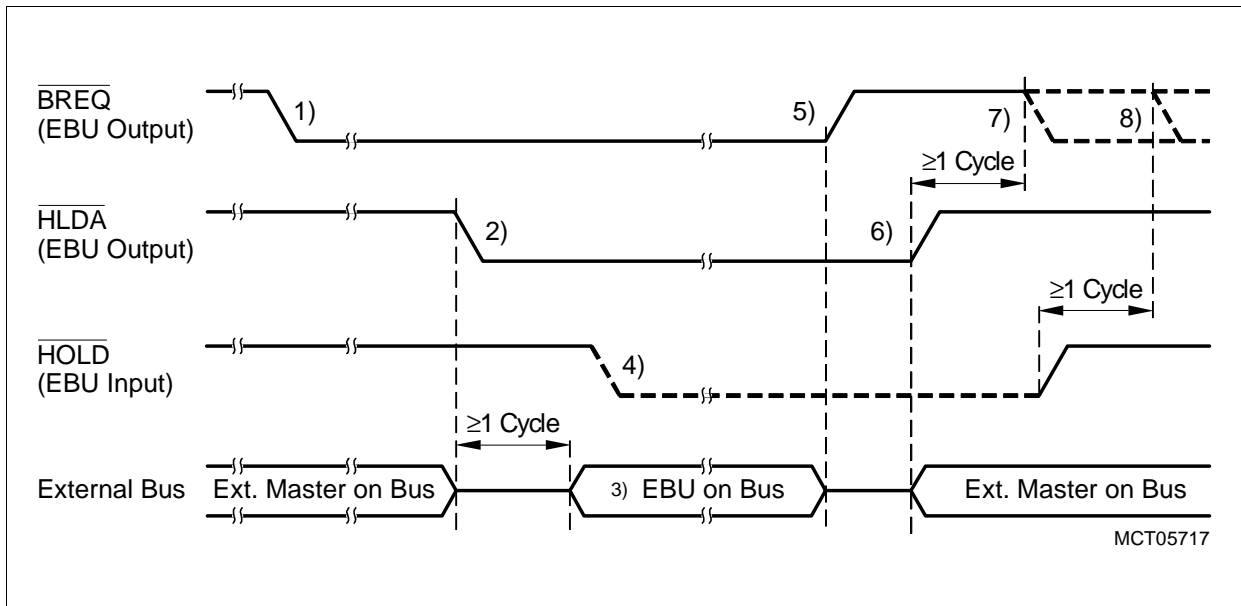


Figure 13-6 Arbitration Sequence with the EBU in Participant Mode

In Participant Mode, the arbitration sequence starts with the EBU in Hold Mode.

1. The EBU detects that it has to perform an external bus access by asserting a low signal on the $\overline{\text{BREQ}}$ output.
2. When the external master is able to release bus ownership, the external master releases the external bus by tri-stating its bus interface lines and drives the $\overline{\text{HLDA}} = 0$.
3. At least one clock (LMBCLK) cycle after issuing $\overline{\text{HLDA}} = 0$, the EBU will start to drive the external bus.
4. When the EBU is in Owner Mode, the external master may optionally drive $\overline{\text{HOLD}} = 0$ to signal that it wants to regain ownership of the external bus.
5. When the criterias are met for the EBU to release the bus ownership, the EBU enters Hold Mode and drives $\overline{\text{BREQ}} = 1$ output high to signal that it has released the bus.
6. When the external master detects that the EBU has released the bus ($\overline{\text{BREQ}} = 1$), it returns $\overline{\text{HLDA}}$ to high level and takes ownership of the external bus.
7. The EBU will not request ownership of the external bus again ($\overline{\text{BREQ}} = 0$) at least one clock (LMBCLK) cycle after $\overline{\text{HLDA}}$ has been driven high.
8. In Owner Mode, the EBU will not request ownership of the external bus $\overline{\text{BREQ}} = 0$ for at least one clock (LMBCLK) cycle after its $\overline{\text{HOLD}}$ input has been driven high.

The conditions that cause change of bus ownership when the EBU is operating in Participant Mode is shown in [Figure 13-5](#).

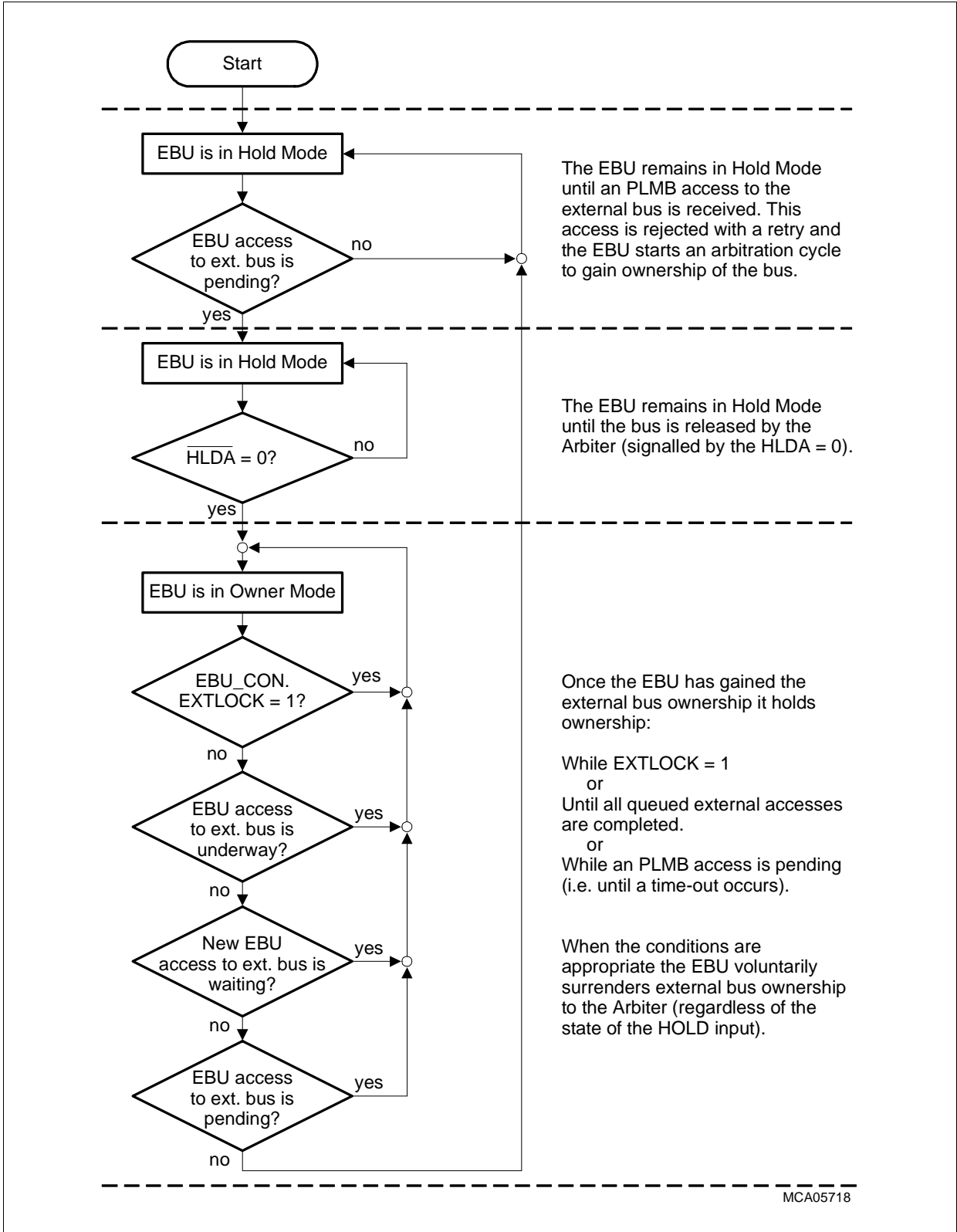


Figure 13-7 Bus Ownership Control with the EBU in Participant Mode

13.3.4 Arbitration Input Signal Sampling

The sampling of the arbitration inputs can be programmed for two modes:

- Synchronous Arbitration
- Asynchronous Arbitration

When synchronous arbitration signal sampling is selected ($ARBSYNC = 0$), the arbitration input signals are sampled and evaluated in the same clock cycle. This mode provides the least overhead during arbitration (i.e. when changing bus ownership). The disadvantage is that the input signals must adhere to setup and hold times with respect to LMBCLK to prevent the propagation of meta-stable signals in the EBU.

When asynchronous arbitration signal sampling is selected ($ARBSYNC = 1$), the arbitration signals are sampled and then fed to an additional latch to be evaluated in the cycle following that in which they were sampled. This provides the EBU with good immunity to signals changing state at or around the time at which they are sampled. The disadvantage is the introduction of additional latency during arbitration (i.e. when changing bus ownership).

13.3.5 Locking the External Bus

The external bus can be locked to allow the EBU to perform uninterrupted sequences of external bus accesses. The EBU allows two methods of locking the external bus:

- Locked PLMB accesses
- Lock bit EXTLOCK

When the EBU has ownership of the external bus and is performing external bus accesses in response to a locked PLMB access sequence, the ownership of the external bus will not be relinquished until the locked PLMB access sequence has been completed.

When lock bit $EXTLOCK = 1$, the EBU will hold the ownership of the external bus until $EXTLOCK$ is subsequently cleared. If $EXTLOCK$ is written to 1 while the EBU is the owner of the external bus, the EBU is immediately prevented from losing the ownership of the external bus until $EXTLOCK$ is cleared. If $EXTLOCK$ is written to 1 while the EBU is not the owner of the external bus, no immediate effect occurs. When the EBU gains the ownership of the external bus the next time, the external master is prevented from regaining ownership of the external bus until $EXTLOCK$ is again cleared.

Note: There is no time-out mechanism available for the EXTLOCK bit. When the EBU is owner of the external bus with EXTLOCK bit set, the external master will remain locked off the bus until the EXTLOCK bit is cleared by software.

13.3.6 Reaction to an PLMB Access to the External Bus

The reaction of the to a external bus request from an PLMB master is controlled as shown in [Figure 13-8](#).

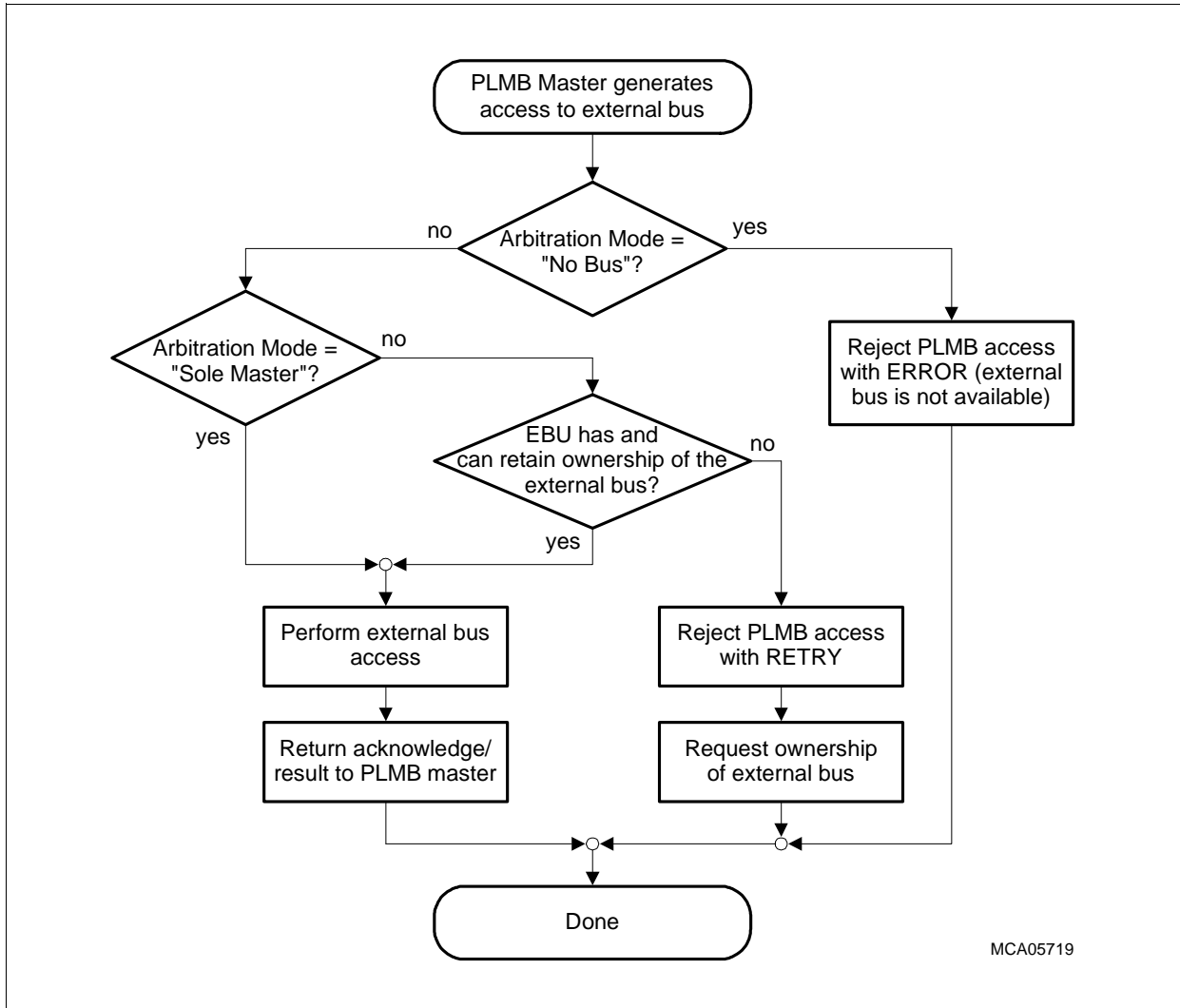


Figure 13-8 EBU Reaction to PLMB to External Bus Access

If the EBU is operating in No Bus Mode, it is not possible for an PLMB master to access the external bus. For this reason, the EBU generates an PLMB error whenever an attempt is made to access the external bus while in No Bus Mode.

If the EBU is operating in Sole Master Mode, it has access to the external bus at all times and as a result it is possible for the EBU to immediately perform the required external bus access.

If the EBU is operating in Arbiter or Participant Modes and receives a request for an external access from an PLMB or FPI bus master when it is not the owner of the external bus (or is not able to retain ownership of the bus), the request is rejected with a “retry”.

As shown in [Figure 13-8](#), this event also triggers the EBU to arbitrate with the external master in order to attempt to gain ownership of the external bus so that the request can be serviced when it is re-submitted by the master. This strategy ensures that the PLMB remains available while the EBU arbitrates for the external bus.

13.3.6.1 Pending Access Time-Out

The strategy of issuing a retry (when the EBU is not the owner of the external bus) as described in the previous section may result in the occurrence of a time-out condition.

When a PLMB master issues a request for an external bus access that is rejected by the EBU with a retry (in order to retain PLMB availability) and arbitration for external bus ownership is started, the external bus ownership is given once to the EBU. Now the EBU waits until the next external bus access occurs on the PLMB. If the requesting PLMB master or any other PLMB master subsequently performs no external bus accesses (e.g. fails to re-submit the original access request), the EBU has ownership of the bus for an indefinite time and it would become impossible for an external master to access the external bus.

To avoid such a bus-locking condition, the EBU contains a time-out mechanism. When the EBU has gained ownership of the external bus, it will retain ownership only until a PLMB-to-external bus access occurs or a programmable number of LMBCLK clock cycles has elapsed. If one of these conditions has occurred, the pending access is cancelled and the EBU will continue to arbitrate the external bus in the normal fashion. The desired time-out time (number of LMBCLK cycles) is programmed using bit field TIMEOUTC. The time-out value can be in the range 1×8 up to 255×8 LMBCLK clock cycles.

13.4 Start-Up/Boot Process

The EBU can start up in one of three modes after a reset or boot operation.

Table 13-8 EBU Start-Up Modes

Start-Up Modes	Arbitration Mode
Disabled EBU is disabled	No Bus Mode
Emulation Mode The emulator region is selected for all external bus accesses.	Participant Mode Arbiter Mode
External Boot Mode Region 0 is selected for all external bus accesses. Region 0 device characteristics are auto-configured by performing an external Boot Configuration Value fetch from a “Standard” non-multiplexed asynchronous device (see Page 13-21).	Participant Mode Arbiter Mode

Further details about EBU start-up mode selection after reset are defined in [Table 4-3](#) on [Page 4-12](#).

13.4.1 Disabled

The EBU will come up with access to the external bus disabled after reset (i.e. no access from PLMB to external memory is possible without EBU re-configuration).

13.4.2 Emulation Mode

The Emulation Mode of the EBU allows a system to start fetching code from a region reserved for emulator hardware. The EBU will come up with access to the external bus enabled after reset. All accesses to the external bus are directed via the emulator region. The fixed default configuration (contents of registers EBU_EMUBC and EBU_EMUBAP) of the emulator region is used as a setting that will provide reasonable code execution performance while not placing too great a restriction on the access time of the memory device connected to the emulator region (i.e. to the $\overline{\text{CSEMU}}$ chip select output, see [Page 13-29](#)).

In Emulation Mode, the EBU can be configured for either Arbiter or Participant arbitration modes. When configured as Participant, the EBU must be granted the bus by an external master before any external bus access can be made. When configured as Arbiter, the EBU owns the bus immediately after reset and can, if required, perform immediate external bus accesses.

Note: In TC1796, only Arbiter arbitration mode is supported in emulation mode. Furthermore, the CSEMU signal is connected to the CSCOMB chip select output by default after reset.

13.4.3 External Boot Mode

The External Boot Mode of the EBU allows the EBU to boot (i.e. run all start-up code) from external memory. Immediately after reset a system may have no knowledge as to the type of memory connected to the external bus. When External Boot Mode is selected, the EBU will automatically read a 32-bit Boot Configuration Value from an external memory (connected to CS0, chip select region 0). The Boot Configuration Value in the external boot memory makes it possible to initialize the EBU with appropriate configuration values for the external boot memory. These configuration values will, in turn, be used for the subsequent read accesses from the external boot memory (i.e. instruction fetches).

In External Boot Mode, the EBU can be configured for either Arbiter or Participant Arbitration modes. When configured as Participant, the EBU must be granted the bus by an external master before the Boot Configuration Value access can be made. When configured as Arbiter, the EBU owns the bus immediately after reset and can, therefore, perform the Boot Configuration Value access immediately after reset.

13.4.3.1 Boot Process

If External Boot Mode is selected, the EBU will perform one external bus read access to the dedicated address 000004_H of the memory device attached to chip select line CS0. The data read by this read access is used to configure the EBU with parameters (see [Page 13-23](#)). Any internal PLMB requests will be acknowledged with RETRY code until the external bus boot read access is complete. The boot read access itself is performed as an asynchronous access cycle with all timing parameters set to their maximum values. This access scheme supports demultiplexed ROMs, EPROMs or Flash memories. [Figure 13-9](#) shows a timing example of booting from a standard demultiplexed asynchronous memory device.

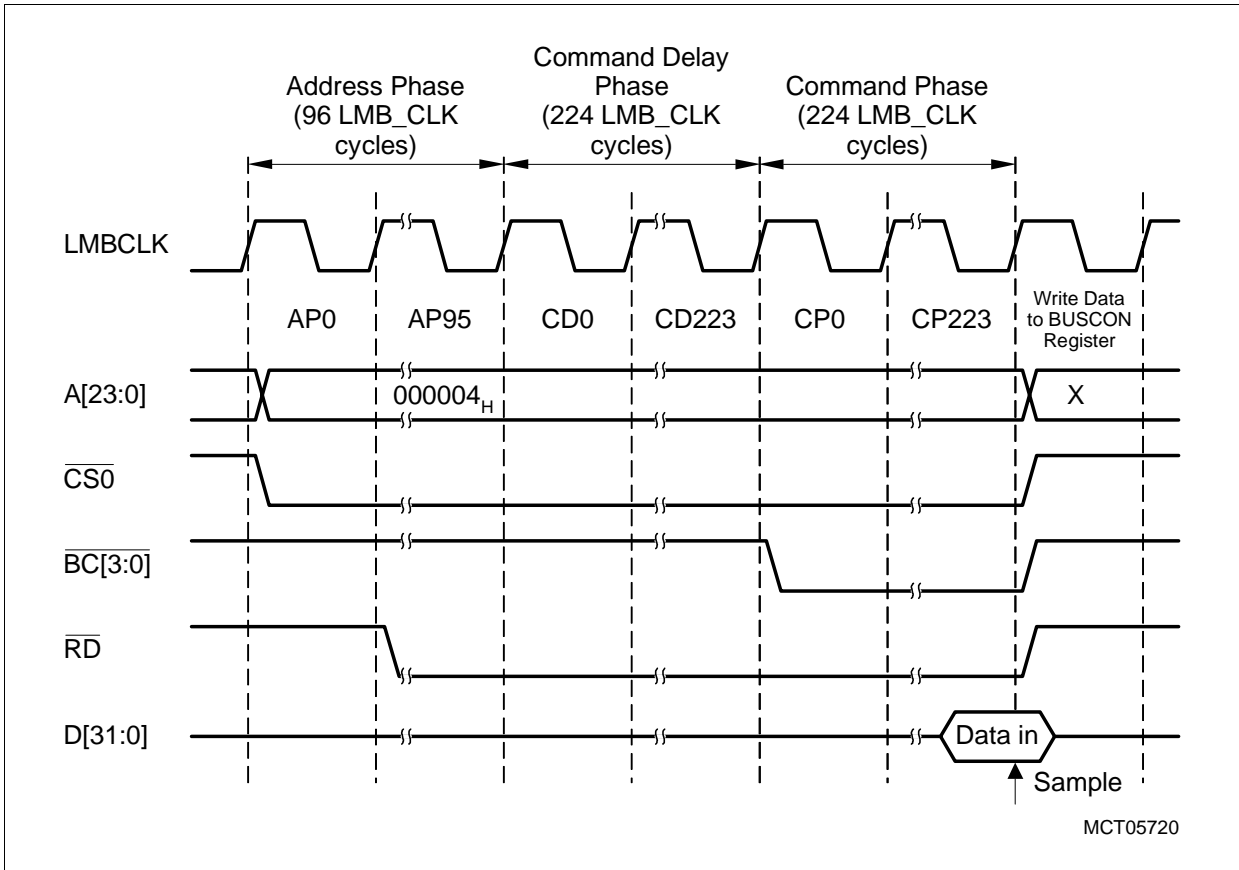


Figure 13-9 Boot Read Access Cycle

When a reset has become inactive, the EBU waits for a gap of 256 LMBCLK clock cycles until the access is initiated, as shown in [Figure 13-9](#). This gap is inserted to fulfill the recovery times needed by external synchronous devices such as Flash ROMs. During the read access, the maximum number of programmable LMBCLK cycles is inserted ($WAITRDC \times CMULT = 7 \times 32 = 224$), and the evaluation of the \overline{WAIT} signal is inhibited.

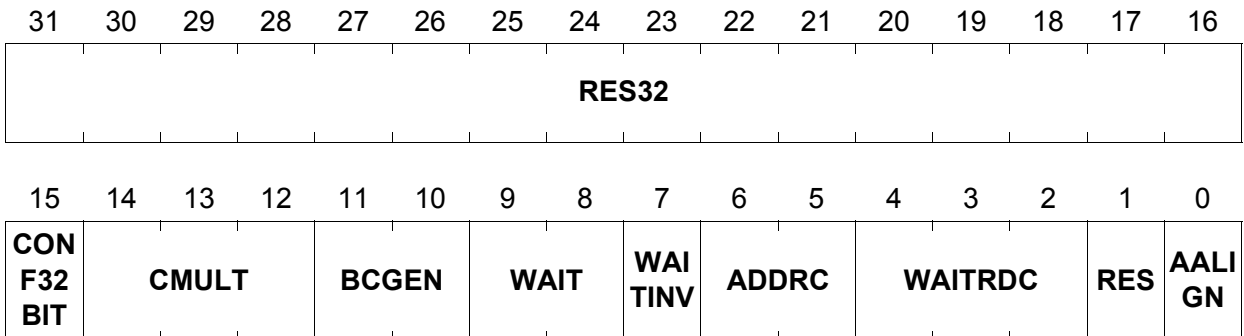
Note: The boot memory must be connected to chip select line $\overline{CS0}$. The EBU assumes that the boot memory is 32-bit wide and, therefore, always reads a 32-bit configuration word at the “Data in” point shown above. The encoding of this 32-bit configuration word subsequently signals whether the memory is actually 32-bit or 16-bit wide. In the case of 16-bit wide memory, the EBU will discard the most significant 16 bits of the configuration word.

Note: If $FFFF_H$ is returned as configuration word during the boot read cycle (e.g. by reading the configuration word from an erased external boot memory device), the arbitration mode is set to No Bus Mode ($ARBMODE = 00_B$, see [“No Bus Arbitration Mode” on Page 13-10](#)).

13.4.3.2 Boot Configuration Value

The EBU can support boot operation from 16-bit or 32-bit wide memories. The format of the Boot Configuration Value is as follows:

Boot Configuration Value



Field	Bits	Description
AALIGN	0	Address Alignment Loaded into EBU_BUSCON0.AALIGN.
RES	1	Reserved Reserved for future use. For TC1796, this bit must be set to 0.
WAITRDC	[4:2]	Number of Wait States for Read Accesses Loaded into EBU_BUSAP0.WAITRDC.
ADDRC	[6:5]	Number of Cycles in the Address Phase Loaded into EBU_BUSAP0.ADDRC.
WAITINV	7	WAIT Input Polarity Control Loaded into EBU_BUSCON0.WAITINV.
WAIT	[9:8]	External Wait State Control Loaded into EBU_BUSCON0.WAIT.
BCGEN	[11:10]	Byte Control Signal Control Loaded into EBU_BUSCON0.BCGEN.
CMULT	[14:12]	Cycle Multiplier Control Loaded into EBU_BUSCON0.CMULT.

LMB External Bus Unit

Field	Bits	Description
CONF32BIT	15	<p>32-Bit Configuration Word</p> <p>0_B The memory connected to $\overline{CS0}$ is 16-bit wide. Bit field RES32 (bits [31:16]) of the configuration word are discarded.</p> <p>1_B The memory connected to $\overline{CS0}$ is 32-bit wide. Bit field RES32 (bits [31:16]) of the configuration word are potentially available for use as configuration information.</p> <p><i>Note: In the TC1796, the EBU does not actually use bits 31 to 16 of the configuration word when CONF32BIT is 1. However, to allow future expansion this bit should be programmed to 0.</i></p>
RES32	[31:16]	<p>Reserved</p> <p>This bit field is reserved for future use. In the TC1796, RES32 should always be set to 0000_H.</p>

13.5 Master Mode Operation

In master mode, the EBU supports interconnection to a wide variety of memory/peripheral devices with flexible programming of the access parameters. In the following sections, the basic features for these access modes are described. The types of external access cycles provided by the EBU are:

- Asynchronous devices with demultiplexed access
 - INTEL-style peripherals (separate \overline{RD} and \overline{WR} control signals)
 - Motorola-style peripherals (MR/ \overline{W} signals)
 - ROMs, EPROMs
 - Static RAMs
- Burst Mode Flash devices

Each TC1796 internal PLMB master can access external devices via the EBU. The EBU provides four user-programmable external memory regions. Each of these regions is provided with a set of registers that determine the parameters of the external bus transaction and one chip select signal. A PLMB transaction that matches one of these user-programmable external memory regions is translated by the EBU to the appropriate external access(es).

In the TC1796, the EBU responds to the address ranges as defined in [Table 13-9](#).

Table 13-9 EBU External Address Ranges

Address Range	Description	Action
8080 0000 _H - 8FDF FFFF _H	External memory space (cached area)	compare
A080 0000 _H - AFDF FFFF _H	External memory space (non-cached area)	
D800 0000 _H - DDFF FFFF _H	External Peripheral Space (non-cached area)	
DE00 0000 _H - DEFF FFFF _H	External Emulator Space (non-cached area)	
E000 0000 _H - E7FF FFFF _H	External Peripheral Space (non-cached area)	
F800 0000 _H - F800 03FF _H	EBU Registers	access register

The “compare” action means that the EBU compares the supplied PLMB address to all its external regions. If a match is found, the EBU performs the appropriate external bus access. Otherwise, the EBU generates a PLMB Error Acknowledge.

The “access registers” action means that the EBU is selected for a control/status register access. The EBU performs the requested register access (or generates an PLMB Error Acknowledge if there is no register at the supplied address).

For all address ranges that not listed in [Table 13-9](#) the EBU is not selected and PLMB requests are ignored.

13.5.1 External Memory Regions

The EBU of the TC1796 supports four (plus one for emulation purposes) memory regions, which have its own associated chip select outputs $\overline{CS}[3:0]$ (and \overline{CSEMU}). Each of these regions has a set of control registers to specify the type of memory/peripheral device and the access parameters.

Each of the four user-programmable regions (x = 0-3 is the numbering index of these regions) is configurable through registers EBU_ADDRSELx, EBU_BUSCONx, and EBU_BUSAPx. The access parameters for each of the regions can be programmed individually to accommodate different types of external devices.

The emulator region is programmable in the same way as the user-programmable regions through registers EBU_EMUAS, EBU_EMUBC, and EBU_EMUBAP. Each of the regions can be defined for either asynchronous demultiplexed or synchronous Burst Flash access.

Table 13-10 EBU Address Regions, Registers and Chip Selects

Region	Associated Chip Select	Address Select Registers	Bus Configuration Registers	Bus Access Parameters Registers
Region 0	$\overline{CS0}$	EBU_ADDRSEL0	EBU_BUSCON0	EBU_BUSAP0
Region 1	$\overline{CS1}$	EBU_ADDRSEL1	EBU_BUSCON1	EBU_BUSAP1
Region 2	$\overline{CS2}$	EBU_ADDRSEL2	EBU_BUSCON2	EBU_BUSAP2
Region 3	$\overline{CS3}$	EBU_ADDRSEL3	EBU_BUSCON3	EBU_BUSAP3
Emulator region	\overline{CSEMU}	EBU_EMUAS	EBU_EMUBC	EBU_EMUBAP

Table 13-11 lists the programmable parameters that are available for the five external regions (region 0 to 3 and the emulator memory region).

Table 13-11 Programmable Parameters of Regions

Register	Parameter (Bit/Bit field)	Function
EBU_ADDRSELx EBU_EMUAS	ALTSEG	Alternate segment of region to be compared to PLMB address bits [31:28].
	BASE	Region base address to be compared with PLMB address in conjunction with the MASK parameter.
	MASK	Address mask for each external region. Specifies the number of right-most bits in the base address starting from bit 26.
	ALTENAB	Alternate segment comparison enable of a region: Determines whether or not parameter ALTSEG is always compared to PLMB address.
	REGENAB	Enable bit for each region. A disabled region will always generate a miss during address comparison.
EBU_BUSCONx EBU_EMUBC	AGEN	Region access type: Demultiplexed (asynch.) or Burst Flash (sync.) access
	ENDIAN	Region endian mode: Little- or big-endian access mode
	PORTW	External region data bus width: 16-bit or 32-bit
	WRITE	Region write protection: ON or OFF
	AALIGN	Region address alignment enable: ON or OFF
EBU_CON	GLOBALCS	Global chip select signal selection: Selects the <u>CSx</u> chip select lines that generate the internal <u>CSSLB</u> signal (see Page 13-29).
EBU_EMUOVL	OVERLAY	<u>Overlay</u> chip select signal selection: Selects the <u>CSx</u> chip select lines that generate the internal <u>CSOVL</u> signal (see Page 13-29).

13.5.2 Chip Select Control

The EBU generates several chip select signals (see [Figure 13-10](#)):

- \overline{CSx} , which is activated by accesses to the memory regions.
- \overline{CSEMU} , which is activated by accesses to the emulator memory region.
- \overline{CSOVL} , which is generated internally when the overlay memory region is accessed.
- \overline{CSGLB} , which is generated internally when one of \overline{CSx} becomes active.
- \overline{CSCOMB} , which is a combination of \overline{CSEMU} , \overline{CSGLB} , and \overline{CSOVL} .
- Details about the chip select control logic as implemented in the TC1796 are shown in [Figure 13-10](#).

The four chip select lines \overline{CSx} are all available at dedicated chip select outputs. The internal \overline{CSGLB} signal is controlled by bit field EBU_CON.GLOBALCS. The bits in this bit field make it possible to individually enable/disable each of the \overline{CSx} lines for \overline{CSGLB} generation. In a similar way, \overline{CSOVL} is controlled by the bits of bit field EBU_EMUOVL.OVERLAY. Finally, \overline{CSEMU} , \overline{CSGLB} , and \overline{CSOVL} can be individually enabled/disabled for the combined chip select output signal \overline{CSCOMB} generation.

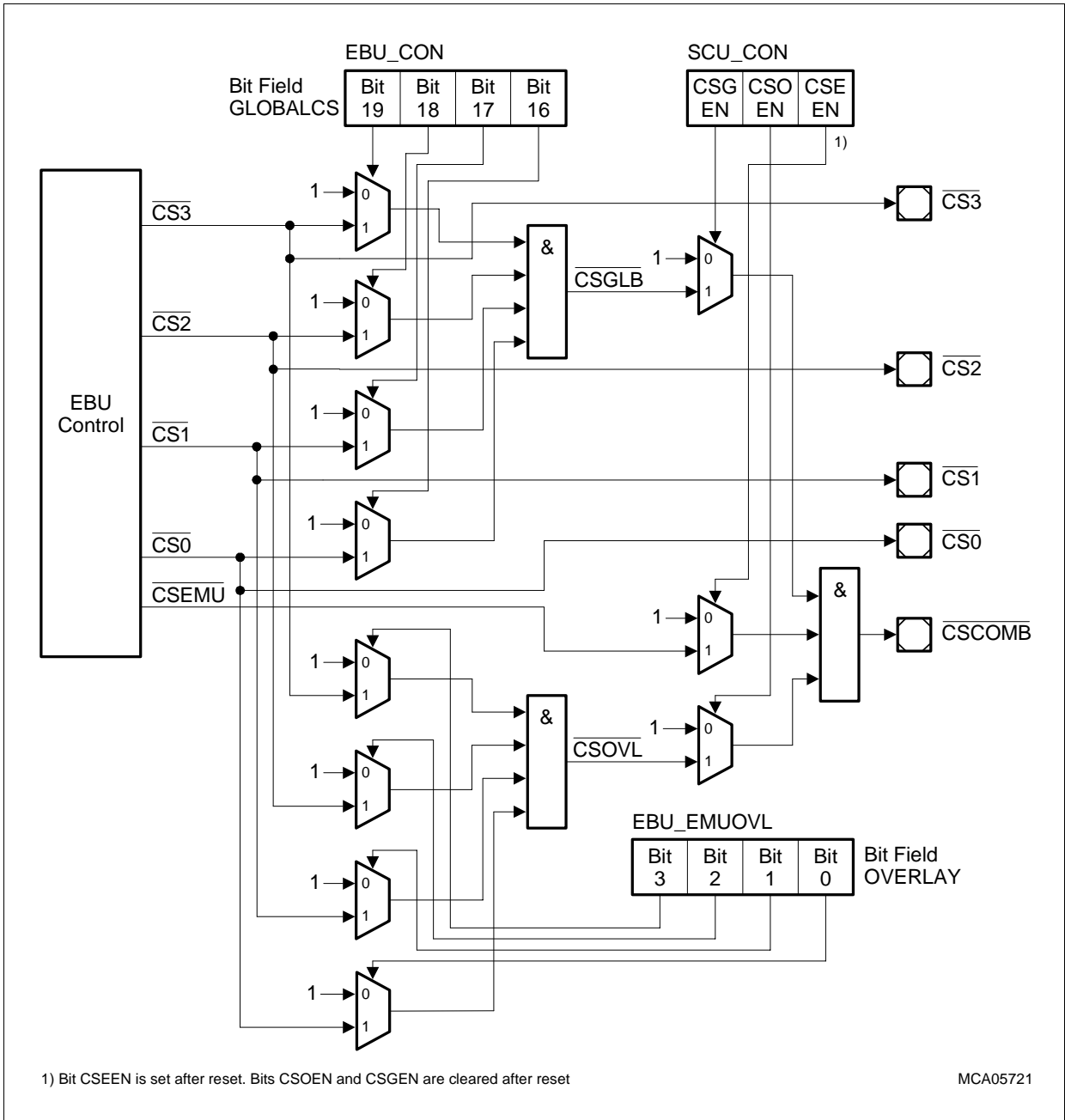


Figure 13-10 Chip Select Control Logic

13.5.3 Address Comparison

Standard Address Comparison Mode

When an Addressing Override Mode is not active, each of the five EBU regions can be programmed for independent base addresses and lengths by bits and bit fields in registers EBU_ADDRSELx and EBU_EMU_AS.

- Bit REGEN is the enable control of a region. If the region is disabled (REGEN = 0), no address comparison will take place for the region.
- Bit field BASE specifies address bits A[31:12] of region x, where A[31:28] must only point to segments 8, 10, 13, and 14 (see [Table 13-9](#) on [Page 13-25](#)).
- Bit field MASK determines the length of a region. It specifies how many bits of a PLMB address must match the contents of the BASE(x) bit field (to a maximum of 15, starting with A[26]). Note that address bits A[31:27] must always match.
- Bit field ALTSEG determines the number of an alternate segment that can be used for address comparison with A[31:28] (if enabled by ALTENAB = 1).
- Bit ALTENAB determines whether an additional alternate segment number as defined by ALTSEG is used for address comparison.

The Emulator Address Select register EBU_EMUAS has identical functionality as the EBU_ADDRSELx register.

The address comparison scheme is shown in [Figure 13-11](#).

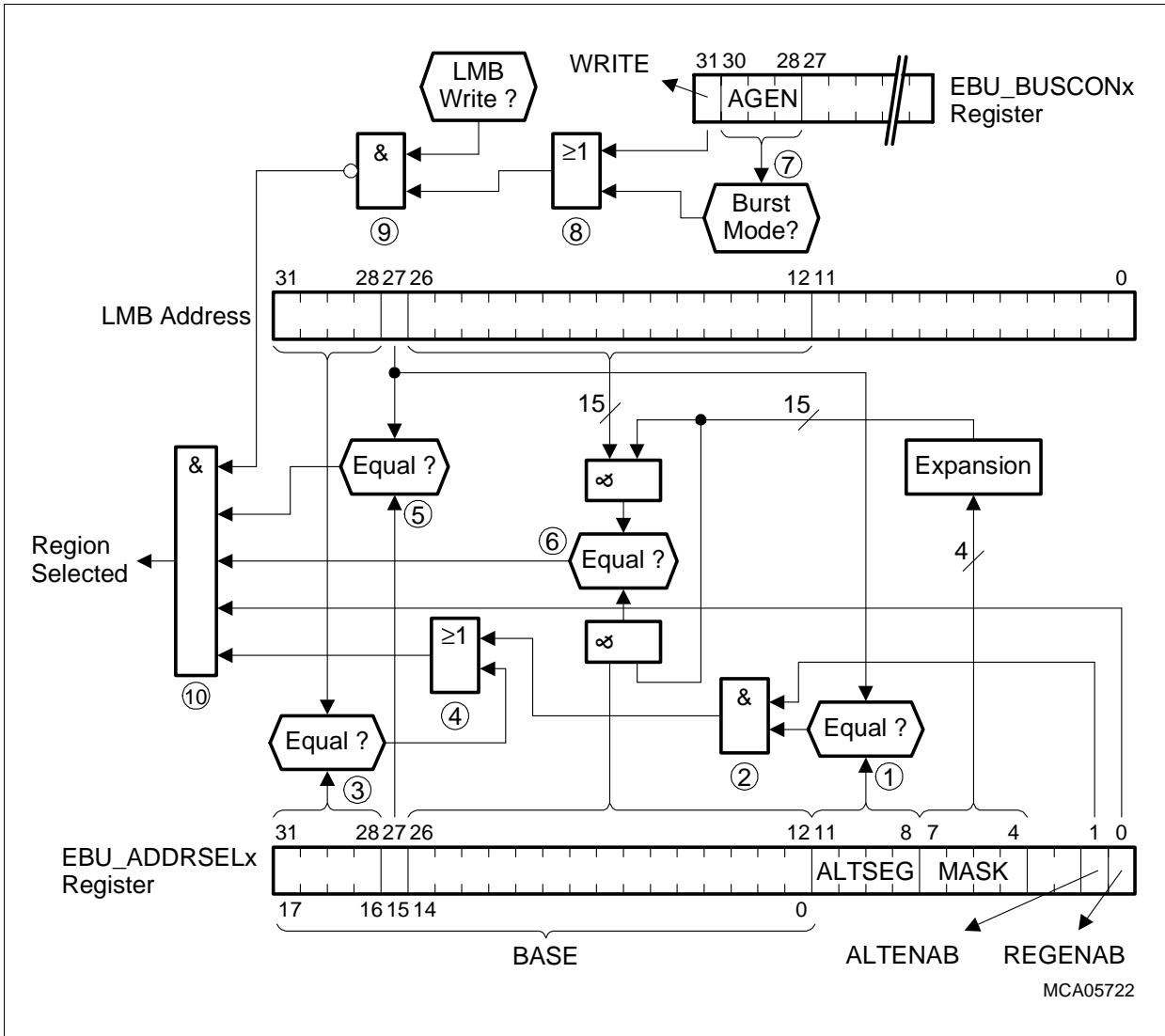


Figure 13-11 Address Comparison to Detect Access to External Region

When the EBU is processing an PLMB access, the address is compared in parallel to the parameters of all five regions. The address comparison process for one region is shown in **Figure 13-11**. This process is as follows:

1. The most significant four bits of the PLMB address are compared with the ALTSEG bit field (“alternate” segment address). The result of the comparison (1 if equal, otherwise 0) is fed to an AND gate.
2. If ALTENAB = 0, the alternate segment function is disabled and the output of the AND gate is 0. With ALTENAB = 1, the alternate segment function is enabled, and the result of the comparison between ALTSEG and the segment part of the PLMB address is fed to an OR gate.

LMB External Bus Unit

3. The most significant four bits of the PLMB address (“main” segment address) are compared to the most significant four bits of the BASE bit field. The result of the comparison (1 if equal, otherwise 0) is fed to the OR gate.
4. The OR gate combines the result of the “main” and “alternate” segment comparisons generates a 1 if the PLMB address is in the selected segment(s) for the region, otherwise it generates 0. The output of the OR gate is fed to the final AND gate.
5. Bit 27 of the PLMB address is (unconditionally) compared with bit 15 of the BASE bit field. The result of the comparison (1 if equal, otherwise 0) is fed to the final AND gate.
6. The appropriate number of PLMB address bits from bit 26 downwards is compared with the corresponding bits from bit field BASE bit 14 downwards. The number of bits used for the comparison is controlled by the MASK bit field. The result of the comparison (1 if the appropriate bits are equal, otherwise 0) is fed to the final AND gate.
7. The comparator checks whether the access is a Burst Flash access of Type 0 (AGEN = 010_B) or of Type 1 (AGEN = 101_B). The output of the comparator is fed to an OR gate.
8. The OR gate produces a 1 if the region is a read-only region (for example a Burst Flash access) that is write-protected via the write bit. The output of the OR gate is fed to a NAND gate.
9. The NAND gate delivers a 0 if a write is performed to a read-only region, and prevents the region from being selected. The output of the NAND gate is fed to the final AND gate.
10. The final AND gate delivers a 1 if a match occurs at the address comparison, and the region x is enabled by REGENAB = 1, and the access is not a write access when the region is defined as read-only access.

This address decoding scheme has the following effects:

- The smallest possible address region is 2¹² bytes (4 Kbyte)
- The largest possible address region is 2²⁷ bytes (128 Mbyte)
- The start address of a region depends on the size of the region. It must be at an address that is a multiple of the size of a region; for example, the smallest region can be placed on any 4-Kbyte boundary, while the largest region can be placed on 8-Mbyte boundaries only.

Table 13-12 shows the possible region sizes and start granularity, as determined by the programming of the MASK bit field. The range of the offset address within such a region is also given.

Table 13-12 EBU Address Regions Size and Start Address Relations

MASK	No. of Address Bits compared to BASE[26:12]	Range of Address Bits compared to BASE[26:12]	Region Size and Start Address Granularity	Range of Offset Address Bits within Region
1111 _B	15	A[26:12]	4 Kbyte	A[11:0]
1110 _B	14	A[26:13]	8 Kbyte	A[12:0]
1101 _B	13	A[26:14]	16 Kbyte	A[13:0]
1100 _B	12	A[26:15]	32 Kbyte	A[14:0]
1011 _B	11	A[26:16]	64 Kbyte	A[15:0]
1010 _B	10	A[26:17]	128 Kbyte	A[16:0]
1001 _B	9	A[26:18]	256 Kbyte	A[17:0]
1000 _B	8	A[26:19]	512 Kbyte	A[18:0]
0111 _B	7	A[26:20]	1 Mbyte	A[19:0]
0110 _B	6	A[26:21]	2 Mbyte	A[20:0]
0101 _B	5	A[26:22]	4 Mbyte	A[21:0]
0100 _B	4	A[26:23]	8 Mbyte	A[22:0]
0011 _B	3	A[26:24]	16 Mbyte	A[23:0]
0010 _B	2	A[26:25]	32 Mbyte ¹⁾	A[24:0]
0001 _B	1	A[26]	64 Mbyte ¹⁾	A[25:0]
0000 _B	0	–	128 Mbyte ¹⁾	A[26:0]

1) These region size selections do not affect the external address pins of the TC1796 because A24, A25, and A26 are not output at pins (only A[23:0] are available at TC1796 pins).

The EBU uses the five region select outputs from the above scheme, in conjunction with its own address decode logic, to react to PLMB accesses as follows:

1. **Address is in the EBU register space:**
An EBU register access is executed, and in case of a illegal register address, PLMB Error Acknowledge is returned.
2. **Address matches exactly one enabled external region:**
The requested access is performed to external memory.
3. **Address matches more than one enabled region (overlapping regions):**
The requested access is performed using the parameters from the region with highest priority. The region with the lowest region number x (region 0) has the highest priority. The emulator region has lowest priority).
4. **Address matches disabled region(s) or no address match:**
The EBU returns a PLMB Error Acknowledge.

When defining mirrored segments, the user is responsible for ensuring that there is no collision. There is no checking mechanism in hardware that ensures that each segment defined (either in BASE[31:28] or ALTSEG[11:8] or both) is exclusive. Therefore, the user must ensure that each mapping from region 0 to 3 and the emulator region does not interfere with any other; otherwise, only the mapping with the highest priority will take effect.

Chip Select 0 Address Override Mode

The Chip Select 0 Address Override Mode is selected when bit EBU_CON.CS0FAM is set (CS0 Fills Address Map). With CS0FAM = 1, all external bus accesses are directed to region 0 (CS0 becomes active) and executed with the parameter settings of region 0. This mode is automatically selected when external boot mode has been detected during a reset (see [Table 13-8](#)). In this case, the region 0 access parameters are loaded automatically by the boot read access (see [Page 13-21](#)).

Note: When performing a boot sequence from the device connected to CS0, software must be written carefully to ensure that external code access is never unintentionally disabled. This can be achieved when the regions are configured as required with CS0FAM = 1 after reset, and afterwards CS0FAM is set to 0 to activate the new address map.

Emulator Chip Select Address Override Mode

Emulator Chip Select Address Override Mode is selected when bit EBU_CON.EMUFAM (CSEMU Fills Address Map) is set (and bit CS0FAM = 0). With EMUFAM = 1, all external bus accesses are directed to the emulator region (CSEMU becomes active) and executed with the parameter settings of the emulator region. This mode is automatically selected when emulation mode has been detected during a reset (see [Table 13-8](#)).

Note: When performing a boot sequence from the device connected to CSEMU, software must be written carefully to ensure that external code access is never unintentionally disabled. This can be achieved when all regions are configured as required with EMUFAM = 1 after reset, and afterwards CS0FAM is set to 0 to activate the new address map.

13.5.4 Access Parameter Selection

Selection of the appropriate access parameters is shown in [Figure 13-12](#):

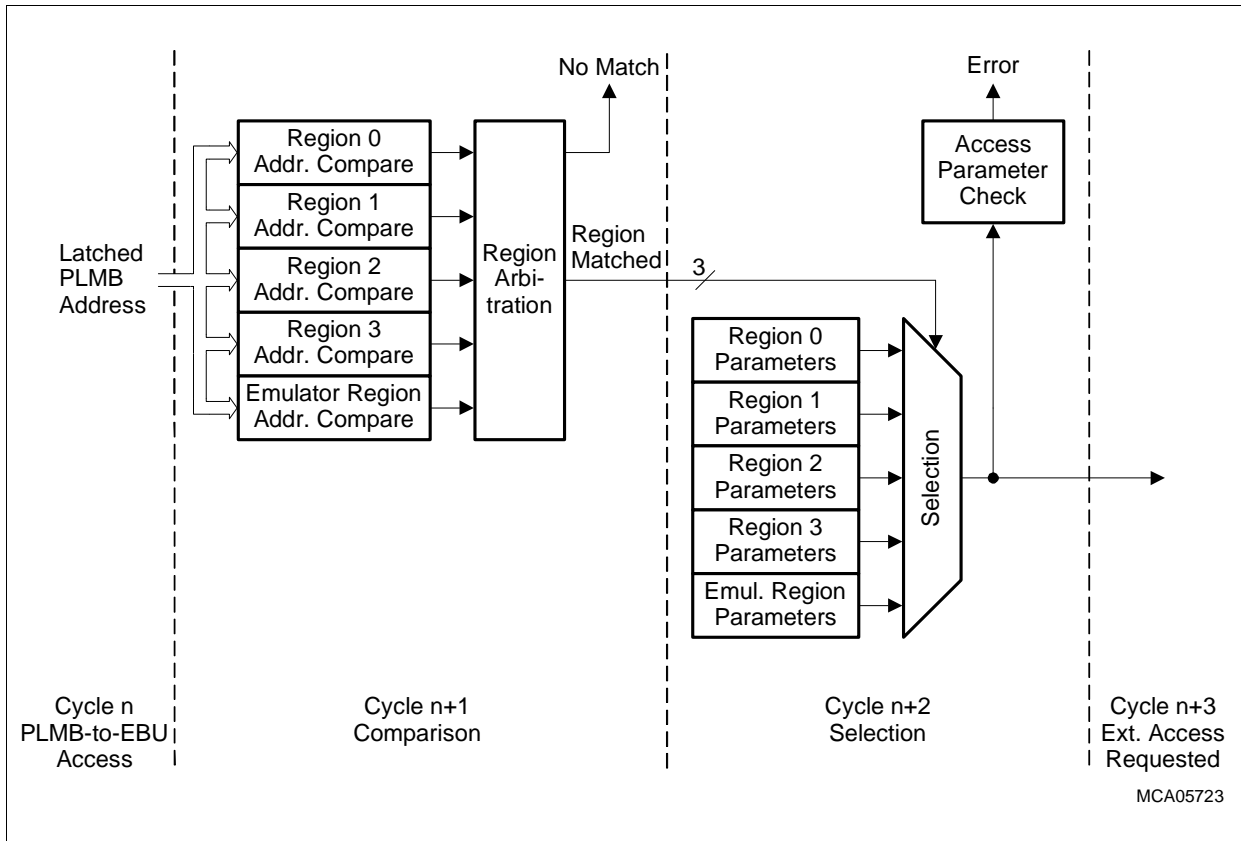


Figure 13-12 Access Parameter Selection Phases

At the end of the PLMB address phase (Cycle n), the 32-bit PLMB address is made available to the region comparison logic.

In the second phase (Cycle $n + 1$), the region comparison logic determines whether the PLMB address lies within an active region (or regions). If the address is not matched (i.e. is not within an active region), the “No Match” signal causes bits [31:28] or ALTSEG[11:8] to return an PLMB Error Acknowledge and discard the access. Otherwise the “Region” signal passes the region index number (in the case of multiple region matches this is the index number of the lowest matching region) to the parameter selection logic.

During the third phase (Cycle $n + 2$), the parameter selection logic selects the appropriate access cycle parameters. At the end of this cycle, the access parameters associated with the highest priority region are checked for an invalid access (e.g. a write access that does not lie within a defined, writable region). If the access is invalid, the EBU returns an PLMB Error Acknowledge and discards the access. Otherwise the parameters are made available to the external bus driver logic.

LMB External Bus Unit

The next cycle (Cycle $n + 3$) sees these parameters being passed to the appropriate external access cycle state machine (used to select/initialize the appropriate external access cycle, and also to select the external bus pins to be used for the access cycle).

Write Protection

Each address region has an associated bit to provide write protection (programmable separately for each region). Write protection is controlled by the WRITE bit in the EBU_BUSCONx and EBU_EMUBC registers.

Writing to the area occupied by a write-protected region will only generate an error if there is not a lower-priority region programmed for write accesses at the matching address (i.e. if the access causes no regions to be selected).

13.5.5 Little-/Big-Endian Access Modes

The native mode of the EBU is little-endian (in line with the architecture of TriCore). Additionally, the EBU provides limited support for big-endian access modes. Big-endian accesses are supported by modifying the least significant two address bits. This feature is programmable separately for each chip select region by bits ENDIAN in registers EBU_BUSCONx and EBU_EMUBC.

Big-endian access mode is available only for single accesses (8-bit, 16-bit, or 32-bit) to regions programmed for 16-bit wide operation. There is no support for burst accesses to regions that have been programmed for big-endian operation. There is no concept of little- and big-endian for accesses to 32-bit wide external regions.

Table 13-13 shows how the lower two bits of the PLMB address are changed before being issued as an address on the external bus.

Table 13-13 Little- and Big-endian Address Translation

Internal (PLMB) Address	External Address			
	Little-Endian		Big-Endian	
	16-bit Access	8-bit Access	16-bit Access	8-bit Access
...XXX00 _B	...XXX00 _B	...XXX00 _B	...XXX10 _B	...XXX11 _B
...XXX01 _B	¹⁾	...XXX01 _B	¹⁾	...XXX10 _B
...XXX10 _B	—	...XXX10 _B	...XXX00 _B	...XXX01 _B
...XXX11 _B	¹⁾	...XXX11 _B	¹⁾	...XXX00 _B

1) This is a non-aligned 16-bit access that can never be generated on the PLMB.

13.5.6 PLMB Bus Width Translation

If the internal access width is wider than the external bus width specified for the selected external region (programmed via the EBU_BUSCONx.PORTW bit field), the internal access is split in the EBU into several external accesses. For example, if the PLMB requests to read a 64-bit word and the external device is only 16-bit wide, the EBU will automatically perform four external 16-bit accesses. When multiple accesses are generated in this way, external bus arbitration is blocked until the multiple access is complete. This means that the EBU remains the owner of the external bus for the duration of the access sequence. The external accesses are performed in ascending PLMB address order.

To allow proper bus width translation, the EBU has the capability to re-align data between the external bus and the PLMB as shown in [Figure 13-13](#).

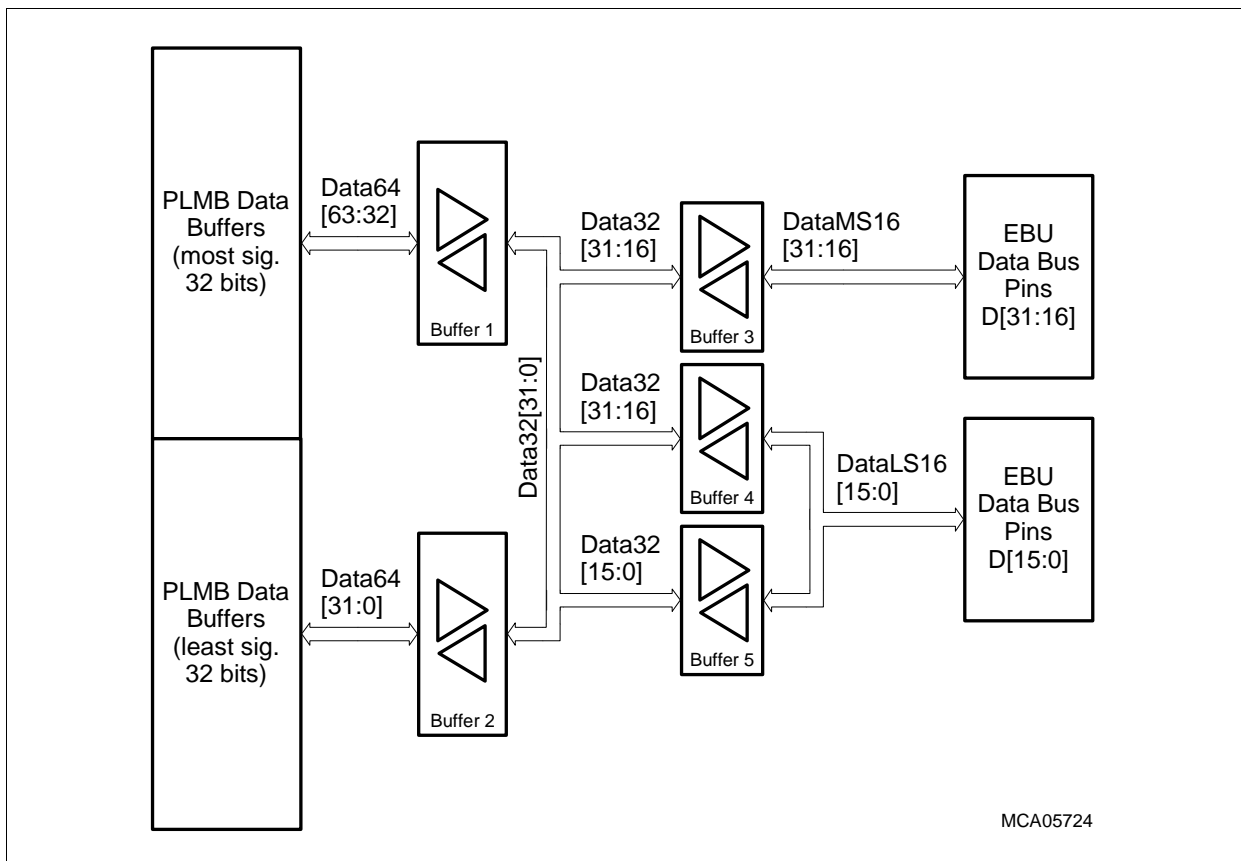


Figure 13-13 PLMB to External Bus Data Re-Alignment

- During an access to a 32-bit wide external region, either Buffer 1 or Buffer 2 is enabled (according to bit 2 of the PLMB address being accessed) to perform the required 64-bit (PLMB) data to 32-bit bus alignment (signified by Data32[31:0] above). To generate a 32-bit access to the external data bus D[31:0], Buffer 3 and Buffer 5 are enabled together.

LMB External Bus Unit

- During an access to a 16-bit wide external region, either Buffer 1 or Buffer 2 is enabled (according to bit 2 of the PLMB address being accessed) and either Buffer 4 or Buffer 5 is enabled (according to bit 1 of the PLMB address being accessed). This allows any PLMB channel byte pair (i.e. any properly aligned 16-bit data) to be re-aligned to the lower 16 bits of the external data bus D[15:0].

Note: When accessing a 16-bit wide external region programmed for big-endian operation, the PLMB address is modified prior to being driven to the external bus (see [Table 13-13](#) on [Page 13-36](#)). For a 64-bit access, this will generate external bus accesses in the address sequence $X010_B$, $X000_B$, $X110_B$, $X100_B$. For a 32-bit access, the external bus address sequence will be $XX10_B$, $XX00_B$.

13.5.7 Address Alignment During Bus Accesses

During an external bus access, the EBU will optionally align the internal byte address to generate the appropriate external word or half-word address aligned to the external address pins A[23:0]. This address alignment is enabled separately for each region by the AALIGN bit in the EBU_BUSCONx or EBU_EMUBC registers. When enabled, alignment will be performed according to the selected bus width (i.e. the value of the appropriate PORTW bit field in the EBU_BUSCONx or EBU_EMUBC registers). The address alignment will be done as defined in [Table 13-14](#).

Table 13-14 EBU External Bus Address Alignment

Parameter PORTW	Selected Bus Width	Address Alignment
00 _B	Reserved	–
01 _B	16-bit	External address A[23:0] = PLMB address [24:1]
10 _B	32-bit	External address A[23:0] = PLMB address [25:2]
11 _B	Reserved	–

When alignment is not enabled (default after reset) for a region, the EBU will issue a byte address to the external bus.

13.6 PLMB Data Buffering

The EBU contains three data buffers that are used to adapt accesses from the internal 64-bit wide PLMB to the 16-bit or 32-bit wide external bus. **Figure 13-14** illustrates the data-buffering scheme.

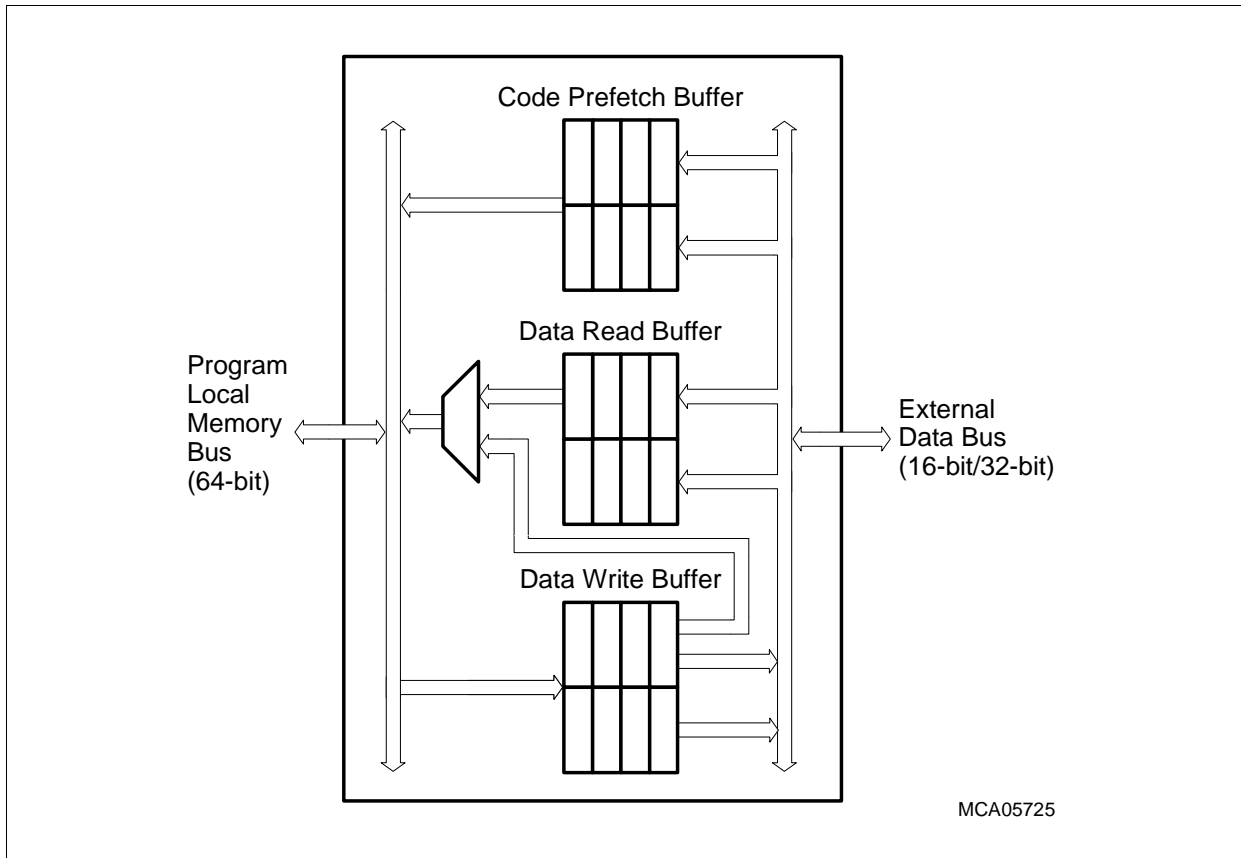


Figure 13-14 EBU PLMB Data Buffering

13.6.1 Data Read Buffer

This buffer consists of four 64 bit words (8×32 -bit) and is used whenever a PLMB master performs a read from a device on the external bus. The buffer is used to assemble the appropriate data width (e.g. 32-bit or 64-bit) to be delivered to the PLMB from the appropriate number of 16-bit or 32-bit data fetched from the external bus.

The EBU provides a bypass feature for the case where a PLMB read access is received which matches a pending write in the Data Write Buffer. In this case the required data is returned directly from the Data Write Buffer. In all other cases the request is translated in to the appropriate number of external read accesses. The bypass feature can be disabled separately for each chip select region via the DLOAD bit in registers EBU_BUSCON or EBU_EMUBC. In the case of an external device at which reading a location does not necessarily return the last value written to that location (e.g. hardware

modified bits in an external peripheral), the user's software must ensure data coherency through the use of the DLOAD bit(s).

When the EBU receives an external bus read access that represents a code fetch (i.e. was generated by the PMU) to a memory region with "pre-fetch" enabled (via bit PREFETCH in registers EBU_BUSCON or EBU_EMUBC), the EBU will perform a code prefetch into the Code Prefetch Buffer.

13.6.2 Code Prefetch Buffer

This buffer consists of four 64-bit words (8×32 -bit) which is enough to service a PLMB four-word burst read request (BTR4). This four-word burst read is sufficient to load the code cache line size of the TriCore CPU. When the code is highly sequential, the prefetch buffer can help to speed up the instruction fetch process. This feature is programmable through the PREFETCH bit in registers EBU_BUSCON or EBU_EMUBC. On an instruction fetch cycle, a transaction originated from the PMU (most likely a cache line refill) takes place on the PLMB. This transaction is handled via the Data Read Buffer. At the end of the transaction, a prefetch activity is triggered to fill up the Code Prefetch Buffer by reading the next four consecutive 64-bit words.

When using Burst Flash memories, the prefetch is performed by extending the access by the appropriate number of cycles to fetch the next four 64-bit words. For asynchronous memories, the prefetch is performed by generating the appropriate number of device read accesses.

The result of the prefetch will be that the Code Prefetch Buffer will be ready with the instructions when the next request for sequential instructions arrives.

This allows the cancellation of a pending prefetch if a new data access starts during the code prefetch access. This feature is separately selectable for each chip select region, and controlled by bit WEAKPREF in registers EBU_BUSCON or EBU_EMUBC.

Note: A code prefetch is only triggered by a PMU read access that is handled via the Data Read Buffer. A PMU read access that is serviced by the Code Prefetch Buffer will not generate a subsequent code prefetch.

13.6.3 Data Write Buffer

This buffer consists of four 64-bit words (8×32 -bit). This is enough to service PLMB write transactions up to and including a PLMB four-word burst write request (BTR4). This four-word burst write is sufficient to support the data cache line size of the TriCore CPU. When any PLMB-to-external-bus data write transaction occurs, this transaction will be translated to the appropriate number of external bus write accesses. The Data Write Buffer allows the EBU to accept data from the PLMB (from a single PLMB transaction) faster than data can be written to the external bus, and also allows the EBU to complete the PLMB request associated with one PLMB-to-external-bus data write transaction while the external bus is busy (e.g. in the case of an ongoing code prefetch).

Note: The Data Write Buffer can only store the data associated with a single write access. If the EBU receives a PLMB request for a write to the external bus and the Data Write Buffer is not available (i.e. a previous write is still pending or underway), the EBU will return an PLMB Retry acknowledge.

For the case where the Data Write Buffer holds the requested read data, a programmable bypass feature is provided from the Data Write Buffer to the Data Read Buffer.

13.7 Standard Access Phases

Accesses to asynchronous devices and Burst Flash devices are composed of a number of standard access phases (according to the type of device and the type of access).

There are six access phases defined:

- Address Phase AP (mandatory for read and write cycles of both device types)
- Command Delay Phase CD (optional)
- Command Phase CP (mandatory for read and write cycles of both device types)
- Data Hold Phase DH (optional, only applies to write cycles)
- Recovery Phase RP (optional)
- Burst Phase BP (mandatory, only applies to burst read cycles)

Throughout the remainder of this document, a short-hand notation is adopted to represent any clock cycle in any phase. This notation consists of two or three letters followed by a number. The letters identify the access phase within which the clock cycle is located (e.g. AP for Address Phase). The number denotes the number of LMBCLK clock cycles within the phase (i.e. 1 = first, etc.). In the case of delays that can be extended by external control inputs the lower case letters “e” and “i” are inserted following the two letter phase identifier to differentiate between internally (“i”) and externally (“e”) generated delays. For example, AP2 identifies the second clock in the Address Phase. CPe3 identifies the third clock in the Command Phase which is being extended by external wait-states.

13.7.1 Address Phase (AP)

The Address Phase is mandatory. It always consists of at least one or more LMBCLK cycles. The phase can be optionally extended to accommodate slower devices.

At the start of the Address Phase, the EBU:

- Selects the device to be accessed by asserting the appropriate \overline{CSx} signal,
- Issues the address which is to be accessed on the address bus,
- Asserts the \overline{ADV} signal low,
- Asserts the appropriate \overline{BCx} signals if these are programmed to be asserted with the \overline{CSx} signal,

LMB External Bus Unit

- Asserts the $\overline{MR/\overline{W}}$ signal according to the type of access to be performed (low in the case of a write access, not used in burst read cycles). This level is retained until the start of the next Address Phase.
- At the end of the Address Phase the EBU returns the \overline{ADV} signal to high.

The length (number of LMBCLK cycles) of the Address Phase is programmed via the EBU_BUSAPx.ADDRC bit field parameter. Additionally, the length of the Address Phase selected via ADDRC can be multiplied by the EBU_BUSCONx.CMULT parameter, but only if bit EBU_BUSCONx.MULTMAP[0] is set (ADDRC multiplier function enabled).

The calculation of the number of LMBCLK cycles in the Address Phase is described in the following table:

MULTMAP	Number of Address Phase Cycles
XXXXXX0 _B	= ADDRC
XXXXXX1 _B	= ADDRC × CMULT

The equivalent control capability is available for bit field EBU_EMUBAP.ADDRC, that can be multiplied by EBU_EMUBCx.CMULT if bit EBU_EMUBC.MULTMAP[0] is set.

When an access is performed at a burst read cycle, the start of the Address Phase is always synchronized to a rising edge of the appropriate BFCLK signal (BFCLK0 for accesses to Type 0 devices and BFCLK1 for accesses to Type 1 devices). This allows the programming of the EBU to correctly latch the data issued by the Burst Flash. BFCLK0 is used during a burst access to cause a Flash device to output the next sequential data value.

13.7.2 Command Delay Phase (CD)

The Command Delay phase is optional. This means that it can also be programmed for a length of zero LMBCLK clock cycles. The CD phase allows for the insertion of a delay between Address Phase and Command Phase(s). This phase accommodates devices that are not fast enough to receive commands immediately after getting the address.

The length (number of LMBCLK cycles) of the Command Delay phase is programmed via the EBU_BUSAPx.CMDDELAY bit field. This parameter makes it possible to select between zero to seven Command Delay phases. Additionally, the number of Command Delay phases as selected via ADDRc can be multiplied by the EBU_BUSCONx.CMULT parameter, but only if bit EBU_BUSCONx.MULTMAP[2] is set (CMDDELAY multiplier function enabled).

The calculation of the number of LMBCLK cycles in the Command Delay phase is defined in the following table:

MULTMAP	Number of Command Delay Phase Cycles
XXXX0XX _B	= CMDDELAY
XXXX1XX _B	= CMDDELAY × CMULT

The equivalent control capability is available for bit field EBU_EMUBAP.CMDDELAY, which can be multiplied by EBU_EMUBC.CMULT if bit EBU_EMUBC.MULTMAP[2] is set.

13.7.3 Command Phase (CP)

The Command Phase is mandatory. It always consists of at least one or more LMBCLK cycles. The phase can optionally be extended to accommodate slower devices.

The length (number of LMBCLK cycles) of the Command Phase is separately programmable for read and write accesses. Bit field EBU_BUSAPx.WAITRDC determines the basic length of Command Phases during read cycles and bit field EBU_BUSAPx.WAITWRC determines the basic length of Command Phases during write cycles. One to seven LMBCLK cycles can be programmed for these two parameters. The total number of LMBCLK cycles in Command Phases during read and write accesses is calculated by multiplying WAITRDC and WAITWRC by the parameter as selected by bit field EBU_BUSCONx.CMULT (independently of bit field EBU_BUSCONx.MULTMAP).

The calculation of the number of Command Phase LMBCLK cycles is described in the following table:

Cycle Type	Number of Command Phase Cycles
Read Cycle	= WAITRDC × CMULT
Write Cycle	= WAITWRC × CMULT

The equivalent control capability is available for bit fields EBU_EMUBAP.WAITRDC and EBU_BUSAPx.WAITWRC, which are multiplied by EBU_EMUBC.CMULT.

Additionally, when accessing asynchronous devices, a Command Phase can also be extended externally using the WAIT signal when the region being accessed is programmed for external command delay control via bit EBU_BUSCONx.WAIT or EBU_EMUBC.WAIT.

The Command Phase is further subdivided into:

- CPi (= internally-programmed Command Phase)
- CPe (= externally-prolonged Command Phase, i.e. prolonged by the assertion of the WAIT signal).

At the start of the Command Phase, the EBU:

- Asserts the appropriate control signal \overline{RD} or RD/\overline{WR} low according to the access type (read or write),
- Issues the data to be written on the data bus D[31:0] (in the case of a write cycle),
- Asserts the appropriate \overline{BCx} low (in the case where \overline{BCx} is programmed to be asserted with the \overline{RD} or RD/\overline{WR} signals).

At the end of the Command Phase during an asynchronous access, the EBU:

- Returns the appropriate control signal \overline{RD} or RD/\overline{WR} high according to the type of access type (read or write),
- Latches the data from the data bus D[31:0] (in the case of a read cycle),
- Returns the appropriate \overline{BCx} high (in the case where \overline{BCx} is programmed to be asserted with the \overline{RD} or RD/\overline{WR} signals).

The EBU takes no action at the end of the Command Phase during an access to a Burst Flash device (i.e. no signals are changed or sampled).

During accesses to Burst Flash devices, the Command Phase is always extended by zero or more LMBCLK cycles to ensure that the end of the Command Phase coincides with a positive edge of the appropriate BFCLKO (Burst Flash clock output) signal.

13.7.4 Data Hold Phase (DH)

The Data Hold phase is optional. This means that it can also be programmed for a length of zero LMBCLK clock cycles. Furthermore, it is only available for asynchronous write accesses. The Data Hold phase extends the amount of time for which data is still held on the bus after the rising edge of the RD/ $\overline{\text{WR}}$ signal occurred. The Data Hold phase is used to accommodate external devices that require a data hold time after the rising edge of the RD/ $\overline{\text{WR}}$ signal. The length (number of LMBCLK cycles) of the Data Hold phase is programmed via the EBU_BUSAPx.DATAC bit field. The total number of Data Hold phase clock cycles is calculated by multiplying the DATAC parameter (independently of bit field EBU_BUSCONx.MULTMAP) with EBU_BUSCONx.CMULT.

The calculation of the number of LMBCLK cycles in the Data Hold phase is described in the following table:

Access Type	Number of Data Hold Phase Cycles
Asynchronous Write Cycle	= DATAC × CMULT

The equivalent control capability is available for bit field EBU_EMUBAP.DATAC, which is multiplied by EBU_EMUBC.CMULT.

Additionally, when accessing Nand Flash Devices, a Data Hold phase can also be extended externally by asserting the $\overline{\text{WAIT}}$ signal when the region being accessed is programmed for external data hold delay control via bit field EBU_BUSCONx.WAIT.

13.7.5 Burst Phase (BP)

The Burst Phase is mandatory during burst read accesses. At the end of the Burst Phase the EBU reads data from the data bus. During a burst read access, Burst Phases are repeated as many times as required in order to read the required amount of data from the Burst Flash device.

At the start of the first Burst Phase during a burst read access, the EBU:

- Drives the $\overline{\text{BAA}}$ signal low to cause the Burst Flash device to advance the address with each subsequent BFCLKO positive edge.

At the end of the last Burst Phase during a burst read access, the EBU:

- Returns the $\overline{\text{BAA}}$ signal high,
- Returns the $\overline{\text{CSx}}$ signal high,
- Returns the $\overline{\text{RD}}$ signal high.

The total number of Burst Phase cycles is calculated by multiplying the EBU_BUSAPx.BURSTC parameter with EBU_BUSCONx.CMULT but only if bit EBU_BUSCONx.MULTMAP[3] is set (BURSTC multiplier function enabled).

The calculation of the number of Burst Phase cycles is defined by the following table:

MULTMAP	Number of Burst Phase Cycles
XXX0XXX _B	= BURSTC
XXX1XXX _B	= BURSTC × CMULT

The equivalent control capability is available for bit field EBU_EMUBAP.BURSTC, which is multiplied by EBU_EMUBC.CMULT.

During accesses to Burst Flash devices the length of the Burst Phase must be programmed such that the end of the Burst Phase always coincides with a positive edge of the appropriate BFCLKO (Burst Flash Clock) signal.

13.7.6 Recovery Phase (RP)

The Recovery Phase is optional. This means that it can also be programmed for a length of zero LMBCLK clock cycles. This phase allows the insertion of a delay following an external bus access that delays the start of the Address Phase for the next external bus access. This permits flexible adjustment of the delay between accesses to the various external devices. The following individually programmable delays are provided on a region by region basis for the following conditions:

- Bit fields EBU_BUSAPx.RDRECOVC and EBU_EMUBAPx.RDRECOVC determine the basic length of the Recovery Phase after a read access.
- Bit fields EBU_BUSAPx.WRRECOVC and EBU_EMUBAPx.WRRECOVC determine the basic length of the Recovery Phase after a write access.
- Bit fields EBU_BUSAPx.DTARDWR and EBU_EMUBAPx.DTARDWR determine the length (basic number of LMBCLK clock cycles) of the Recovery Phase after a write access that is followed by a read access, or after a read access that is followed by a write access.
- Bit fields EBU_BUSAPx.DTACS and EBU_EMUBAPx.DTACS determine the length (basic number of LMBCLK clock cycles) of the Recovery Phase after a read/write access of one region that is followed by a read/write access of another region.

The calculation of the total numbers of Recovery Phase clock cycles for each of the four Recovery Phase parameters is described in the following table:

Parameter	MULTMAP	Number of Burst Phase Cycles
RDRECOVC	X0XXXXX _B	= RDRECOVC
	X1XXXXX _B	= RDRECOVC × CMULT
WRRECOVC	0XXXXXX _B	= RDRECOVC
	1XXXXXX _B	= RDRECOVC × CMULT
DTARDWR	–	= DTARDWR × CMULT
DTACS	–	= DTACS × CMULT

The EBU implements a “highest wins” algorithm to ensure that the longest applicable recovery delay is always used between consecutive accesses to the external bus. [Table 13-15](#) shows the scheme for determining this delay for all possible circumstances. For example, if a read access to a region associated with CS1 is followed by a write to a region associated with CS2, the delay will be the highest of DTARDWR, DTACS and RDRECOVC. In this case, if DTARDWR is greater than DTACS and RDRECOVC, then the number of recovery cycles between the two accesses is DTARDWR.

Table 13-15 Parameters for Recovery Phase

Region	Case		Parameter(s) used to calculate “Highest Wins” Recovery Phase
	Current Access	Next Access	
Same \overline{CSn}	Read	Read	RDRECOVC
	Write	Write	WRRECOVC
	Read	Write	RDRECOVC, DTARDWR
	Write	Read	WRRECOVC, DTARDWR
Different \overline{CSn}	Read	Read	DTACS, RDRECOVC
	Write	Write	DTACS, WRRECOVC
	Read	Write	DTACS, RDRECOVC, DTARDWR
	Write	Read	DTACS, WRRECOVC, DTARDWR

13.7.7 Multiplication Factor for Access Phase Length

As discussed in the previous sections, the length of each access phase is programmable as a multiple of the LMBCLK clock period. Since the LMB clock runs significantly faster than most external devices, some devices may need the EBU to be programmed to use a large number of LMBCLK clock cycles in a particular phase (or phases). Programming phases for a large number of LMBCLK clock cycles is supported by provision of a multiplication factor (or prescaler) scheme. This allows configurations in which one (or more) of the phases is relatively much longer than the others, and provides a flexible scheme that allows the user to optimize device accesses by fine tuning the individual phases for the specific device access characteristics.

A specific multiplier (effectively a clock prescaler) is provided for each chip select region *x*. The multiplier values are controlled by the EBU_BUSCON*x*.CMULT (and EBU_EMUBC.CMULT) bit fields, providing multiplication factors of 1, 4, 8, 16, and 32.

As some types of device timing requirements for a number of the access phases tend to be larger than others, these phase length settings are always multiplied by the appropriate multiplier value. These phases length settings are:

- Number of Command Phase cycles during read accesses (WAITRDC)
- Number of Command Phase cycles during write accesses (WAITWRC)
- Number of Recovery Phase cycles when switching from a read access to a write access or vice versa (DTARDWR)
- Number of Recovery Phase cycles when switching between different regions (DTACS)

Each of the remaining phase length settings can be individually programmed whether or not they are multiplied by CMULT. These phase length settings are:

- Number of Address Phase cycles (ADDRC)
- Number of Command Delay Phase cycles (CMDDELAY)
- Number of Data Hold Phase cycles (DATAC)
- Number of Recovery Phase cycles following a read access (RDRECOVC)
- Number of Recovery Phase cycles following a write access (WRRECOVC)
- Number of Burst Phases cycles (BURSTC)

The selection (enable/disable control) which of these phase length settings are multiplied by the multiplication factor is selected by the bits in the multiplication factor bit fields EBU_BUSCON*x*.MULTMAP or EBU_EMUBC.MULTMAP.

13.8 Asynchronous Read/Write Accesses

Asynchronous read/write access of the EBU support the following features:

- LMBCLK clock-synchronous signal generation
- Support for 16-bit and 32-bit bus width
Performing an PLMB access with a data width greater than that of the external device automatically triggers a sequence of the appropriate number of external accesses to match the PLMB access width.
- Demultiplexed address/data lines
- Programmable access parameters
 - Internal control of command delay cycles
 - External and/or internal control of wait states
 - Variable data hold cycles for write operation (to allow flexible hold time adjustment)
 - Variable inactive/recovery cycles when:
 - Switching between different memory regions (CS),
 - Switching between read and write operations,
 - After each read cycle,
 - After each write cycle.

Software driver routines are required in order to support Nand Flash devices using asynchronous device accesses. A single Nand Flash access sequence is performed by generating the appropriate sequence of discrete asynchronous device accesses in software.

The EBU does not provide support 8-bit bus width. When 8-bit SRAM devices are used, they must be used in pairs to implement either a 16-bit or 32-bit wide memory region.

13.8.1 Signal List

The following signals of the EBU are used for asynchronous accesses:

Table 13-16 Asynchronous Mode Signal List

Signal/Pin	Type	Function
D[31:0]	O	Data bus lines 0-31
A[23:0]	O	Address bus lines 0-23
CS[3:0]	O	Chip select 0-3
RD	O	Read control line
RD/W \bar{R}	O	Write control line
MR/W \bar{R}	O	Motorola-style read/write control line
BC[3:0]	O	Byte control lines 0-3
WAIT	I	Wait input

13.8.2 Demultiplexed Device Configurations

The EBU supports two different configurations of demultiplexed memory/peripheral devices: 16-bit demultiplexed mode and 32-bit demultiplexed mode.

The 16-bit demultiplexed mode is selected by:

- EBU_BUSCONx.PORTW = 01_B
- EBU_BUSCONx.AGEN = 000_B

For 16-bit demultiplexed devices, the complete 24-bit address is driven at the EBU pins for the duration of the complete external bus cycle. The 16-bit data is driven to/read from the D[15:0] pins during the data phase. D[31:16] are not used.

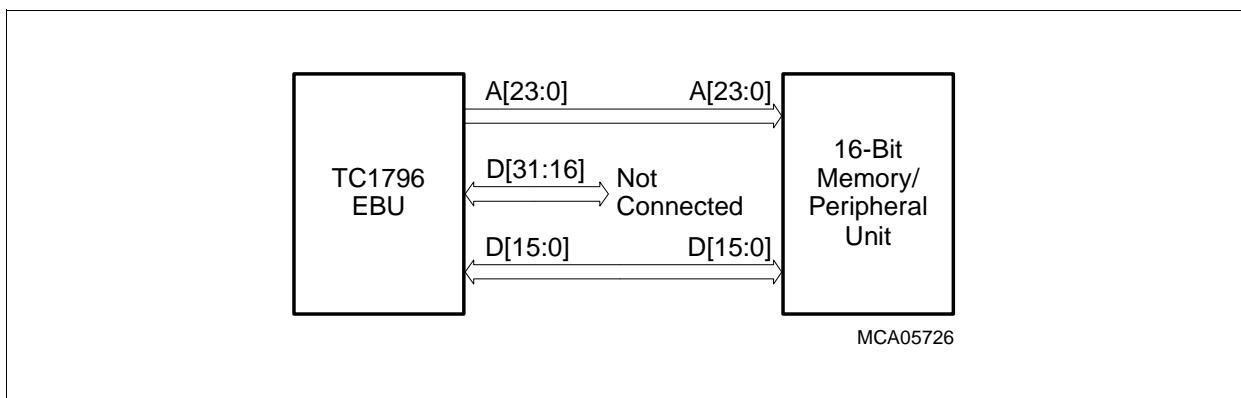


Figure 13-15 Connection of a 16-bit Demultiplexed Device

The 32-bit demultiplexed mode is selected by:

- EBU_BUSCONx.PORTW = 10_B
- EBU_BUSCONx.AGEN = 000_B

For 32-bit demultiplexed devices, the complete 24-bit address is driven at the EBU pins for the duration of the complete external bus cycle. The 32-bit data is driven to/read from the D[31:0] pins during the data phase.

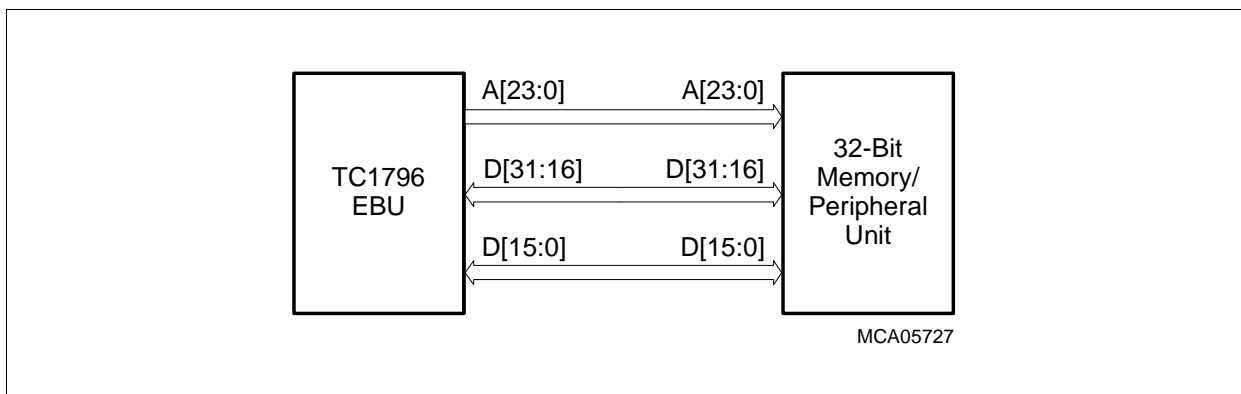


Figure 13-16 Connection of a 32-bit Demultiplexed Device

LMB External Bus Unit

Figure 13-17 shows a typical connection for Intel-style and Motorola-style peripherals. The MR/W signal indicates the data direction for the current transfer, and can be used to control the data direction through the buffer for the D[31:0] bus (as well as controlling whether an access to a Motorola-Style device is read or write).

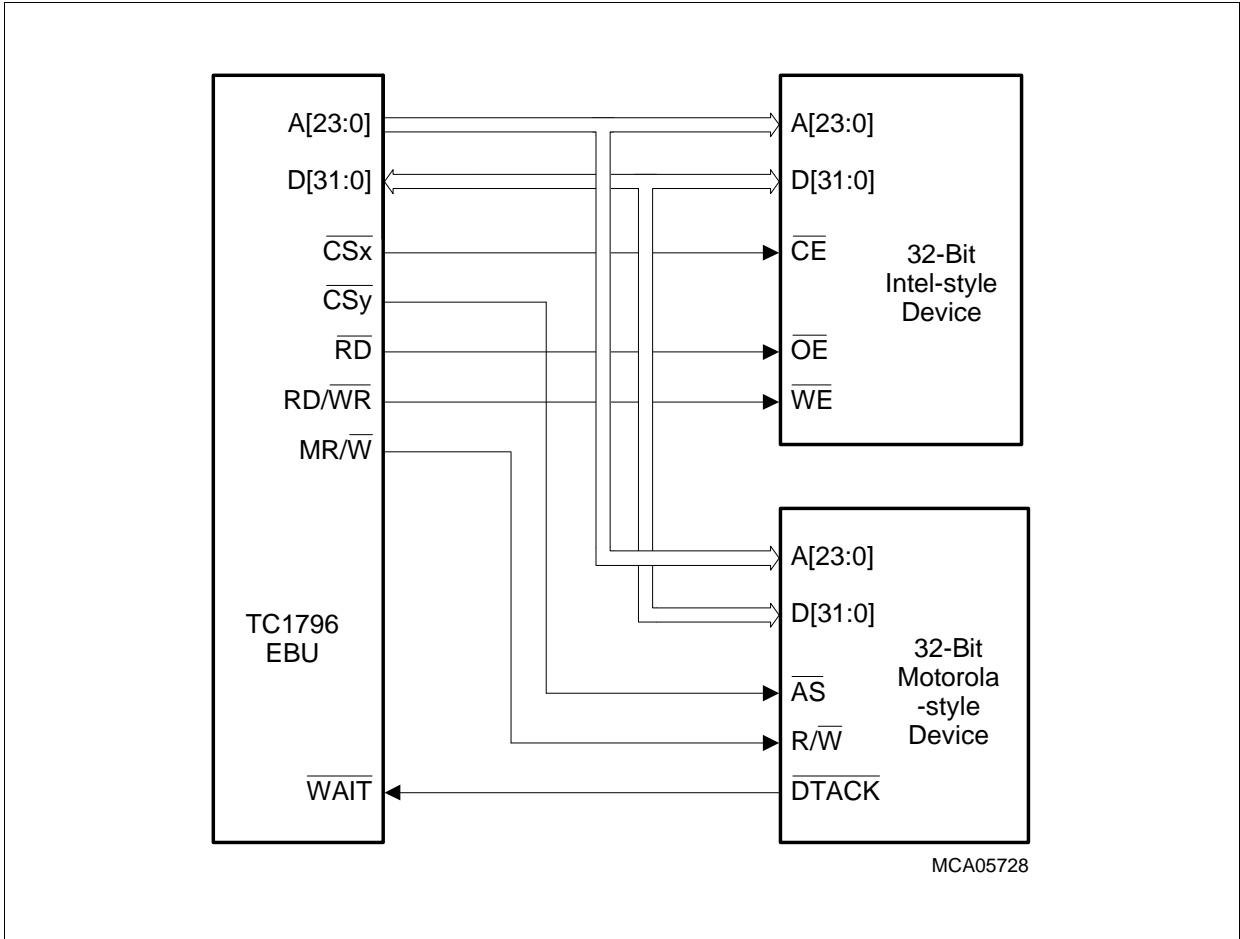


Figure 13-17 Typical Connection of Asynchronous Devices

13.8.3 Standard Asynchronous Access Phases

Accesses to asynchronous devices are composed of a subset of the standard access phases which are detailed in [Section 13.7](#). The standard access phases for asynchronous devices are:

- AP: Address Phase (compulsory - see [Page 13-41](#))
- CD: Command Delay Phase (optional - see [Page 13-43](#))
- CP: Command Phase (compulsory - see [Page 13-43](#))
- DH: Data Hold Phase (optional - see [Page 13-45](#))
- RP: Recovery Phase (optional - see [Page 13-47](#))

13.8.4 Programmable Parameters

[Table 13-17](#) lists the programmable parameters for asynchronous accesses. These parameters only apply to asynchronous devices when `EBU_BUSCONx.AGEN = 000B`. Note that emulation registers “EBU_EMU...” include parameters that control the emulator chip select region (`CSEMU` output), while “EBU_BUS...x” registers include parameters that control the four `CS[3:0]` chip select regions x. The equivalent registers contain identical bits and bit fields.

Table 13-17 Asynchronous Access Programmable Parameters

Register	Parameter (Bit/Bit field)	Function
EBU_BUSAPx EBU_EMUBAP	ADDRC	Number of cycles in address phase; can be multiplied by CMULT.
	CMDDELAY	Number of programmed command delay cycles; can be multiplied by CMULT.
	WAITRDC	Number of programmed wait states for read accesses; always multiplied by CMULT.
	WAITWRC	Number of programmed wait states for write accesses; always multiplied by CMULT.
	DATAAC	Number of Data Hold cycles; always multiplied by CMULT.

LMB External Bus Unit

Table 13-17 Asynchronous Access Programmable Parameters (cont'd)

Register	Parameter (Bit/Bit field)	Function
EBU_BUSAPx EBU_EMUBAP	RDRECOVC	Number of minimum recovery cycles after a read access; can be multiplied by CMULT.
	WRRECOVC	Number of minimum recovery cycles after a write access; can be multiplied by CMULT.
	DTARDWR	Number of minimum recovery cycles between a read access and a write access; always multiplied by CMULT.
	DTACS	Number of minimum recovery cycles when the next access going to a different memory region; always multiplied by CMULT.
EBU_BUSCONx EBU_EMUBC	WAIT	External Wait State control (OFF, asynchronous, synchronous)
	WAITINV	Reversed polarity at $\overline{\text{WAIT}}$: active low or active high
	CMULT	Common (cycle) multiplier control ($\times 1$, $\times 4$, $\times 8$, $\times 16$, $\times 32$)
	MULTMAP	Multiplier map, each bit enables multiplier function of bits and bit fields ADDR, CMDDELAY, DATA, RDRECOVC, and WRRECOVC.

13.8.5 Accesses to Demultiplexed Devices

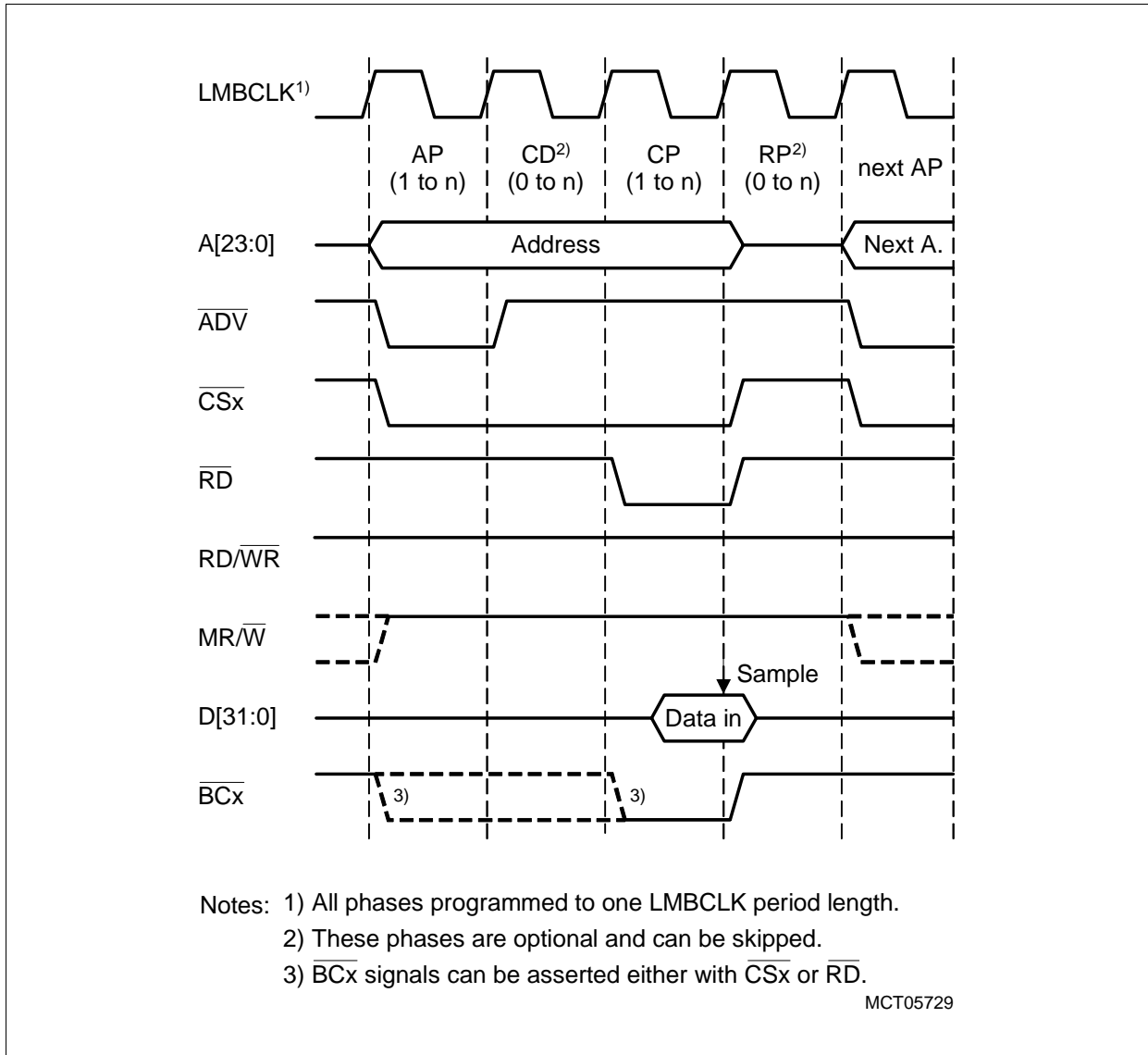


Figure 13-18 Demultiplexed External Bus Read Access Cycle

Figure 13-18 above shows an example of a read access to a demultiplexed device. This type of access cycle consists of two to four phases as follows:

- Address Phase (compulsory)
- Command Delay Phase (optional)
- Command Phase (compulsory)
- Recovery Phase (optional)

The example shows an access cycle with all four phases programmed for a duration of one LMBCLK cycle.

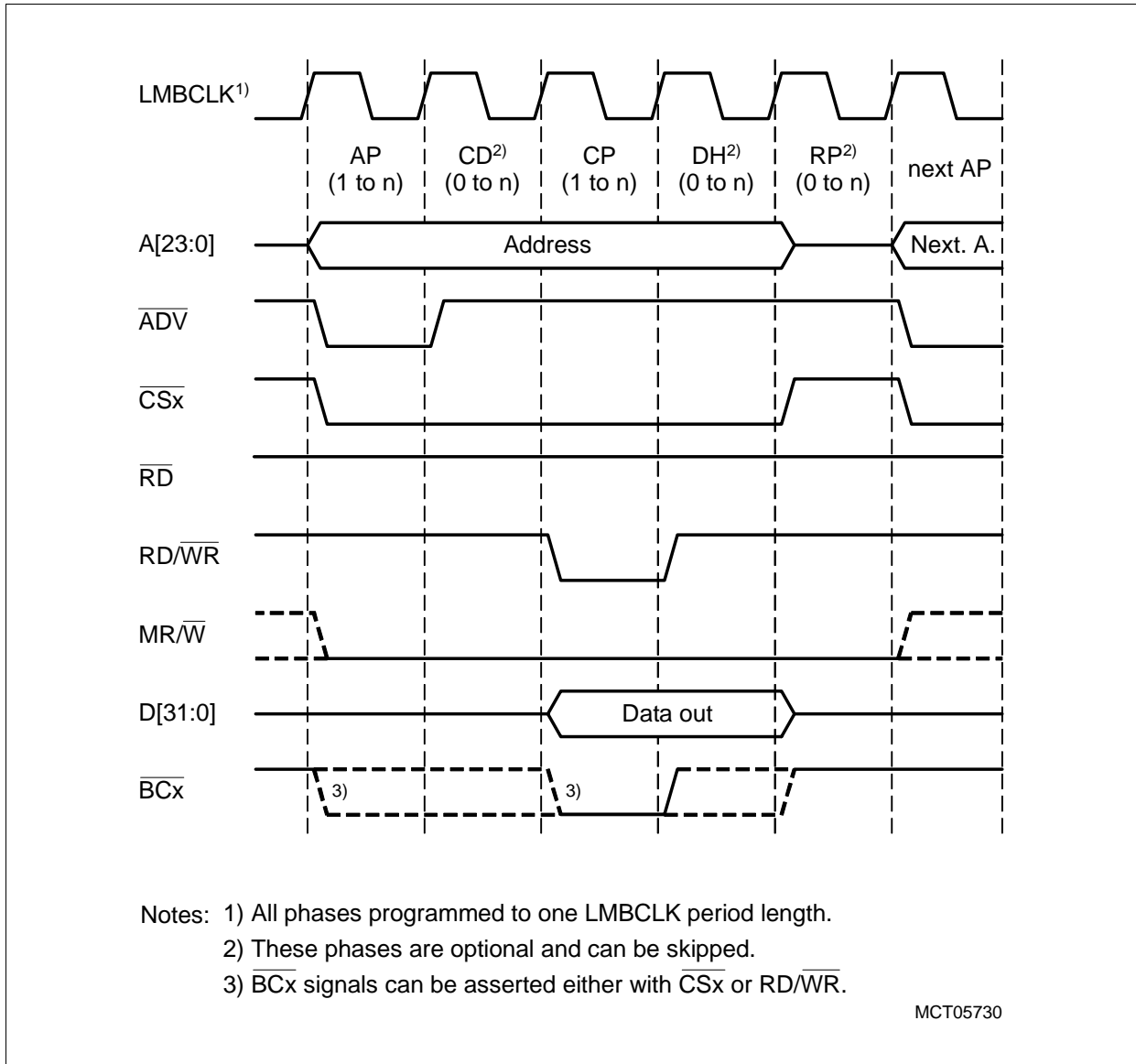


Figure 13-19 Demultiplexed External Bus Write Access Cycle

Figure 13-19 above shows an example of a write access to a demultiplexed device. This type of access cycle consists of two to five phases as follows:

- Address Phase (compulsory)
- Command Delay Phase (optional)
- Command Phase (compulsory)
- Data Hold Phase (optional)
- Recovery Phase (optional)

The example shows an access cycle with all five phases programmed for a duration of one LMBCLK cycle.

13.8.6 Dynamic Command Delay and Wait State Insertion

In general, there are two critical phases during asynchronous device accesses. These phases are:

- **Command Delay Phase** (see [Page 13-43](#)).
- **Command Phase** (see [Page 13-43](#)).

In the EBU, internal length programming for these two phases is available via bit field EBU_BUSAPx.CMDDELAY. This bit field determines the basic number of Command Delay phase clock cycles. The total number of Command Delay phase clock cycles further depends on the settings of bit field EBU_BUSCONx.CMULT and EBU_BUSCONx.MULTMAP[2].

The equivalent control capability is available for bit fields EBU_EMUBAP.CMDDELAY, EBU_EMUBCx.CMULT, and EBU_EMUBCx.MULTMAP[2]. The $\overline{\text{WAIT}}$ functionality as described in the next two subsections, has also an equivalent control capability for the bit fields EBU_EMUBC.CMDDELAY, EBU_EMUAP.WAITRDC, and EBU_EMUAP.WAITWRC.

13.8.6.1 External Extension of the Command Phase by $\overline{\text{WAIT}}$

The $\overline{\text{WAIT}}$ input can be used to cause the EBU to extend the Command Phase by inserting additional cycles prior to deactivation of the $\overline{\text{RD}}$ and RD/WR lines. This signal can be programmed separately for each region to be ignored or sampled either synchronously or asynchronously (selected via the EBU_BUSCONx.WAIT or EBU_EMUBC.WAIT bit field). Additionally, the polarity of $\overline{\text{WAIT}}$ can be programmed for active low (default after reset) or active high function via bit EBU_BUSCONx.WAITINV or EBU_EMUBC.WAITINV. The signal will only take effect after the programmed number of Command Phase cycles has passed. This means that the signal can only be used to extend the phase, not to shorten it.

When programmed for synchronous operation, $\overline{\text{WAIT}}$ is sampled on every rising edge of LMBCLK during the Command Phase. The sampled value is then used on the next rising edge of LMBCLK to decide whether to prolong the Command Phase or to start the next phase. [Figure 13-20](#) shows an example of $\overline{\text{WAIT}}$ used in Synchronous Mode.

Note: Due to the one-cycle delay in Synchronous Mode between the sampling of the $\overline{\text{WAIT}}$ input and its evaluation by the EBU, the Command Phase must always be programmed to be at least one LMBCLK cycle (via EBU_BUSAPx.WAITRDC or EBU_BUSAPx.WAITWRC) in this mode.

When programmed for asynchronous operation, $\overline{\text{WAIT}}$ is also sampled at each rising edge of LMBCLK during the Command Phase. However, an extra synchronization cycle is inserted prior to the use of the sampled value. This means that the sampled value is not used until the second following rising edge of LMBCLK. [Figure 13-21](#) shows an example of $\overline{\text{WAIT}}$ used in Asynchronous Mode.

LMB External Bus Unit

Note: Due to the two-cycle delay in Asynchronous Mode between the sampling of the $\overline{\text{WAIT}}$ input and its evaluation by the EBU, the Command Phase must always be programmed to be at least two LMBCLK cycles (via `EBU_BUSAPx.WAITRDC` or `EBU_EMUBAP.WAITWRC`) in this mode.

Figure 13-20 shows an example of the extension of the Command Phase through the $\overline{\text{WAIT}}$ input in synchronous mode:

- At LMBCLK edge 1 (at the end of the Address Phase), the EBU samples the $\overline{\text{WAIT}}$ input as low and starts the first cycle of the Command Phase (C*P*_i1 - internally programmed).
- At LMBCLK edge 2, the EBU samples the $\overline{\text{WAIT}}$ input as low and starts an additional Command Phase cycle (C*P*_e2 - externally generated) as a result of the $\overline{\text{WAIT}}$ input sampled as low at LMBCLK edge 1.
- At LMBCLK edge 3, the EBU samples the $\overline{\text{WAIT}}$ input as high and starts an additional Command Phase cycle (C*P*_e3 - externally generated) as a result of the $\overline{\text{WAIT}}$ input sampled as low at LMBCLK edge 2.
- Finally at LMBCLK edge 4, as a result of the $\overline{\text{WAIT}}$ input sampled as high at point 3, the EBU terminates the Command Phase, reads the input data from D[31:0] and starts the Recovery Phase.

Note: Synchronous operation means that even though access to the device may be asynchronous, the control logic generating the control signals must meet setup and hold time requirements with respect to LMBCLK.

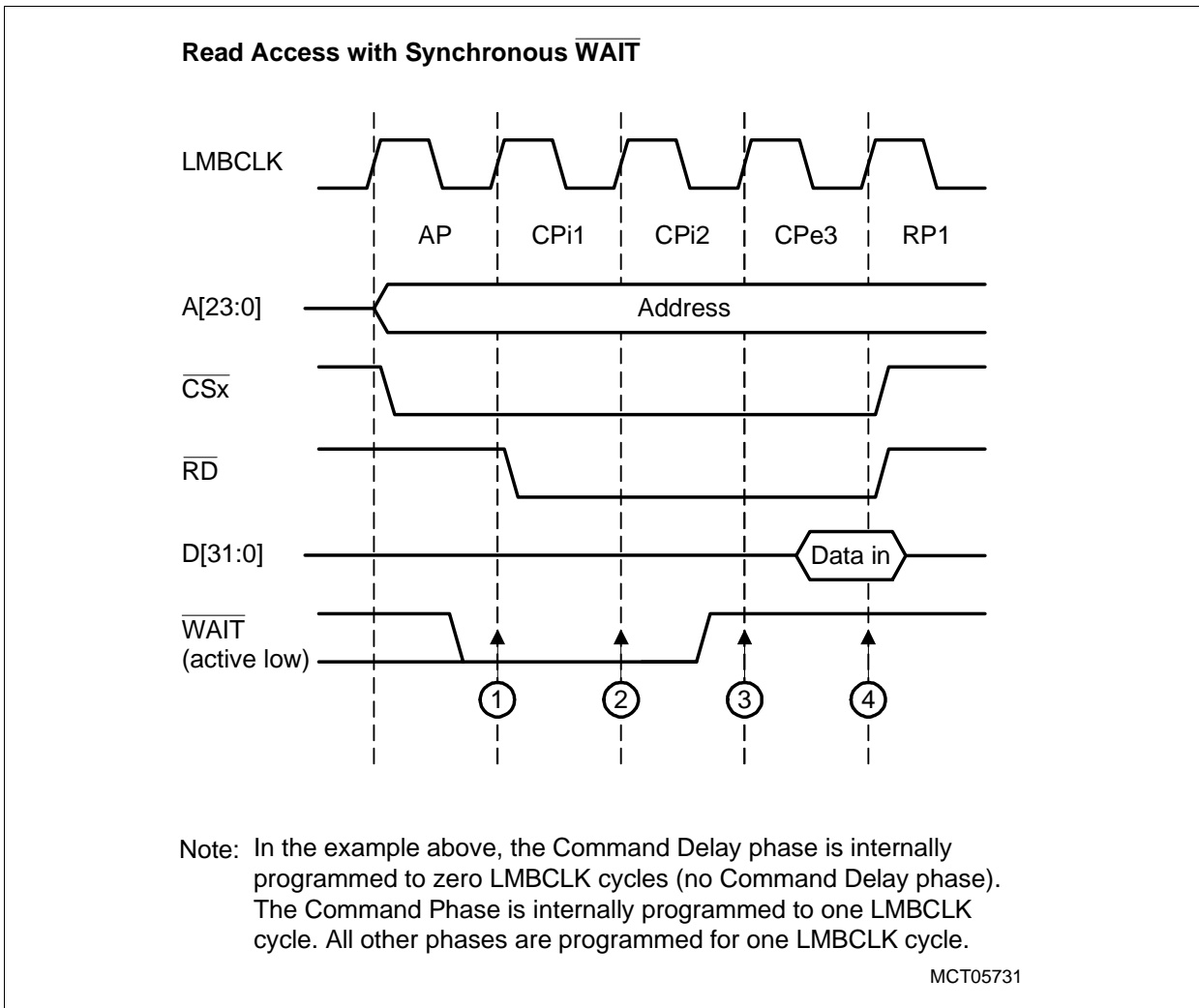


Figure 13-20 External Wait Insertion (Synchronous Mode)

Figure 13-21 shows an example of the extension of the Command Phase through the $\overline{\text{WAIT}}$ input in asynchronous mode:

- At LMBCLK edge 1 (at the end of the Address Phase), the EBU samples the $\overline{\text{WAIT}}$ input as low and starts the first cycle of the Command Phase (CPI1 - internally programmed).
- At LMBCLK edge 2, the EBU samples the $\overline{\text{WAIT}}$ input as low and starts the second cycle of the Command Phase (CPI2 - internally programmed).
- At LMBCLK edge 3, the EBU samples the $\overline{\text{WAIT}}$ input as high and starts an additional Command Phase cycle (CPE3 - externally generated) as a result of the $\overline{\text{WAIT}}$ input sampled as low at LMBCLK edge 1.
- At LMBCLK edge 4, the EBU starts an additional Command Phase cycle (CPE4 - externally generated) as a result of the $\overline{\text{WAIT}}$ input sampled as low at LMBCLK edge 2.

LMB External Bus Unit

- Finally at LMBCLK edge 5, as a result of the $\overline{\text{WAIT}}$ input sampled as high at LMBCLK edge 3, the EBU terminates the Command Phase, reads the input data from D[31:0], and starts the Recovery Phase.

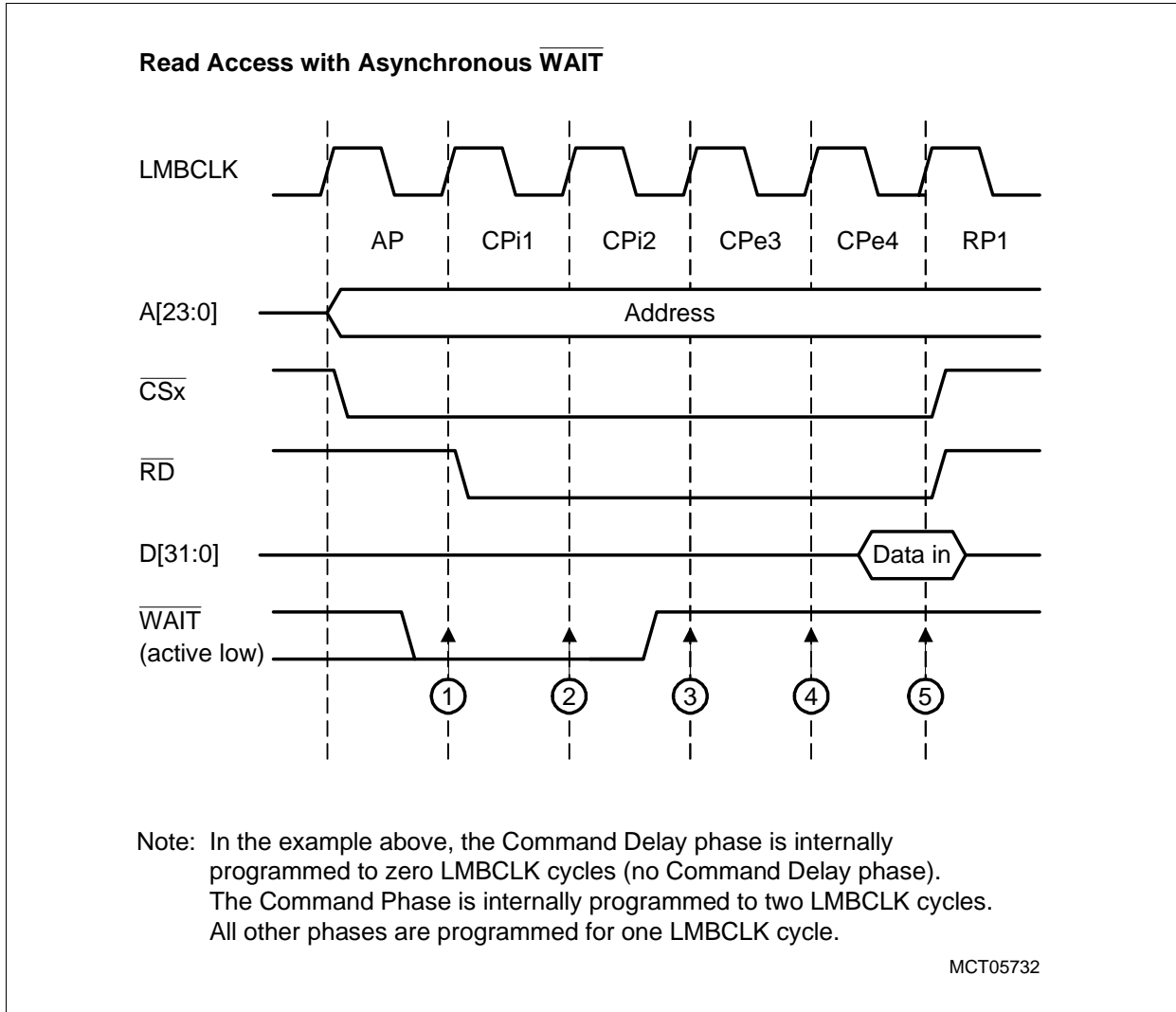


Figure 13-21 External Wait Insertion (Asynchronous Mode)

13.8.6.2 Interfacing to INTEL-style Devices

Figure 13-22 shows an example of accessing an INTEL-style demultiplexed device for read and write accesses. It shows the insertion of delay cycles (shown shaded) to adjust the access cycle to the device's timing requirements.

Both read and write accesses start with a two-cycle Address Phase followed by a two-cycle Command Delay Phase.

For the read access, the Command Delay Phase is followed by a three-cycle Command Phase. At the end of the Command Phase, the data is read (latched) by the EBU. A one-cycle Recovery Phase is inserted at the end of the cycle. At the start of this Recovery Phase, all control signals return to their non-active levels.

For the write access, the Command Delay Phase is followed by a two cycle Command Phase. During a write access, it is possible to insert a Data Hold Phase to satisfy the data hold time requirements of the device. In the example the Data Hold Phase consists of two cycles (DH1 and DH2). During the Data Hold Phase, the RD/WR control signal is driven to the non-active state, but the data and address are still driven on the bus.

LMB External Bus Unit

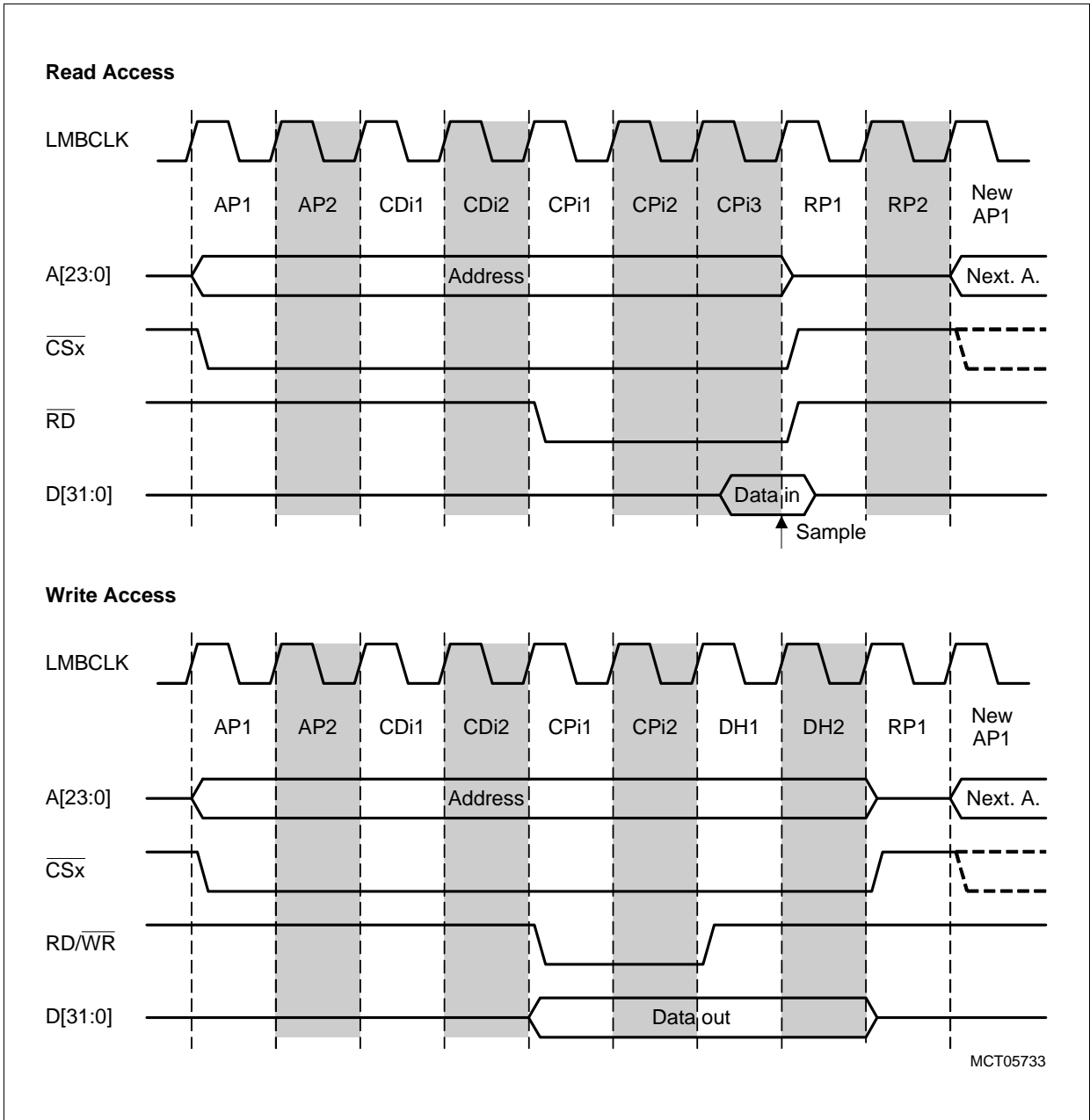


Figure 13-22 Example of an INTEL-style Demultiplexed Device Access

13.8.6.3 Interfacing to Motorola-style Devices

Figure 13-23 gives an example of accessing a Motorola-style demultiplexed device for read and write accesses. The chip select signal \overline{CS}_x is used to generate the \overline{AS} input for the Motorola-style device. The $\overline{MR}/\overline{W}$ signal maintains its level (according on whether the cycle is read or write) from the Address Phase to the Recovery Phase during each access.

Externally controlled wait states (see [Page 13-57](#)) are used to synchronize with the \overline{DTACK} output from the device. A Motorola-style device asserts \overline{DTACK} during a read access to signal that the data is available on the data bus. During a write access, the falling edge of \overline{DTACK} indicates that the Motorola-style device has completed the write access and the data can be removed from the data bus. This can be accomplished by use of the \overline{WAIT} signal with inverse polarity (selected by $EBU_BUSCONx.WAITINV = 1$) with the following limitations:

1. During read accesses, a minimum command phase of three LMBCLK cycles must be used to ensure correct recognition of the \overline{WAIT} input.
2. Only the falling edge of \overline{DTACK} is recognized by the \overline{WAIT} pin, and the user must insert sufficient recovery cycles to ensure that the \overline{DTACK} pin has returned high before the next external bus cycle starts (i.e. before the next falling edge of any \overline{CS}_x signal). Typically, Motorola-style devices de-assert \overline{DTACK} after the rising edge of their \overline{AS} input (which is connected to a EBU \overline{CS}_x output).

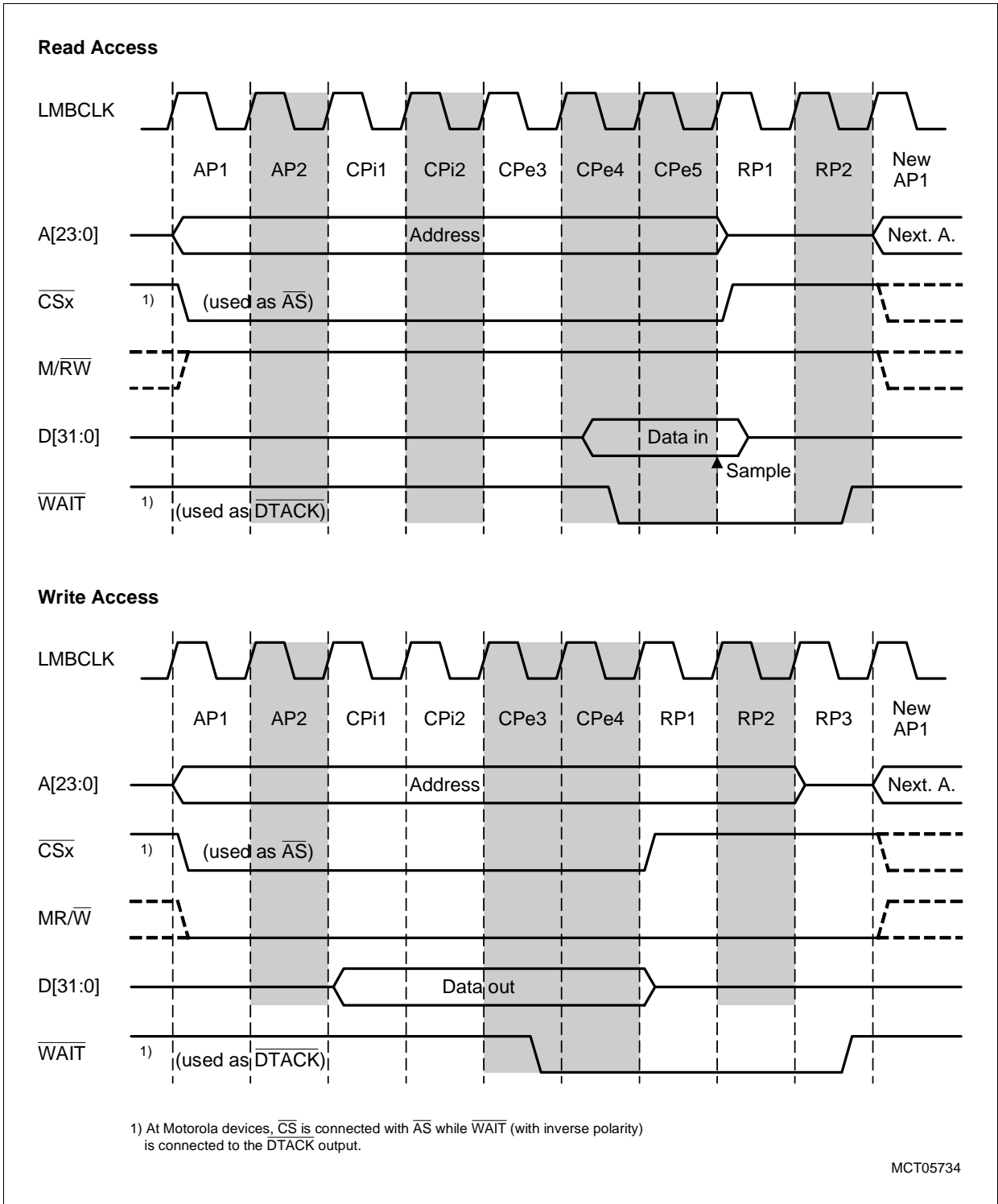


Figure 13-23 Example of a Motorola-style Demultiplexed Device Access

13.9 Burst Mode Read Accesses

Synchronous burst read accesses of the EBU support the following features:

- Fully synchronous timing with flexible programmable timing parameters (address cycles, read wait cycles, data cycles)
- Simultaneous support for two different Burst Flash Types
- Programmable $\overline{\text{WAIT}}$ operation
- Programmable Burst Mode and burst length
- 16-bit or 32-bit data bus width
- Re-synchronization of read data to a feedback clock to maximize the frequency of operation

When it is necessary to perform non-burst accesses to Burst Flash devices, it is necessary to reprogram the characteristics of the corresponding chip select region for the appropriate asynchronous access.

13.9.1 Signal List

The following signals of the EBU are used for the burst read accesses:

Table 13-18 Burst Flash Mode Signal List

Signal	Type	Function
D[31:0]	I	Data bus lines 0-31
A[23:0]	O	Address bus outputs 0-23
$\overline{\text{ADV}}$	O	Address valid strobe output
$\overline{\text{CS}}[3:0]$	O	Chip select outputs 0-3
$\overline{\text{RD}}$	O	Read control output
$\overline{\text{BAA}}$	O	Burst address advance control output
$\overline{\text{WAIT}}$	I	Wait/terminate burst control input
BFCLKO	O	Burst Flash clock output
BFCLKI	I	Burst Flash clock input (feedback)

13.9.2 Burst Flash Memory Configurations

The EBU supports two different configurations of Burst Flash memory devices: 16-bit mode and 32-bit mode. Both modes are demultiplexed modes with separate address and data lines.

The 16-bit Burst Flash mode is selected by:

- EBU_BUSCONx.PORTW = 01_B
- EBU_BUSCONx.AGEN = 010_B (Burst Flash Type 0) or 101_B (Burst Flash Type 1)

For 16-bit Burst Flash devices, the complete 24-bit address is driven on the EBU pins during the address phase(s). The 16-bit data is read from the D[15:0] pins during the burst phases. D[31:16] are not used/connected.

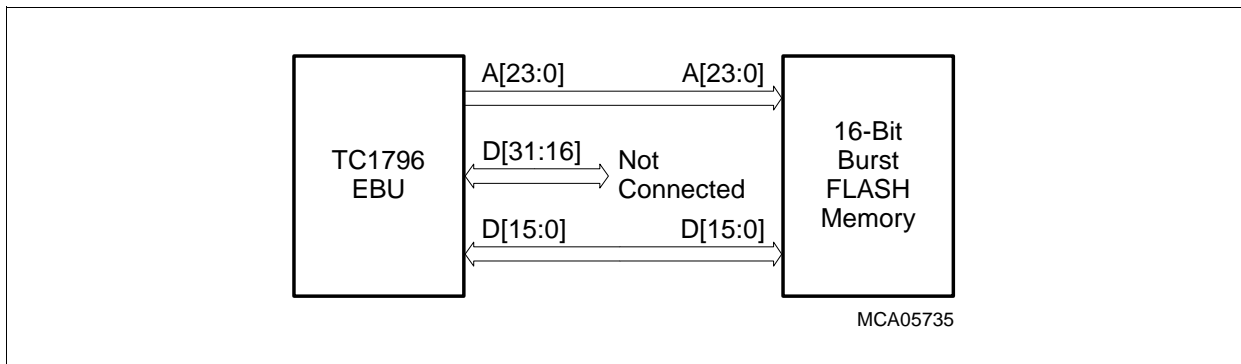


Figure 13-24 Connection of a 16-bit Burst Flash Memory

The 32-bit Burst Flash mode is selected by:

- EBU_BUSCONx.PORTW = 10_B
- EBU_BUSCONx.AGEN = 010_B (Burst Flash Type 0) or 101_B (Burst Flash Type 1)

For 32-bit Burst Flash devices, the complete 24-bit address is driven at the EBU pins during the address phase(s). The 32-bit data is driven to/read from the D[31:0] pins during the burst phases.

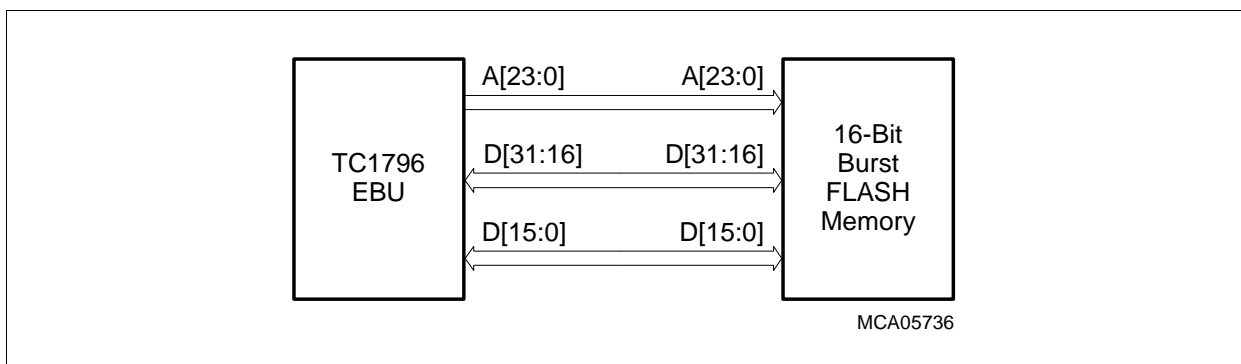


Figure 13-25 Connection of a 32-bit Burst Flash Memory

Figure 13-26 shows an example of two basic configurations for external Burst Flash memory connections.

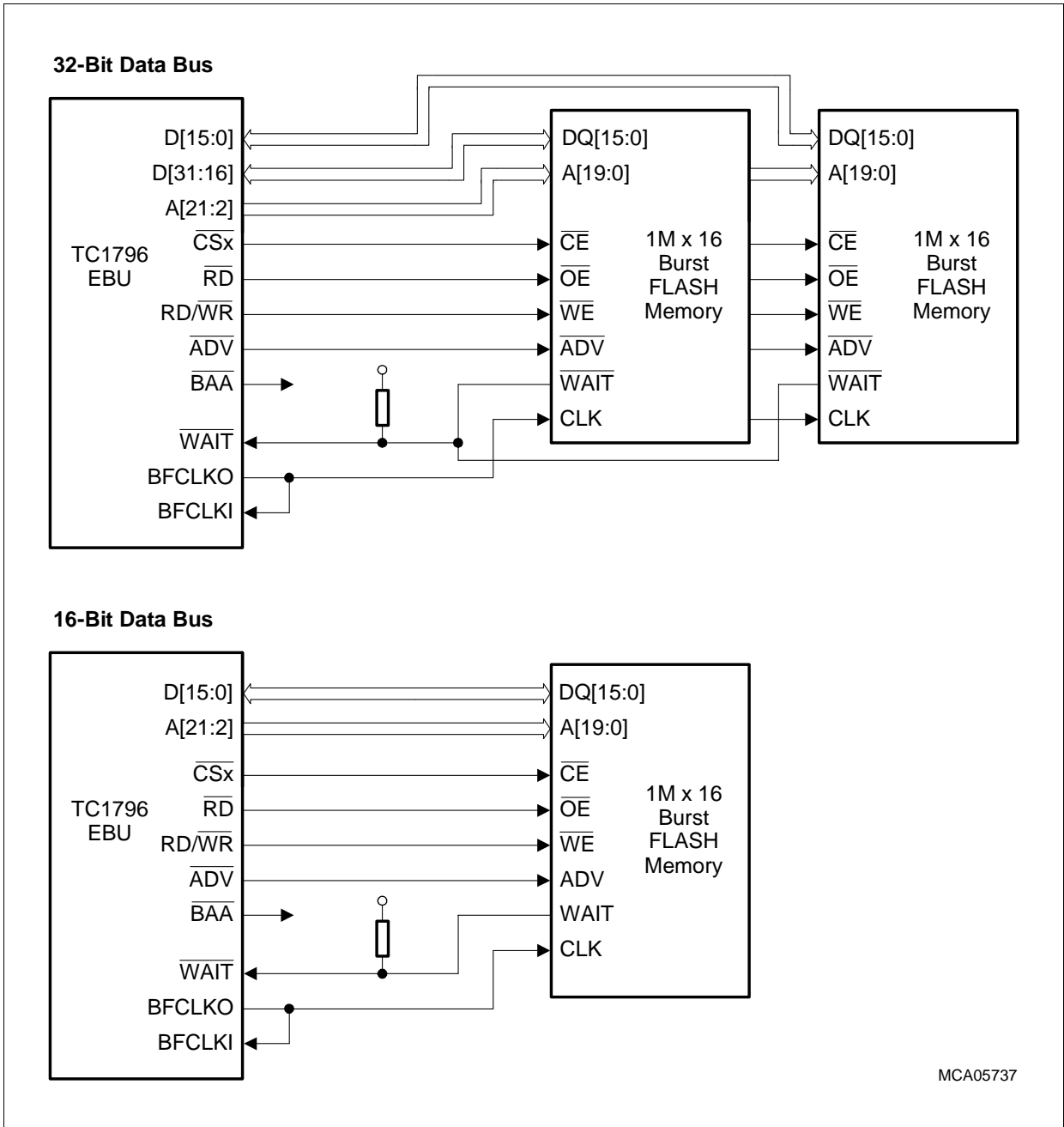


Figure 13-26 Example Configuration for Connection with Intel/AMD Flash Devices

After reset, the Burst Flash memory devices are normally configured as asynchronous devices. For operation in Burst Mode, the user's software must first perform the appropriate actions to initialize the Burst Flash memory devices for burst operation (using asynchronous access mode), and then reconfigure the EBU to use the devices in Burst Mode.

Note: The EBU will issue an LMB Error Acknowledge if an attempt is made to write to an address that is programmed as Burst Flash unless a lower priority write-enabled region exists at the same address.

13.9.3 Burst Mode Access Phases

Accesses to asynchronous devices are composed of a number of standard Access Phases (which are detailed in [Section 13.7](#)). The standard access phases for Burst Flash devices are:

- AP: Address Phase (compulsory - see [Page 13-41](#))
- CD: Command Delay Phase (optional - see [Page 13-43](#))
- CP: Command Phase (compulsory - see [Page 13-43](#))
- BP: Burst Phase (compulsory - see [Page 13-46](#))
- RP: Recovery Phase (optional - see [Page 13-47](#))

Note: During a burst access, the Burst Phase (BP) is repeated the required number of times to complete the burst length.

13.9.4 Programmable Parameters

[Table 13-19](#) lists the programmable parameters for Burst Flash accesses. These parameters only apply only when the AGEN bit field in the EBU_BUSCONx or EBU_EMUBC register for a particular memory region is set to 010_B or 101_B (Burst Flash access).

Table 13-19 Burst Flash Access Programmable Parameters

Register	Parameter (Bit/Bit field)	Function
EBU_BUSAPx EBU_EMUBAP	ADDRC	Number of cycles in address phase; can be multiplied by CMULT.
	CMDDELAY	Number of programmed command delay cycles; can be multiplied by CMULT.
	WAITRDC	Number of programmed wait states for read accesses; always multiplied by CMULT.
	BURSTC	Number of data cycles; can be multiplied by CMULT.
	RDRECOVC	Number of minimum recovery cycles after a read access; can be multiplied by CMULT.
	DTACS	Number of minimum recovery cycles when the next access points to a different memory region; always multiplied by CMULT.
EBU_BUSCONx EBU_EMUBC	WAIT	External Wait State control (OFF, asynchronous, synchronous)
	CMULT	Common (cycle) multiplier control ($\times 1$, $\times 4$, $\times 8$, $\times 16$, $\times 32$)
	MULTMAP	Multiplier map, each bit enables multiplier function of bits and bit fields ADDRRC, CMDDELAY, BURSTC (see next chapter), DATAC, RDRECOVC, and WRRECOVC.

LMB External Bus Unit

Table 13-19 Burst Flash Access Programmable Parameters (cont'd)

Register	Parameter (Bit/Bit field)	Function
EBU_BFCON	WAITFUNC0	Function of $\overline{\text{WAIT}}$ input during Type 0 reads: Wait for data or terminate burst
	WAITFUNC1	Function of $\overline{\text{WAIT}}$ input during Type 1 reads: Wait for data or terminate burst
	FBBMSEL0	Flash Burst Mode select for Type 0 device: Continuous or burst length defined in FETBLEN
	FBBMSEL1	Flash Burst Mode select for Type 1 device: Continuous or burst length defined in FETBLEN
	FETBLEN0	Number of burst data cycles for Type 0 devices: 1, 2, 4, or 8
	FETBLEN1	Number of burst data cycles for Type 1 devices: 1, 2, 4, or 8
	EXTCLOCK	BFCLKO frequency in relation to LMBCLK: Equal, $\div 2$, $\div 3$ or $\div 4$ of LMBCLK
	FDBKEN	Burst clock feedback control
	DBA0	Address Alignment/Wrapping control for Type 0 device
	DBA1	Address Alignment/Wrapping control for Type 1 device
	EBSE0	Early $\overline{\text{ADV}}/\overline{\text{BAA}}$ signal control for type 0 device
	EBSE1	Early $\overline{\text{ADV}}/\overline{\text{BAA}}$ signal control for type 1 device
	BFCMSEL	BFCLKO clock mode selection

13.9.5 Support for two Burst Flash Device Types

Support is provided for the use of two different Burst Flash configurations. This means, two sets of burst access parameters (e.g. for different Burst Flash devices) can be defined and stored in register EBU_BFCON, and depending on bit EBU_BUSCONx.AGEN, one of these two parameter sets is selected. With AGEN = 0, Burst Flash Type 0 access parameters are selected. With AGEN = 1, Burst Flash Type 1 access parameters are selected.

Bits DBA0, EBSE0, WAITFUNC0, FBBMSEL0, and bit field FETBLEN0 in the lower half of register EBU_BFCON determine the characteristics for a Burst Flash Type 0 access.

Bits DBA1, EBSE1, WAITFUNC1, FBBMSEL1, and bit field FETBLEN1 in the upper half of register EBU_BFCON determine the characteristics for a Burst Flash Type 1 access.

The control functions of bit field EXTLOCK (burst clock frequency), bit BFCMSEL (burst clock mode), bit FDBKEN (burst clock feedback control), and bit field DTALTNCY (latency cycle control) are valid for both Burst Flash access types, independently of bit AGEN.

13.9.6 BFCLKO Output

The EBU provides a clock signal BFCLKO (Burst Flash Clock Output) suitable for clocking Burst Flash devices during burst read accesses. The BFCLKO signal can be programmed to operate in one of two modes:

- Ungated Mode: EBU_BFCON.BFCSEL = 0
- Gated Mode: EBU_BFCON.BFCSEL = 1

In Ungated Mode, the BFCLKO clock signal is generated all the time, whether or not burst cycles occur.

In Gated Mode, the BFCLKO clock signal is only active and generated during Burst Flash access cycles. This is the default selection after reset. BFCLKO is at low level(0) in inactive state. BFCLKO is also at low level during asynchronous read/writes of/to Burst Flash devices. Operation in this mode provides power savings and ensures correct asynchronous accesses of Burst Flash device(s).

Bit field EBU_BFCON.EXTLOCK determines the frequency relation between LMBCLK clock and of the BFCLKO output for both Burst Flash access types. [Figure 13-27](#) shows all possible BFCLKO configurations that can be selected.

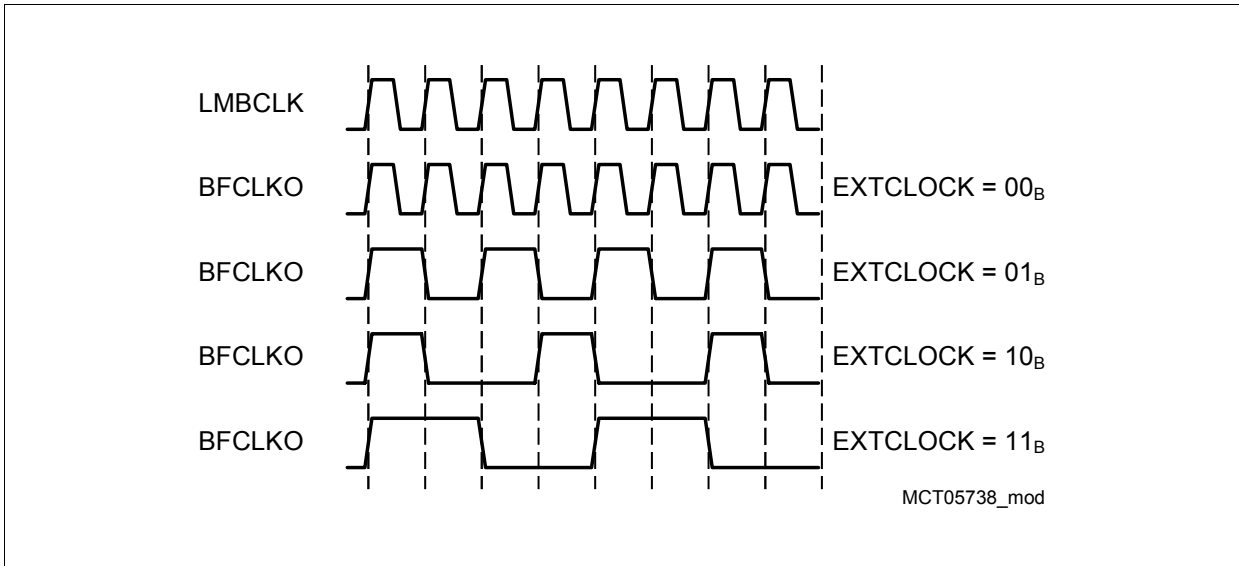


Figure 13-27 Possible BFCLKO Configurations

13.9.7 BFCLKI Input and Burst Flash Clock Feedback

The EBU Burst Mode control logic can be configured to use a clock feedback signal at BFCLKI to maximize the operating frequency for a given Flash device. Clock feedback functionality is enabled by setting `EBU_BFCON.FDBKEN = 1`.

When enabled, the first sampling stage for the read data is clocked by BFCLKI instead of by BFCLKO. The data is then re-synchronized to BFCLKO internally before being passed to the normal logic. When the clock feedback mode is used, BFCLKI can therefore be skewed by almost an entire BFCLKO cycle relative to the internal LMBCLK clock without losing data integrity.

A side effect of using clock feedback mode is an increase in data latency of 2 cycles of BFCLKO. The value for this increased latency in LMBCLK clock cycles must be programmed into bit field `EBU_BFCON.DTALTNCY` to ensure that data is sampled on the correct clock edge.

13.9.8 Burst Length Control

The maximum number of Burst Phases that are generated in a single access is controlled by two parameters of register EBU_BFCON:

- Flash burst length mode select bit FBBMSEL_n ($n = 0, 1$)
- Fetch burst length bit field FETBLEN_n ($n = 0, 1$)

Index n refers to type 0 or type 1 Burst Flash device parameters.

With FBBMSEL_n = 0, the EBU generates Burst Flash access cycles with an unlimited number of Burst Phases. This means, Burst Phases are generated in a Burst Flash access cycle as long as data with consecutive addresses can be requested by the EBU. Note that the length in number of LMBCLK cycles of one Burst Phase is defined by parameter BURSTC.

With FBBMSEL_n = 1, the number of Burst Phases within one Burst Flash access cycle is defined by bit field FETBLEN_n. In this mode, one, two, four, or eight Burst Phases can be selected for one Burst Flash access cycle. Bit field FETBLEN_n determines the maximum number of Burst Phases. If more data is requested than can be delivered by one Burst Flash access cycle (as defined by FETBLEN_n), the EBU will automatically generate the appropriate number of continuous Burst Flash access cycles to supply the required amount of data.

13.9.9 Control of \overline{ADV} and \overline{BAA} Delays During Burst Flash Access

The \overline{ADV} and \overline{BAA} signals are delayed by half an LMBCLK clock cycle with respect to the other control signals of the EBU. This delay can be removed via bit BFCON.EBSEN ($n = 0, 1$). The default setting after reset has the delay enabled (EBSEN = 0). Index n refers to type 0 or type 1 Burst Flash device parameters.

Note: If EBSEN = 0, it must be regarded that the $\overline{ADV}/\overline{BAA}$ active/inactive time in respect to BFCLKO can be negative. More details are defined in the AC timings of the TC1796 Data Sheet.

13.9.10 Burst Flash Access Cycle

Figure 13-28 shows an example of a burst read access (burst length of four) with a Burst Flash device.

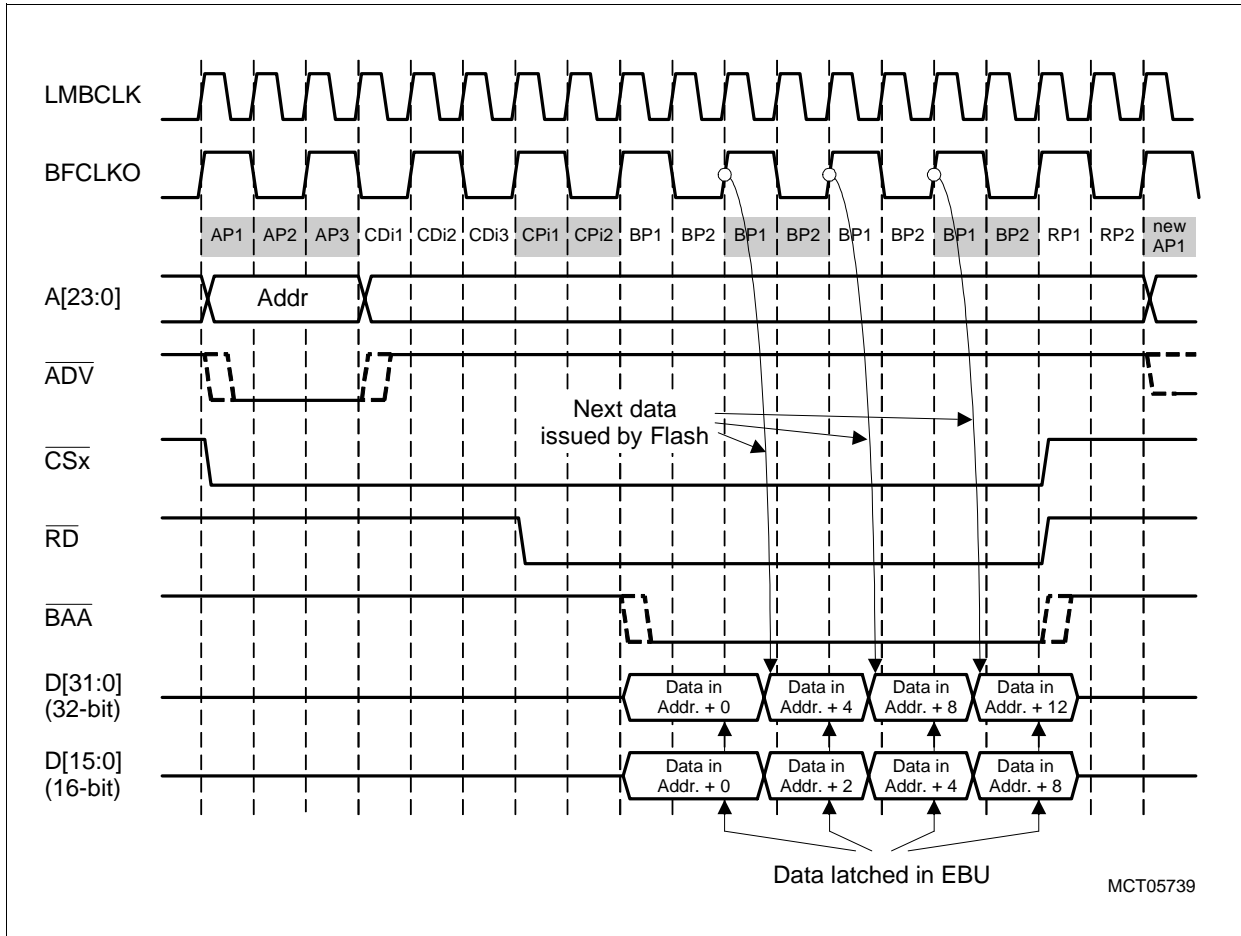


Figure 13-28 Burst Flash Read Operation Example (Burst Length of 4)

The Burst Flash access cycle shown in Figure 13-28 is based on the following settings:

- BFCLKO frequency = 1/2 of LMBCLK frequency (EXTCLOCK = 01_B)
- Address Phase length = 3 LMBCLK cycles (ADDRC = 01_B)
- Command Delay Phase length = 3 LMBCLK cycles (CMDDELAY = 011_B)
- Command Phase length = 2 LMBCLK cycles (WAITRDC = 010_B)
- Burst Phase length = 2 LMBCLK cycles (BURSTC = 010_B)
- Recovery Phase length = 2 LMBCLK cycles (RDRECOVC = 010_B)
- Burst Length = 4 (FETBLENN_n = 010_B)
- Bus Width = 16-bit (PORTW = 01_B) or 32-bit (PORTW = 10_B)

Note that the cycle multiplier control capability is disabled in the example (MULTMAP = 00_H).

Figure 13-29 shows a burst read access (burst length of four) with a BFCLKO frequency of 1/3 of LMBCLK frequency (EXTCLOCK = 01_B).

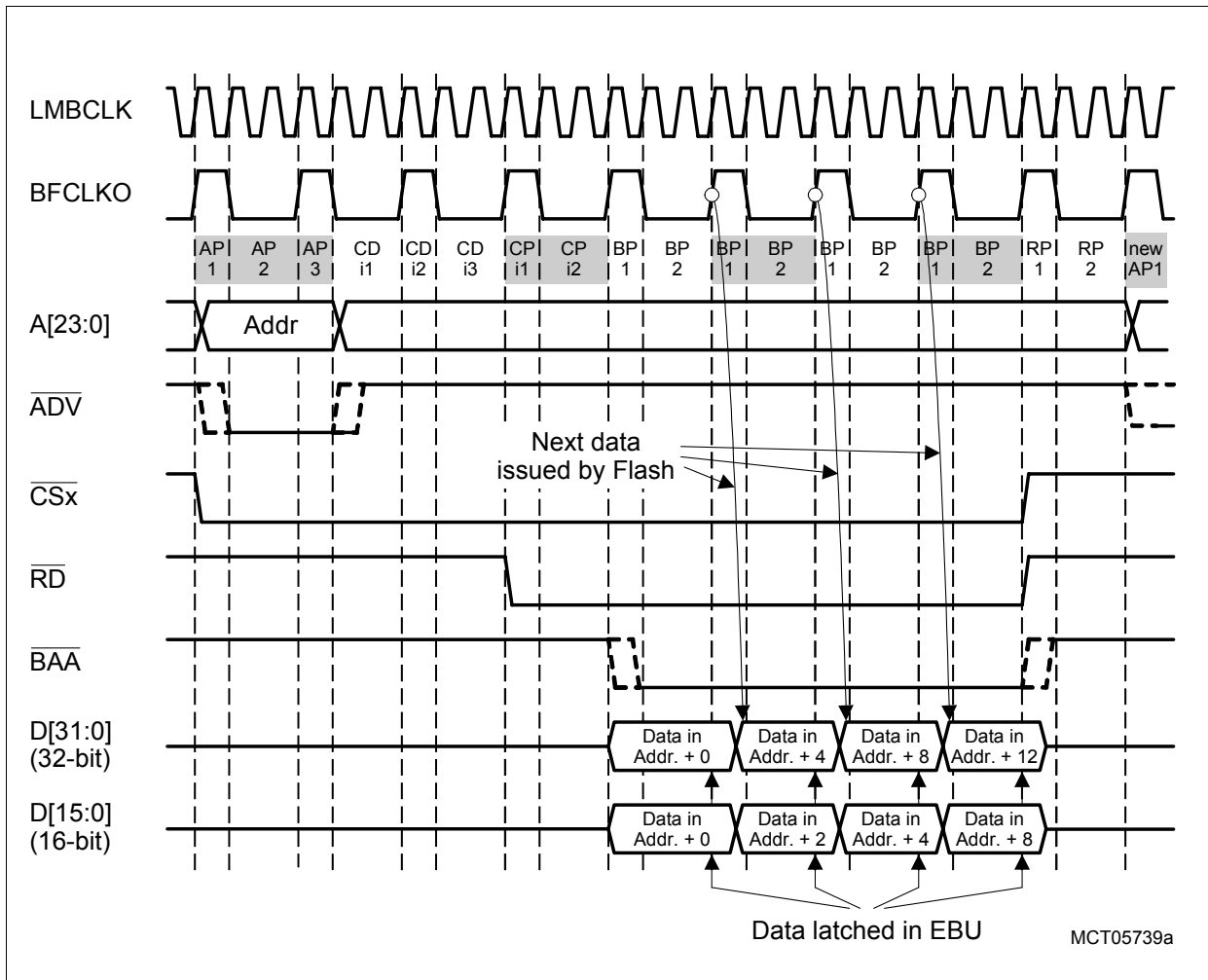


Figure 13-29 Burst Flash Read Operation Example (Burst Length of 4)

The lengths of the standard accesses phases during Burst Flash accesses are programmed as multiples of LMBCLK periods independent of the BFCLKO frequency. It is the users responsibility to ensure (by programming the access phases accordingly) that the sampling of data by the EBU guarantees valid sampling of the data from the Burst Flash device.

13.9.11 External Cycle Control via the WAIT Input

The EBU controls the Burst Flash device via the $\overline{\text{WAIT}}$ input. This allows the EBU to support operation of Burst Flash while crossing Burst Flash page boundaries. During a Burst Flash access, the $\overline{\text{WAIT}}$ input operates in one of the five modes shown in Table 13-20.

Table 13-20 WAIT Mode Selection

EBU_BUSCON. WAIT	EBU_BFCON. WAITFUNC	Selected WAIT Mode
00 _B	–	$\overline{\text{WAIT}}$ function disabled
01 _B	0	Asynchronous Wait for Page Load
	1	Asynchronous Terminate and Start New Burst
10 _B	0	Synchronous Wait for Page Load
	1	Synchronous Terminate and Start New Burst
11 _B	0	Synchronous Data Handshake
	1	$\overline{\text{WAIT}}$ function disabled

13.9.11.1 Wait for Page Load Mode

Note: The description in this section assumes a low active $\overline{\text{WAIT}}$ input ($\text{WAITINV} = 0$).

In Wait for Page Load Mode, the $\overline{\text{WAIT}}$ input being asserted low during a burst access causes the EBU to complete the current Burst Phase and then to perform an additional Burst Phase (if the previous Burst Phase was not the last of the current burst access cycle). Following this, the next Burst Phase (again if this Burst Phase required) will not be started until the $\overline{\text{WAIT}}$ input is de-asserted. Additionally, the start of the next Burst Phase is also synchronized to the next rising edge of the BFCLKO signal to ensure that the EBU continues to sample data from the device at the correct time with respect to BFCLKO. This mode allows, for example, an Intel Burst Flash device to temporarily suspend the burst sequence in order to perform a page load when a page boundary is crossed during the access.

The Wait for Page Load Mode supports the use of Intel Burst Flash devices (and compatibles) configured for Early Wait Generation Mode ($\text{EBU_BUSCON.WAIT} = 01_{\text{B}}$) and standard wait generation ($\text{EBU_BUSCON.WAIT} = 10_{\text{B}}$).

In operation, the Burst Flash controller loads a counter with the required number of samples at the start of each burst. At the end of each Burst Phase, the Burst Flash controller samples the $\overline{\text{WAIT}}$ input and the data bus. If $\overline{\text{WAIT}}$ is inactive, the sample is valid, the sample counter is decremented, and the sampled data is passed to the data path of the EBU. This synchronous sampling means that the validity of the sample can not be determined until the clock cycle after the end of the burst phase. The Burst Flash controller in the EBU will therefore overrun and generate extra burst phases until the sample counter is decremented to zero. Extra data samples returned after the sample counter is zero will be discarded.

This mode of operation is compatible with the use of clock feedback as, with feedback enabled, $\overline{\text{WAIT}}$ is fed through the same re-synchronization signals as the data bus. The

only effect on operation is that the number of overrun cycles will increase as the decrementing of the sample counter will be lagged by the re-synchronization stages.

13.9.11.2 Terminate and Start New Burst Mode

In Terminate and Start New Burst Mode, the $\overline{\text{WAIT}}$ input being asserted low during a burst access causes the EBU to complete the current Burst Phase and then to terminate the current access. If additional data is required, the EBU will start a new access cycle (i.e. Address Phase, Command Delay Phase, Command Phase, Burst Phase etc.). Furthermore, the EBU calculates the address to be issued during the new Address Phase to ensure correct sequential reading of the Burst Flash device(s). This mode allows, for example, an AMD Burst Flash device to cause the EBU to terminate the current access when a page boundary is crossed, and to issue a complete new burst read access. This allows the Burst Flash device to perform a page load at the start of the new access.

All current known devices require EBU_BUSCON.WAIT to be set to 10_B (standard wait). There are no devices currently known requiring BUSCON.WAIT to be set to 01_B (early wait).

Note: The Terminate and Start New Burst Mode cannot be used with clock feedback.

13.9.12 Termination of a Burst Mode Read Access

A burst read operation is terminated by de-asserting the \overline{CSx} signal, followed by the appropriate length Recovery Phase. **Figure 13-30** shows termination of a burst access following the read of two locations (i.e. two Burst Phases) from the Burst Flash device(s).

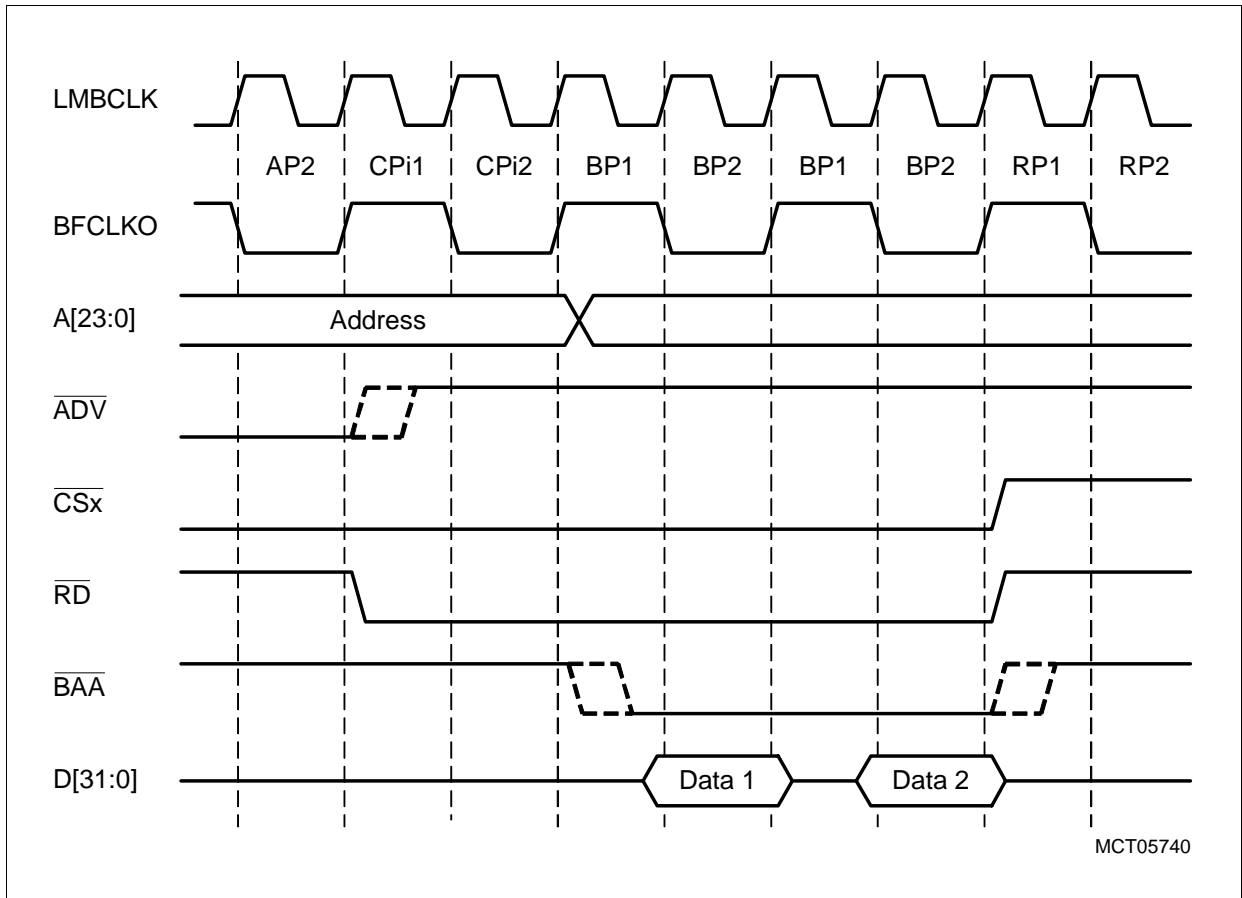


Figure 13-30 Terminating a Burst by De-asserting CS

13.10 EBU Registers

This section describes the registers and programmable parameters of the EBU. **Figure 13-31** shows all registers associated with the EBU. All these registers can be read in User Mode, but can only be written in Supervisor Mode.

EBU Registers Overview

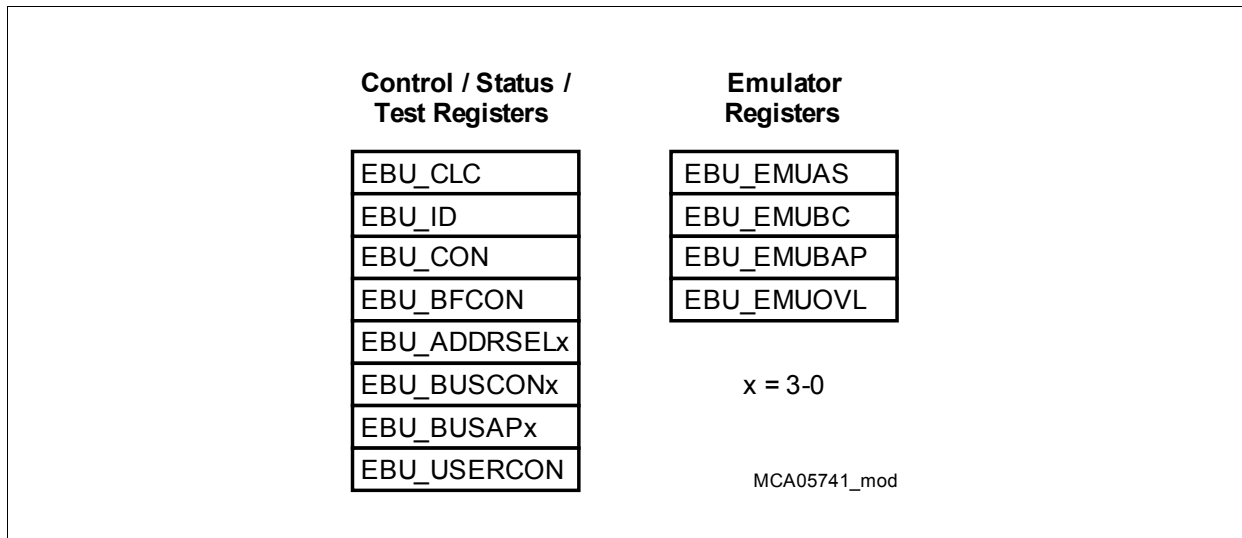


Figure 13-31 EBU Registers

The complete and detailed address map of the PMU module with its registers is shown in **Table 18-34** on **Page 18-112** of this TC1796 System Units (Vol. 1 of 2) User’s Manual.

Table 13-21 Registers Address Space -Flash Registers

Module	Base Address	End Address	Note
EBU	F800 0000 _H	F800 03FF _H	—

Table 13-22 Registers Overview PMU Overlay Control Registers

Register Short Name	Register Long Name	Offset Address	Description see
EBU_CLC	EBU Clock Control Register	000 _H	Page 13-82
EBU_ID	EBU Module Identification Register	008 _H	Page 13-81
EBU_CON	EBU Configuration Register	010 _H	Page 13-83
EBU_BFCON	EBU Burst Flash Control Register	020 _H	Page 13-86

Table 13-22 Registers Overview PMU Overlay Control Registers

Register Short Name	Register Long Name	Offset Address	Description see
EBU_ADDRSEL0	EBU Address Select Register 0	080 _H	Page 13-90
EBU_ADDRSEL1	EBU Address Select Register 1	088 _H	
EBU_ADDRSEL2	EBU Address Select Register 2	090 _H	
EBU_ADDRSEL3	EBU Address Select Register 3	098 _H	
EBU_BUSCON0	EBU Bus Configuration Register 0	0C0 _H	Page 13-92
EBU_BUSCON1	EBU Bus Configuration Register 1	0C8 _H	
EBU_BUSCON2	EBU Bus Configuration Register 2	0D0 _H	
EBU_BUSCON3	EBU Bus Configuration Register 3	0D8 _H	
EBU_BUSAP0	EBU Bus Access Parameter Register 0	100 _H	Page 13-97
EBU_BUSAP1	EBU Bus Access Parameter Register 1	108 _H	
EBU_BUSAP2	EBU Bus Access Parameter Register 2	110 _H	
EBU_BUSAP3	EBU Bus Access Parameter Register 3	118 _H	
EBU_EMUAS	EBU Emulator Address Select Register	160 _H	Page 13-101
EBU_EMUBC	EBU Emulator Bus Configuration Register	168 _H	Page 13-102
EBU_EMUBAP	EBU Emulator Bus Access Parameter Register	170 _H	Page 13-106
EBU_EMUOVL	EBU Emulator Overlay Register	178 _H	Page 13-110
EBU_USERCON	EBU Test/Control Configuration Register	190 _H	Page 13-111

Note: The EBU registers are accessible only through word accesses. Half-word and byte accesses on EBU registers will generate a bus error.

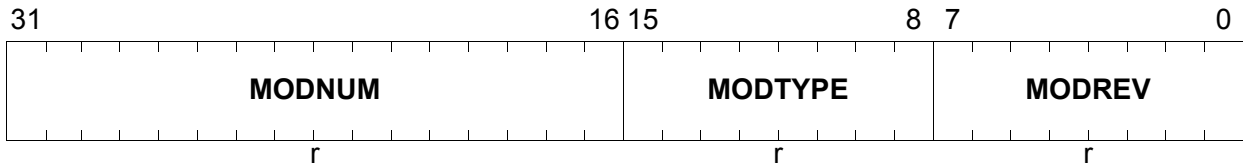
13.10.1 Identification Register, ID

The Module Identification Register ID contains read-only information about the EBU module version.

EBU_ID

EBU Module Identification Register (008_H)

Reset Value: 0014 C0XX_H



Field	Bits	Type	Description
MODREV	[7:0]	r	Module Revision Number MODREV defines the module revision number. The value of a module revision starts with 01 _H (first revision).
MODTYPE	[15:8]	r	Module Type This bit field defines the module as a 32-bit module: C0 _H
MODNUM	[31:16]	r	Module Number Value This bit field defines the module identification number for the EBU: 0014 _H

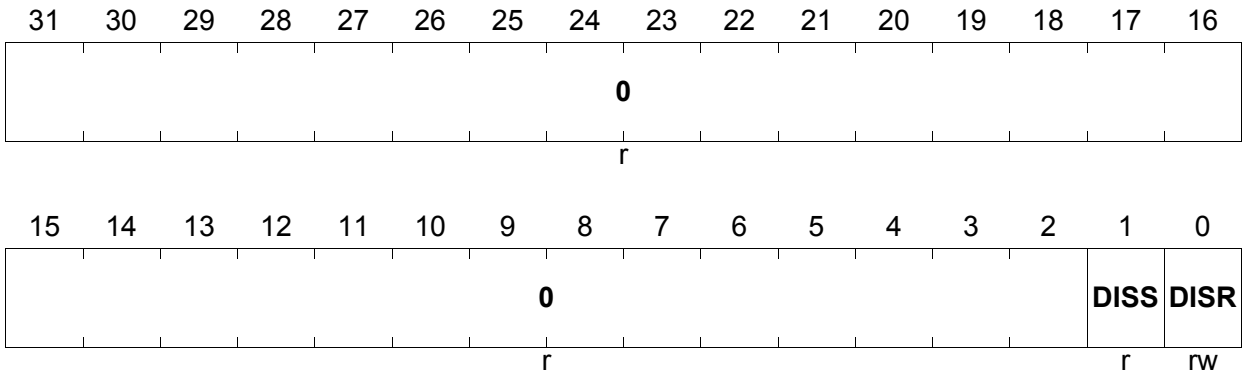
13.10.2 Clock Control Register, CLC

EBU_CLC

EBU Clock Control Register

(000_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
DISR	0	rw	EBU Disable Request Bit This bit is used for enable/disable control of the EBU. 0 _B EBU disable is not requested 1 _B EBU disable is requested
DISS	1	r	EBU Disable Status Bit 0 _B EBU is enabled (default after reset) 1 _B EBU is disabled In the TC1796, DISS is always read as 0, meaning that the EBU is always enabled.
0	[31:2]	r	Reserved Read as 0; should be written with 0.

Note: While the DISR bit is implemented in the EBU, a request to disable the module will be ignored. This register is Endinit-protected after initialization. Writing to this register in this state will cause the EBU to generate an LMB Error.

13.10.3 Configuration Register, CON

EBU_CON

EBU Configuration Register

(010_H)

Reset Value (Internal boot): 0000 0028_H

Reset Value (External boot): 0801 0068_H

0801 008A_H

Reset Value (Emulation mode): 1001 0068_H

1001 00A8_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
0		BF SSS	EMU FAM	CS0 FAM	0						GLOBALCS					
r		rw		rw		rw						rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
TIMEOUTC							ARB MODE		ARB SYN C	EXT LOC K	1	0	0			
rw							rw		rw	rw	rw	rw	rw	r		

Field	Bits	Type	Description
EXTLOCK	4	rw	External Bus Lock Control 0 _B External bus is not locked after the EBU gains ownership 1 _B External bus is locked after the EBU gains ownership
ARBSYNC	5	rw	Arbitration Signal Synchronization Control 0 _B Arbitration inputs are synchronous 1 _B Arbitration inputs are asynchronous
ARBMODE	[7:6]	rw	Arbitration Mode Selection 00 _B 00No Bus arbitration mode selected 01 _B 01Arbiter Mode arbitration mode selected 10 _B 10Participant arbitration mode selected 11 _B 11Sole Master arbitration mode selected

LMB External Bus Unit

Field	Bits	Type	Description
TIMEOUTC	[15:8]	rw	<p>Bus Time-out Control</p> <p>This bit field determines the number of inactive cycles leading to a bus time-out after the EBU gains ownership.</p> <p>00_H Time-out is disabled. 01_H Time-out is generated after 1 × 8 clock cycles. ..._H ... FF_H Time-out is generated after 255 × 8 clock cycles.</p>
GLOBALCS	[19:16]	rw	<p>Global Chip Select Signal</p> <p>The bits of this bit field are used to enable the <u>EBU</u> chip select lines for the global chip select <u>CSGLB</u> generation. With x = 0-3, the GLOBALCS bits are defined as follows (see also Page 13-29):</p> <p>0_B <u>CSGLB</u> is not activated by an active CS_x. 1_B <u>CSGLB</u> is activated when CS_x becomes active.</p>
CS0FAM	27	rw	<p>CS0 Fills Address Map</p> <p>0_B Normal region/chip select logic is in operation. 1_B Normal region/chip select logic is disabled. The Chip Select 0 Address Override Mode is selected. All external bus accesses are directed to Region 0 (CS0).</p> <p>This bit is set following reset in external boot mode (see Page 13-21) which ensures that the CPU to which the EBU is connected can boot from external memory, regardless of the CPU specific boot address.</p>
EMUFAM	28	rw	<p>CSEMU Fills Address Map</p> <p>0_B Normal region/chip select logic is in operation. 1_B Normal region/chip select logic is disabled. The Emulator Chip Select Address Override Mode is selected. All external bus accesses are directed to the Emulator Region (CSEMU).</p> <p>This bit has no effect when bit CS0FAM is 1. It is set following reset in emulation mode (see Page 13-20), which ensures that the CPU to which the EBU is connected can boot from emulation memory regardless of the CPU-specific boot address.</p>

LMB External Bus Unit

Field	Bits	Type	Description
BFSSS	29	rw	Burst Flash Single Stage Synchronization This bit reduces the number of synchronization stages used in the pad logic of EBU pads. 0 _B Two stages of synchronization used. 1 _B Single stage of synchronization used.
1	3	rw	Reserved Read as 1 after reset; must be written with 0.
0	[2:1], [26:20]	rw	Reserved Read as 0; must be written with 0.
0	0, [31:30]	r	Reserved Read as 0; should be written with 0.

13.10.4 Burst Flash Control Register, BFCON

EBU_BFCON

EBU Burst Flash Control Register (020_H)

Reset Value: 0010 01D0_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				DBA 1	EBS E1	0			WAI TFU NC1	FBB MSE L1	FETBLEN1				
r				rw	rw	r			rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTALTNCY				FDB KEN	DBA 0	EBS E0	BFC MSE L	EXT CLOCK		WAI TFU NC0	FBB MSE L0	FETBLEN0			
rw				rw	rw	rw	rw	rw		rw	rw	rw			

Field	Bits	Type	Description
FETBLEN0	[3:0]	rw	Fetch Burst Length for Burst Flash Type 0 This bit field determines the maximum number of burst data cycles which are executed by the EBU during a burst access to Burst Flash Type 0. 000 _B 1 data access (default after reset). 001 _B 2 data accesses. 010 _B 4 data accesses. 011 _B 8 data accesses. 1XX _B Reserved.
FBBMSEL0	4	rw	Flash Burst Buffer Mode Select for Burst Flash Type 0 0 _B Continuous mode selected 1 _B Flash burst buffer length defined by value in FETBLEN0 (default after reset)
WAITFUNC0	5	rw	Function of WAIT input for Burst Flash Type 0 0 _B The $\overline{\text{WAIT}}$ input operates as a wait data bus function on bursts (default after reset). 1 _B The $\overline{\text{WAIT}}$ input operates as a terminate burst function.

LMB External Bus Unit

Field	Bits	Type	Description
EXTCLOCK	[7:6]	rw	<p>Frequency of External Clock at Pin BFCLKO This bit field sets the frequency of pin BFCLKO for accesses to both types of Burst Flash devices.</p> <p>00_B Equal to LMBCLK frequency. In the TC1796, this selection is not allowed for LMBCLK (f_{CPU}) frequencies above 75 MHz.</p> <p>01_B 1/2 of LMBCLK frequency.</p> <p>10_B 1/3 of LMBCLK frequency.</p> <p>11_B 1/4 of LMBCLK frequency (default after reset).</p>
BFCMSEL	8	rw	<p>Burst Flash Clock Mode Select</p> <p>0_B Clock signal is continuously available at pin BFCLKO.</p> <p>1_B Clock signal is only available at pin BFCLKO during a Burst Flash access (default after reset). BFCLKO is at low level in inactive state.</p>
EBSE0	9	rw	<p>Early Burst Signal Enable for Burst Flash Type 0</p> <p>0_B Outputs ADV and BAA are delayed by 1/2 LMBCLK clock period.</p> <p>1_B Output lines ADV and BAA are not delayed. (see Page 13-73)</p>
DBA0	10	rw	<p>Disable Burst Address Wrapping</p> <p>0_B The EBU automatically re-aligns any non-aligned burst access to a Type 0 device so that data can be fetched from the device in a single burst transaction.</p> <p>1_B The EBU always starts any burst access to a Type 0 device at the address specified by the LMB request. Any required address wrapping must be automatically provided by the Burst Flash device.</p> <p><i>Note: Care must be taken with the use of this feature. The address at which wrapping should take place varies with the LMB burst access size, while Burst Flash devices wrap at fixed address boundaries. Therefore, it is not possible to guarantee correct behavior for all LMB accesses when this bit is 1.</i></p>

LMB External Bus Unit

Field	Bits	Type	Description
FDBKEN	11	rw	<p>Burst Flash Clock Feedback Enable</p> <p>0_B BFCLK feedback not used. In this case, bit field DTALTNCY must be set to 0000_B.</p> <p>1_B Incoming data and control signals (from the Burst Flash device) are re-synchronized to the BFCLKI input.</p>
DTALTNCY	[15:12]	rw	<p>Latency Cycle Control</p> <p>This bit field determines the number of additional LMBCLK cycles of latency to be used when sampling inputs from the Burst Flash devices.</p>
FETBLEN1	[19:16]	rw	<p>Fetch Burst Length for Burst Flash Type 1</p> <p>This bit field determines the maximum number of burst data cycles which are executed by the EBU during a Burst Flash Type 1 burst access.</p> <p>000_B 1 data access (default after reset). 001_B 2 data accesses. 010_B 4 data accesses. 011_B 8 data accesses. 1XX_B Reserved.</p>
FBBMSEL1	20	rw	<p>Flash Burst Buffer Mode Select for Burst Flash Type 1</p> <p>0_B Continuous mode. 1_B Flash burst buffer length is defined by the value of FETBLEN1 (default after reset).</p>
WAITFUNC1	21	rw	<p>Function of $\overline{\text{WAIT}}$ input for Burst Flash Type 1</p> <p>0_B The $\overline{\text{WAIT}}$ input operates as a wait data bus function on bursts (default after reset). 1_B The $\overline{\text{WAIT}}$ input operates as a terminate burst function.</p>
EBSE1	25	rw	<p>Early Burst Signal Enable for Burst Flash Type 1</p> <p>0_B Outputs $\overline{\text{ADV}}$ and $\overline{\text{BAA}}$ are delayed by 1/2 LMBCLK period (default after reset). 1_B Outputs $\overline{\text{ADV}}$ and $\overline{\text{BAA}}$ are not delayed. (see Page 13-73)</p>

LMB External Bus Unit

Field	Bits	Type	Description
DBA1	26	rw	<p>Disable Burst Address Wrapping</p> <p>0_B The EBU automatically re-aligns any non-aligned burst access to a Type 1 device so that data can be fetched from the device in a single burst transaction.</p> <p>1_B The EBU always starts any burst access to a Type 1 device at the address specified by the LMB request. Any required address wrapping must be automatically provided by the Burst Flash device.</p> <p><i>Note: Care must be taken with the use of this feature. The address at which wrapping should take place varies with the LMB burst access size, while Burst Flash devices wrap at fixed address boundaries. Therefore, it is not possible to guarantee correct behavior for all LMB accesses when this bit is 1.</i></p>
0	[31:27], [24:22]	0	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

13.10.5 Address Select Register, ADDRSELx

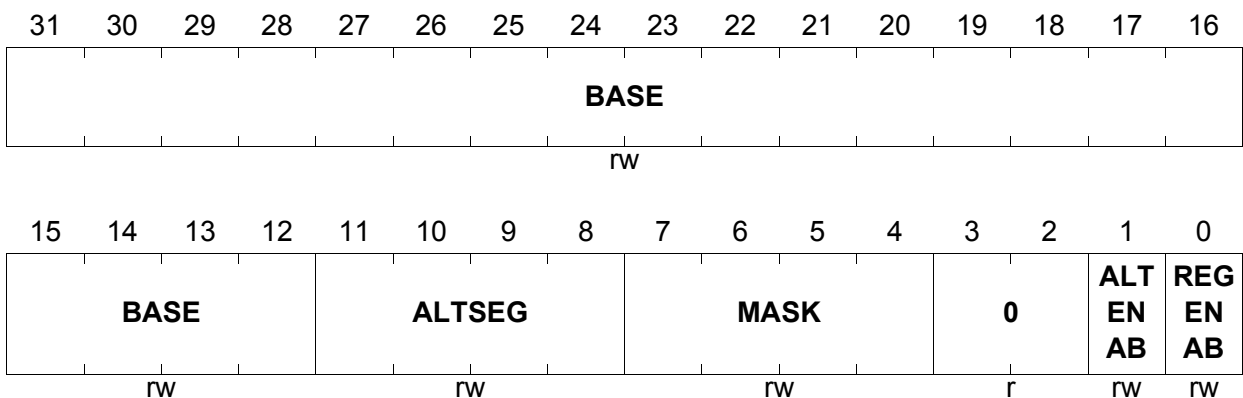
EBU_ADDRSELx (x = 0-3)

EBU Address Select Register x (080_H+x*8_H)

ADDRSEL[3:1] - Reset Value: 0000 0000_H

ADDRSEL0 - Reset Value (internal boot): 0000 0000_H

ADDRSEL0 - Reset Value (external boot): A000 0001_H



Field	Bits	Type	Description
REGENAB	0	rw	Memory Region Enable 0 _B Memory region is disabled (default after reset except for ADDRSEL0 - see below). 1 _B Memory region is enabled. When the EBU is in external boot mode, REGENAB in register ADDRSEL0 is 1 after reset.
ALTENAB	1	rw	Alternate Segment Comparison Enable 0 _B ALTSEG is never compared to LMB address (default after reset). 1 _B ALTSEG is always compared to LMB address.
MASK	[7:4]	rw	Memory Region Address Mask Specifies the number of right-most bits in the base address starting at bit 26, which should be included in the address comparison. Bits [31:27] will always be part of the comparison.
ALTSEG	[11:8]	rw	Memory Region Alternate Segment Alternate segment to be compared to PLMB address bit [31:28].

LMB External Bus Unit

Field	Bits	Type	Description
BASE	[31:12]	rw	Memory Region Base Address Base address to be compared to PLMB address in conjunction with the mask control.
0	[3:2]	r	Reserved Read as 0; should be written with 0.

*Note: The actual reset value of the ADDRSEL0 register is not relevant during external boot modes, as the EBU_CON.CS0FAM bit is set to ensure that region 0 (i.e. $\overline{CS0}$) is activated for **all** external bus accesses. This allows external boot to be performed from region 0 regardless of the actual boot address of the CPU to which EBU is connected.*

13.10.6 Bus Configuration Register, BUSCONx

EBU_BUSCONx (x = 0-3)

EBU Bus Configuration Register (0C0_H+x*8_H)

Reset Value (internal boot): 8092 8000_H

Reset Value (external boot): 8092 807F_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRI TE	AGEN		0	WAIT		PORTW		BCGEN		WAI TINV	PRE FET CH	DLO AD	ENDI AN		
rw	rw		rw	rw		rw		rw		rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMULT		0	CTYPE		AALI GN	WEA KPR EF	0	MULTMAP							
rw		r	rw		rw	rw	r	rw							

Field	Bits	Type	Description
MULTMAP	[6:0]	rw	<p>Multiplier Map</p> <p>This bit field is a mask value that determines whether or not a specific bus access parameter as defined in register BUSAPx is multiplied with the value of bit field CMULT.</p> <p>XXXXXX0_BADDRC is not multiplied. XXXXXX1_BADDRC is multiplied by CMULT. XXXXX0X_BReserved, don't care. XXXXX1X_BReserved, don't care. XXXX0XX_BCMDDDELAY is not multiplied. XXXX1XX_BCMDDDELAY is multiplied by CMULT. XXX0XXX_BBURSTC is not multiplied. XXX1XXX_BBURSTC is not multiplied. XX0XXXX_BDATAAC is not multiplied. XX1XXXX_BDATAAC is not multiplied. X0XXXXX_BRDRECOVC is not multiplied. X1XXXXX_BRDRECOVC is multiplied by CMULT. 0XXXXXX_BWRRECOVC is not multiplied. 1XXXXXX_BWRRECOVC is multiplied by CMULT.</p> <p>Note that the bus access parameters WAITRDC, WAITWRC, DTARDWR, and DTACS in register BUSAPx are not programmable and always use the CMULT multiplier.</p>
WEAKPREF	8	rw	<p>Weak Prefetch</p> <p>0_B Code prefetch cannot be aborted by an interrupting data access. 1_B Code prefetch can be aborted by an interrupting data access.</p>
AALIGN	9	rw	<p>Address Alignment</p> <p>0_B The EBU always issues a byte address when performing an external bus access via this chip select. 1_B The EBU aligns the address according to the setting of the PORTW bit field. (see Section 13.5.7).</p>
CTYPE	[11:10]	rw	<p>Cycle Type</p> <p>This bit field is 00_B after reset. In the TC1796, CTYPE must always be written with 00_B.</p>

LMB External Bus Unit

Field	Bits	Type	Description
CMULT	[15:13]	rw	<p>Cycle Multiplier Control This bit field specifies a multiplier for the cycles specified via MULTMAP. WAITRDC, WAITWRC, DTARDWR and DTACS always use this multiplier.</p> <p>000_B Multiplier is 1. 001_B Multiplier is 4. 010_B Multiplier is 8. 011_B Multiplier is 16. 100_B Multiplier is 32 (default after reset). 101_B Reserved. 11X_B Reserved.</p>
ENDIAN	16	rw	<p>Endian Mode</p> <p>0_B Little-endian access mode selected (default after reset). 1_B Big-endian access mode selected.</p>
DLOAD	17	rw	<p>Enforce Data Upload from External Bus</p> <p>0_B Data access is fed from data write buffer if it is available (default after reset). 1_B Data access is always fed from the external bus access.</p>
PREFETCH	18	rw	<p>Prefetch Mechanism for Each Code Access</p> <p>0_B Code access never uses prefetch buffer mechanism (default after reset). 1_B Code access always uses prefetch buffer mechanism.</p>
WAITINV	19	rw	<p>Reversed Polarity at <u>WAIT</u></p> <p>0_B Input signal at the <u>WAIT</u> pin is active low (default after reset). 1_B Input signal at the <u>WAIT</u> pin is active high (reversed polarity).</p> <p>This bit has no effect when using Burst Flash Data Handshake Mode.</p>

LMB External Bus Unit

Field	Bits	Type	Description
BCGEN	[21:20]	rw	<p>Byte Control Signal Control This bit field selects the timing mode of the byte control signals.</p> <p>00_B Byte control signals follow chip select timing. 01_B <u>Byte</u> control <u>signals</u> follow control signal timing (RD, RD/W\overline{R}) (default after reset). 10_B Byte control <u>signals</u> follow write enable signal timing (RD/W\overline{R} only). 11_B Reserved.</p>
PORTW	[23:22]	rw	<p>External Device Data Width Control This bit field selects the data bus width for bus region x.</p> <p>00_B Reserved 01_B 16-bit 10_B 32-bit (default after reset) 11_B Reserved</p>
WAIT	[25:24]	rw	<p>External Wait State Control This bit determines the function of the \overline{WAIT} input. This is specific to the device type (i.e. the AGEN field). For Asynchronous devices (AGEN = 000_B): 00_B OFF (default after reset). 01_B Asynchronous input at \overline{WAIT}. 10_B Synchronous input at \overline{WAIT}. 11_B Nand Flash mode. For Burst Flash devices (AGEN = 010_B or 101_B): 00_B OFF (default after reset). 01_B Early synchronous input at \overline{WAIT}. 10_B Synchronous input at \overline{WAIT}. 11_B Reserved.</p>
0	[27:26]	rw	<p>Reserved Read as 0 after reset; <u>must</u> be always written with 0.</p>
AGEN	[30:28]	rw	<p>Address Generation Control/Selection This bit field selects the access type for bus region x.</p> <p>000_B Demultiplexed (asynch.) access selected (default after reset). 010_B Burst Flash Type 0 (sync.) access selected. 101_B Burst Flash Type 1(sync.) access selected. Other values of AGEN are reserved and must not be used.</p>

LMB External Bus Unit

Field	Bits	Type	Description
WRITE	31	rw	Memory Region Write Protection 0 _B Writes to the memory region are enabled. 1 _B Writes to the memory region are disabled (default after reset).
0	12, 7	r	Reserved Read as 0; should be written with 0.

Note: When in external boot mode (see [Page 13-21](#)) the reset value of BUSCON0 is overwritten automatically (subsequent to the release of reset) as a result of the external Boot Configuration Value fetch.

13.10.7 Bus Access Parameter Register, BUSAPx

EBU_BUSAPx (x = 0-3)

EBU Bus Access Parameter Register

(100_H+x*8_H)

Reset Value: FFFF FFFF_H



Field	Bits	Type	Description
DTACS	[3:0]	rw	<p>Recovery Cycles between Different Regions</p> <p>This bit field determines the number of clock cycles of the Recovery Phase between consecutive accesses directed to different regions. The total number of this type of Recovery Phase clock cycles is defined by DTACS multiplied by the factor selected by EBU_BUSCONx.CMULT (see also Page 13-47).</p> <p>0000_B No Recovery Phase clock cycles available. 0001_B 1 clock cycle selected. ..._B ... 1110_B 14 clock cycles selected. 1111_B 15 clock cycles selected.</p>

Field	Bits	Type	Description
DTARDWR	[7:4]	rw	<p>Recovery Cycles between Read and Write Accesses</p> <p>This bit field determines the basic number of clock cycles of the Recovery Phase between a read and write access, and vice versa. The total number of this type of Recovery Phase clock cycles is defined by DTARDWR multiplied by the factor selected by EBU_BUSCONx.CMULT (see also Page 13-47).</p> <p>0000_B No Recovery Phase clock cycles available. 0001_B 1 clock cycle selected. ..._B ... 1110_B 14 clock cycles selected. 1111_B 15 clock cycles selected.</p>
WRRECOVC	[10:8]	rw	<p>Recovery Cycles after Write Accesses</p> <p>This bit field determines the basic number of clock cycles of the Recovery Phase at the end of write accesses. The total number of this type of Recovery Phase clock cycles further depends on bit fields EBU_BUSCONx.CMULT and EBU_BUSCONx.MULTMAP[6] (see also Page 13-47).</p> <p>000_B No Recovery Phase clock cycles available. 001_B 1 clock cycle selected. ..._B ... 110_B 6 clock cycles selected. 111_B 7 clock cycles selected.</p>
RDRECOVC	[13:11]	rw	<p>Recovery Cycles after Read Accesses</p> <p>This bit field determines the basic number of clock cycles of the Recovery Phase at the end of read accesses. The total number of this type of Recovery Phase clock cycles further depends on bit fields EBU_BUSCONx.CMULT and BUSCONx.MULTMAP[5] (see also Page 13-47).</p> <p>000_B No Recovery Phase clock cycles available. 001_B 1 clock cycle selected. ..._B ... 110_B 6 clock cycles selected. 111_B 7 clock cycles selected.</p>

LMB External Bus Unit

Field	Bits	Type	Description
DATA_C	[15:14]	rw	<p>Data Hold Cycles for Write Accesses</p> <p>This bit field determines the basic number of Data Hold phase clock cycles during write accesses. The total number of Data Hold phase cycles further depends on bit fields EBU_BUSCONx.CMULT and EBU_BUSCONx.MULTMAP[4] (see also Page 13-45).</p> <p>00_B No Data Hold Phase clock cycles available. 01_B 1 clock cycle selected. 10_B 2 clock cycles selected. 11_B 3 clock cycles selected.</p>
BURST_C	[18:16]	rw	<p>Data Cycles during Burst Accesses</p> <p>This bit field determines the basic number of data cycles during burst accesses. The total number of burst data cycles further depends on bit fields EBU_BUSCONx.CMULT and EBU_BUSCONx.MULTMAP[3] (see also Page 13-46).</p> <p>000_B No burst data cycle selected. 001_B 1 clock cycle selected. ..._B ... 110_B 6 clock cycles selected. 111_B 7 clock cycles selected.</p>
WAITWRC	[21:19]	rw	<p>Write Command Phase Cycles</p> <p>This bit field determines the basic number of Command Phase clock cycles during write accesses. The total number of Command Phase clock cycles for write accesses is defined by WAITWRC multiplied by EBU_BUSCONx.CMULT (see also Page 13-43).</p> <p>000_B Reserved; do not use this combination. 001_B 1 clock cycle selected. 010_B 2 clock cycle selected. ..._B ... 111_B 7 clock cycle selected.</p>

Field	Bits	Type	Description
WAITRDC	[24:22]	rw	<p>Read Command Phase Cycles</p> <p>This bit field determines the basic number of Command Phase clock cycles during read accesses. The total number of Command Phase clock cycles for read accesses is defined by WAITRDC multiplied by EBU_BUSCONx.CMULT (see also Page 13-43).</p> <p>000_B Reserved; do not use this combination. 001_B 1 clock cycle selected. 010_B 2 clock cycle selected. ..._B ... 111_B 7 clock cycle selected.</p>
CMDDELAY	[27:25]	rw	<p>Command Delay Cycles</p> <p>This bit field determines the basic number of Command Delay phase clock cycles. The total number of Command Delay phase clock cycles further depends on bit fields EBU_BUSCONx.CMULT and EBU_BUSCONx.MULTMAP[2] (see also Page 13-43).</p> <p>000_B 0 clock cycle selected. 001_B 1 clock cycle selected. ..._B ... 110_B 6 clock cycles selected. 111_B 7 clock cycles selected.</p>
1	[29:28]	rw	<p>Reserved</p> <p>Rading these bits will return the value last written; read as 1 after reset.</p>
ADDRC	[31:30]	rw	<p>Address Cycles</p> <p>This bit field determines the basic number of clock cycles of the address phase. The total number of address cycles further depends on bit fields BUSCONx.CMULT and BUSCONx.MULTMAP[0] (see also Page 13-41).</p> <p>00_B 1 clock cycle selected 01_B 1 clock cycle selected 10_B 2 clock cycles selected 11_B 3 clock cycles selected</p>

LMB External Bus Unit

Note: When in external boot mode (see [Page 13-21](#)), the reset value of BUSAP0 is overwritten automatically (subsequent to the release of reset) as a result of the external Boot Configuration Value fetch.

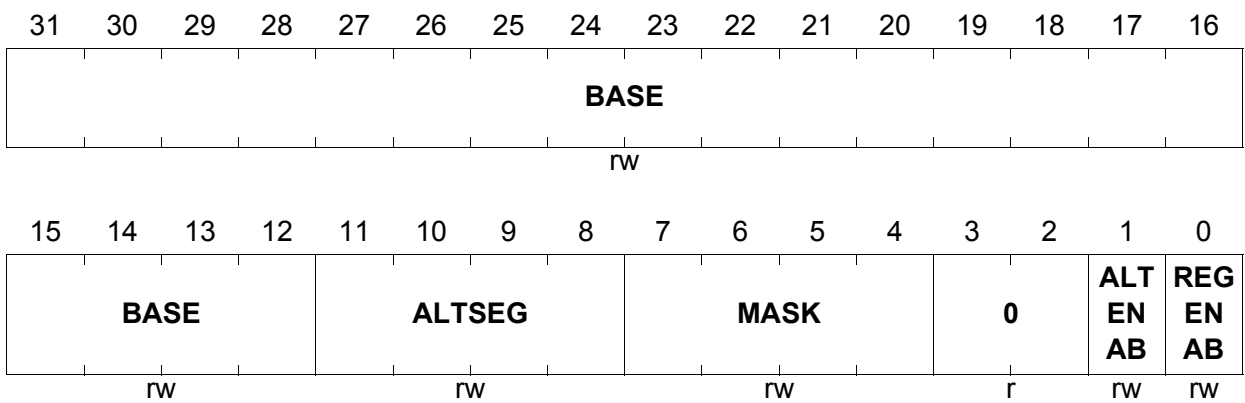
13.10.8 Emulator Address Select Register, EMUAS

EBU_EMUAS

EBU Emulator Address Select Register

(160_H)

Reset Value: DE00 0031_H



Field	Bits	Type	Description
REGENAB	0	rw	Memory Region Enable 0 _B Memory region is disabled. 1 _B Memory region is enabled (default after reset).
ALTENAB	1	rw	Alternate Segment Comparison Enable 0 _B ALTSEG is never compared to LMB address (default after reset). 1 _B ALTSEG is always compared to LMB address.
MASK	[7:4]	rw	Memory Region Address Mask Specifies the number of right-most bits in the base address starting at bit 26, which should be included in the address comparison. Bits [31:27] will always be part of the comparison.
ALTSEG	[11:8]	rw	Memory Region Alternate Segment Alternate segment to be compared to LMB address bit [31:28].
BASE	[31:12]	rw	Memory Region Base Address Base address to be compared to LMB address in conjunction with the mask control.

LMB External Bus Unit

Field	Bits	Type	Description
0	[3:2]	r	Reserved Read as 0; should be written with 0.

13.10.9 Emulator Bus Configuration Register, EMUBC

EBU_EMUBC

EBU Emulator Bus Configuration Register

(168_H)

Reset Value: 0190 2077_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRI	AGEN		0	WAIT		PORTW		BCGEN		WAI	PRE	DLO	ENDI		
TE										TINV	FET	AD	AN		
rW	rW		rW	rW		rW		rW		rW	rW	rW	rW	rW	rW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMULT		0	CTYPE		AALI	WEA	0		MULTMAP						
					GN	KPR									
					rW	rW	r		rW						
rW		r	rW		rW	rW	r		rW						

Field	Bits	Type	Description
MULTMAP	[6:0]	rw	<p>Multiplier Map</p> <p>This bit field is a mask value that determines whether or not a specific bus access parameter as defined in register EMUBAP is multiplied by the value of bit field CMULT.</p> <p>XXXXXX0_BADDRC is not multiplied. XXXXX1_BADDRC is multiplied by CMULT. XXXXX0X_BReserved, don't care. XXXXX1X_BReserved, don't care. XXXX0XX_BCMDDELAY is not multiplied. XXXX1XX_BCMDDELAY is multiplied by CMULT. XXX0XXX_BBURSTC is not multiplied. XXX1XXX_BBURSTC is not multiplied. XX0XXXX_BDATAAC is not multiplied. XX1XXXX_BDATAAC is not multiplied. X0XXXXX_BRDRECOVC is not multiplied. X1XXXXX_BRDRECOVC is multiplied by CMULT. 0XXXXXX_BWRRECOVC is not multiplied. 1XXXXXX_BWRRECOVC is multiplied by CMULT.</p> <p>Note that the bus access parameters WAITRDC, WAITWRC, DTARDWR, and DTACS in register BUSAPx are not programmable and always use the CMULT multiplier.</p>
WEAKPREF	8	rw	<p>Weak Prefetch</p> <p>0_B Code prefetch cannot be aborted by an interrupting data access. 1_B Code prefetch can be aborted by an interrupting data access.</p>
AALIGN	9	rw	<p>Address Alignment</p> <p>0_B The EBU always issues a byte address when performing an external bus access via this chip select. 1_B The EBU aligns the address according to the setting of the PORTW bit field. (see Section 13.5.7)</p>
CTYPE	[11:10]	rw	<p>Cycle Type</p> <p>This bit field is 00_B after reset. In the TC1796, CTYPE must always be written with 00_B.</p>

LMB External Bus Unit

Field	Bits	Type	Description
CMULT	[15:13]	rw	<p>Cycle Multiplier Control This bit field specifies a multiplier for the cycles specified via MULTMAP. WAITRDC, WAITWRC, DTARDWR and DTACS always use this multiplier.</p> <p>000_B Multiplier is 1. 001_B Multiplier is 4 (default after reset). 010_B Multiplier is 8. 011_B Multiplier is 16. 100_B Multiplier is 32. 1010_B Reserved. 11X_B Reserved.</p>
ENDIAN	16	rw	<p>Endian Mode 0_B Little-endian access mode selected (default after reset). 1_B Big-endian access mode selected.</p>
DLOAD	17	rw	<p>Enforce Data Upload from External Bus 0_B Data access is fed from data write buffer if it is available (default after reset). 1_B Data access is always fed from the external bus access.</p>
PREFETCH	18	rw	<p>Prefetch Mechanism for Each Code Access 0_B Code access never uses prefetch buffer mechanism (default after reset). 1_B Code access always uses prefetch buffer mechanism.</p>
WAITINV	19	rw	<p>Reversed Polarity at <u>WAIT</u> 0_B Input signal at the <u>WAIT</u> pin is active low (default after reset). 1_B Input signal at the <u>WAIT</u> pin is active high (reversed polarity). This bit has no effect when using Burst Flash Data Handshake Mode.</p>

LMB External Bus Unit

Field	Bits	Type	Description
BCGEN	[21:20]	rw	<p>Byte Control Signal Control This bit field selects the timing mode of the byte control signals.</p> <p>00_B Byte control signals follow chip select timing. 01_B <u>Byte control</u> signals follow control signal timing (<u>RD</u>, <u>RD/WR</u>) (default after reset). 10_B Byte control <u>signals</u> follow write enable signal timing (<u>RD/WR</u> only). 11_B Reserved.</p>
PORTW	[23:22]	rw	<p>Port Width</p> <p>00_B Reserved. 01_B 16-bit. 10_B 32-bit (default after reset). 11_B Reserved.</p>
WAIT	[25:24]	rw	<p>External Wait State Control This bit determines the function of the <u>WAIT</u> input. This is specific to the device type (i.e. the AGEN field).</p> <p>00_B OFF (default after reset). 01_B Asynchronous input at <u>WAIT</u> (default after reset). 10_B Synchronous input at <u>WAIT</u>. 11_B Data Handshake input at <u>WAIT</u> (Burst Flash devices only).</p>
0	[27:26]	rw	<p>Reserved Read as 0 after reset; must be always written with 0.</p>
AGEN	[30:28]	rw	<p>Address Generation Control This bit field selects the access type for the emulator chip select region.</p> <p>000_B Demultiplexed (asynch.) access selected (default after reset). 010_B Burst Flash access selected. Other values of AGEN are reserved and must not be used!</p>
WRITE	31	rw	<p>Memory Region Write Protection</p> <p>0_B Writes to the memory region are enabled (default after reset). 1_B Writes to the memory region are disabled.</p>

LMB External Bus Unit

Field	Bits	Type	Description
0	7, 12	r	Reserved Read as 0; should be written with 0.

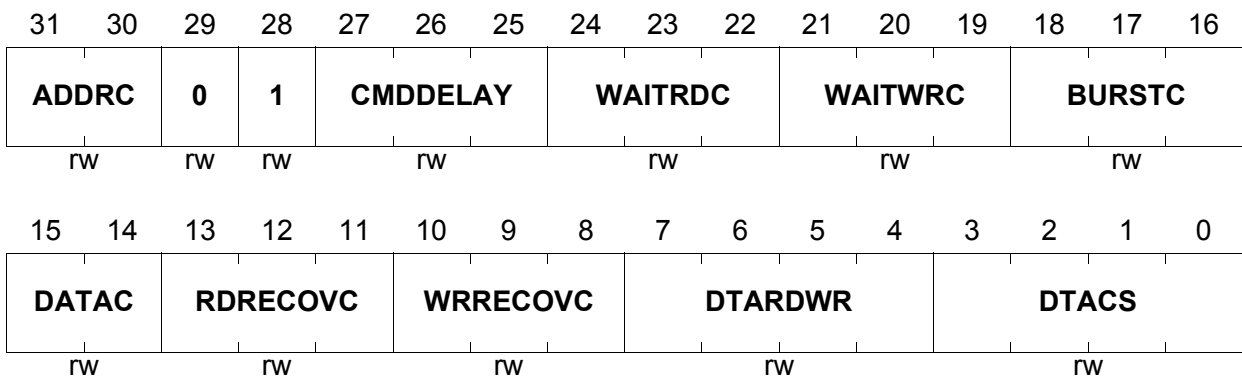
13.10.10 Emulator Bus Access Parameter Register, EMUBAP

EBU_EMUBAP

EBU Emulator Bus Access Parameter Register

(170_H)

Reset Value: 5248 4911_H



Field	Bits	Type	Description
DTACS	[3:0]	rw	<p>Recovery Cycles between Different Regions</p> <p>This bit field determines the basic number of clock cycles of the Recovery Phase between consecutive accesses directed to different regions. The total number of this type of Recovery Phase clock cycles is defined by DTACS multiplied by the factor selected by EBU_EMUBC.CMULT (see also Page 13-47).</p> <p>0000_B No Recovery Phase clock cycles available. 0001_B 1 clock cycle selected. ..._B ... 1110_B 14 clock cycles selected. 1111_B 15 clock cycles selected.</p>

Field	Bits	Type	Description
DTARDWR	[7:4]	rw	<p>Recovery Cycles between Read and Write Accesses</p> <p>This bit field determines the basic number of clock cycles of the Recovery Phase between a read and write access, and vice versa. The total number of this type of Recovery Phase clock cycles is defined by DTARDWR multiplied by the factor selected by EBU_EMUBC.CMULT (see also Page 13-47).</p> <p>0000_B No Recovery Phase clock cycles available. 0001_B 1 clock cycle selected. ..._B ... 1110_B 14 clock cycles selected. 1111_B 15 clock cycles selected.</p>
WRRECOVC	[10:8]	rw	<p>Recovery Cycles after Write Accesses</p> <p>This bit field determines the basic number of clock cycles of the Recovery Phase at the end of write accesses. The total number of this type of Recovery Phase clock cycles further depends on bit fields EBU_EMUBC.CMULT and EBU_EMUBC.MULTMAP[6] (see also Page 13-47).</p> <p>000_B No Recovery Phase clock cycles available. 001_B 1 clock cycle selected. ..._B ... 110_B 6 clock cycles selected. 111_B 7 clock cycles selected.</p>
RDRECOVC	[13:11]	rw	<p>Recovery Cycles after Read Accesses</p> <p>This bit field determines the basic number of clock cycles of the Recovery Phase at the end of read accesses. The total number of this type of Recovery Phase clock cycles further depends on bit fields EBU_EMUBC.CMULT and EBU_EMUBC.MULTMAP[5] (see also Page 13-47).</p> <p>000_B No Recovery Phase clock cycles available. 001_B 1 clock cycle selected. ..._B ... 110_B 6 clock cycles selected. 111_B 7 clock cycles selected.</p>

LMB External Bus Unit

Field	Bits	Type	Description
DATA_C	[15:14]	rw	<p>Data Hold Cycles for Write Accesses</p> <p>This bit field determines the basic number of Data Hold phase clock cycles during write accesses. The total number of Data Hold phase cycles further depends on bit fields EBU_EMUBC.CMULT and EBU_EMUBC.MULTMAP[4] (see also Page 13-45).</p> <p>00_B No Data Hold Phase clock cycles available. 01_B 1 clock cycle selected. 10_B 2 clock cycles selected. 11_B 3 clock cycles selected.</p>
BURST_C	[18:16]	rw	<p>Data Cycles during Burst Accesses</p> <p>This bit field determines the basic number of data cycles during burst accesses. The total number of burst data cycles further depends on bit fields EBU_EMUBC.CMULT and EBU_EMUBC.MULTMAP[3] (see also Page 13-46).</p> <p>000_B No burst data cycle selected. 001_B 1 clock cycle selected. ..._B ... 110_B 6 clock cycles selected. 111_B 7 clock cycles selected.</p>
WAITWRC	[21:19]	rw	<p>Write Command Phase Cycles</p> <p>This bit field determines the basic number of Command Phase clock cycles during write accesses. The total number of Command Phase clock cycles for write accesses is defined by WAITWRC multiplied by EBU_EMUBC.CMULT (see also Page 13-43).</p> <p>000_B Reserved; do not use this combination. 001_B 1 clock cycle selected. 010_B 2 clock cycle selected. ..._B ... 111_B 7 clock cycle selected.</p>

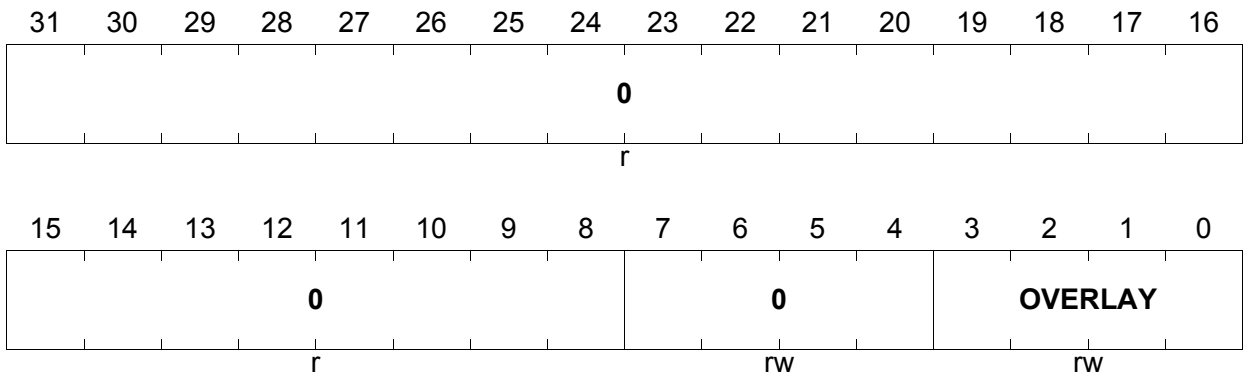
LMB External Bus Unit

Field	Bits	Type	Description
WAITRDC	[24:22]	rw	<p>Read Command Phase Cycles</p> <p>This bit field determines the basic number of Command Phase clock cycles during read accesses. The total number of Command Phase clock cycles for read accesses is defined by WAITRDC multiplied by EBU_EMUBC.CMULT (see also Page 13-43).</p> <p>000_B Reserved; do not use this combination. 001_B 1 clock cycle selected. 010_B 2 clock cycle selected. ..._B ... 111_B 7 clock cycle selected.</p>
CMDDELAY	[27:25]	rw	<p>Command Delay Cycles</p> <p>This bit field determines the basic number of Command Delay phase clock cycles. The total number of Command Delay phase clock cycles further depends on bit fields EBU_EMUBC.CMULT and EBU_EMUBC.MULTMAP[2] (see also Page 13-43).</p> <p>000_B 0 clock cycle selected. 001_B 1 clock cycle selected. ..._B ... 110_B 6 clock cycles selected. 111_B 7 clock cycles selected.</p>
1	28	r	<p>Reserved</p> <p>Reading these bits will return the value last written; read as 1 after reset.</p>
0	29	r	<p>Reserved</p> <p>Reading these bits will return the value last written; read as 0 after reset.</p>
ADDRC	[31:30]	rw	<p>Address Cycles</p> <p>This bit field determines the basic number of clock cycles of the address phase. The total number of address cycles further depends on bit fields EBU_EMUBC.CMULT and EBU_EMUBC.MULTMAP[0] (see also Page 13-41).</p> <p>00_B 1 clock cycle selected 01_B 1 clock cycle selected 10_B 2 clock cycles selected 11_B 3 clock cycles selected</p>

13.10.11 Emulator Overlay Register, EMUOVL

EBU_EMUOVL

EBU Emulator Overlay Register (178_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
OVERLAY	[3:0]	rw	<p>Overlay Chip Select Signal</p> <p>The bits of this bit field are <u>used</u> to enable the $\overline{CS[3:0]}$ chip select lines for the \overline{CSOVL} overlay chip select generation. With $x = 0-3$, the OVERLAY bits are defined as follows (see also Page 13-29):</p> <p>0_B \overline{CSOVL} is not activated by an active \overline{CSx}.</p> <p>1_B \overline{CSOVL} is activated when \overline{CSx} becomes active.</p> <p>Bits [3:0] are assigned to $\overline{CS[3:0]}$.</p>
0	[7:4]	rw	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>
0	[31:8]	r	<p>Reserved</p> <p>Read as 0.</p>

14 Interrupt System

The TC1796 interrupt system provides a flexible and time-efficient means for processing interrupts. This chapter describes the interrupt system for the TC1796. Topics covered include the architecture of the interrupt system, interrupt system configuration, and the interrupt operations of the TC1796 peripherals and Central Processing Unit (CPU). General information is also given about the Peripheral Control Processor (PCP). For details about that unit, see [Chapter 11](#).

This chapter also discusses the Non-Maskable Interrupt (NMI) (see [Section 14.10 on Page 14-25](#)).

14.1 Overview

An interrupt request can be serviced either by the CPU or by the PCP. These units are called “service providers”. Interrupt requests are called “service requests” rather than “interrupt requests” in this document because they can be serviced by either of the service providers.

Each peripheral unit in the TC1796 can generate service requests. Additionally, the Bus Control Units, the Debug Unit, the PCP, and even the CPU itself can generate service requests to either of the two service providers.

As shown in [Figure 14-1](#), each TC1796 unit that can generate service requests is connected to one or more Service Request Nodes (SRN). Each SRN contains a Service Request Control Register `mod_SRCx`, where “mod” is the identifier of the service requesting unit and “x” an optional index. Two arbitration buses connect the SRNs with two Interrupt Control Units (ICUs), which handle interrupt arbitration among competing interrupt service requests, as follows:

- The Interrupt Control Unit (ICU) arbitrates service requests for the CPU and administers the CPU interrupt arbitration bus.
- The Peripheral Interrupt Control Unit (PICU) arbitrates service requests for the PCP and administers the PCP interrupt arbitration bus.

The PCP can make service requests directly to itself (via the PICU), or it can make service requests to the CPU. The Debug Unit can generate service requests to the PCP or the CPU. The CPU can make service requests directly to itself (via the ICU), or it can make service requests to the PCP. The CPU Service Request Nodes are activated through software.

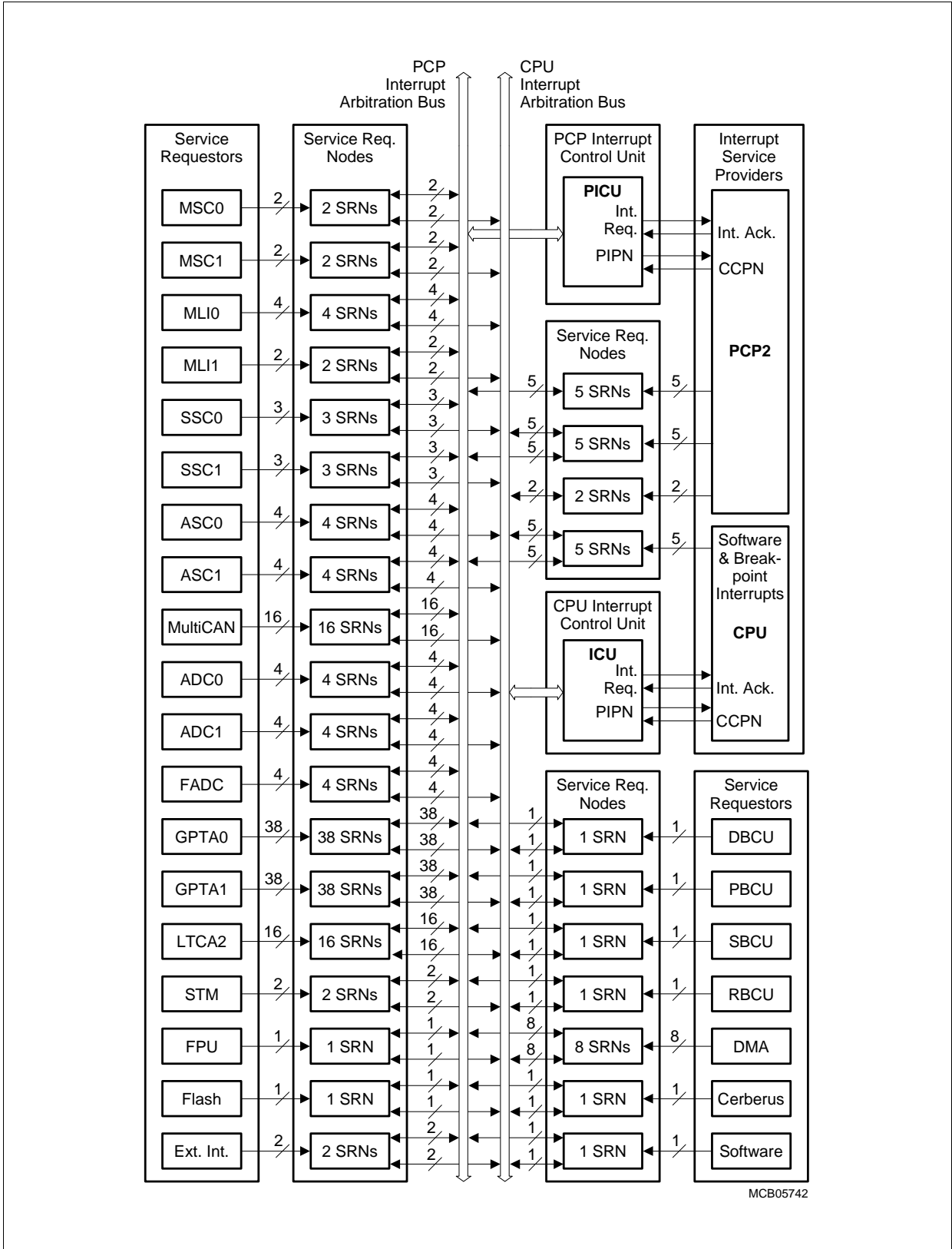


Figure 14-1 Block Diagram of the TC1796 Interrupt System

14.2 Service Request Nodes

In total, there are 181 Service Request Nodes available in the TC1796. Each SRN contains a Service Request Control Register and interface logic that connects it to the triggering unit and to the two interrupt arbitration buses. Some peripheral units of the TC1796 have multiple SRNs.

14.2.1 Service Request Control Registers

All Service Request Control Registers in the TC1796 have the same format. In general, these registers contain:

- Enable/disable information
- Priority information
- Service provider destination
- Service request status bit
- Software-initiated service request set and clear bits

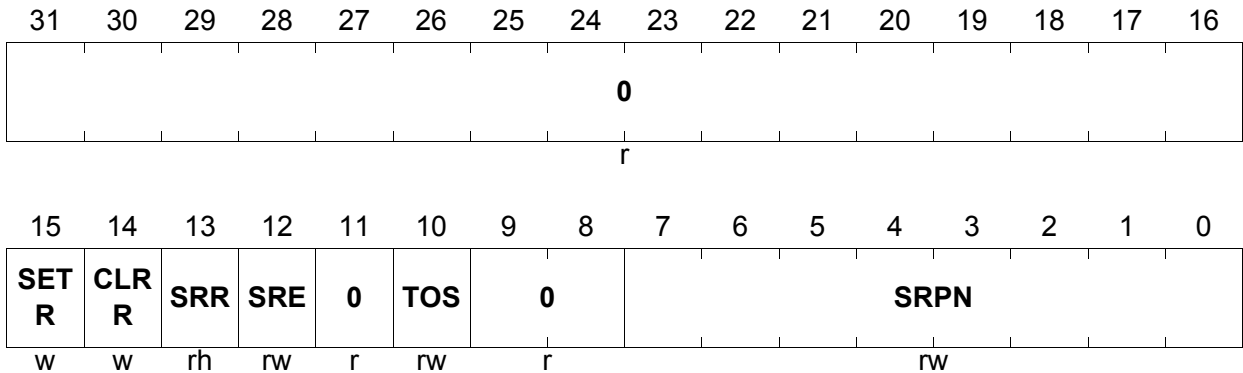
Besides being activated by the associated triggering unit through hardware, each SRN can also be set or cleared by software via two software-initiated service request control bits.

Note: The description given in this chapter characterizes all Service Request Control Registers of the TC1796. Further Information on peripheral module interrupt functions such as enable or request flags is provided in the corresponding sections of the module chapters.

14.2.1.1 General Service Request Control Register Layout

mod_SRC

Service Request Control Register (00_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
SRPN	[7:0]	rw	Service Request Priority Number 00 _H Service request is never serviced 01 _H Service request is on lowest priority ... _H ... FF _H Service request is on highest priority
TOS	10	rw	Type of Service Control 0 _B CPU service is initiated 1 _B PCP request is initiated
SRE	12	rw	Service Request Enable 0 _B Service request is disabled 1 _B Service request is enabled
SRR	13	rh	Service Request Flag 0 _B No service request is pending 1 _B A service request is pending
CLRR	14	w	Request Clear Bit CLRR is required to clear SRR. 0 _B No action 1 _B Clear SRR; bit value is not stored; read always returns 0; no action if SETR is set also.

Field	Bits	Type	Description
SETR	15	w	Request Set Bit SETR is required to set SRR. 0 _B No action 1 _B Set SRR; bit value is not stored; read always returns 0; no action if CLRR is set also.
0	[9:8], 11, [31:16]	r	Reserved Read as 0; should be written with 0.

14.2.1.2 Request Set and Clear Bits (SETR, CLRR)

The SETR and CLRR bits allow software to set or clear the service request bit SRR. Writing a 1 to SETR causes bit SRR to be set to 1. Writing a 1 to CLRR causes bit SRR to be cleared to 0. If hardware attempts to modify SRR during a read-modify-write software operation (such as the bit-set or bit-clear instructions), the software operation will succeed and the hardware operation will have no effect.

The value written to SETR or CLRR is not stored. Writing a 0 to these bits has no effect. These bits always return 0 when read. If both, SETR and CLRR, are set to 1 at the same time, SRR is not changed.

14.2.1.3 Enable Bit (SRE)

The SRE bit enables an interrupt to take part in the arbitration for the selected service provider. It does not enable or disable the setting of the request flag SRR; the request flag can be set by hardware or by software (via SETR) independent of the state of the SRE bit. This allows service requests to be handled automatically by hardware or through software polling.

If SRE = 1, pending service requests are passed on to the designated service provider for interrupt arbitration. The SRR bit is automatically set to 0 by hardware when the service request is acknowledged and serviced. It is recommended that in this case, software should not modify the SRR bit to avoid unexpected behavior due to the hardware controlling this bit.

If SRE = 0, pending service requests are not passed on to service providers. Software can poll the SRR bit to check whether a service request is pending. To acknowledge the service request, the SRR bit must then be cleared by software by writing a 1 to CLRR.

Note: In this document, “active source” means an SRN whose Service Request Control Register has its request enable bit SRE set to 1 to allow its service requests to participate in interrupt arbitration.

14.2.1.4 Service Request Flag (SRR)

When set, the SRR flag indicates that a service request is pending. It can be set or cleared directly by hardware or indirectly through software using the SETR and CLRR bits. Writing directly to this bit via software has no effect.

The SRR status bit can be directly set or cleared by the associated hardware. For instance, in the General Purpose Array Unit, an associated timer event can cause this bit to be set to 1. The details of how hardware events can cause the SRR bit to be set are defined in the individual peripheral/module chapters.

The acknowledgment of the service request by either the Interrupt Control Unit (ICU) or the PCP Interrupt Control Unit (PICU) causes the SRR bit to be cleared.

SRR can be set or cleared either by hardware or by software regardless of the state of the enable bit SRE. However, the request is only forwarded for service if the enable bit is set. If $SRE = 1$, a pending service request takes part in the interrupt arbitration of the service provider selected by the device's TOS bit. If $SRE = 0$, a pending service request is excluded from interrupt arbitrations.

SRR is automatically cleared by hardware when the service request is acknowledged and serviced. Software can poll SRR to check for a pending service request. SRR must be cleared by software in this case by writing a 1 to CLRR.

It is advised not to clear a pending service request flag SRR (writing $CLRR = 1$) and to enable the corresponding service request node SRN (writing $SRE = 1$) simultaneously at the same write access to the Service Request Control Register. When doing this, an unintended interrupt request might be generated. Instead of executing one write access, it is recommended to split the two actions into two consecutive write accesses to the corresponding Service Request Control Register, starting first clearing the pending interrupt flag and afterwards enabling the service request node.

14.2.1.5 Type-Of-Service Control (TOS)

There are two service providers for service requests in the TC1796, the CPU and the PCP. The TOS bit is used to select whether a service request generates an interrupt to the CPU ($TOS = 0$) or to the PCP ($TOS = 1$). Bit 11 of the Service Request Control Register is read-only, returning 0 when read. Writing to this bit position has no effect. However, to ensure compatibility with future extensions, bit 11 should always be written with a 0.

Note that several Service Request Control Registers (e.g. in the PCP) have a hardwired TOS bit (0 or 1) that cannot be written. These registers can only generate an interrupt to one dedicated service provider (PCP or CPU).

Note: Before modifying the content of a TOS bit, the corresponding SRN must be disabled ($SRE = 0$).

14.2.1.6 Service Request Priority Number (SRPN)

The 8-bit Service Request Priority Number (SRPN) indicates the priority of a service request with respect to other sources requesting service from the same service provider, and with respect to the priority of the service provider itself.

Each active source selecting the same service provider must have a unique SRPN value to differentiate its priority. The special SRPN value of 00_H excludes an SRN from taking part in arbitration, regardless of the state of its SRE bit. The SRPN values for active sources selecting different service providers (CPU vs. PCP) may overlap. If a source is not active – meaning its SRE bit is 0 – no restrictions are applied to the service request priority number.

The SRPN is used by service providers to select an Interrupt Service Routine (ISR) or Channel Program (in case of the PCP) to service the request. ISRs are associated with Service Request Priority Numbers by an Interrupt Vector Table located in each service provider. This means that the TC1796 Interrupt Vector Table is ordered by priority number. This is unlike traditional interrupt architectures in which their interrupt vector tables are ordered by the source of the interrupt. The TC1796 Interrupt Vector Table allows a single peripheral can have multiple priorities for different purposes.

The range of values for SRPNs used in a system depends on the number of possible active service requests and the user-definable organization of the Interrupt Vector Table. The 8-bit SRPNs permit up to 255 sources to be active at one time (remembering that the special SRPN value of 00_H excludes an SRN from taking part in arbitration).

Note: Before modifying the content of a SRPN bit field, the corresponding SRN must be disabled (SRE = 0).

SRPNs in the TC1796

In the TC1796, interrupt sources selecting the same service provider are also allowed to have identical SRPN values. In this case, the software (interrupt service routine) must check which of the interrupt sources with identical SRPN has become active.

Note that module-specific interrupt request flags must be available because the SRR flags cannot be used for this check. SRR flags (meaning all SRR flags of interrupts with identical SRPN values) are in general automatically cleared by hardware when a service request is acknowledged and serviced.

Note: This practice with identical SRPN values is not recommended as it is not portable to other TriCore devices.

14.3 Interrupt Control Units

The Interrupt Control Units manage the interrupt system, arbitrate incoming service requests, and determine whether and when to interrupt the service provider. The TC1796 contains two interrupt control units, one for the CPU (called ICU), and one for the PCP (called PICU). Each one controls its associated interrupt arbitration bus and manages the communication with its service provider (see [Figure 14-1](#)).

14.3.1 Interrupt Control Unit (ICU)

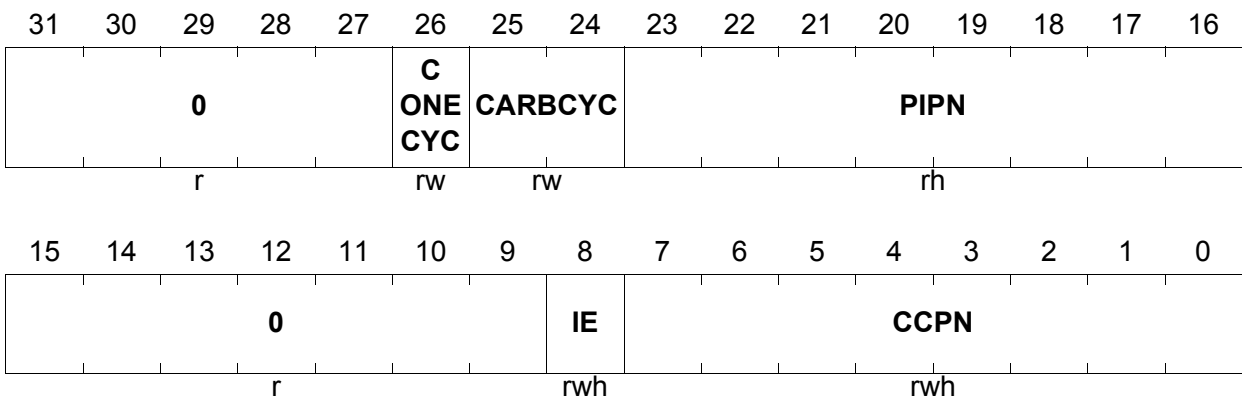
This section describes the interrupt control unit (ICU) for the CPU.

14.3.1.1 ICU Interrupt Control Register (ICR)

The ICU Interrupt Control Register ICR holds the current CPU priority number (CCPN), the global interrupt enable/disable bit (IE), the pending interrupt priority number (PIPn), and bit fields which control the interrupt arbitration process.

ICR

ICU Interrupt Control Register (F7E1FE2C_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
CCPN	[7:0]	rwh	<p>Current CPU Priority Number</p> <p>The Current CPU Priority Number (CCPN) bit field indicates the current priority level of the CPU. It is automatically updated by hardware on entry and exit of interrupt service routines, and through the execution of a BISR instruction. CCPN can also be updated through an MTCR instruction.</p>

Interrupt System

Field	Bits	Type	Description
IE	8	rwh	<p>Global Interrupt Enable Bit</p> <p>The interrupt enable bit globally enables the CPU service request system. Whether or not a service request is delivered to the CPU depends on the individual Service Request Enable Bits (SRE) in the SRNs, and the current state of the CPU.</p> <p>IE is automatically updated by hardware on entry and exit of an Interrupt Service Routine (ISR).</p> <p>IE is cleared to 0 when an interrupt is taken, and is restored to the previous value when the ISR executes an RFE instruction to terminate itself.</p> <p>IE can also be updated through the execution of the ENABLE, DISABLE, MTCR, and BISR instructions.</p> <p>0_B Interrupt system is globally disabled 1_B Interrupt system is globally enabled</p>
PIPN	[23:16]	rh	<p>Pending Interrupt Priority Number</p> <p>PIPN is a read-only bit field that is updated by the ICU at the end of each interrupt arbitration process. It indicates the priority number of the pending service request. PIPN is set to 0 when no request is pending, and at the beginning of each new arbitration process.</p> <p>00_H No valid pending request YY_H A request with priority YY_H is pending</p>
CARBCYC	[25:24]	rw	<p>Number of Arbitration Cycles</p> <p>CARBCYC controls the number of arbitration cycles used to determine the request with the highest priority.</p> <p>00_B 4 arbitration cycles (default) 01_B 3 arbitration cycles 10_B 2 arbitration cycles 11_B 1 arbitration cycle</p>
CONECYC	26	rw	<p>Number of Clocks per Arbitration Cycle Control</p> <p>The CONECYC bit determines the number of system clocks per arbitration cycle. This bit should be set to 1 only for system designs utilizing low system clock frequencies.</p> <p>0_B 2 clocks per arbitration cycle (default) 1_B 1 clock per arbitration cycle</p>
0	[15:9], [31:27]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

14.3.1.2 Operation of the Interrupt Control Unit (ICU)

Service-request arbitration is performed in the ICU in parallel with normal CPU operation. When a triggering event occurs in one or more interrupt sources, the associated SRNs, if enabled, send service requests to the CPU through the ICU. The ICU determines which service request has the highest priority. The ICU will then forward the service request to the CPU. The service request will be acknowledged by the CPU and serviced, depending upon the state of the CPU.

The ICU arbitration process takes place in one or more arbitration cycles over the CPU interrupt arbitration bus. The ICU begins a new arbitration process when a new service request is detected. At the end of the arbitration process, the ICU will have determined the service request with the highest priority number. This number is stored in the ICR.PIPN bit field and becomes the pending service request.

After the arbitration process, the ICU forwards the pending service request to the CPU by attempting to interrupt it. The CPU can be interrupted only if interrupts are enabled globally (that is, ICR.IE = 1) and if the priority of the service request is higher than the current processor priority (ICR.PIPN > ICR.CCPN). Also, the CPU may be temporarily blocked from taking interrupts, for example, if it is executing a multi-cycle instruction such as an atomic read-modify-write operation. The full list of conditions which could block the CPU from immediately responding to an interrupt request generated by the ICU is:

- Current CPU priority, ICR.CCPN, is equal to or higher than the pending interrupt priority, ICR.PIPN
- Interrupt system is globally disabled (ICR.IE = 0)
- CPU is in the process of entering an interrupt- or trap-service routine
- CPU is executing non-interruptible trap services
- CPU is executing a multi-cycle instruction
- CPU is executing an instruction which modifies the conditions of the global interrupt system, such as modifying the ICR
- CPU detects a trap condition (such as context depletion) when trying to enter a service routine

When the CPU is not otherwise prevented from taking an interrupt, the CPU's program counter will be directed to the Interrupt Service Routine entry point associated with the priority of the service request. Next, the CPU saves the value of ICR.PIPN internally, and acknowledges the ICU. The ICU then forwards the acknowledge signal back to the SRN that is requesting service in order to inform it that it will be serviced by the CPU. The SRR bit in this SRN is then cleared to 0.

After sending the acknowledgement, the ICU clears ICR.PIPN to 0 and may start a new arbitration process if there is another pending interrupt request. If not, ICR.PIPN remains at 0 and the ICU enters an idle state, waiting for the next interrupt request to awaken it. If there is a new service request waiting, the priority number of the new request will be written to ICR.PIPN at the end of the new arbitration process and the ICU will deliver the pending interrupt to the CPU according to the rules described in this section.

Interrupt System

If a new service request is received by the ICU before the CPU has acknowledged the pending interrupt request, the ICU deactivates the pending request and starts a new arbitration process. This reduces the latency of service requests posted before the current request is acknowledged. The ICU deactivates the current pending interrupt request by setting the ICR.PIPN bit field to 0, indicating that the ICU has not yet found a new valid pending request. It then executes its arbitration process again. If the new service request has a higher priority than the previous one, its priority will be written to ICR.PIPN. If the new interrupt has a lower priority, the priority of the previous interrupt request will again be written to ICR.PIPN. In any case, the ICU will deliver a new interrupt request to the CPU according to the rules described in this section.

Once the CPU has acknowledged the current pending interrupt request, any new service request generated by an SRN must wait at least until the end of the next service request arbitration process to be serviced.

Essentially, arbitration in the ICU is performed whenever a new service request is detected, regardless of whether or not the CPU is servicing interrupts. Because of this, the ICR.PIPN bit field always reflects the pending service request with the highest priority. This can, for example, be used by software polling techniques to determine high-priority requests while leaving the interrupt system disabled.

14.3.2 PCP Interrupt Control Unit (PICU)

The PCP Interrupt Control Unit (PICU) is closely coupled with the PCP and its Interrupt Control Register (PCP_PICR). The operation of the PICU is very similar to the ICU of the CPU with respect to the overall scheme. However, the PCP cannot handle nested interrupts; an interrupt request to the PCP cannot interrupt the service of another interrupt request. Thus, schemes such as interrupt priority grouping, are not feasible in the PCP.

Note: Details of the PCP_ICR register are described on [Page 11-61](#). The PICU is described on [Page 11-33](#).

14.4 Arbitration Process

The arbitration process implemented in the TC1796 uses a number of arbitration cycles to determine the pending interrupt request with the highest priority number, SRPN. In each of these cycles, two bits of the SRPNs of all pending service requests are compared against each other. The sequence starts with the high-order bits of the SRPNs and works downwards, such that in the last cycle, bits[1:0] of the SRPNs are compared. Thus, to perform an arbitration through all 8 bits of an SRPN, four arbitration cycles are required. There are two factors determining the duration of the arbitration process:

- Number of arbitration cycles, and
- Duration of arbitration cycles.

Both of these can be controlled by the user.

14.4.1 Controlling the Number of Arbitration Cycles

In a real-time system where responsiveness is critical, arbitration must be as fast as possible. However, to maintain flexibility, the TC1796 system is designed to have a large range of service priorities. If not all priorities are needed in a system, arbitration can be speeded up by not examining all the bits used to identify all 255 unique priorities.

For instance, if a 6-bit number is enough to identify all priority numbers used in a system, (meaning that bits [7:6] of all SRPNs are always 0), it is not necessary to perform arbitration on these two bits. Three arbitration cycles will be enough to find the highest number in bits [5:0] of the SRPNs of all pending requests. Similarly, the number of arbitration cycles can be reduced to two if only bits [3:0] are used in all SRPNs, and the number of arbitration cycles can be reduced to one cycle if only bits [1:0] are used.

The ICR.CARBCYC bit field controls the number of cycles in the arbitration process. Its default value is 0, which selects four arbitration cycles. [Table 14-1](#) gives the options for arbitration cycle control.

Table 14-1 Arbitration Cycle Control

Number of Arbitration Cycles	4	3	2	1
ICR.CARBCYC	00 _B	01 _B	10 _B	11 _B
Relevant bits of the SRPNs	[7:0]	[5:0]	[3:0]	[1:0]
Range of priority numbers covered	1..255	1..63	1..15	1..3

Note: If less than four arbitration cycles are selected, the corresponding upper bits of the SRPNs are not examined, even if they do not contain zeros.

14.4.2 Controlling the Duration of Arbitration Cycles

During each arbitration cycle, the rate of information flow between the SRNs and the ICU can become limited by propagation delays within the TC1796 when it is executing at high system clock frequencies. At high frequencies, arbitration cycles may require two system clocks to execute properly. In order to optimize the arbitration scheme at lower system frequencies, an additional control bit, ICR.CONECYC, is implemented. The default value of 0 of this bit selects two clock cycles per arbitration cycle. Setting this bit to 1 selects one clock cycle per arbitration cycle. This bit should only be set to 1 for lower system frequencies. Setting this bit for system frequencies above the specified limit leads to unpredictable behavior of the interrupt system. Correct operation is not then guaranteed.

14.5 Entering an Interrupt Service Routine

When an interrupt request from the ICU is pending and all conditions are met such that the CPU can now service the interrupt request, the CPU performs the following actions in preparation for entering the designated Interrupt Service Routine (ISR):

1. Upper context of the current task is saved¹⁾. The current CPU priority number, ICR.CCPN, and the state of the global interrupt enable bit, ICR.IE, are automatically saved with the PCXI register (bit field PCPN and bit PIE).
2. Interrupt system is globally disabled (ICR.IE is set to 0).
3. Current CPU priority number (ICR.CCPN) is set to the value of ICR.PIPN.
4. PSW is set to a default value:
 - a) All permissions are enabled, that is, PSW.IO = 10_B.
 - b) Memory protection is switched to PRS0, that is, PSW.PRS = 0.
 - c) The stack pointer bit is set to the interrupt stack, that is, PSW.IS = 1.
 - d) The call depth counter is cleared, the call depth limit is set to 63, that is, PSW.CDC = 0.
5. Stack pointer, A10, is reloaded with the contents of the Interrupt Stack Pointer, ISP, if the PSW.IS bit of the interrupted routine was set to 0 (using the user stack); otherwise it is left unaltered.
6. CPU program counter is assigned an effective address consisting of the contents of the BIV register OR-ed with the ICR.PIPN number left-shifted by 5. This indexes the Interrupt Vector Table entry corresponding to the interrupt priority.
7. Contents at the effective address of the program counter in the Interrupt Vector Table are fetched as the first instruction of the Interrupt Service Routine (ISR). Execution continues linearly from there until the ISR branches or exits.

1) Note that, if a context-switch trap occurs while the CPU is in the process of saving the upper context of the current task, the pending ISR will not be entered, the interrupt request will be left pending, and the CPU will enter the appropriate trap handling routine instead.

As explained, receipt of further interrupts is disabled ($ICR.IE = 0$) when an Interrupt Service Routine is entered. At the same time, the current CPU priority $ICR.CCPN$ is set by hardware to the priority of the interrupting source ($ICR.PIPN$).

Clearly, before the processor can receive any more interrupts, the ISR must eventually re-enable the interrupt system again by setting $ICR.IE = 1$. Furthermore, the ISR can also modify the priority number $ICR.CCPN$ to allow effective interrupt priority levels. It is up to the user to enable the interrupt system again and optionally modify the priority number $CCPN$ to implement interrupt priority levels or handle special cases.

To simply enable the interrupt system again, the `ENABLE` instruction can be used, which sets $ICR.IE$ bit to 1. The `BISR` instruction offers a convenient way to re-enable the interrupt system, to set $ICR.CCPN$ to a new value, and to save the lower context of the interrupted task. It is also possible to use an `MTCR` instruction to modify $ICR.IE$ and $ICR.CCPN$. However, this should be performed together with an `ISYNC` instruction (which synchronizes the instruction stream) to ensure completion of this operation before the execution of following instructions.

Note: The lower context can also be saved through execution of a `SVLCX` (Save Lower Context) instruction.

14.6 Exiting an Interrupt Service Routine

When an ISR exits with an `RFE` (Return From Exception) instruction, the hardware automatically restores the upper context. Register `PCXI`, which holds the Previous CPU Priority Number ($PCPN$) and the Previous Global Interrupt Enable Bit (PIE), is a part of this upper context. The value saved in $PCPN$ is written to $ICR.CCPN$ to set the CPU priority number to the value before the interruption, and bit PIE is written to $ICR.IE$ to restore the state of this bit. The interrupted routine then continues.

Note: There is no automatic restoring of the lower context on an exit from an Interrupt Service Routine. If the lower context was saved during the execution of the ISR, either through execution of the `BISR` instruction or a `SVLCX` instruction, the ISR must restore the lower context again via the `RSLCX` (Restore Lower Context) instruction before it exits through `RFI` execution.

14.7 Interrupt Vector Table

Interrupt Service Routines (ISRs) are associated with interrupts at a particular priority by way of the Interrupt Vector Table. The Interrupt Vector Table is an array of ISR entry points.

When the CPU takes an interrupt, it calculates an address in the Interrupt Vector Table that corresponds with the priority of the interrupt (the ICR.PIPN bit field). This address is loaded in the program counter. The CPU begins executing instructions at this address in the Interrupt Vector Table. The code at this address is the start of the selected ISR. Depending on the code size of the ISR, the Interrupt Vector Table may only store the initial portion of the ISR, such as a jump instruction that vectors the CPU to the rest of the ISR elsewhere in memory.

The Interrupt Vector Table is stored in code memory. The BIV register specifies the base address of the Interrupt Vector Table. Interrupt vectors are ordered in the table by increasing priority.

The Base of Interrupt Vector Table register (BIV) stores the base address of the Interrupt Vector Table. It can be assigned to any available code memory. Its default on power-up is fixed at 0000 0000_H. However, the BIV register can be modified using the MTCR instruction during the initialization phase of the system, before interrupts are enabled. With this arrangement, it is possible to have multiple Interrupt Vector Tables and switch between them by changing the contents of the BIV register.

Note: The BIV register is protected by the ENDINIT bit (see [Section 16.3](#)). Modifications should only be done while the interrupt system is globally disabled (ICR.IE = 0). Also, an ISYNC instruction should be issued after modifying BIV to ensure completion of this operation before execution of following instructions.

When interrupted, the CPU calculates the entry point of the appropriate ISR from the PIPN and the contents of the BIV register. The PIPN is left-shifted by five bits and OR-ed with the address in the BIV register to generate a pointer into the Interrupt Vector Table. Execution of the ISR begins at this address. Due to this operation, it is recommended that bits [12:5] of register BIV are set to 0 (see [Figure 14-2](#)). Note that bit 0 of the BIV register is always 0 and cannot be written to (instructions have to be aligned on even byte boundaries).

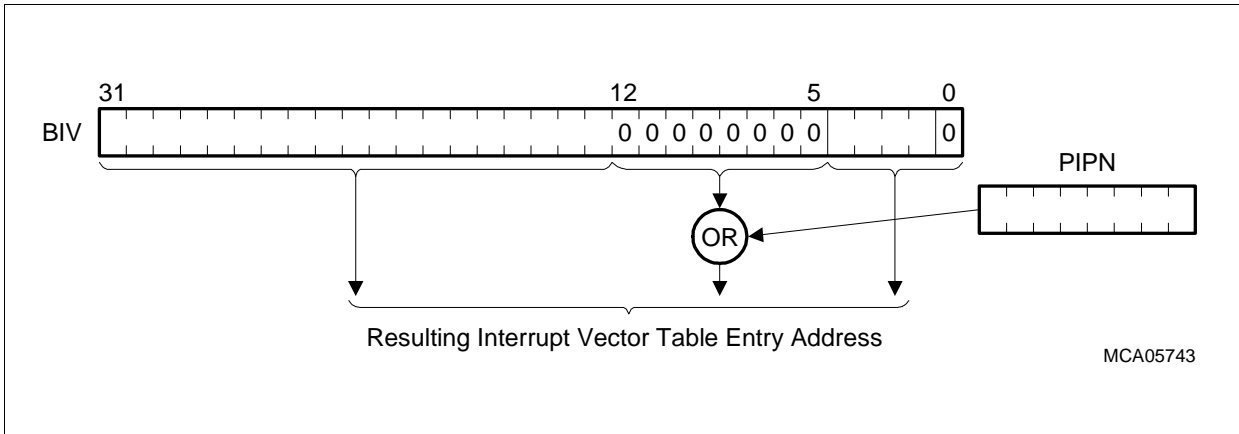


Figure 14-2 Interrupt Vector Table Entry Address Calculation

Left-shifting the PIPN by 5 bits creates entries into the Interrupt Vector Table which are evenly spaced 8 words apart. If an ISR is very short, it may fit entirely within the eight words available in the vector table entry. Otherwise, the code at the entry point must ultimately cause a jump to the rest of the ISR residing elsewhere in memory. Due to the way the vector table is organized according to the interrupt priorities, the TC1796 offers an additional option by allowing spanning several Interrupt Vector Table entries so long as those entries are otherwise unused. [Figure 14-3](#) illustrates this.

The required size of the Interrupt Vector Table depends only on the range of priority numbers actually used in a system. Of the 256 vector entries, 255 may be used. Vector entry 0 is never used, because if ICR.PIPN is 0, the CPU is not interrupted. Distinct interrupt handlers are supported, but systems requiring fewer entries need not dedicate the full memory area required by the largest configurations.

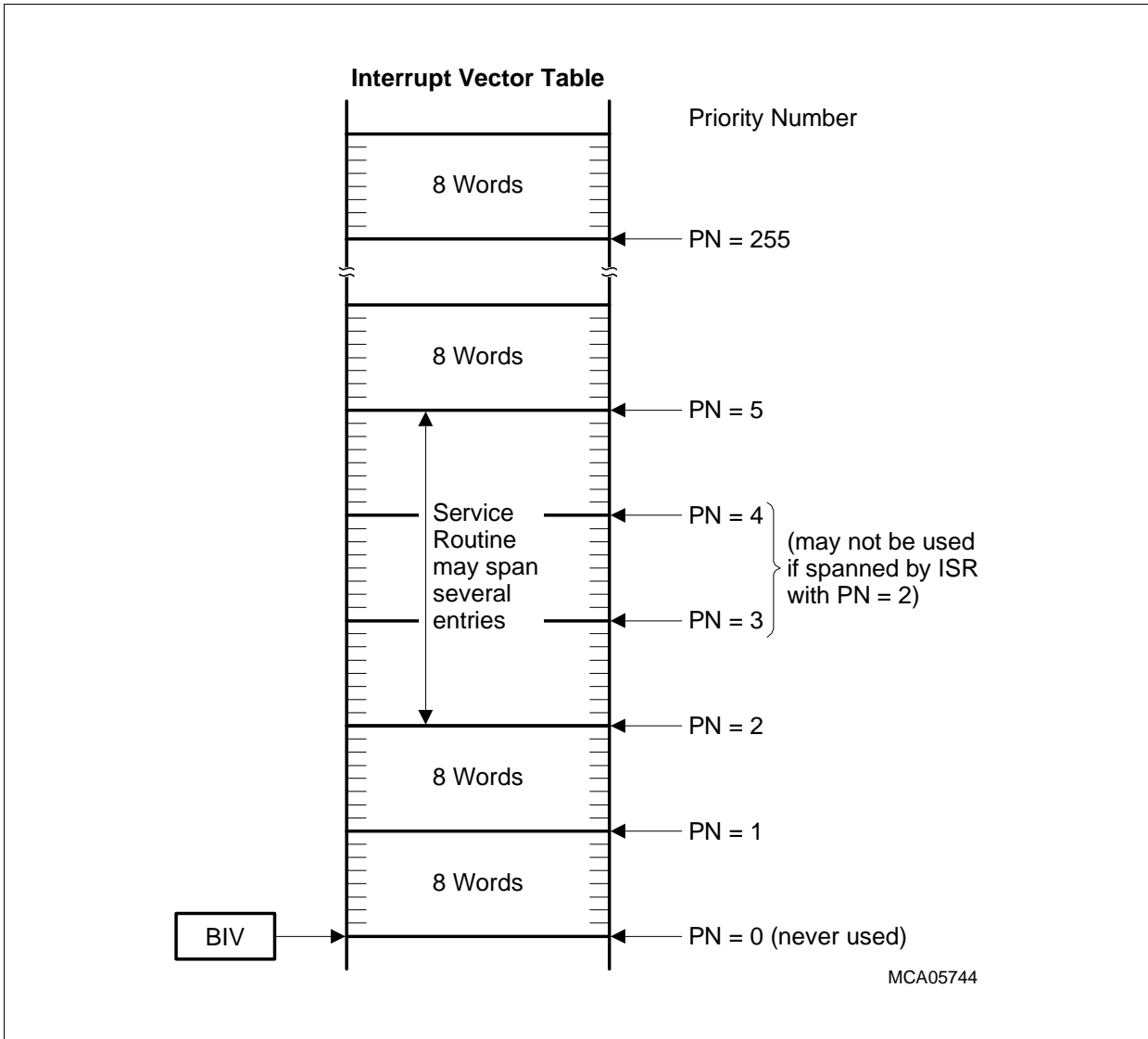


Figure 14-3 Interrupt Vector Table

14.8 Usage of the TC1796 Interrupt System

The following sections provide some examples of using the TC1796 interrupt system to solve both typical and special application requirements.

14.8.1 Spanning Interrupt Service Routines Across Vector Entries

Each Interrupt Vector Table entry consists of eight words of memory. If an ISR can be made to fit directly in the Interrupt Vector Table there is no need for a jump instruction to vector to the rest of the interrupt handler elsewhere in memory. However, only the simplest ISRs can fit in the eight words available to a single entry in the table. But it is easy to arrange for ISRs to span across multiple entries, since the Interrupt Vector Table is ordered not by the interrupt source but by interrupt priority. This technique is explained in this section.

In the example of [Figure 14-3](#), entry locations 3 and 4 are occupied by the ISR for entry 2. In [Figure 14-3](#), the next available entry after entry 2 is entry 5. Of course, if this technique is used, it would be improper to allow any SRN to request service at any of the spanned vector priorities. Thus, priority levels 3 and 4 must not be assigned to SRNs requesting CPU service. They can, however, be used to request PCP service.

There is a performance trade-off that may arise when using this technique because the range of priority numbers used increases. This may have an impact on the number of arbitration cycles required to perform arbitration. Consider the case in which a system uses only three active interrupt sources, that is, where there are only three SRNs enabled to request service. If these three active sources are assigned to priority numbers 1, 2, and 3, it would be sufficient to perform the arbitration in just one cycle. However, if the ISR for interrupt priority 2 is spanned across three Interrupt Vector Table entries as shown in [Figure 14-3](#), the priority numbers 1, 2 and 5 would have to be assigned. Thus, two arbitration cycles would have to be used to perform the full arbitration process.

The trade-off between the performance impact of the number of arbitration cycles and the performance gain through spanning service routines can be made by the system designer depending on system needs. Reducing the number of arbitration cycles reduces the service request arbitration latency - spanning service routines reduces the run time of service routines (and therefore also the latency for further interrupts at that priority level or below). For example, if there are multiple fleeting measurements to be made by a system, reducing arbitration latency may be most important. But if keeping total interrupt response time to a minimum is most urgent, spanning Interrupt Vector Table entries may be a solution.

14.8.2 Configuring Ordinary Interrupt Service Routines

When the CPU starts to service an interrupt, the interrupt system is globally disabled and the CPU priority ICR.CCPN is set to the priority of the interrupt now being serviced. This blocks all further interrupts from being serviced until the interrupt system is enabled again.

After an ordinary ISR begins execution, it is usually desirable for the ISR to re-enable global interrupts so that higher-priority interrupts (that is, interrupts that are greater than the current value of ICR.CCPN) can be serviced even during the current ISR's execution. Thus, such an ISR may set ICR.IE = 1 again with, for instance, the ENABLE instruction.

If the ISR enables the interrupt system again by setting ICR.IE = 1 but does not change ICR.CCPN, the effect is that from that point on the hardware can be interrupted by higher-priority interrupts but will be blocked from servicing interrupt requests with the same or lower priority than the current value of bit field ICR.CCPN. Since the current ISR is clearly also at this priority level, the hardware is also blocked from delivering further interrupts to it as well. (This condition is clearly necessary so that the ISR can service the interrupt request automatically.)

When the ISR is finished, it exits with an RFE instruction. Hardware then restores the values of ICR.CCPN and ICR.IE to the values of the interrupted program.

14.8.3 Interrupt Priority Groups

It is sometimes useful to create groups of interrupts at the same or different interrupt priorities that cannot interrupt each other's ISRs. For instance, devices that can generate multiple interrupts may need to have interrupts at different priorities interlocked in this way. The TC1796 interrupt architecture can be used to create such interrupt priority groups. It is effected by managing the current CPU priority level ICR.CCPN in a way described in this section.

For example, in order to make an interrupt priority group out of priority numbers 11 and 12, one would not want an ISR executing at priority 11 to be interrupted by a service request at priority 12, since this would be in the same priority group. Only interrupts above 12 should be allowed to interrupt the ISRs in this interrupt priority group. However, under ordinary ISR usage, the ISR at priority 11 would be interrupted by any request with a higher priority number, including priority 12.

If, however, all ISRs in the interrupt priority group set the value of ICR.CCPN to the highest priority level within their group before they re-enable interrupts, then the desired interlocking will occur.

Figure 14-4 shows an example for interrupt priority grouping. The interrupt requests with the priority numbers 11 and 12 form one group, while the requests with priority numbers 14 through 17 form another group. Each ISR in group 1 sets the value of ICR.CCPN to 12, the highest number in that group, before re-enabling the interrupt system. Each ISR in group 2 sets the value of ICR.CCPN to 17 before re-enabling the interrupt system. If,

Interrupt System

for example, interrupt 14 is serviced, it can only be interrupted by requests with a priority number higher than 17; therefore it will not be interrupted by requests from its own priority group or requests with lower priority.

In **Figure 14-4**, the interrupt request with priority number 13 can be said to form an interrupt priority group with just itself as a member.

Setting ICR.CCPN to the maximum value 255 in each service routine has the same effect as not re-enabling the interrupt system; all interrupt requests can then be considered to be in the same group.

Interrupt priority groups demonstrate the power of the TC1796 priority-based interrupt-ordering system. Thus the flexibility of interrupt priority levels ranges from all interrupts in one group to each interrupt request building its own group, and to all possible combinations in between.

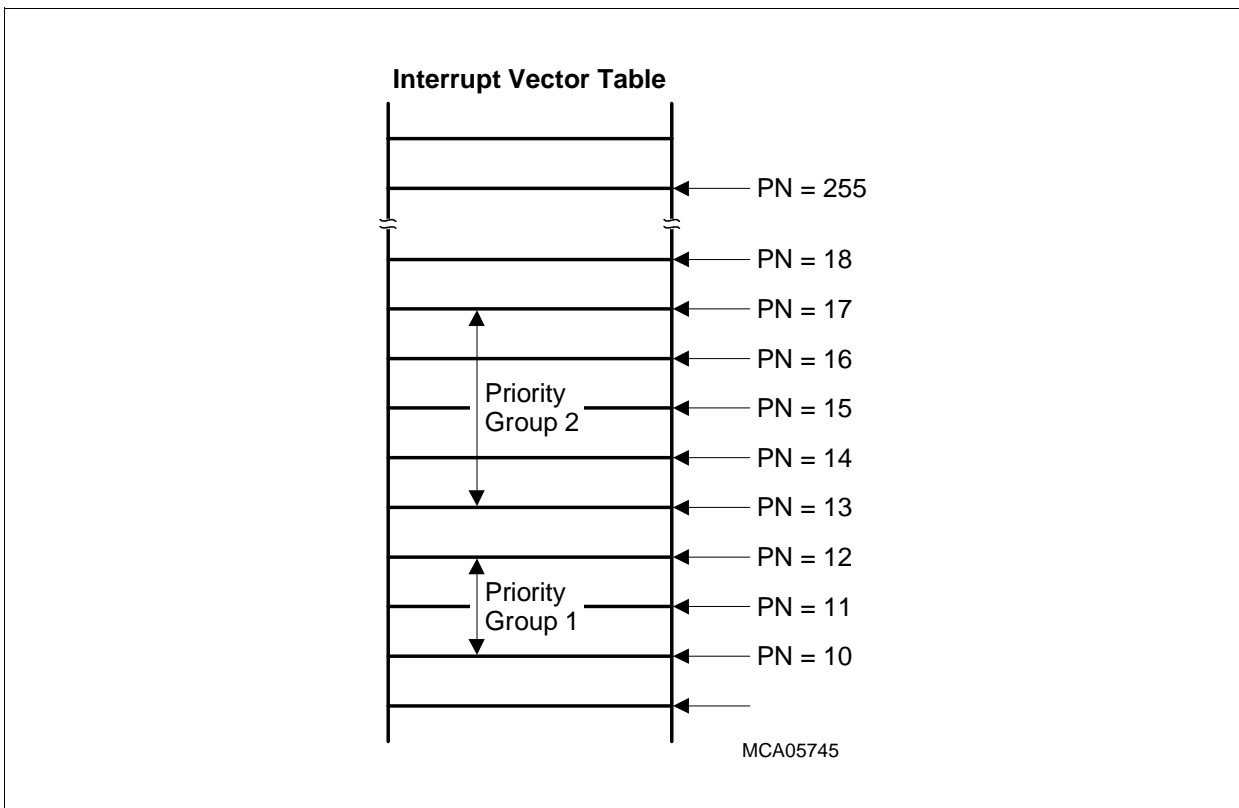


Figure 14-4 Interrupt Priority Groups

14.8.4 Splitting Interrupt Service Across Different Priority Levels

Interrupt service can be divided into multiple ISRs that execute at different priority levels. For example, the beginning stage of interrupt service may be very time-critical, such as reading a data value within a limited time window after the interrupt request activation. However, once the time-critical phase is past, there may still be more to do – for instance,

to process the observation. During this second phase, it may be acceptable for this ISR to be interrupted by lower-level interrupts. This can be performed as follows.

Say, for example, the initial interrupt priority is fixed very high because response time is critical. The necessary actions are carried out immediately by the ISR at that high-priority level. Then the ISR prepares to invoke another ISR at a lower priority level through software to perform the lower-priority actions.

To invoke an ISR through software, the high-priority ISR directly sets an interrupt request bit in an SRN that will invoke the appropriate low-priority ISR. Then the high-priority ISR exits.

When the high-priority ISR exits, the pending low-priority interrupt will eventually be serviced (depending on the priority of other pending interrupts). When the low-priority ISR eventually executes, the low-priority actions of the interrupt will be performed.

The inverse of this method can also be employed, wherein a low-priority ISR raises its own priority level, or leaves interrupts turned off while it executes. For instance, the priority of a service request might be low because the time to respond to the event is not critical, but once it has been granted service, this service should not be interrupted. In this case, the ISR could raise the value of ICR.CCPN to a priority that would exclude some or all other interrupts, or simply leave interrupts disabled.

14.8.5 Using different Priorities for the same Interrupt Source

For some applications, the urgency of a service request may vary, depending on the current state of the system. To handle this, different priority numbers (SRPNs) can be assigned at different times to a service request depending on the application needs.

Of course, Interrupt Service Routines must be placed in the Interrupt Vector Table at all addresses corresponding to the range of priorities used. If service remains the same at different priorities, copies of the ISR can be placed at the possible different entries, or the entries can all vector to a common ISR. If the ISR should execute different code depending on its priority, one need merely put the appropriate ISR in the appropriate entry of the Interrupt Vector Table.

This flexibility is another advantage of the TC1796 interrupt architecture. In traditional interrupt systems where the interrupt vectors are ordered by interrupting source, the ISR would have to check the current priority of the interrupt request and perform a branch to the appropriate code section, causing a delay in the response to the request. In the TC1796, however, the extra check and branch in the ISR are not necessary, which reduces the interrupt latency.

Because this approach may necessitate an increase in the range of interrupt priorities, the system designer must trade off this advantage against any possible increase in the number of arbitration cycles.

14.8.6 Interrupt Priority 1

Interrupt Priority 1 is the first and lowest-priority entry in the Interrupt Vector Table. It is generally reserved for ISRs which perform task management. ISRs whose actions cause software-managed tasks to be created post a software interrupt request at priority level 1 to signal the event.

The ISR that triggers this event can then execute a normal return from interrupt. There is no need for it to check whether the ISR is returning to the background-task priority level (priority 0) or is returning to a lower-priority ISR that it interrupted. When there is a pending interrupt at a priority higher than the return context for the current interrupt, this interrupt will then be serviced. When a return to the background-task priority level (level 0) is performed, the software-posted interrupt at priority level 1 will be serviced automatically.

14.8.7 Software-Initiated Interrupts

Software can set the service request bit (SRR) in a SRN by writing to its Service Request Control Register. Thus, software can initiate interrupts that are handled by the same mechanism as hardware interrupts.

After the SRR bit is set in an active SRN, there is no way to distinguish between a software-initiated interrupt request and a hardware interrupt request. For this reason, software should only use SRNs and interrupt priority numbers that are not being used for hardware interrupts.

The TC1796 contains four SRNs that support software-initiated interrupts. These SRNs are not connected to peripheral modules and can only cause interrupts when software sets its SRR bit. These SRNs are called the CPU Service Request Nodes (CPU_SRC[3:0]). The PCP can also cause these four SRNs to generate service requests. See also [Page 2-14](#) for TC1796-specific implementation details of the four CPU Service Request Control Registers.

Additionally, any otherwise unused SRN can be employed to generate software interrupts.

14.8.8 External Interrupts

Two SRNs, DMA_SYSSRC2 and DMA_SYSSRC3, are reserved to handle external interrupts. The setup for external GPIO port input signals (edge/level triggering, gating etc.) that are able to generate an interrupt request is controlled in the External Request Unit (ERU). The ERU functionality is described in detail in [Section 5.3 on Page 5-9](#).

14.9 Service Request Node Table

Table 14-2 shows all TC1796 Service Request Nodes.

Table 14-2 Service Request Nodes in the TC1796

Module	No. of Nodes	Description	SRC Register
CPU	4	CPU Service Request Nodes [3:0]	CPU_SRC[3:0] ¹⁾
	1	Software Breakpoint Request Node	CPU_SBSRC ¹⁾
Cerberus	1	Cerberus/OCDS Request Node	CBS_SRC
PBCU	1	PBCU Request Node	PBCU_SRC ¹⁾
DBCUC	1	DBCUC Request Node	DBCUC_SRC ¹⁾
SBCUC	1	SBCUC Request Node	SBCUC_SRC
RBCUC	1	RBCUC Request Node	RBCUC_SRC ¹⁾
DMA	8	DMA Service Request Nodes [7:0]	DMA_SRC[7:0]
	1	FPU Service Request Node	DMA_SYSSRC0
	1	FLASH Service Request Node	DMA_SYSSRC1
	2	External Interrupt Nodes [1:0]	DMA_SYSSRC2 DMA_SYSSRC3
	1	Software Interrupt Request Node	DMA_SYSSRC4
	4	MLI0 Service Request Nodes [3:0]	DMA_MLI0SRC [3:0]
	2	MLI1 Service Request Nodes [1:0]	DMA_MLI1SRC [1:0]
PCP	12	PCP Service Request Nodes [11:0]	PCP_SRC[11:0] ¹⁾
STM	2	STM Service Request Nodes [1:0]	STM_SRC[1:0]
ASC0	4	ASC0 Transmit Interrupt Service Request Node	ASC0_TSRC
		ASC0 Receive Interrupt Service Request Node	ASC0_RSRC
		ASC0 Error Interrupt Service Request Node	ASC0_ESRC
		ASC0 Transmit Buffer Interrupt Service Request Node	ASC0_TBSRC

Interrupt System

Table 14-2 Service Request Nodes in the TC1796 (cont'd)

Module	No. of Nodes	Description	SRC Register
ASC1	4	ASC1 Transmit Interrupt Service Request Node	ASC1_TSRC
		ASC1 Receive Interrupt Service Request Node	ASC1_RSRC
		ASC1 Error Interrupt Service Request Node	ASC1_ESRC
		ASC1 Transmit Buffer Interrupt Service Request Node	ASC1_TBSRC
SSC0	3	SSC0 Transmit Interrupt Service Request Node	SSC0_TSRC ¹⁾
		SSC0 Receive Interrupt Service Request Node	SSC0_RSRC ¹⁾
		SSC0 Error Interrupt Service Request Node	SSC0_ESRC ¹⁾
SSC1	3	SSC1 Transmit Interrupt Service Request Node	SSC1_TSRC ¹⁾
		SSC1 Receive Interrupt Service Request Node	SSC1_RSRC ¹⁾
		SSC1 Error Interrupt Service Request Node	SSC1_ESRC ¹⁾
MSC0	2	MSC0 Service Request Nodes [1:0]	MSC0_SRC[1:0]
MSC1	2	MSC1 Service Request Nodes [1:0]	MSC1_SRC[1:0]
CAN	16	CAN Service Request Nodes [15:0]	CAN_SRC[15:0] ¹⁾
GPTA	38	GPTA0 Service Request Nodes [37:00]	GPTA0_SRC [37:00]
	38	GPTA1 Service Request Nodes [37:00]	GPTA1_SRC [37:00]
	16	LTCA2 Service Request Nodes [15:00]	LTCA2_SRC [15:00]
ADC0	4	ADC0 Service Request Nodes [3:0]	ADC0_SRC[3:0] ¹⁾
ADC1	4	ADC1 Service Request Nodes [3:0]	ADC1_SRC[3:0] ¹⁾
FADC	4	FADC Service Request Nodes [3:0]	FADC_SRC[3:0] ¹⁾

181 = Total Number of Request Nodes

1) These service request registers are not bit-addressable because its register address is outside the first 16 Kbyte of a segment.

14.10 Non-Maskable Interrupt

Although called an interrupt, the Non-Maskable Interrupt (NMI) is actually serviced as a trap, since it is not interruptible and does not follow the standards for regular interrupts.

In the TC1796, four different events can generate an NMI trap:

- A transition on the $\overline{\text{NMI}}$ input pin
- An error or wake-up signal from the Watchdog Timer
- The PLL becomes unlocked
- The occurrence of an SRAM parity error in an on-chip memory block

The type of NMI trap is indicated in the NMI Status Register (NMISR).

14.10.1 External $\overline{\text{NMI}}$ Input

An external NMI event is generated when a one-to-zero transition is detected at the external $\overline{\text{NMI}}$ input pin. NMISR.NMIEXT is set in this case. The $\overline{\text{NMI}}$ pin is sampled at the system clock frequency. A transition is recognized when one sample shows a 1 and the next sample shows a 0. Subsequent 0-samples or a 0-to-1 transition do not trigger any action.

NMI is equipped with a noise suppression filter which suppresses glitches below 10 ns pulse width. NMI pulses with a width above 100 ns are safely recognized as a valid signal. The noise suppression filter is switched-off when pin BYPASS = 1.

14.10.2 Phase-Locked Loop NMI

The PLL clock generation unit sets the NMIPLL flag when it detects a loss in the synchronization with the external oscillator clock input. This condition means that the PLL clock frequency is no longer stable, and that the PLL will now decrease to its VCO clock base frequency.

14.10.3 Watchdog Timer NMI

The Watchdog Timer sets the NMIWDT flag for two conditions:

- A Watchdog Timer error has occurred
- Bit 15 of the Watchdog Timer is set while the CPU is in idle mode

A Watchdog Timer error can produce an NMI event because:

- Access to register WDT_CON0 was attempted improperly, or
- The Watchdog Timer overflowed either in Time-Out Mode or in Normal Watchdog Timer Mode.

When the CPU is in Idle Mode and the Watchdog Timer is not disabled, an increment of the Watchdog Timer counter from $7FFF_H$ to 8000_H (that is, when bit 15 of the timer is set to 1) sets the NMIWDT bit to wake up the CPU.

14.10.4 SRAM Parity Error NMI

If an SRAM parity error is detected in an SRAM memory block that is enabled for parity error detection (corresponding enable bit SCU_PETCR.PENx is set), the NMIPER flag in register NMISR is set together with related parity error flag SCU_PETSR.PFLx. In contrast to the other NMI flags, the NMIPER flag remains set when reading register NMISR. NMIPER is cleared only with an NMISR read operation when no error flag PFLx is set during the NMISR read operation. This parity error flag mechanism has some effects on a NMI trap handling routine. **Figure 14-5** shows a trap handler flow diagram that especially handles a typical SRAM parity error NMI trap recognition.

Additional details about SRAM parity error control are described in section **“SRAM Parity Control” on Page 5-37**.

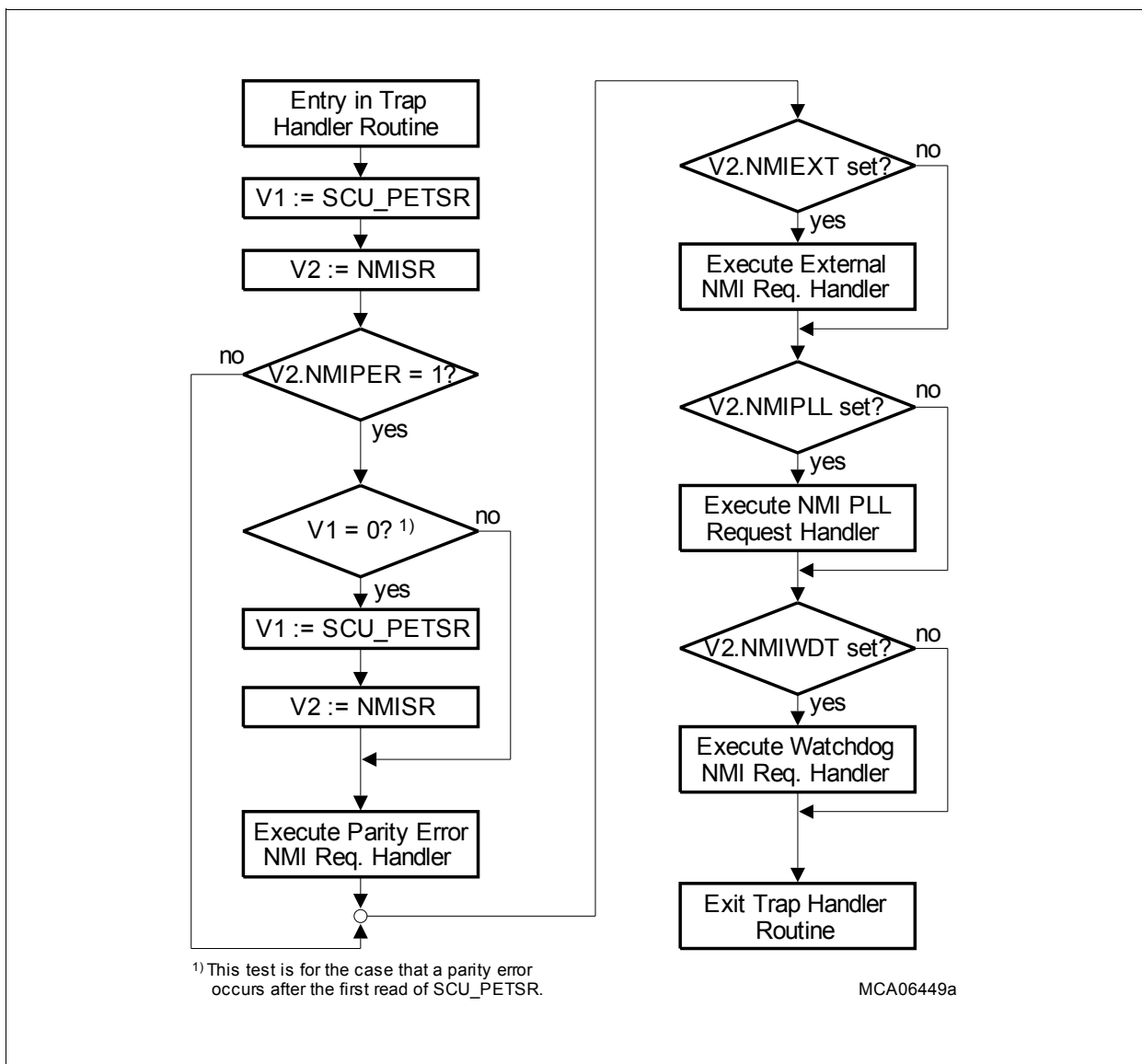


Figure 14-5 NMI Trap Handler Routine for Parity Error Handling

14.10.5 NMI Enable

After reset, the NMI is disabled. It must be enabled by the user program when the NMI handler routine has been setup. The NMI is enabled by setting bit SCU_CON.NMIEN. Once NMIEN has been set, it cannot be cleared again by software but only by a reset operation (except Watchdog reset).

14.10.6 NMI Status Register

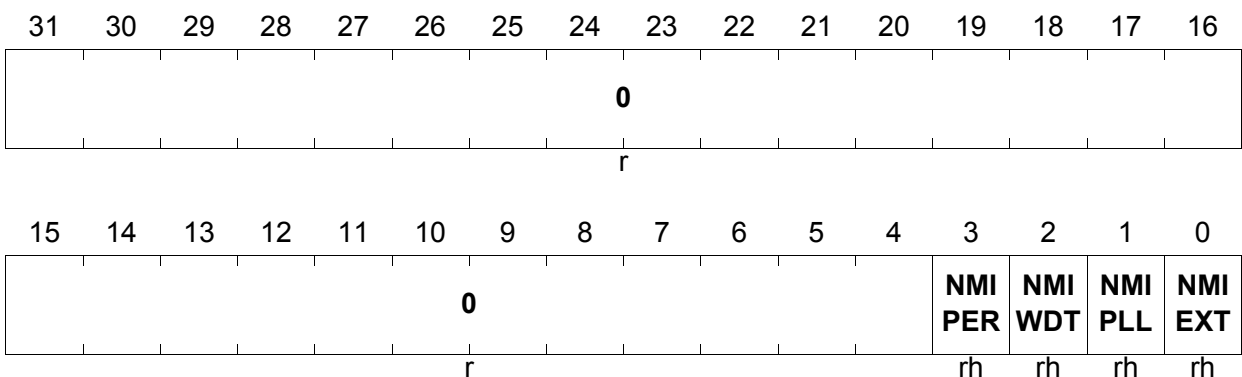
The source of an NMI trap can be identified through three status bits in NMISR. The bits in NMISR are read-only; writing to them has no effect.

The CPU detects a one-to-zero transition of the $\overline{\text{NMI}}$ input signal as indicating a NMI trap event. It then sets NMISR.NMIEXT. If the Watchdog Timer times out, it sets NMISR.NMIWDT. If the PLL loses its clock signal, it sets NMISR.PLL.

The bits in NMISR are OR-ed together to generate an NMI trap request to the CPU. If one of the NMISR bits is newly asserted while another bit is set, no new NMI trap request is generated. All flags are cleared automatically after a read of NMISR. Therefore, after reading NMISR, the NMI TSR must check all bits in NMISR to determine whether there have been multiple causes of an NMI trap.

NMISR

NMI Status Register (F000002C_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
NMIEXT	0	rh	External NMI Flag This flag indicates whether or not an external NMI request has occurred. 0 _B No external NMI request has occurred. 1 _B An external NMI request has been detected.

Interrupt System

Field	Bits	Type	Description
NMIPLL	1	rh	<p>PLL NMI Flag This flag indicates whether or not a PLL NMI request has occurred.</p> <p>0_B No PLL NMI has occurred. 1_B The PLL has lost the lock to the external crystal (becomes unlocked).</p>
NMIWDT	2	rh	<p>Watchdog Timer NMI Flag This flag indicates whether or not a Watchdog Timer NMI request has occurred.</p> <p>0_B No watchdog NMI occurred. 1_B The Watchdog Timer has entered the pre-warning phase due to a watchdog error.</p>
NMIPER	3	rh	<p>Parity Error NMI Flag This flag indicates whether or not SRAM parity error NMI request has occurred.</p> <p>0_B No SRAM parity error NMI occurred. 1_B An SRAM parity error NMI has been detected. The SRAM module where the parity error occurred can be checked by reading SCU_PETSR. NMIPER is only cleared with a NMISR read operation when the bits in SCU_PETSR are no more set.</p>
0	[31:4]	r	<p>Reserved Read as 0.</p>

Note: The NMISR register is located in the address range of the System Control Unit (see [Page 5-61](#)).

15 System Timer

15.1 Overview

This chapter describes the System Timer (STM). The TC1796's STM is designed for global system timing applications requiring both high precision about long periods. The STM has the following features:

- Free-running 56-bit counter
- All 56 bits can be read synchronously
- Different 32-bit portions of the 56-bit counter can be read synchronously
- Flexible interrupt generation based on compare match with partial STM content
- Driven by max. 75 MHz clock ($= f_{SYS}$, default after reset $= f_{SYS}/2$)
- Counting starts automatically after a reset operation
- STM is reset by:
 - Watchdog reset
 - Software reset (RST_REQ.RRSTM must be set)
 - Power-on reset
- STM (and clock divider STM_CLC.RMC) is not reset at a hardware reset ($\overline{HDRST} = 0$)
- STM can be halted in debug/suspend mode (via STM_CLC register)

Special STM register semantics provide synchronous views of the entire 56-bit counter, or 32-bit subsets at different levels of resolution.

The maximum clock period is $2^{56} \times f_{STM}$. At $f_{STM} = 75$ MHz, for example, the STM counts 30.47 years before overflowing. Thus, it is capable of continuously timing the entire expected product life-time of a system without overflowing.

15.2 Operation

The STM is an upward counter, running either at the system clock frequency f_{SYS} or at a fraction of it. The STM clock frequency is $f_{STM} = f_{SYS}/RMC$ with $RMC = 0-7$ (default after reset is $f_{STM} = f_{SYS}/2$, selected by $RMC = 010_B$). RMC is a bit field in register STM_CLC. In case of a power-on reset, a watchdog reset, or a software reset, the STM is reset. After one of these reset conditions, the STM is enabled and immediately starts counting up. It is not possible to affect the content of the STM during normal operation of the TC1796. The timer registers can only be read but not written to.

The STM can be optionally disabled for power-saving purposes, or suspended for debugging purposes via its clock control register. In suspend mode of the TC1796 (initiated by writing an appropriate value to STM_CLC register), the STM clock is stopped but all registers are still readable.

Due to the 56-bit width of the STM, it is not possible to read its entire content with one instruction. It needs to be read with two load instructions. Since the timer would continue

System Timer

to count between the two load operations, there is a chance that the two values read are not be consistent (due to possible overflow from the low part of the timer to the high part between the two read operations). To enable a synchronous and consistent reading of the STM content, a capture register (STM_CAP) is implemented. It latches the content of the high part of the STM each time when one of the registers STM_TIM0 to STM_TIM5 is read. Thus, STM_CAP holds the upper value of the timer at exactly the same time when the lower part is read. The second read operation would then read the content of the STM_CAP to get the complete timer value.

The STM can also be read in sections from seven registers, STM_TIM0 through STM_TIM6, that select increasingly higher-order 32-bit ranges of the STM. These can be viewed as individual 32-bit timers, each with a different resolution and timing range.

The content of the 56-bit System Timer can be compared against the content of two compare values stored in the STM_CMP0 and STM_CMP1 registers. Interrupts can be generated on a compare match of the STM with the STM_CMP0 or STM_CMP1 registers.

Figure 15-1 is an overview of the STM module. It shows the options for reading parts of STM content.

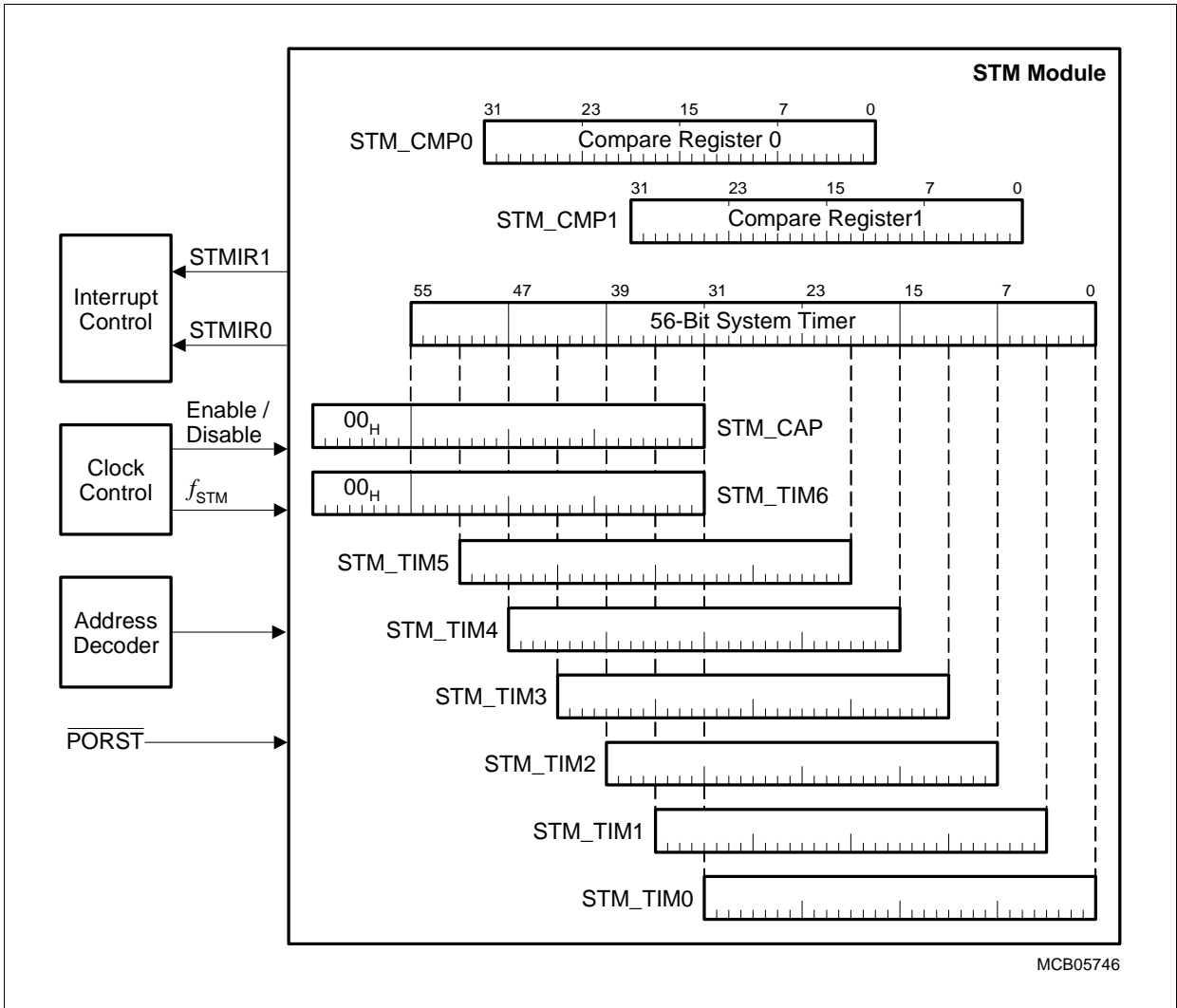


Figure 15-1 General Block Diagram of the STM Module Registers

15.2.1 Resolution and Ranges

Table 15-1 is an overview on the individual timer registers with their resolutions and timing ranges. As an example, the values for a 75 and 37.5 MHz STM input clock frequency are given.

Table 15-1 System Timer Resolutions and Ranges

Register	STM Bits	Resolution [s]	Range [s]	Resolution	Range	f_{STM} [MHz]
STM_TIM0	[31:0]	f_{STM}	$2^{32} / f_{STM}$	13.3 ns	57.3 s	75
STM_TIM1	[35:4]	$16 / f_{STM}$	$2^{36} / f_{STM}$	213 ns	916.2 s	
STM_TIM2	[39:8]	$256 / f_{STM}$	$2^{40} / f_{STM}$	3.41 μ s	244.3 min	
STM_TIM3	[43:12]	$4096 / f_{STM}$	$2^{44} / f_{STM}$	54.6 μ s	65.1 h	
STM_TIM4	[47:16]	$65536 / f_{STM}$	$2^{48} / f_{STM}$	0.874 ms	43.44 days	
STM_TIM5	[51:20]	$2^{20} / f_{STM}$	$2^{52} / f_{STM}$	13.98 ms	1.90 yr	
STM_TIM6	[55:32]	$2^{32} / f_{STM}$	$2^{56} / f_{STM}$	57.3 s	30.47 yr	
STM_CAP	[55:32]	$2^{32} / f_{STM}$	$2^{56} / f_{STM}$	57.3 s	30.47 yr	
STM_TIM0	[31:0]	f_{STM}	$2^{32} / f_{STM}$	26.7 ns	114.5 s	
STM_TIM1	[35:4]	$16 / f_{STM}$	$2^{36} / f_{STM}$	426 ns	1832.5 s	
STM_TIM2	[39:8]	$256 / f_{STM}$	$2^{40} / f_{STM}$	6.8 μ s	488.7 min	
STM_TIM3	[43:12]	$4096 / f_{STM}$	$2^{44} / f_{STM}$	109 μ s	130.3 h	
STM_TIM4	[47:16]	$65536 / f_{STM}$	$2^{48} / f_{STM}$	1.75 ms	86.87 days	
STM_TIM5	[51:20]	$2^{20} / f_{STM}$	$2^{52} / f_{STM}$	28.0 ms	3.81 yr	
STM_TIM6	[55:32]	$2^{32} / f_{STM}$	$2^{56} / f_{STM}$	114.5 s	60.93 yr	
STM_CAP	[55:32]	$2^{32} / f_{STM}$	$2^{56} / f_{STM}$	114.5 s	60.93 yr	

Note: The maximum input clock f_{STM} is 75 MHz.

15.2.2 Compare Register Operation

The content of the 56-bit STM can be compared to the content of two compare values stored in the STM_CMP0 and STM_CMP1 registers. Interrupts can be generated on a compare match of the STM with the STM_CMP0 or STM_CMP1 registers.

Two parameters are programmable for the compare operation:

1. The width of the relevant bits in registers STM_CMP0/STM_CMP1 (compare width MSIZE_x) that is taken for the compare operation can be programmed from 1 to 32.
2. The first bit location in the 56-bit STM that is taken for the compare operation can be programmed from 0 to 24.

These programming capabilities make compare functions very flexible. It even makes it possible to detect bit transitions of a single bit *n* (*n* = 0 to 24) within the 56-bit STM by setting MSIZE = 0 and MSTART = *n*.

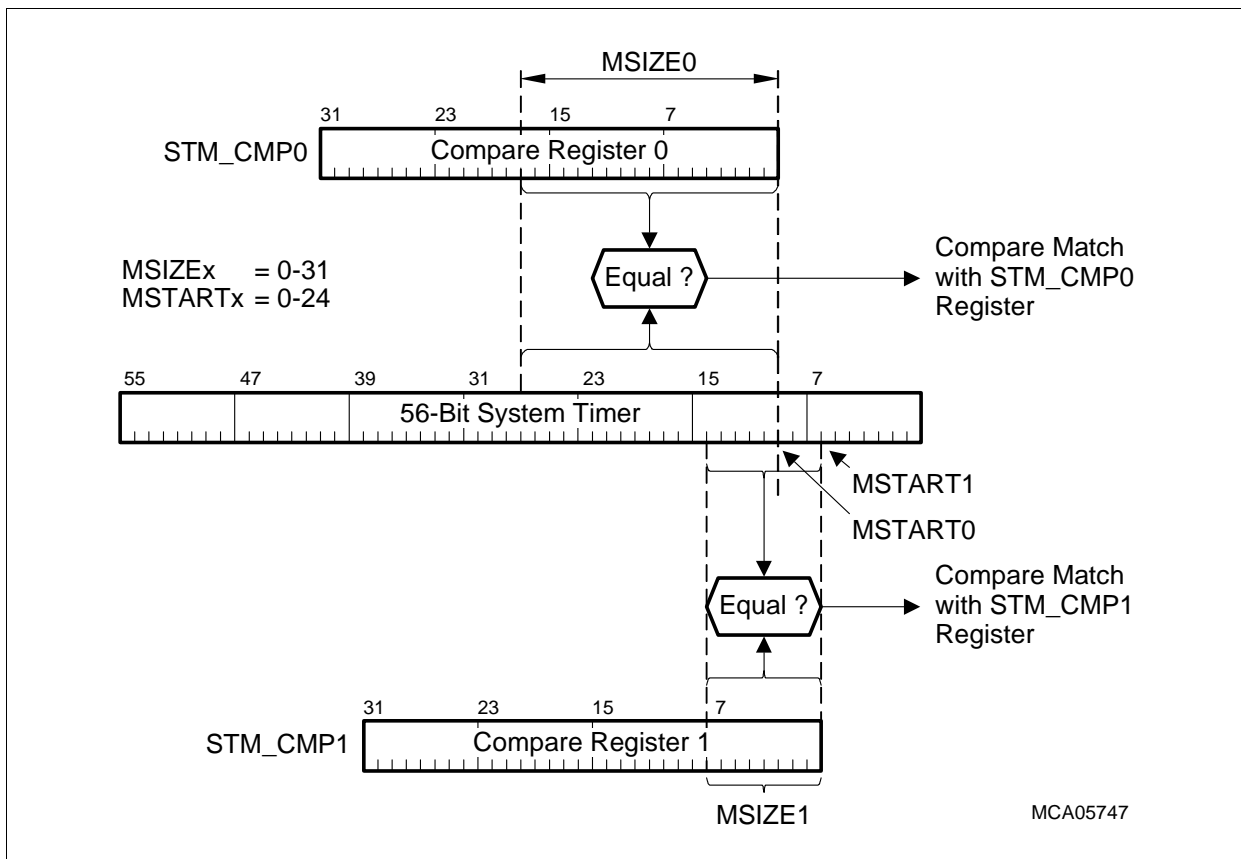


Figure 15-2 Compare Mode Operation

Figure 15-2 shows an example of the compare operation. In this example the following parameters are programmed:

- MSIZE0 = 10001_B = 17_D; MSTART0 = 01010_B = 10_D
- MSIZE1 = 00111_B = 7_D; MSTART1 = 00111_B = 7_D

A compare operation with MSIZE not equal 0 always implies that the compared value stored in the CMP register is right-extended with zeros. This means that in the example of **Figure 15-2**, the compare register content STM_CMP0[17:0] plus ten zero bits right-extended is compared with STM[27:0] with STM[9:0] = 000_H. In case of register STM_CMP1, STM[14:0] with STM[6:0] = 00_H are compared to STM_CMP1[7:0] plus seven zero bits right-extended.

15.2.3 Compare Match Interrupt Control

The compare match interrupt control logic is shown in **Figure 15-3**. Each STM_CMPx register has its compare match interrupt request flag (STM_ICR.CMPxIR) that is set by hardware on a compare match event. The interrupt request flags can be set (STM_ISSR.CMPxIRS) or cleared (STM_ISSR.CMPxIRR) by software. Note that setting STM_ICR.CMPxIR by writing a 1 into STM_ISSR.CMPxIRS does not generate an interrupt at STMIRx. The compare match interrupts from CMP0 and CMP1 can be further directed by STM_ICR.CMPxOS to either output signal STMIR0 or STMIR1. The STMIR0 and STMIR1 outputs are each connected to an interrupt service request control registers, STM_SRC0 and STM_SCR1. These registers control the general interrupt handling and processing as described in **Chapter 14** of this TC1796 System Units (Vol. 1 of 2) User's Manual.

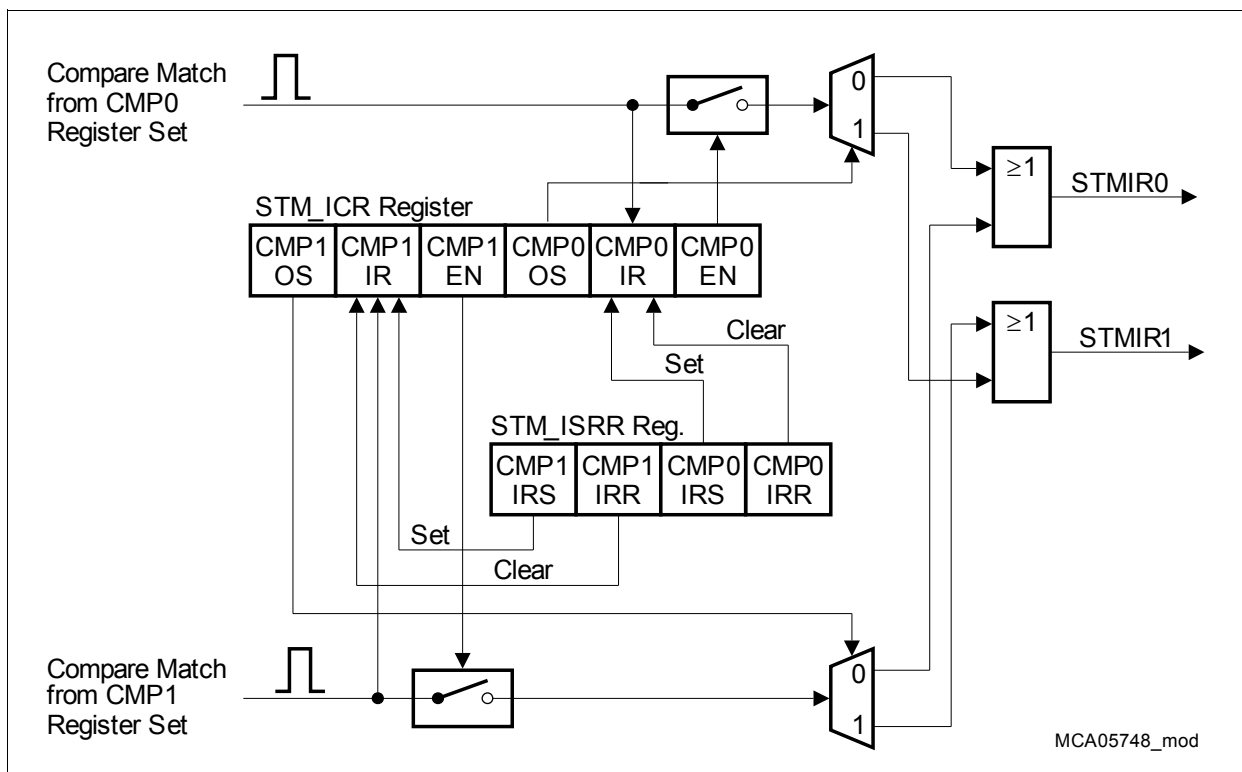


Figure 15-3 STM Interrupt Control

The compare match interrupt flags STM_ICR.CMPxIR are immediately set after an STM reset operation, caused by a compare match event with the reset values of the STM and the compare registers STM_CMPx. This setting of the CMPxIR flags does not directly generate compare match interrupts because the compare match interrupts are automatically disabled after an STM reset (CMPxEN = 0). Therefore, before enabling a compare match interrupt after an STM reset, the CMPxIR flags should be cleared by software (writing register STM_ISSR with CMPxIRR set). Otherwise, undesired compare match interrupt events are triggered accidentally.

15.3 Kernel Registers

This section describes the kernel registers of the STM. The STM registers can be divided into four types, as shown in [Figure 15-4](#).

Note: In the TC1796, all kernel registers are readable in suspend mode.

STM Registers Overview

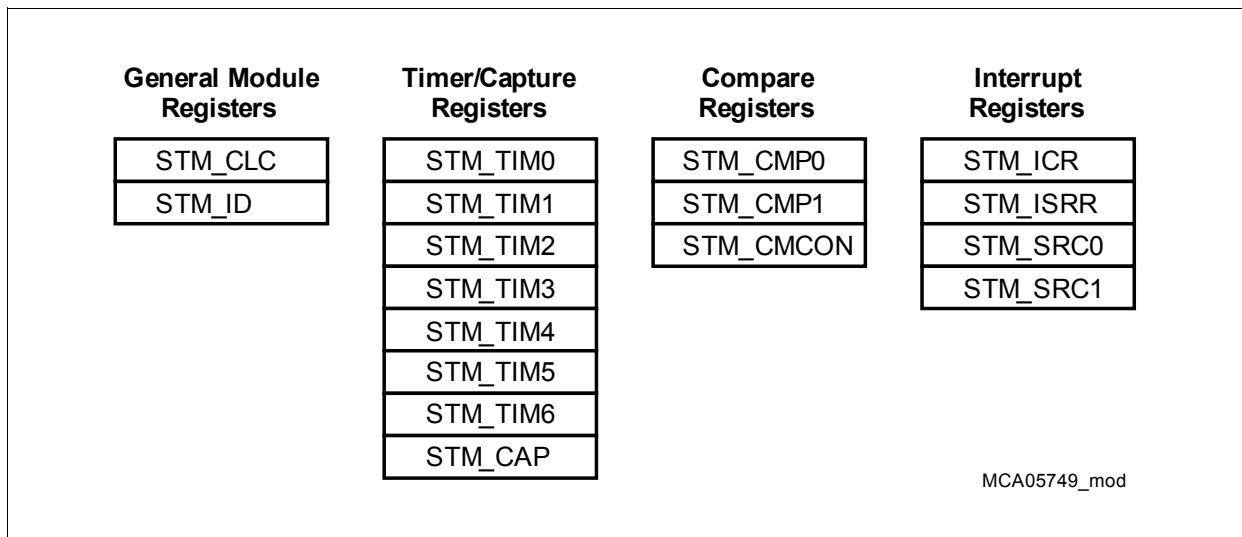


Figure 15-4 STM Registers

The complete and detailed address map of the STM module with its registers is shown in [Table 18-5](#) on [Page 18-13](#).

Table 15-2 Registers Address Space

Module	Base Address	End Address	Note
STM	F000 0200 _H	F000 02FF _H	-

Table 15-3 Registers Overview - STM Registers

Register Short Name	Register Long Name	Offset Address	Description see
STM_CLC	STM Clock Control Register	00 _H	Page 15-9
STM_ID	STM Module Identification Register	08 _H	Page 15-10
STM_TIM0	STM Timer Register 0	10 _H	Page 15-11
STM_TIM1	STM Timer Register 1	14 _H	Page 15-11
STM_TIM2	STM Timer Register 2	18 _H	Page 15-12
STM_TIM3	STM Timer Register 3	1C _H	Page 15-12
STM_TIM4	STM Timer Register 4	20 _H	Page 15-12
STM_TIM5	STM Timer Register 5	24 _H	Page 15-13
STM_TIM6	STM Timer Register 6	28 _H	Page 15-13
STM_CAP	STM Timer Capture Register	2C _H	Page 15-14
STM_CMP0	STM Compare Register 0 Low Part	30 _H	Page 15-15
STM_CMP1	STM Compare Register 0 High Part	34 _H	
STM_CMCON	STM Compare Match Control Register	38 _H	Page 15-16
STM_ICR	STM Interrupt Control Register	3C _H	Page 15-18
STM_ISRR	STM Interrupt Set/Reset Register	40 _H	Page 15-20
STM_SRC1	STM Interrupt Service Request Control Register 1	F8 _H	Page 15-21
STM_SRC0	STM Interrupt Service Request Control Register 0	FC _H	

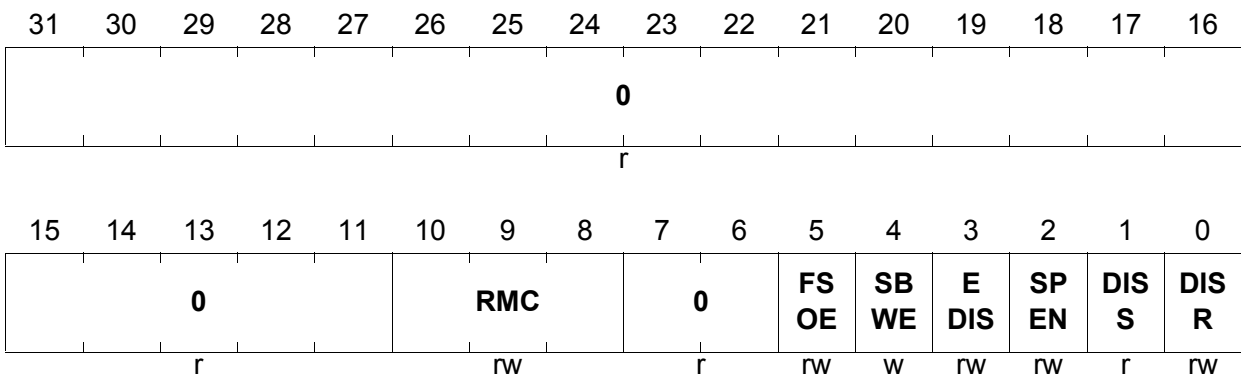
15.3.1 General Module Register

The STM Clock Control Register is used to switch the STM on or off and to control its input clock rate. After a power-on reset, the STM is always enabled and starts counting. The STM can be disabled by setting bit DISR to 1.

STM_CLC

System Timer Clock Control Register (00_H)

Reset Value: 0000 0200_H



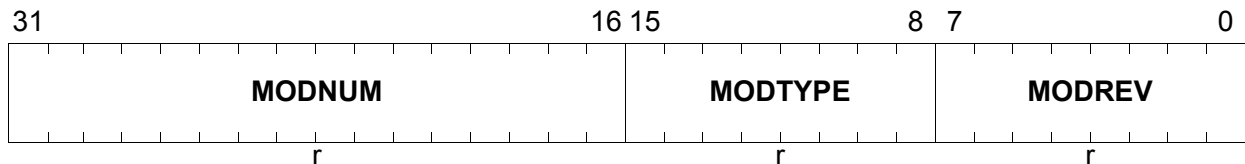
Field	Bits	Type	Description
DISR	0	rw	Module Disable Request Bit Used for enable/disable control of the STM module. 0 _B No disable requested 1 _B Disable requested
DISS	1	r	Module Disable Status Bit Bit indicates the current status of the STM module. 0 _B STM module is enabled 1 _B STM module is disabled
SPEN	2	rw	Module Suspend Enable for OCDS Used for enabling the suspend mode.
EDIS	3	rw	External Request Disable Used for controlling the external clock disable request.
SBWE	4	w	Module Suspend Bit Write Enable for OCDS Determines whether SPEN and FSOE are write-protected.
FSOE	5	rw	Fast Switch Off Enable Used for fast clock switch off in OCDS suspend mode.

Field	Bits	Type	Description
RMC	[10:8]	rw	Clock Divider in Run Mode 000 _B No clock signal f_{STM} generated 001 _B Clock signal $f_{STM} = f_{SYS}$ selected (default after reset) 010 _B Clock signal $f_{STM} / 2$ selected 011 _B Clock signal $f_{STM} / 3$ selected ... _B ... 111 _B Clock signal $f_{STM} / 7$ selected <i>Note: This bit field is not affected by a hardware reset operation ($HDRST = 0$).</i>
0	[7:6], [31:11]	r	Reserved Read as 0; should be written with 0.

The STM Module Identification Register ID contains read-only information about the STM module version.

STM_ID

STM Module Identification Register (08_H) **Reset Value: 0000 C0XX_H**



Field	Bits	Type	Description
MODREV	[7:0]	r	Module Revision Number MODREV defines the module revision number. The value of a module revision starts with 01 _H (first revision).
MODTYPE	[15:8]	r	Module Type This bit field defines the module as a 32-bit module: C0 _H
MODNUM	[31:16]	r	Module Number Value This bit field defines the module identification number for the STM: 0000 _H

15.3.2 Timer/Capture Registers

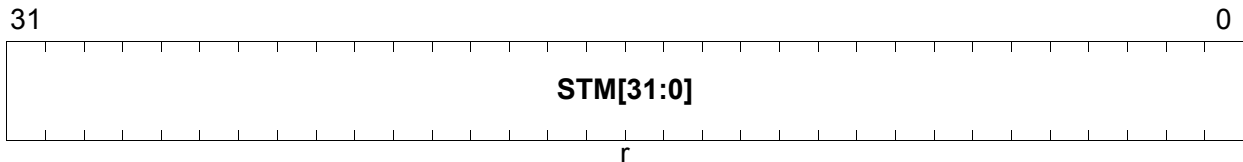
Registers STM_TIM1 to STM_TIM6 provide 32-bit views at varying resolutions of the underlying STM counter.

STM_TIM0

STM Timer Register 0

(10_H)

Reset Value: 0000 0000_H



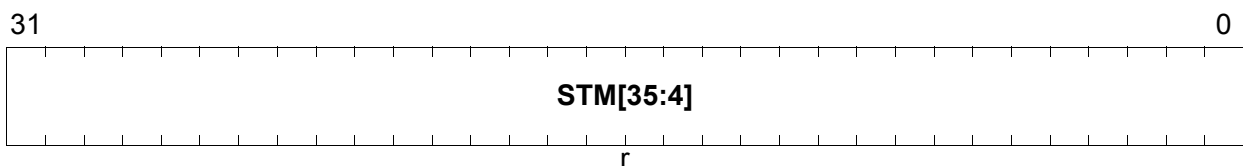
Field	Bits	Type	Description
STM[31:0]	[31:0]	r	System Timer Bits [31:0] This bit field contains bits [31:0] of the 56-bit STM.

STM_TIM1

STM Timer Register 1

(14_H)

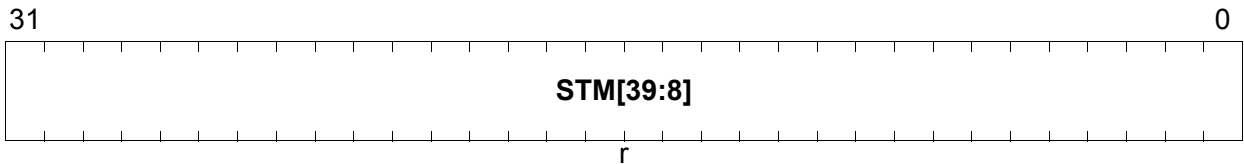
Reset Value: 0000 0000_H



Field	Bits	Type	Description
STM[35:4]	[31:0]	r	System Timer Bits [35:4] This bit field contains bits [35:4] of the 56-bit STM.

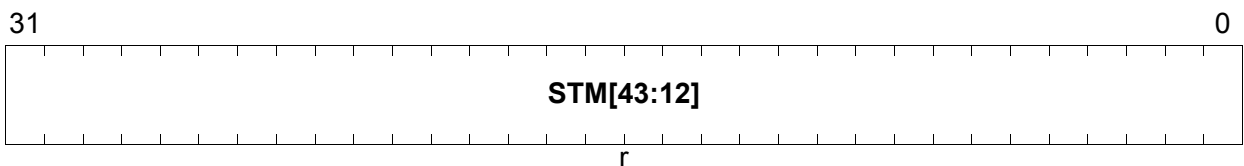
System Timer

STM_TIM2
STM Timer Register 2 (18_H) **Reset Value: 0000 0000_H**



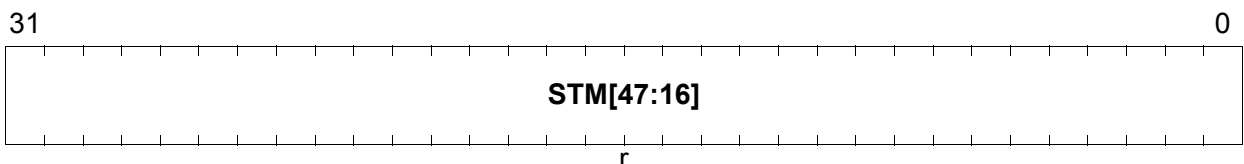
Field	Bits	Type	Description
STM[39:8]	[31:0]	r	System Timer Bits [39:8] This bit field contains bits [39:8] of the 56-bit STM.

STM_TIM3
STM Timer Register 3 (1C_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
STM[43:12]	[31:0]	r	System Timer Bits [43:12] This bit field contains bits [43:12] of the 56-bit STM.

STM_TIM4
STM Timer Register 4 (20_H) **Reset Value: 0000 0000_H**



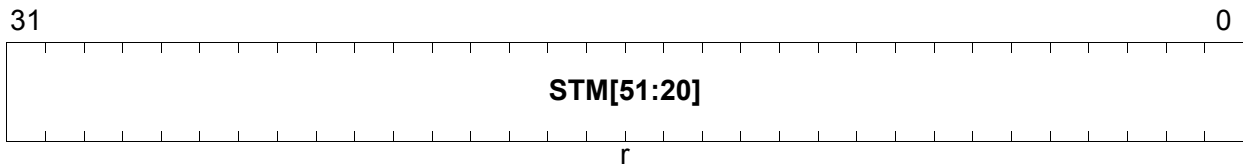
Field	Bits	Type	Description
STM[47:16]	[31:0]	r	System Timer Bits [47:16] This bit field contains bits [47:16] of the 56-bit STM.

STM_TIM5

STM Timer Register 5

(24_H)

Reset Value: 0000 0000_H



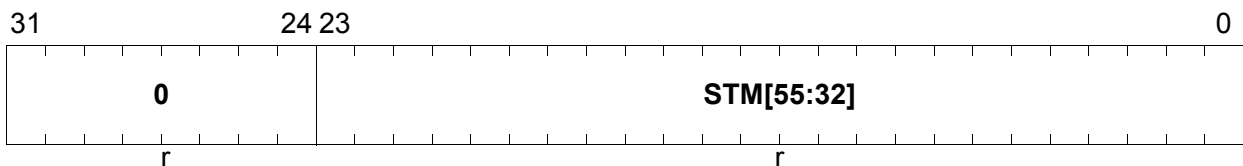
Field	Bits	Type	Description
STM[51:20]	[31:0]	r	System Timer Bits [51:20] This bit field contains bits [51:20] of the 56-bit STM.

STM_TIM6

STM Timer Register 6

(28_H)

Reset Value: 0000 0000_H



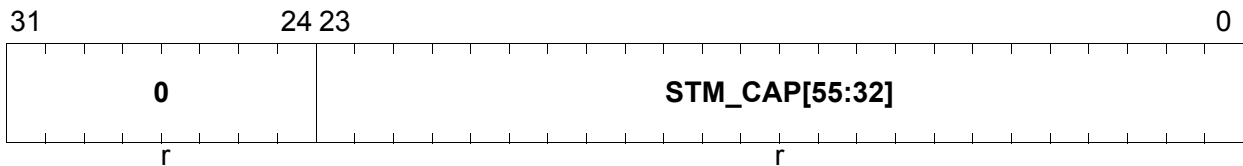
Field	Bits	Type	Description
STM[55:32]	[23:0]	r	System Timer Bits [55:32] This bit field contains bits [55:32] of the 56-bit STM.
0	[31:24]	r	Reserved Read as 0.

STM_CAP

STM Timer Capture Register

(2C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
STM[55:32]	[23:0]	r	Captured System Timer Bits [55:32] The capture register STM_CAP always captures the STM bits [55:32] when one of the registers STM_TIM0 to STM_TIM5 is read. This capture operation is performed in order to enable software to operate with a coherent value of all the 56 STM bits at one time stamp. This bit field contains bits [55:32] of the 56-bit STM.
0	[31:24]	r	Reserved Read as 0.

Note: The bits in registers STM_CAP to STM_TIM0 are all read-only bits.

15.3.3 Compare Registers

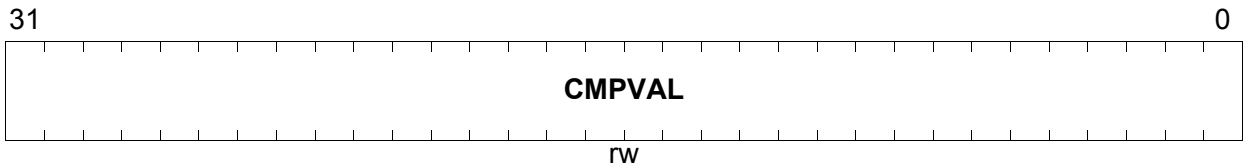
The STM Compare Register holds up to 32-bits; its value is compared to the value of the STM.

STM_CMPx (x = 0-1)

STM Compare Register x

(30_H+x*4_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CMPVAL	[31:0]	rw	Compare Value of Compare Register x This bit field holds up to 32 bits of the compare value (right-adjusted).

System Timer

The STM Compare Match Control Register controls the parameters of the compare logic.

STM_CMCON

STM Compare Match Control Register

(38_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
MSIZE0	[4:0]	rw	<p>Compare Register Size for CMP0</p> <p>This bit field determines the number of bits in register CMP0 (starting from bit 0) that are used for the compare operation with the STM.</p> <p>0000_B CMP0[0] used for compare operation 00001_B CMP0[1:0] used for compare operation ..._B ... 11110_B CMP0[30:0] used for compare operation 11111_B CMP0[31:0] used for compare operation</p>
MSTART0	[12:8]	rw	<p>Start Bit Location for CMP0</p> <p>This bit field determines the lowest bit number of the 56-bit STM that is compared with the content of register CMP0 bit 0. The number of bits to be compared is defined by bit field MSIZE0.</p> <p>00000_B STM[0] is the lowest bit number 00001_B STM[1] is the lowest bit number ..._B ... 10111_B STM[23] is the lowest bit number 11000_B STM[24] is the lowest bit number Bit combinations 11001_B to 11111_B are reserved and must not be used.</p>

Field	Bits	Type	Description
MSIZE1	[20:16]	rw	<p>Compare Register Size for CMP1 This bit field determines the number of bits in register CMP1 (starting from bit 0) that are used for the compare operation with the STM.</p> <p>00000_B CMP1[0] used for compare operation 00001_B CMP1[1:0] used for compare operation ..._B ... 11110_B CMP1[30:0] used for compare operation 11111_B CMP1[31:0] used for compare operation</p>
MSTART1	[28:24]	rw	<p>Start Bit Location for CMP1 This bit field determines the lowest bit number of the 56-bit STM that is compared with the content of register CMP1 bit 0. The number of bits to be compared is defined by bit field MSIZE1.</p> <p>00000_B STM[0] is the lowest bit number 00001_B STM[1] is the lowest bit number ..._B ... 10111_B STM[23] is the lowest bit number 11000_B STM[24] is the lowest bit number Bit combinations 11001_B to 11111_B are reserved and must not be used.</p>
0	[7:5], [15:13], [23:21], [31:29]	r	<p>Reserved Read as 0; should be written with 0.</p>

15.3.4 Interrupt Registers

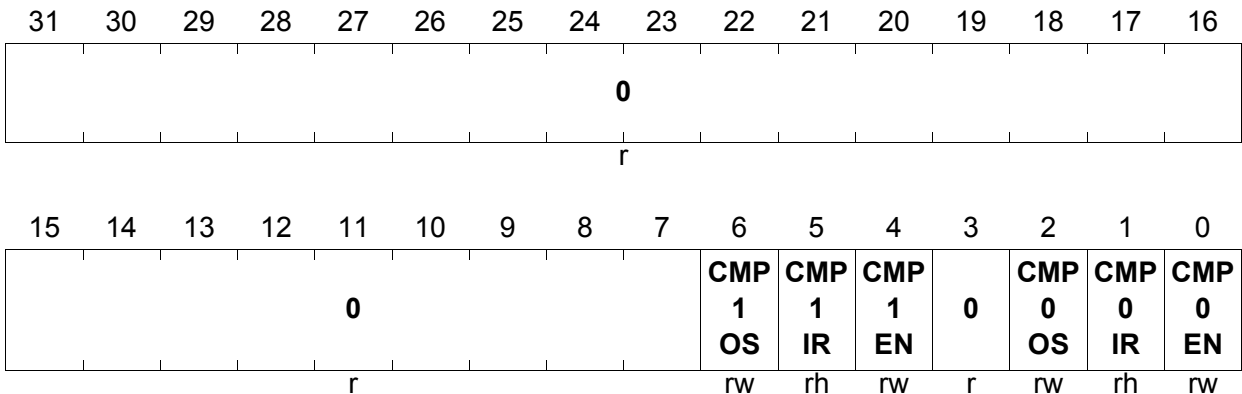
The two compare match interrupts of the STM are controlled by the STM Interrupt Control Register.

STM_ICR

STM Interrupt Control Register

(3C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CMP0EN	0	rw	<p>Compare Register CMP0 Interrupt Enable Control This bit enables the compare match interrupt with compare register CMP0.</p> <p>0_B Interrupt on compare match with CMP0 disabled 1_B Interrupt on compare match with CMP0 enabled</p>
CMP0IR	1	rh	<p>Compare Register CMP0 Interrupt Request Flag This bit indicates whether or not a compare match interrupt request of compare register CMP0 is pending. CMP0IR must be cleared by software.</p> <p>0_B A compare match interrupt has not been detected since the bit has been cleared for the last time. 1_B A compare match interrupt has been detected. CMP0IR must be cleared by software and can be set by software, too (see CMPISRR register). After an STM reset, CMP0IR is immediately set as the result of a compare match event with the reset values of the STM and the compare register CMP0.</p>

Field	Bits	Type	Description
CMP0OS	2	rw	<p>Compare Register CMP0 Interrupt Output Selection This bit determines the interrupt output that is activated on a compare match event of compare register CMP0.</p> <p>0_B Interrupt output STMIR0 selected 1_B Interrupt output STMIR1 selected</p>
CMP1EN	4	rw	<p>Compare Register CMP1 Interrupt Enable Control This bit enables the compare match interrupt with compare register CMP1.</p> <p>0_B Interrupt on compare match with CMP1 disabled 1_B Interrupt on compare match with CMP1 enabled</p>
CMP1IR	5	rh	<p>Compare Register CMP1 Interrupt Request Flag This bit indicates whether or not a compare match interrupt request of compare register CMP1 is pending. CMP1IR must be cleared by software.</p> <p>0_B A compare match interrupt has not been detected since the bit has been cleared for the last time. 1_B A compare match interrupt has been detected. CMP1IR must be cleared by software and can be set by software, too (see CMPISRR register). After an STM reset CMP1IR is immediately set as the result of a compare match event with the reset values of the STM and the compare register CMP1.</p>
CMP1OS	6	rw	<p>Compare Register CMP1 Interrupt Output Selection This bit determines the interrupt output that is activated on a compare match event of compare register CMP1.</p> <p>0_B Interrupt output STMIR0 selected 1_B Interrupt output STMIR1 selected</p>
0	3, [31:7]	r	<p>Reserved Read as 0; should be written with 0.</p>

System Timer

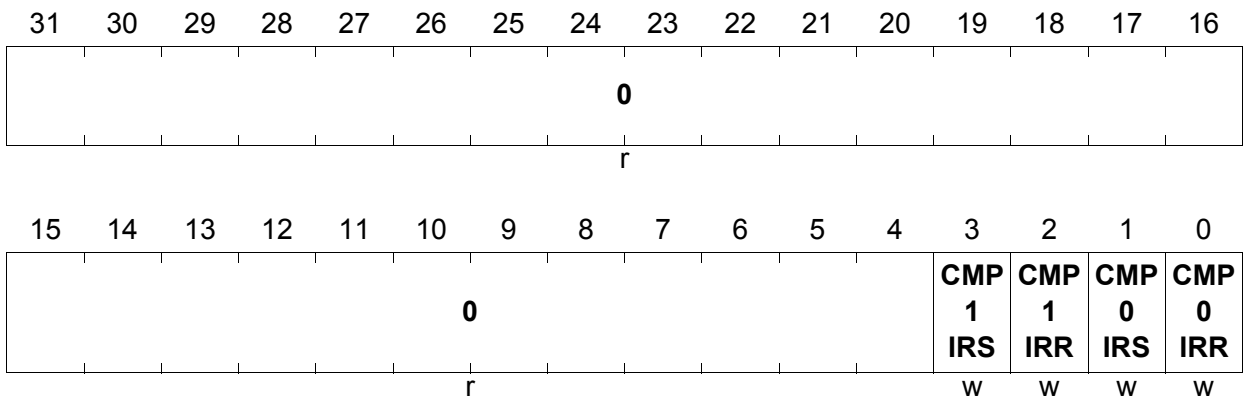
The bits in the STM Interrupt Set/Reset Register make it possible to set or clear the compare match interrupt request status flags of register ICR.

STM_ISRR

STM Interrupt Set/Reset Register

(40_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CMP0IRR	0	w	Reset Compare Register CMP0 Interrupt Flag 0 _B Bit ICR.CMP0IR is not changed 1 _B Bit ICR.CMP0IR is cleared
CMP0IRS	1	w	Set Compare Register CMP0 Interrupt Flag 0 _B Bit ICR.CMP0IR is not changed 1 _B Bit ICR.CMP0IR is set. The state of bit CMP0IRR is “don’t care” in this case
CMP1IRR	2	w	Reset Compare Register CMP1 Interrupt Flag 0 _B Bit ICR.CMP0IR is not changed 1 _B Bit ICR.CMP0IR is cleared
CMP1IRS	3	w	Set Compare Register CMP1 Interrupt Flag 0 _B Bit ICR.CMP1IR is not changed 1 _B Bit ICR.CMP1IR is set. The state of bit CMP1IRR is “don’t care” in this case
0	[31:4]	r	Reserved Read as 0; should be written with 0.

Note: Reading register CMISRR always returns 0000 0000_H.

System Timer

In the TC1796, the compare match interrupt output signals of the STM, STMIR0 and STMIR1, are controlled by the STM Service Request Control Registers STM_SRC0 and STM_SRC1.

STM_SRC0

STM Service Request Control Register 0

(FC_H)

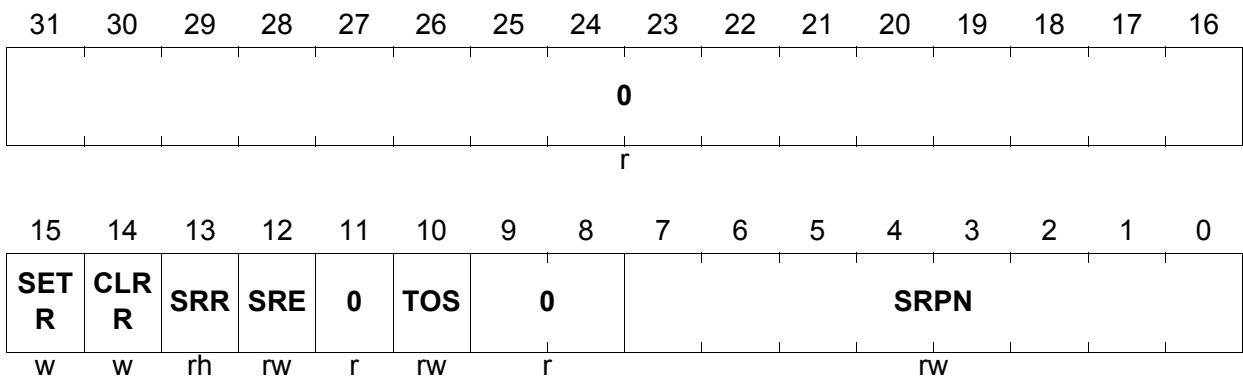
Reset Value: 0000 0000_H

STM_SRC1

STM Service Request Control Register 1

(F8_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SRPN	[7:0]	rw	Service Request Priority Number
TOS	10	rw	Type of Service Control
SRE	12	rw	Service Request Enable
SRR	13	rh	Service Request Flag
CLRR	14	w	Request Clear Bit
SETR	15	w	Request Set Bit
0	[9:8], 11, [31:16]	r	Reserved Read as 0; should be written with 0.

Note: Further details of interrupt handling and processing are described in [Chapter 14](#) of this TC1796 System Units (Vol. 1 of 2) User's Manual

16 Watchdog Timer

This chapter describes the TC1796 Watchdog Timer (WDT). Topics include an overview of the WDT function and descriptions of the registers, the password-protection scheme, accessing registers, modes, and initialization.

16.1 Watchdog Timer Overview

The WDT provides a highly reliable and secure way to detect and recover from software or hardware failure. The WDT helps to abort an accidental malfunction of the TC1796 in a user-specified time period. When enabled, the WDT will cause the TC1796 system to be reset if the WDT is not serviced within a user-programmable time period. The CPU must service the WDT within this time interval to prevent the WDT from causing a TC1796 system reset. Hence, routine service of the WDT confirms that the system is functioning properly.

In addition to this standard “Watchdog” function, the WDT incorporates the End-of-Initialization (Endinit) feature and monitors its modifications. A system-wide line is connected to the WDT_CON0.ENDINIT bit, serving as an additional write-protection for critical registers (besides Supervisor Mode protection). Registers protected via this line can only be modified when Supervisor Mode is active and bit ENDINIT = 0.

Because servicing the Watchdog and modifications of the ENDINIT bit are critical functions that must not be allowed in case of a system malfunction, a sophisticated scheme is implemented that requires a password and guard bits during accesses to the WDT control register. Any write access that does not deliver the correct password or the correct value for the guard bits is regarded as a malfunction of the system, and a Watchdog reset is triggered. In addition, even after a valid access has been performed and the ENDINIT bit has been cleared to provide access to the critical registers, the Watchdog imposes a time limit for this access window. If ENDINIT has not been properly set again before this limit expires, the system is assumed to have malfunctioned, and a Watchdog reset is triggered. These stringent requirements, although not a guarantee, nonetheless provide a high degree of assurance of the robustness of system operation.

A further enhancement in the TC1796’s WDT is its reset prewarning operation. Instead of immediately resetting the device on the detection of an error (the way that standard Watchdogs do), the WDT first issues a Non-Maskable Interrupt (NMI) to the CPU before finally resetting the device at a specified time period later. This gives the CPU a chance to save system state to memory for later examination of the cause of the malfunction, an important aid in debugging.

16.2 Features of the Watchdog Timer

The major features of the WDT are summarized here. The WDT is implemented in the System Control Unit (SCU) module of the TC1796. **Figure 16-1** gives an overview of its interface signals.

- 16-bit Watchdog counter
- Selectable input frequency: $f_{SYS}/256$ or $f_{SYS}/16384$
- 16-bit user-definable reload value for normal Watchdog operation, fixed reload value for Time-Out and Prewarning Modes
- Incorporation of the ENDINIT bit and monitoring of its modifications
- Sophisticated Password Access mechanism with fixed and user-definable password fields
- Proper access always requires two write accesses. The time between the two accesses is monitored by the WDT and limited.
- Access Error Detection: Invalid password (during first access) or invalid guard bits (during second access) trigger the Watchdog reset generation.
- Overflow Error Detection: An overflow of the counter triggers the Watchdog reset generation.
- Watchdog function can be disabled; access protection and ENDINIT monitor function remain enabled.
- Double Reset Detection: If a Watchdog induced reset occurs twice, a severe system malfunction is assumed and the TC1796 is held in reset until a power-on or hardware reset occurs. This prevents the device from being periodically reset if, for instance, connection to the external memory has been lost such that even system initialization could not be performed.
- Important debugging support is provided through the reset prewarning operation by first issuing an NMI to the CPU before finally resetting the device after a certain period of time.

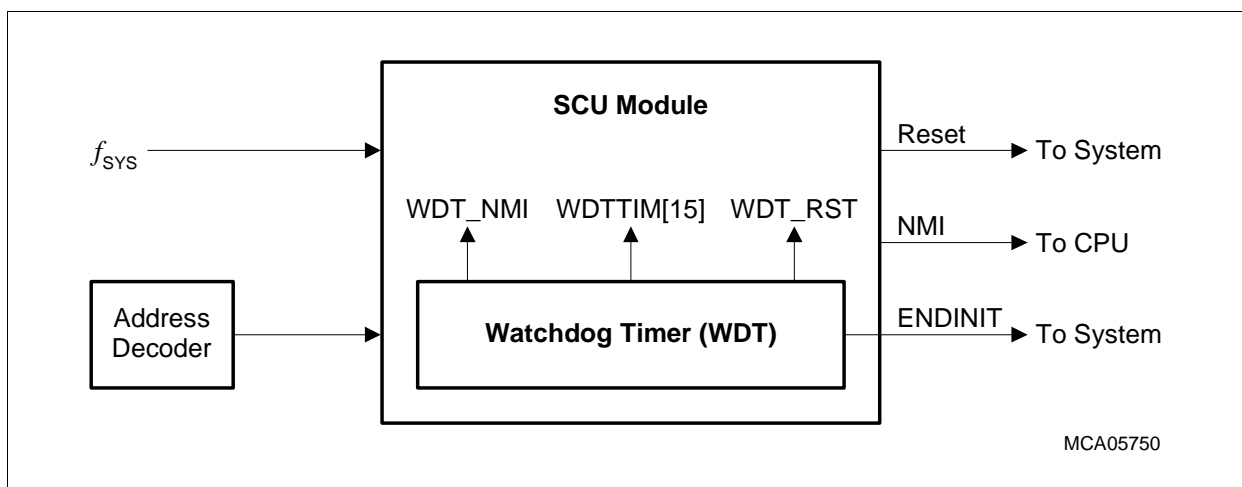


Figure 16-1 Interface of the WDT Inside and Outside the SCU Module

16.3 The Endinit Function

Because understanding of the ENDINIT bit and its function is an important prerequisite for the descriptions in the following sections, its function is explained first.

There are a number of registers in the TC1796 that are usually programmed only once during the initialization sequence of the application. Modification of such registers during normal application run can have a severe impact on the overall operation of modules or the entire system.

While the Supervisor Mode, which allows writes to registers only when it is active, provides a certain level of protection against unintentional modifications, this might not provide enough security for system-critical registers.

The TC1796 provides one more level of protection for such registers via the Endinit feature. This is a highly secure write-protection scheme that makes unintentional modifications of registers protected by this feature nearly impossible.

The Endinit feature consists of an ENDINIT bit incorporated in the WDT control register, WDT_CON0. A system-wide line is connected to this bit. Registers protected via Endinit use the state of this line to determine whether or not writes are enabled. Writes are only enabled if ENDINIT = 0 and Supervisor Mode is active. Write attempts if this condition is not true will be discarded and the register contents will not be modified in this case. In this case, an interrupt in the corresponding bus control units of the TC1796 is generated instead of a bus error trap. There is an exception: If read-modify-write instructions are used for the write access, a bus error trap is generated instead of an interrupt.

An additional line, controlled through a separate bit, makes it possible to protect against unintentional writes, providing an extra level of security. However, to get the highest level of security, this bit is incorporated in the highly secure access protection scheme implemented in the WDT. This is a complex procedure, that makes it nearly impossible for the ENDINIT bit to be modified unintentionally. It is explained in the following sections. In addition, the WDT monitors ENDINIT modifications by starting a time-out sequence each time software opens access to the critical registers through clearing ENDINIT to 0. If the time-out period ends before ENDINIT is set to 1 again, a malfunction of the software and/or the hardware is assumed and the device is reset.

The access-protection scheme and the Endinit time-out operation of the WDT is described in the following sections. [Table 16-1](#) lists the registers that are protected via the Endinit feature in the TC1796.

Note: The clearing of the ENDINIT bit takes some time. Accesses to Endinit-protected registers after the clearing of the ENDINIT bit must only be done after ensuring that ENDINIT has been cleared. As a solution, WDT_CON0 (the register with the ENDINIT bit) should be read back once before Endinit-protected registers are accessed the first time after ENDINIT has been cleared.

Table 16-1 TC1796 Registers Protected via the Endinit Feature

Registers	Description
mod_CLC	All clock control registers of the individual peripheral modules are Endinit-protected.
mod_FDR	All clock fractional divider registers of the individual peripheral modules are Endinit-protected.
BTV, BIV, ISP	Trap and interrupt vector table pointer as well as the interrupt stack pointer are Endinit-protected.
WDT_CON1	The WDT Control Register 1, which controls the disabling and the input frequency of the WDT, is Endinit-protected. In addition, its bits will only have an effect on the WDT when ENDINIT is properly set to 1 again.
RST_REQ OSC_CON PLL_CLC SCU_SCLKFDR SCU_EMSR SCU_TCCON SCU_CON SCU_TCLR0 SCU_TCLR1 SCU_PTCON Px_PDR Px_ESR PCP_CS DMA_OCDSR DMA_SUSPMR FLASH_FCON FLASH_MARP DMI_CON DMI_CON1 PMI_CON0 DMA_MEnAENR DMA_MEnARR MLIx_AER MLIx_ARR	All these registers are also Endinit-protected.

16.4 Watchdog Timer Operation

The following sections describe the registers, the operation, and different modes of the WDT, as well as the Password Access mechanism. **Figure 16-2** gives an example for the operation of the WDT. A rough description of the sequence of events in this figure is provided here. Refer to the following sections for a detailed explanation.

1. Time-Out Mode is automatically entered after reset. Timer counts with slowest input clock.
2. Time-Out Mode terminated and Normal Mode is entered by setting ENDINIT to 1.
3. Normal Mode is terminated and Time-Out Mode is entered through a Password Access to WDT_CON0. The reload value was set to REL_1.
4. Time-Out Mode is terminated and Normal Mode entered again by setting ENDINIT to 1. The reload value WDTREL has been changed to REL_2 and the timer input clock was set to the fast clock.

Events 3) and 4) constitute a WDT service sequence.

5. The WDT was not serviced and continued to count until overflow. Reset Prewarning Mode is entered. Timer counts with selected fast input clock. Watchdog operation cannot be altered or stopped in this mode.
6. Timer continued to count until overflow, generating a WDT reset.
7. Time-Out Mode is automatically entered after reset. Timer counts with slowest input clock.
8. Time-Out Mode is terminated and Normal Mode is entered again.

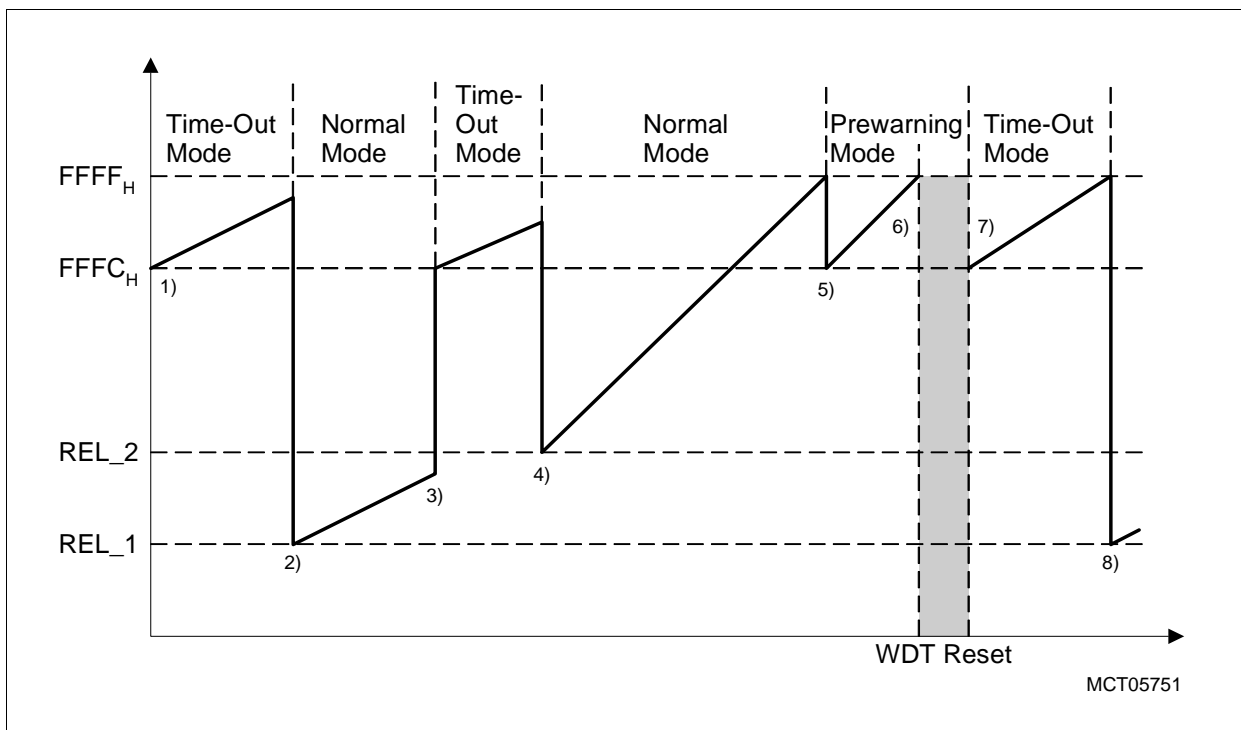


Figure 16-2 Example for an Operation Sequence of the WDT

16.4.1 WDT Register Overview

Two control registers, WDT_CON0 and WDT_CON1, and one status register, WDT_SR, serve for communication of the software with the WDT. This section provides a short overview and describes the access mechanisms of the WDT registers. Detailed layout and bit descriptions of the registers are given on [Page 16-28](#).

Register WDT_CON0 holds the ENDINIT bit, a register lock status bit (WDTLCK), an 8-bit user-definable password field (WDTPW), and the user-definable reload (start) value (WDTREL) for the WDT in Normal Mode.

Register WDT_CON1 contains two bits. Bit WDTIR is a request bit for the WDT input frequency selection, while bit WDTDR is a request bit for the Disable Mode of the WDT. These two bits are only request bits in that they do not actually control the input frequency and disabling of the WDT. They can be modified only when the ENDINIT bit is 0, but they will have an effect only when ENDINIT is properly set to 1 again.

The status register WDT_SR holds information about the current conditions of the WDT. It contains the current timer count value (WDTTIM), three bits indicating the mode of operation (WDTTO for Time-Out Mode, WDTPR for Prewarning Mode, and WDTDS for Disable Mode), and the error indication bits for timer overflow (WDTOE) and access error (WDTAE).

While WDT_SR is a read-only register, the control registers can be read and written. Reading these registers is always possible; a write access, however, must follow certain protocols. Register WDT_CON1 is Supervisor Mode and Endinit-protected, thus, Supervisor Mode must be active and bit ENDINIT must be 0 for a successful write to this register. If one or both conditions are not met, a bus error will be generated, and the bits in WDT_CON1 will be not modified.

Register WDT_CON0 requires a much more complex write procedure as it has a special write-protection mechanism. Proper access to WDT_CON0 always requires two write accesses in order to modify its contents. The first write access requires a password to be written to the register to unlock it. This access is called Password Access. Then, the second access can modify the register's contents. It is called Modify Access. When the Modify Access completes, WDT_CON0 is locked again automatically. (Even if no parameters are changed in the second write access, it is still called a Modify Access.) If the Modify Access sets ENDINIT = 0, then other protected system registers, such as WDT_CON1, are unlocked and can be modified.

Note: WDT_CON0 is automatically re-locked after a Modify Access, so a new Password Access must be performed to modify it again. Note further that the WDT switches to Time-Out Mode as a side-effect of a successful Password Access, so that protected registers can remain unlocked at most for the duration of one Time-out Period. Otherwise, the system will be forced to reset.

16.4.2 Operating Modes of the Watchdog Timer

The WDT can operate in one of four different operating modes:

- Time-Out Mode
- Normal Mode
- Disable Mode
- Prewarning Mode

The following overview describes these modes and how the WDT changes from one mode to the other. As well as these major operating modes, the WDT has special behavior during power-saving and OCDS suspend modes. Detailed discussions of each of the modes can be found on [Page 16-13](#).

Figure 16-3 is a state diagram of the different modes of the WDT and the transition possibilities. Please refer to the description for the conditions for changing from one state to the other.

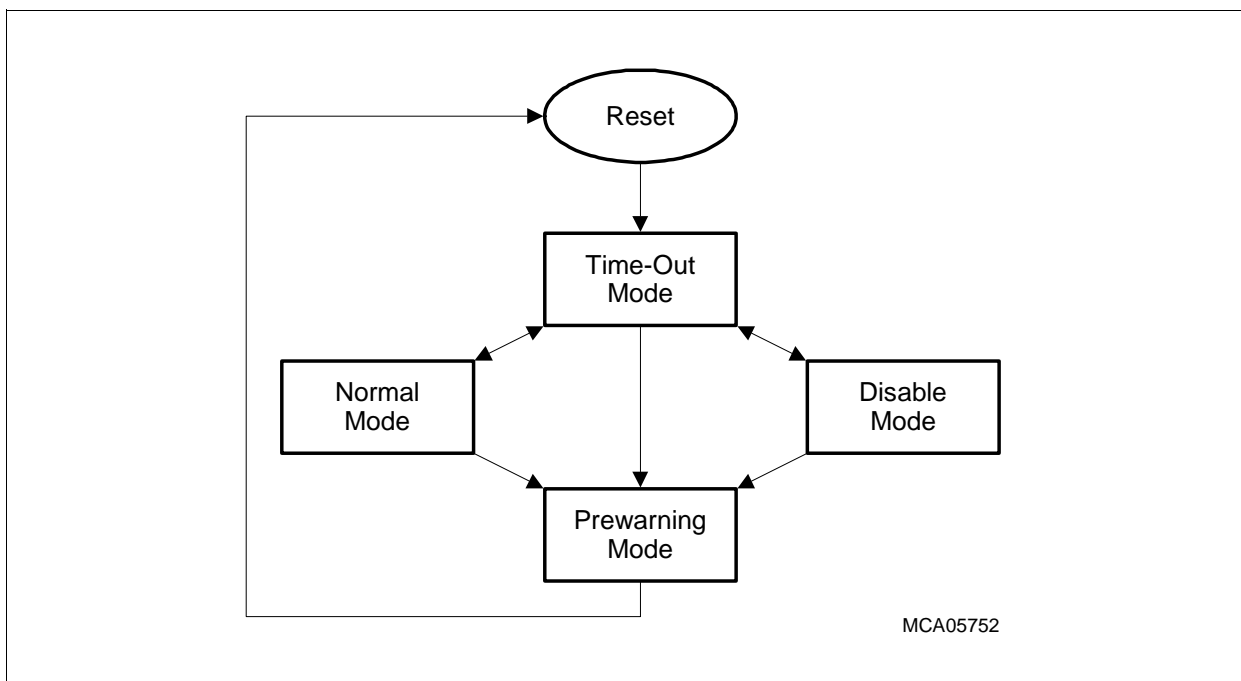


Figure 16-3 State Diagram of the WDT Operating Modes

16.4.2.1 Time-Out Mode

The Time-Out Mode is the default mode after a reset. It is also always entered when a Valid Password Access to register WDT_CON0 is performed (see [Page 16-10](#)). The timer is set to a predefined value and starts counting upwards. Time-Out Mode can only be exited properly by setting ENDINIT to 1 with a correct access sequence. If an improper access to the WDT is performed, or if the timer overflows before ENDINIT is set to 1, a WDT Non-Maskable Interrupt request (WDT_NMI) is requested, and Prewarning Mode is entered. A reset of the TC1796 is imminent and can no longer be stopped.

A proper exit from Time-Out Mode can either be to the Normal or the Disable Mode, depending on the state of the disable request bit, WDTDR, in register WDT_CON1.

16.4.2.2 Normal Mode

In Normal Mode (WDTDR = 0), the WDT operates in a standard Watchdog fashion. The timer is set to a user-defined start value, and begins counting up. It has to be serviced before the counter overflows. Servicing is performed through a proper access sequence to the WDT control register WDT_CON0. This reloads the timer with the start value, and normal operation continues.

If the WDT is not serviced before the timer overflows, or if an invalid access to the WDT is performed, a system malfunction is assumed. Normal Mode is terminated, a WDT NMI request (WDT_NMI) is requested, and Prewarning Mode is entered. A reset of the TC1796 is imminent and can no longer be stopped.

Because servicing the WDT is an access sequence, first requiring a Valid Password Access to register WDT_CON0, the WDT will enter Time-Out Mode until the second proper access is performed.

16.4.2.3 Disable Mode

Disable Mode is provided for applications which truly do not require the WDT function. It can be entered from Time-Out Mode when the disable request bit WDTDR is set to 1. The timer is stopped in this mode. However, disabling the WDT does only stop it from performing the standard Watchdog function (Normal Mode), eliminating the need for timely service of the WDT. It does not disable Time-Out and Prewarning Mode. If an access to register WDT_CON0 is performed in Disable Mode, Time-Out Mode is entered if the access was valid, and Prewarning Mode is entered if the access was invalid. Thus, the ENDINIT monitor function as well as (a part of) the system malfunction detection will still be active.

16.4.2.4 Prewarning Mode

Prewarning Mode is entered always when a Watchdog error is detected. This can be due to an overflow of the timer in Normal or Time-Out Mode, or an invalid access to register WDT_CON0. Instead of immediately generating a reset of the device, as other WDTs do, the TC1796 WDT provides the system with a chance to save important state information before the reset occurs. This is done through first activating an NMI trap request to the CPU, warning it about the coming reset (reset prewarning). If the CPU is still able to do so (depending on the type and severity of the detected malfunction), it can react to the Watchdog NMI request and can save important system state to memory. This saved system state can then be examined during debugging to determine the cause of the malfunction. If the part would be immediately reset on the detection of a Watchdog error, this debugging information would never be available, and investigating the cause of the malfunction would be a very difficult task.

In Prewarning Mode, after having generated the NMI request, the WDT counts for a specified period of time, and then generates a Watchdog reset for the device. This reset generation cannot be avoided in this mode; the WDT does not react anymore to accesses to its registers, nor will it change its state. This is to prevent a malfunction from falsely terminating this mode, disabling the reset, and letting the device to continue to function improperly.

Note: In Prewarning Mode, it is not required for the part to wait for the end of this mode and the reset. After having saved required state in the NMI routine, software can execute a software reset to shorten the time. However, the state of the Watchdog Status Register should also be saved in this case, because the error flags contained in it will be cleared due to the software reset (this is not the case if the Watchdog reset is awaited).

16.4.3 Password Access to WDT_CON0

A correct password must be written to register WDT_CON0 in order to unlock it for modifications. Software must either know the correct password in advance, or it can compute it at runtime. The password required to unlock the register is formed by a combination of bits in registers WDT_CON0 and WDT_CON1, plus a number of guard bits. [Table 16-2](#) summarizes the requirements for the password.

Table 16-2 Password Access Bit Pattern Requirements

Bit Position	Required Value
0	Current state of the ENDINIT bit, WDT_CON0.ENDINIT
1	Fixed; must be written with 0
2	Current state of the input frequency request bit, WDT_CON1.WDTIR
3	Current state of the input frequency request bit, WDT_CON1.WDTDR
[7:4]	Fixed; must be written to 1111 _B
[15:8]	Current value of user-definable password field, WDT_CON0.WDTPW
[31:16]	Current value of user-definable reload value, WDT_CON0.WDTREL

When reading register WDT_CON0, bit positions [7:4] always return 0s. As can be seen from [Table 16-2](#), the password is designed such that it is not possible to just read the contents of a register and use this as the password. The password is never identical to the contents of WDT_CON0 or WDT_CON1, it is always required to modify the read value (at least bits 1 and [7:4]) to get the correct password. This prevents a malfunction from accidentally reading a WDT register's contents and writing it to WDT_CON0 as an unlocking password.

If the password matches the requirements, WDT_CON0 will be unlocked as soon as the Password Access has finished. The unlocked condition will be indicated by WDT_CON0.WDTLCK = 0.

If WDT_CON0 is successfully unlocked, a subsequent write access can modify it, as described on [Page 16-11](#).

If an improper password value is written to WDT_CON0 during the Password Access, a Watchdog Access Error condition exists. Bit WDTAE is set and the Prewarning Mode is entered.

The user-definable password, WDTPW, provides additional options for adjusting the password requirements to the application's needs. It can be used, for instance, to detect unexpected software loops, or to monitor the execution sequence of routines. See [Page 16-24](#).

16.4.4 Modify Access to WDT_CON0

If WDT_CON0 is successfully unlocked as described on [Page 16-10](#), the following write access to WDT_CON0 can modify it. However, this access must also meet certain requirements in order to be accepted and regarded as valid. [Table 16-3](#) lists the required bit patterns. If the access does not follow these rules, a Watchdog Access Error condition is detected, bit WDTAE is set, and the Prewarning Mode is entered.

Table 16-3 Modify Access Bit Pattern Requirements

Bit Position	Value
0	User-definable; desired value for the ENDINIT bit, WDT_CON0.ENDINIT.
1	Fixed; must be written with 1.
2	Fixed; must be written with 0.
3	Fixed; must be written with 0.
[7:4]	Fixed; must be written with 1111 _B .
[15:8]	User-definable; desired value of user-definable password field, WDT_CON0.WDTPW.
[31:16]	User-definable; desired value of user-definable reload value, WDT_CON0.WDTREL.

After the Modify Access has completed, WDT_CON0.WDTLCK is set to 1 again by hardware, automatically re-locking WDT_CON0. Before the register can be modified again, a Valid Password Access must be executed again.

16.4.5 Term Definitions for WDT_CON0 Accesses

To simplify the descriptions in the following sections, a number of terms are defined to indicate the type of access to register WDT_CON0:

Watchdog Access Sequence: Two accesses to register WDT_CON0 consisting of first a Password Access followed by a Modify Access. The two accesses do not have to be adjacent accesses, any number of accesses to other addresses can occur between these accesses as long as the Time-out Period is not exceeded.

Password Access: The first access of a Watchdog Access Sequence to register WDT_CON0 intended to open WDT_CON0 for modifications. This access needs to write a defined password value to WDT_CON0 in order to successfully open WDT_CON0.

Valid Password Access: A Password Access with the correct password value. A Valid Password Access opens register WDT_CON0 for one, and only one, Modify Access. Bit WDTLCK is set to 0 after this access. The WDT is placed into the Time-Out Mode after a Valid Password Access in Normal Mode or Disabled Mode.

Modify Access: The second access of an Watchdog Access Sequence to register WDT_CON0 intended to modify parameters in WDT_CON0. The parameters that can be modified are WDTREL, WDTPW and ENDINIT. Special guard bits in WDT_CON0 must be written with predefined values in order for this access to be accepted.

Valid Modify Access: A Modify Access with the correct guard bit values. The values written to WDTREL, WDTPW, and ENDINIT are in effect after completion of this access. Bit WDTLCK is automatically set to 1 after this access. Register WDT_CON0 is locked until it is re-opened with a Valid Password Access again.

16.4.6 Detailed Descriptions of the WDT Modes

The following subsections provide detailed descriptions of each of the modes of the WDT. The entry conditions and actions, operation in this mode, as well as exit conditions and the succeeding mode are listed for each mode.

16.4.6.1 Time-Out Mode Details

Time-Out Mode is the default after reset, and is entered each time a Valid Password Access to register WDT_CON0 is performed.

Table 16-4 WDT Time-Out Mode

State / Action	Description
Entry	<ul style="list-style-type: none"> • Automatically after any reset • If a valid password was written to WDT_CON0 in Normal or Disable Mode
Actions on Entry	<ul style="list-style-type: none"> • WDTTIM is set to $FFFC_H$; WDTTO is set to 1; WDTDS is set to 0. • ENDINIT = 0 if mode entered through reset; otherwise, it retains its previous value. • Bits WDTAE and WDTOE depend on their state before the reset if the reset was caused by the Watchdog. For any other reset (PORST, HDRST, SRST, PWDRST), they are 0. • WDTIS retains its previous value. • After reset, ENDINIT is 0. Thus, access to Endinit-protected registers is enabled. If Time-Out Mode was entered through other reasons, ENDINIT might or might not be 0.
Operation	<ul style="list-style-type: none"> • Timer starts counting up from $FFFC_H$; increments with clock rate determined through WDTIS (0 after reset, slowest clock). • Access to registers WDT_CON0 is possible. Access to register WDT_CON1 is possible if ENDINIT = 0. • Restarting Time-Out Mode is not possible: A Valid Password Access in this mode does not invoke another Time-out sequence (it does not reload the timer, etc.). A Modify Access to WDT_CON0 writing a 0 to ENDINIT does not terminate Time-Out Mode. • It is not possible to change the reload value or frequency in Time-Out Mode, as this would require setting ENDINIT to 1, which terminates Time-Out Mode. Reload value is not used until Normal Mode is entered.

Table 16-4 WDT Time-Out Mode (cont'd)

State / Action	Description
Exit	<ol style="list-style-type: none"> 1. Writing ENDINIT to 1 with a Valid Modify Access (a Valid Password Access must have been executed first). 2. Timer WDTTIM overflows from FFFF_H to 0000_H. 3. An invalid access to WDT_CON0 (either during the password or the Modify Access).
Next Mode	Depending on the Exit condition: <ol style="list-style-type: none"> 1. If WDTDR = 0 (no disable request), the WDT enters the Normal Mode or if WDTDR = 1 (disable request), the WDT enters the Disable Mode. 2. Bit WDTOE is set to 1, and the WDT enters the Prewarning Mode. 3. Bit WDTAE is set to 1, and the WDT enters the Prewarning Mode.

16.4.6.2 Normal Mode Details

Normal Mode can be entered from Time-Out Mode only if bit WDT_CON1.WDTDR is set to 0 before proper termination of Time-Out Mode. The WDT operates as a standard Watchdog in this mode, requiring timely service to prevent a timer overflow.

Table 16-5 WDT Normal Mode

State / Action	Description
Entry	<ul style="list-style-type: none"> • Only from Time-Out Mode by writing ENDINIT to 1 with a Valid Modify Access (a Valid Password Access must have been executed first), while bit WDTDR = 0.
Actions on Entry	<ul style="list-style-type: none"> • WDTTIM is loaded with the value of WDTREL. • Bits WDTAE, WDTOE, WDTPR, WDTTO, and WDTDS are cleared to 0.
Operation	<ul style="list-style-type: none"> • WDTTIM starts counting up from reload value with frequency selected through WDTIS.
Exit	<ol style="list-style-type: none"> 1. A Valid Password Access to register WDTCON. 2. Timer WDTTIM overflows from FFFF_H to 0000_H. 3. An invalid access to WDT_CON0 (either during the password or the Modify Access).
Next Mode	Depending on Exit condition: <ol style="list-style-type: none"> 1. Time-Out Mode. 2. Prewarning Mode, bit WDTOE is set to 1 (overflow error). 3. Prewarning Mode, bit WDTAE is set to 1 (access error).

16.4.6.3 Disable Mode Details

Disable Mode is provided for applications that truly do not require the WDT function. It can only be entered from Time-Out Mode if bit WDT_CON1.WDTDR is set to 1 before proper termination of Time-Out Mode. The counter stops in this mode, eliminating the need for a WDT service. However, if an access to register WDT_CON0 is performed, the WDT will leave Disable Mode. Disable Mode does not stop the detection of access errors and the entry of Prewarning Mode nor the entry of Time-Out Mode on a Valid Password Access.

Table 16-6 WDT Disable Mode

State / Action	Description
Entry	<ul style="list-style-type: none"> • Only from Time-Out Mode by writing ENDINIT to 1 with a Valid Modify Access (a Valid Password Access must have been executed first), while bit WDTDR = 1.
Actions on Entry	<ul style="list-style-type: none"> • Bits WDTAE, WDTOE, WDTPR, and WDTTO are cleared. Bit WDTDS is set to 1. • Timer WDTTIM is stopped (it retains its current value).
Operation	–
Exit	<ol style="list-style-type: none"> 1. Valid Password Access to register WDTCON. 2. Invalid access to WDT_CON0 (either during the password or the Modify Access).
Next Mode	Depending on Exit condition: <ol style="list-style-type: none"> 1. Time-Out Mode. 2. Prewarning Mode, bit WDTAE is set to 1 (access error).

16.4.6.4 Prewarning Mode Details

Prewarning Mode is always entered immediately after a Watchdog error condition was detected. This can be either an access error to register WDT_CON0 or an overflow of the counter in Normal or Time-Out Mode. This mode indicates that a reset of the device is imminent. Operation of the WDT in this mode can not be altered or stopped, except through a reset.

Table 16-7 WDT Prewarning Mode

State / Action	Description
Entry	Detection of a Watchdog error: <ul style="list-style-type: none"> • Overflow of timer WDTTIM. • Access error to register WDT_CON0 (either on a password or Modify Access) in Time-Out, Normal, or Disable modes.
Actions on Entry	<ul style="list-style-type: none"> • NMIWDT in register NMISR is set (this triggers an NMI request to the CPU). • WDTTIM is set to FFFC_H. • WDTPR is set to 1; WDTDS is set to 0; WDTIS retains its value. • WDTTO retains its previous value: If entry into Prewarning Mode was from Time-Out Mode, WDTTO is 1. In all other cases, WDTTO is 0. • Bits WDTAE and WDTOE indicate whether Prewarning Mode was entered due to an access or an overflow error. They have been set accordingly on exit of the previous mode.
Operation	<ul style="list-style-type: none"> • Timer WDT_TIM starts counting up from FFFC_H with frequency selected through WDTIS. • Register WDT_CON0 can be accessed in this mode as usual. However, the WDT will not change its mode anymore, regardless whether valid or invalid accesses are made to WDT_CON0. For invalid accesses to WDT_CON0 (Password or Modify Access), however, bit WDTAE in WDT_SR will be set. • Register WDT_CON1 can not be written to in Prewarning Mode, even if bit ENDINIT = 0. Write access to WDT_CON1 is totally prohibited.
Exit	<ul style="list-style-type: none"> • Prewarning Mode can not be disabled, prolonged, or terminated (except through a reset). The timer will increment until it overflows from FFFF_H to 0000_H, which then causes a system reset. Bit WDTRST in register RSTSR is set in this case.
Next Mode	Reset

Note: In Prewarning Mode, the device does not have to wait for the end of the Time-out Period and the reset. After having saved required state in the NMI routine,

Watchdog Timer

software can execute a software reset to shorten the time. However, the state of the Watchdog Status Register should also be saved in this case, since the error flags contained in it will be cleared due to the software reset (this is not the case if the Watchdog reset is awaited).

16.4.6.5 WDT Operation During Power-Saving Modes

If the CPU is in Idle Mode or Sleep Mode, it cannot service the WDT because no software is running. Excluding the case where the system is running normally, a strategy for managing the WDT is needed while the CPU is in Idle Mode or Sleep Mode. There are two ways to manage the WDT in these cases. First, the Watchdog can be disabled before idling the CPU. This has the disadvantage that the system will no longer be monitored during the idle period.

A better approach to this problem relies upon a wake-up feature of the WDT. Whenever the CPU is put in Idle or Sleep Mode and the WDT is not disabled, it causes the CPU to be awakened at regular intervals. When the WDT changes its count value (WDT_SR.WD TTIM) from 7FFF_H to 8000_H (when the most significant bit of the WDT counter changes its state from 0 to 1), the CPU is awakened and continues to execute the instruction that follows the instruction that was last executed before entering the Idle or Sleep Mode.

Note: Before switching into a non-running power-management mode, software should perform a Watchdog service sequence. At the Modify Access, the Watchdog reload value, WDT_CON0.WDTREL, should be programmed such that the wake-up occurs after a period which best meets application requirements. The maximum period between two CPU wake-ups is one-half of the maximum WDT period.

16.4.6.6 WDT Operation in OCDS Suspend Mode

When the On-Chip Debugging System (OCDS) is enabled after reset, the WDT will automatically stop when OCDS Suspend Mode is activated. It will resume operation after the Suspend Mode is deactivated.

It is possible that severe system malfunctions may not be corrected even by a system reset. If application code cannot be executed properly because of a system fault, then the WDT initialization code itself might not be able to execute to service the WDT, with the result that two WDT-initiated resets might occur back-to-back. A feature of the WDT detects such Double Watchdog Errors and suspends all system operations after the second reset occurs. This feature prevents the TC1796 from executing random wrong code for longer than the Time-out Period, and prevents the TC1796 from being repeatedly reset by the WDT.

The purpose of the Double Watchdog Error feature is to avoid loops such as: Reset - software does not start correctly - prewarning - watchdog reset - software does not start correctly - ...

The WDT has an internal counter that generates a constant reset after a second watchdog error. This counter can be cleared only by external reset sources (power-on reset or hardware reset).

16.4.7 Determining WDT Periods

The WDT uses the system clock, f_{SYS} . A clock divider in front of the WDT provides two output frequencies, $f_{SYS}/256$ and $f_{SYS}/16384$. Bit WDT_SR.WDTIS selects between these options.

When the WDT is in Normal Mode, the duration of a WDT cycle is defined as a Normal Period, as described on [Page 16-20](#).

When the WDT is in Time-Out Mode or Prewarning Mode, the duration of a WDT cycle is defined as a Time-out Period, as described in the next section.

The general formula to calculate a Watchdog period is:

$$\text{period} = \frac{(2^{16} - \text{startvalue}) \times 256 \times 2^{(1 - \text{WDTIS}) \times 6}}{f_{SYS}} \quad (16.1)$$

The parameter startvalue represents the fixed value FFFC_H for the calculation of the Time-out Period, and the user-programmable reload value WDTREL for the calculation of the Normal Period. Note that the exponent $(1 - \text{WDTIS}) \times 6$ results to 0 if WDTIS is 1, and to 6 if WDTIS is 0. This results in the value 256 being multiplied by either 1 ($2^0 = 1$) or by 64 (2^6), giving the two divider factors 256 and 16384.

Note: Because there is no synchronization of the clock divider to the mode transitions of the Watchdog, the next clock pulse, incrementing the counter, may come after one clock divider period, or immediately after the counter was reloaded. Thus, it is recommended that the reload value is programmed to a value that results in one clock pulse more than the required period.

16.4.7.1 Time-out Period

The duration of Time-Out Mode and Prewarning Mode is determined by the Time-out Period described here. The Time-out Period that occurs immediately after reset is governed entirely by system defaults, as no software is able to run at this point; it is described separately below.

Time-out Period After Reset

After reset, the initial count value for the timer is fixed at $FFFC_H$ when the WDT clock starts running. The WDT counts up at a rate determined by $WDT_SR.WDTIS$, which is 0 after any reset ($f_{SYS}/16384$). Counting up from $FFFC_H$, it takes four clocks for the counter to overflow, so the Time-out Period defaults to a period of $4 \times 16384/f_{SYS} = 65536/f_{SYS}$. This establishes the real-time deadline for software to initialize the Watchdog and critical system registers, and to then set $ENDINIT$. For example, the Time-out Period after reset would correspond to 1.6 ms (@ 40 MHz system frequency).

Changing the input frequency selection via $WDT_CON1.WDTIR$ during this initial Time-out Period has no immediate effect, because frequency selection is actually determined by $WDT_SR.WDTIS$, but $WDT_CON1.WDTIR$ is only copied into $WDT_SR.WDTIS$ after $WDT_CON0.ENDINIT$ has been set to 1, that is, after Time-Out Mode has been properly exited. Hence, the new input frequency will become effective only in a subsequent Time-out Period.

Time-out Period During Normal Operation

As after reset, the WDT counter is initially set to $FFFC_H$ when Time-Out Mode is entered, and Time-Out Mode expires when the counter overflows. However, there are two differences to the Time-out Period after reset. First, the input frequency can be either $f_{SYS}/256$ or $f_{SYS}/16384$, depending on the programmed state of bit $WDT_SR.WDTIS$ before the Time-out Period was entered. Second, because there is no synchronization of the clock divider to the mode transitions of the Watchdog, the next clock pulse, incrementing the counter to $FFFD_H$, may come after one clock divider period, or immediately after the counter was initially set to $FFFC_H$. Thus, the minimum duration of the Time-out Period in the latter case will only be three counter clocks. The possible minimum and maximum periods are given in [Table 16-8](#).

Table 16-8 Time-out Period During Normal Operation

WDTIS	Min./Max.	Period	Example @ $f_{SYS} = 75$ MHz
0	min.	$3 \times 16384/f_{SYS} = 49152/f_{SYS}$	0.66 ms
	max.	$4 \times 16384/f_{SYS} = 65536/f_{SYS}$	0.87 ms
1	min.	$3 \times 256/f_{SYS} = 768/f_{SYS}$	10.2 μ s
	max.	$4 \times 256/f_{SYS} = 1024/f_{SYS}$	13.7 μ s

The WDT input clock rate can not be changed during the Time-out Period. The control bit for the input clock rate, $WDT_SR.WDTIS$, is loaded from $WDT_CON1.WDTIR$ when $WDT_CON0.ENDINIT$ is set to 1, that is, after Time-Out Mode has been properly exited. Hence, the new input frequency will become effective only in the subsequent Time-out Period.

Watchdog Timer

Note: In Prewarning Mode, the device does not need to wait for the end of the Time-out Period and the reset. After having saved required state in the NMI routine, software can execute a software reset to shorten the time. However, the state of the Watchdog Status Register should also be saved in this case, since the error flags contained in it will be cleared due to the software reset (this is not the case if the Watchdog reset is awaited).

16.4.7.2 Normal Period

The duration of Normal Mode can be varied by two parameters, the input clock and the reload value.

The system clock, f_{SYS} , can be divided by either 256 or 16384. WDT_SR.WDTIS selects the input clock divider. The default value of WDTIS after reset is 0, corresponding to a frequency of $f_{SYS}/16384$.

When the WDT is serviced in Normal Mode, it is reloaded with the 16-bit reload value, WDT_CON0.WDTREL.

The WDT Period can be varied over a wide range with these two parameters. Again, since there is no synchronization of the clock divider to the mode transitions of the Watchdog, the next clock pulse, incrementing the counter, may come after one clock divider period, or immediately after the counter was reloaded. Thus, it is recommended that the reload value is programmed to a value that results to one clock pulse more than the required period. Using a reload value of $FFFF_H$ could, therefore, lead to an immediate overflow of the timer. Thus, the examples given in [Table 16-9](#) are only shown with a maximum reload value of $FFFE_H$.

Table 16-9 Timer Periods in Normal Mode

WDTIS	Reload Value	Min./Max.	Period	Example @ $f_{SYS} = 75 \text{ MHz}$
0	0000 _H	min.	$65535 \times 16384/f_{SYS} = 1073725440/f_{SYS}$	14.3 s
		max.	$65536 \times 16384/f_{SYS} = 1073741824/f_{SYS}$	14.3 s
	FFFE _H	min.	$1 \times 16384/f_{SYS} = 16384/f_{SYS}$	218 μs
		max.	$2 \times 16384/f_{SYS} = 32768/f_{SYS}$	437 μs
1	0000 _H	min.	$65535 \times 256/f_{SYS} = 16776960/f_{SYS}$	224 ms
		max.	$65536 \times 256/f_{SYS} = 16777216/f_{SYS}$	224 ms
	FFFE _H	min.	$1 \times 256/f_{SYS} = 256/f_{SYS}$	3.5 μs
		max.	$2 \times 256/f_{SYS} = 512/f_{SYS}$	6.8 μs

16.4.7.3 WDT Period During Power-Saving Modes

Care needs to be taken when programming the WDT reload value before going to Idle or Sleep Mode. As described on [Page 16-17](#), the state of bit 15 of the Watchdog counter is used to wake-up from these modes. Thus, the reload value should be chosen such that it is less than $7FFE_H$ (bit 15 = 0); otherwise an immediate wake-up could occur. Only half of the maximum periods shown in [Table 16-9](#) can be used for the wake-up period.

16.5 Handling the Watchdog Timer

This section describes methods of handling the WDT function.

16.5.1 System Initialization

After any reset, the WDT is put in Time-Out Mode, and WDT_CON0.ENDINIT is 0, providing access to sensitive system registers. Changes to the operation of the WDT controlled by register WDT_CON1 become effective only after WDT_CON0.ENDINIT has been set to 1 again. Thus, changes to the WDT mode bits in WDT_CON1 do not interfere with the Time-out operation of the WDT after reset. [Table 16-10](#) shows the default contents of the WDT registers.

Table 16-10 Watchdog Timer Default Values After Reset

Register	Default Contents	Description
WDT_CON0	FFFC 0002 _H	Reload value is FFFC _H , WDTPW is 0; WDT_CON0 is locked (WDTLCK = 1); ENDINIT is 0.
WDT_CON1	0000 0000 _H	WDT disable request is 0; input clock request set to $f_{SYS}/16384$.
WDT_SR	FFFC 001U _H	The Watchdog counter contains FFFC _H (the initial Time-out value); WDT is operating in Time-Out Mode (WDTTO = 1); WDT is enabled (WDTDS = 0); input clock is $f_{SYS}/16384$. Bits WDTOE and WDTAE are set to 0 after a power-on, a hardware or a software reset. In case of a reset caused by the WDT, these two bits are set depending on the error condition that caused the Watchdog reset.

Because the WDT is in Time-Out Mode after reset, WDT_CON0.ENDINIT must be set to 1 before the Time-out Period expires. This means that initialization of ENDINIT-protected system registers must be complete before the expiration of the Time-out Period, defined on [Page 16-18](#). To set WDT_CON0.ENDINIT to 1, a Valid Password Access to WDT_CON0 must be performed first. During the subsequent Valid Modify Access, WDT_CON0.ENDINIT must be set to 1, which will exit Time-Out Mode. The WDT is switched to the operation determined by the new values of WDTIS and WDTDS.

Note: The action described above must absolutely be performed during initialization of the device to properly terminate this mode. Even if the Watchdog function will not be used in an application and the WDT will be disabled, a valid access sequence to the WDT is mandatory. Otherwise, the Watchdog counter will overflow, Prewarning Mode will be entered, and a Watchdog reset will occur at the end of the Time-out Period.

Bit fields WDT_CON0.WDTREL and WDT_CON0.WDTPW can optionally be changed during the Valid Modify Access, but it is not required. WDT_CON0.ENDINIT can be set to 1 or 0; however, setting ENDINIT to 0 does not stop Time-Out Mode. Any values written to WDTREL, WDTPW, and ENDINIT are stored in WDT_CON0, and WDT_CON0 is automatically locked (WDTLCK = 1) after the Modify Access is finished.

16.5.2 Re-opening Access to Critical System Registers

If some or all of the system's Endinit-protected registers must be changed during run time of an application, access can be re-opened. To do this, WDT_CON0 must first be unlocked with a Valid Password Access. In the subsequent Valid Modify Access, ENDINIT can be set to 0. Access to Endinit-protected registers is now open again. However, when WDT_CON0 is unlocked, the WDT is automatically switched to Time-Out Mode. Thus, the access window is time-limited. Time-Out Mode is only terminated after ENDINIT has been set to 1 again, requiring another Valid Password and Valid Modify Access to WDT_CON0.

If the WDT is not used in an application and is therefore disabled (WDT_SR.WDTDS = 1), the above described case is the only occasion when WDT_CON0 must be accessed again after the system is initialized. If there are no further changes to critical system registers needed, no further accesses to WDT_CON0, WDT_CON1, or WDT_SR are necessary. However, it is always recommended that the WDT be used in an application for safety reasons.

16.5.3 Servicing the Watchdog Timer

If the WDT is used in an application and is enabled (WDT_SR.WDTDS = 0), it must be regularly serviced to prevent it from overflowing.

Service is performed in two steps. a Valid Password Access followed by a Valid Modify Access. The Valid Password Access to WDT_CON0 automatically switches the WDT to Time-Out Mode. Thus, the Modify Access must be performed before the Time-out expires or a system reset will result.

During the next Modify Access, the strict requirement is that WDT_CON0.ENDINIT as well as bit 1 and bits [7:4] are written with 1, while bits [3:2] are written with 0.

Note: ENDINIT must be written with 1 to perform a proper service, even if it is already set to 1.

Changes to the reload value WDTREL, or the user-definable password WDTPW, are not required. However, changing WDTPW is recommended so that software can monitor WDT service operations throughout the duration of an application program (see next section).

If WDT service is properly executed, Time-Out Mode is terminated, and the WDT switches back to its former mode of operation, and WDT service is complete.

16.5.4 Handling the User-Definable Password Field

WDT_CON0.WDTPW is an 8-bit field that can be set by software to any arbitrary value during a Modify Access. Settings of this bit field have no effect on the operation of the WDT, other than the role it plays in forming the password bit pattern, as discussed on [Page 16-10](#).

The purpose of this bit field is to support further enhancements to the password protection scheme. For the following description, it is assumed that software does at least not fully compute the value for the Password Access from the contents of registers WDT_CON0 and WDT_CON1, but uses a predefined constant, embedded in the instruction stream, for the password (this is at least necessary for the user-definable password field WDTPW). For example, software can modify this bit field each time it executes a Watchdog service sequence. The next service sequence needs to take this new value into account for its Password Access, and it again changes the value during its Modify Access. Up to 256 different password values can be used. In this way, each service sequence is unique. If a malfunction occurs that, for instance, would result in the omission of one or more of these service sequences, the next service sequence would most probably not write the correct password. This service sequence would rely on the password value programmed during the normally preceding service sequence. However, if this one was skipped, the password value required by the contents of the Watchdog registers is the one programmed at the last service sequence executed before the malfunction had occurred. A Watchdog error condition would be detected in this case.

In the same manner, the Watchdog would detect the malfunction if a service sequence would be executed twice due to a falsely performed jump. [Figure 16-4](#) illustrates these examples.

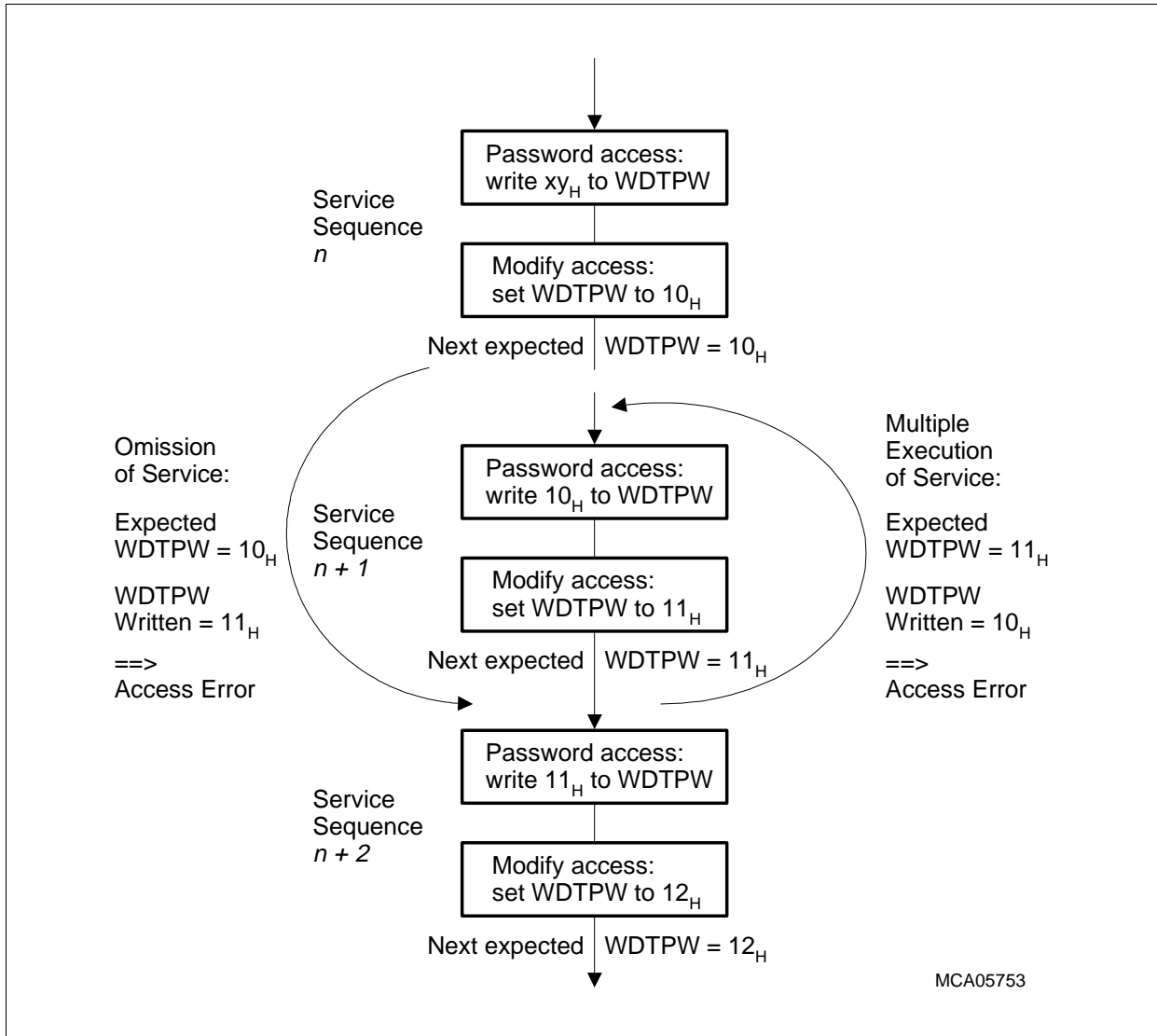


Figure 16-4 Detection of False Jumps and Loops

Other schemes are possible. Consider the case in which a routine determines some conditions that alter the program flow. One of two or more different paths will be executed next, depending on these conditions. Before branching to the appropriate routine(s), software performs a Watchdog service and sets the new password value for WDTPW such that it depends on these conditions, that is, some or all of these condition codes can be incorporated into WDTPW. The next service sequence is performed at the point where the different paths come together again. To determine the correct password, software uses a value returned from the path that was executed. This value must match the value in WDTPW, otherwise the wrong path was executed. [Figure 16-5](#) shows an example.

It is also possible to have the different paths of a program compute the full or partial password to unlock register WDT_CON0. The password will only match at the next

Watchdog Timer

service sequence if all the expected paths and calculation routines have been executed properly. If one or more steps were omitted or a wrong path was executed due to a malfunction, the Watchdog failure mechanism will detect this and issue a reset of the device (after the prewarning phase).

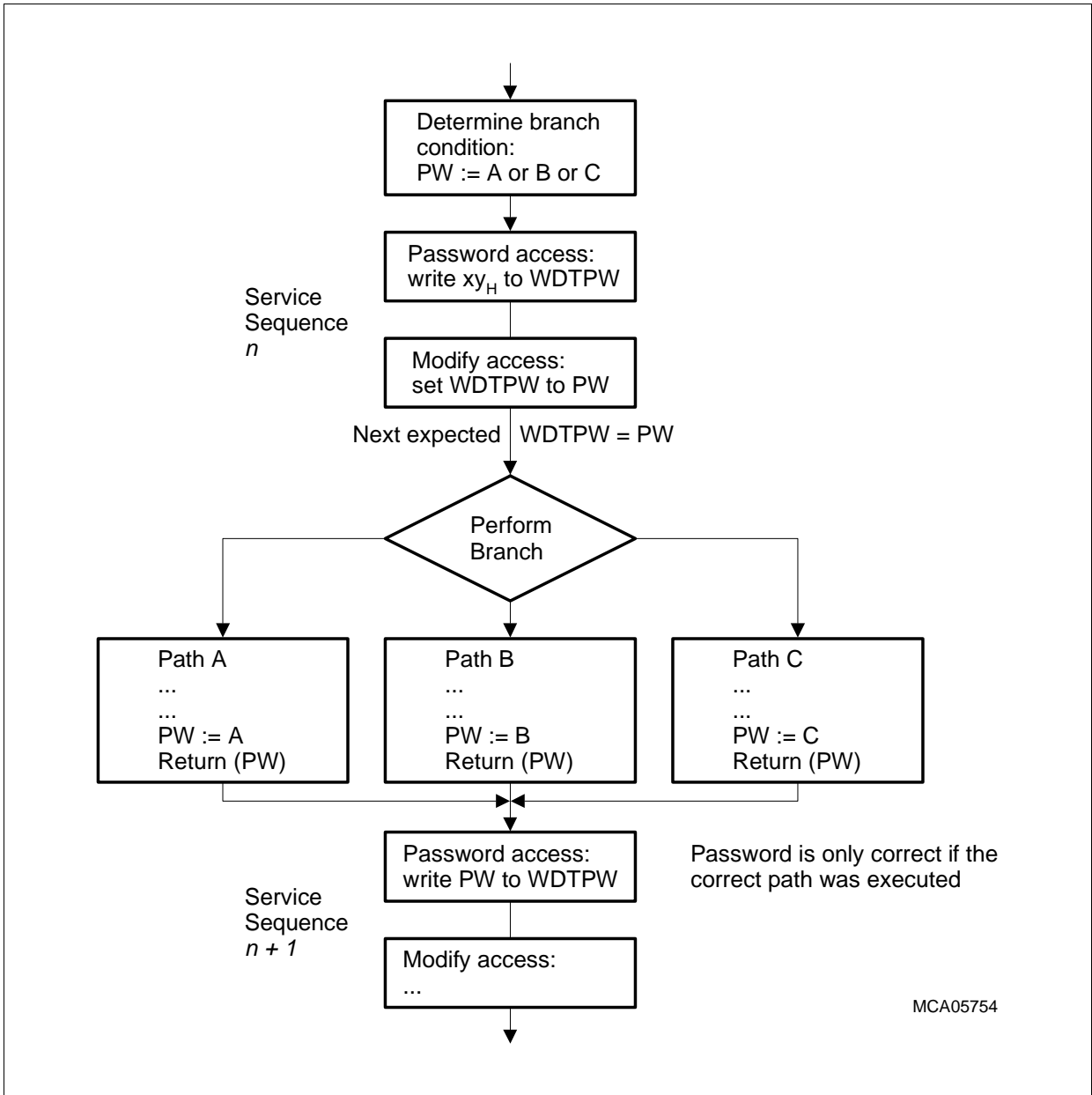


Figure 16-5 Monitoring Program Sequences

16.5.5 Determining the Required Values for a WDT Access

As described on [Page 16-10](#) and [Page 16-11](#), the values required for the password and Modify Accesses to register WDT_CON0 are designed such that they can be derived from the values read from registers WDT_CON0 and WDT_CON1. However, at least some bits have to be modified in order to get the correct write value. This makes it very unlikely that a false operation derives values from reading these registers that inadvertently affect the WDT operation when written back to WDT_CON0. Even if a false write operation would have written the correct password to WDT_CON0, one further, different correct value needs to be written to this register in order to have an effect. In addition, the WDT switches to Time-Out Mode after the Valid Password Access, providing only a time-limited window for the second access.

Although computing the required values from the current contents of the Watchdog registers is one option, using predetermined values that are set at compile-time of the program, may be the better approach in many cases. Usually, handling the WDT is performed by one and only one task. Thus, the problem will not occur that another task might have changed some of the parameters which must not be modified (which would require reading the contents, modifying the value appropriately, and then writing it back). The one task handling the Watchdog Timer function would always “know” how it has programmed the WDT last time, and would therefore also “know” the next password value for opening WDT_CON0. In fact, this method would actually detect the case if another task had illegally modified the Watchdog registers, since the predetermined password might not work anymore, and a Watchdog error condition would thus be generated.

In addition, accessing the WDT with predetermined values has the obvious benefit of shorter code, as no computing steps need to be performed.

16.6 Watchdog Timer Registers

This section describes the WDT registers. The WDT is provided with three registers: WDT_CON0, WDT_CON1, and WDT_SR, as shown in [Figure 16-6](#). They are located inside the SCU module's address range.

WDT Registers Overview

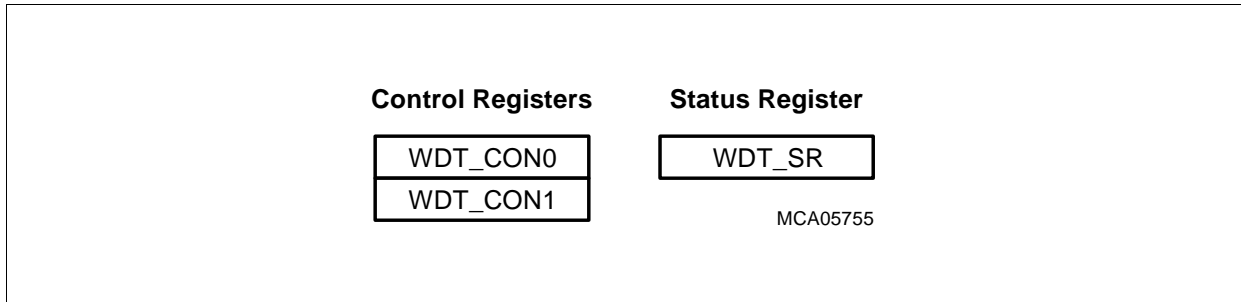


Figure 16-6 Watchdog Registers

The complete and detailed address map of the SCU which includes the WDT registers is shown in [Table 18-3](#) on [Page 18-7](#).

Table 16-11 Registers Address Space - WDT Registers

Module	Base Address	End Address	Note
WDT	F000 0000 _H	F000 00FF _H	—

Table 16-12 Registers Overview - WDT Kernel Registers

Register Short Name	Register Long Name	Offset Address	Description see
WDT_CON0	Watchdog Timer Control Register 0	20 _H	Page 16-29
WDT_CON1	Watchdog Timer Control Register 1	24 _H	Page 16-31
WDT_SR	Watchdog Timer Status Register	28 _H	Page 16-32

Watchdog Timer

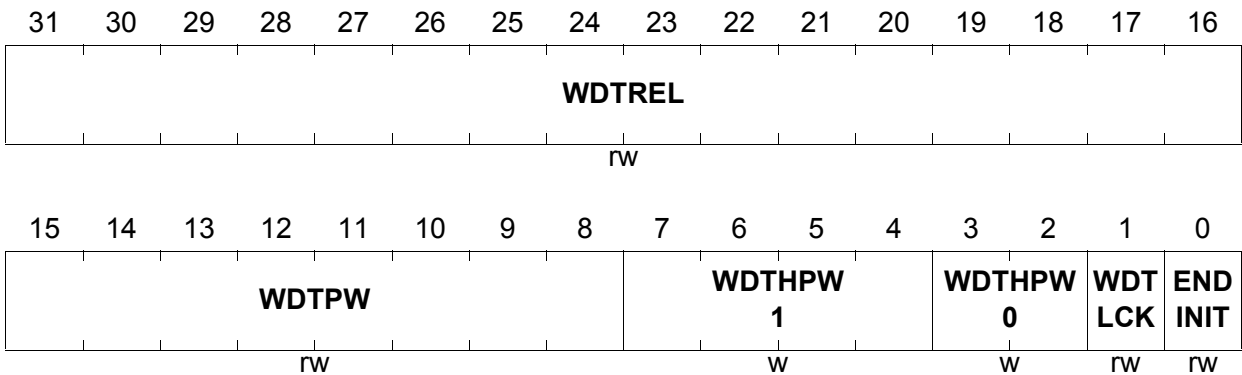
16.6.1 Watchdog Timer Control Register 0

Register WDT_CON0 manages the Password Access to the WDT. It also stores the timer reload value, a user-definable password field, a lock bit, and the End-of-Initialization control bit.

WDT_CON0

Watchdog Timer Control Register 0 (20_H)

Reset Value: FFFC 0002_H



Field	Bits	Type	Description
ENDINIT	0	rw	<p>End-of-Initialization Control Bit</p> <p>0_B Access to Endinit-protected registers is permitted (default after reset).</p> <p>1_B Access to Endinit-protected registers is not permitted.</p> <p>ENDINIT controls the access to critical system registers. During a Password Access, it must be written with its current value. It can be changed during a Modify Access to WDT_CON0.</p>

Watchdog Timer

Field	Bits	Type	Description
WDTLCK	1	rw	<p>Lock Bit to Control Access to WDT_CON0</p> <p>0_B Register WDT_CON0 is unlocked. 1_B Register WDT_CON0 is locked (default after reset).</p> <p>The actual value of WDTLCK is controlled by hardware. It is set to 0 after a successful Password Access to WDT_CON0, and automatically set to 1 again after a successful Modify Access to WDT_CON0. During a write to WDT_CON0, the value written to this bit is only used for the password-protection mechanism and is not stored. This bit must be set to 0 during a Password Access to WDT_CON0, and set to 1 during a Modify Access to WDT_CON0. That is, the inverted value read from WDTLCK always must be written to itself.</p>
WDTHPW0	[3:2]	w	<p>Hardware Password 0</p> <p>This bit field must be written with the value of the bits WDT_CON1.WDTR and WDT_CON1.WDIR during a Password Access. This bit field must be written with 0's during a Modify Access to WDT_CON0. When read, these bits always return 0.</p>
WDTHPW1	[7:4]	w	<p>Hardware Password 1</p> <p>This bit field must be written to 1111_B during both, a Password Access and a Modify Access to WDT_CON0. When read, these bits always return 0.</p>
WDTPW	[15:8]	rw	<p>User-Definable Password Field for Access to WDT_CON0</p> <p>This bit field must be written with its current contents during a Password Access. It can be changed during a Modify Access to WDT_CON0.</p>
WDTREL	[31:16]	rw	<p>Reload Value for the WDT</p> <p>If the WDT is enabled and in Normal Timer Mode, it will start counting from this value after a correct Watchdog service. This bit field must be written with its current contents during a Password Access. It can be changed during a Modify Access to WDT_CON0 (FFFC_H = default after reset).</p>

Watchdog Timer

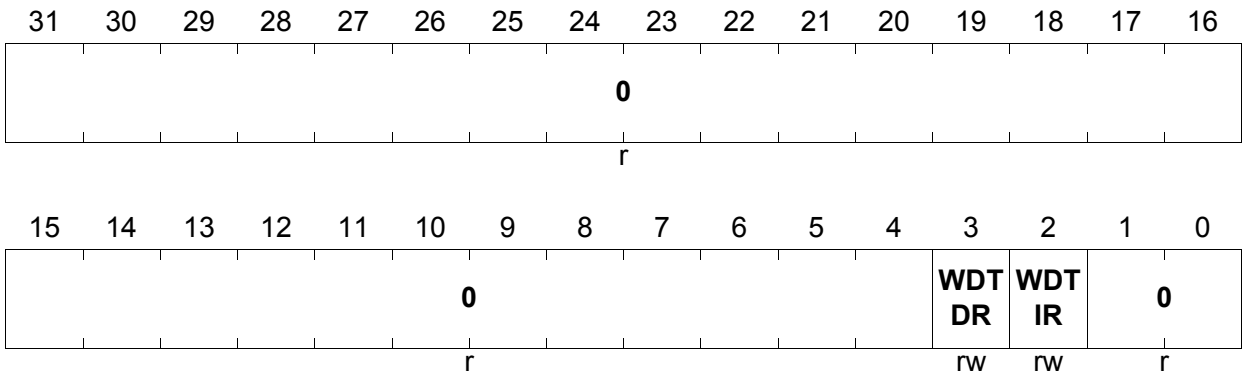
16.6.2 Watchdog Timer Control Register 1

WDT_CON1 manages operation of the WDT. It includes the disable request and frequency selection bits. It is ENDINIT-protected.

WDT_CON1

Watchdog Timer Control Register 1 (24_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
WDTIR	2	rw	<p>WDT Input Frequency Request Control Bit</p> <p>0_B Request to set input frequency to $f_{SYS}/16384$.</p> <p>1_B Request to set input frequency to $f_{SYS}/256$.</p> <p>This bit can only be modified if WDT_CON0.ENDINIT is set to 0. WDT_SR.WDTIS is updated by this bit only when ENDINIT is set to 1 again. As long as ENDINIT is left at 0, WDT_SR.WDTIS controls the current input frequency of the WDT. When ENDINIT is set to 1 again, WDT_SR.WDTIS is updated with the state of WDTIR.</p>
WDTDR	3	rw	<p>Watchdog Timer Disable Request Control Bit</p> <p>0_B Request to enable the WDT.</p> <p>1_B Request to disable the WDT.</p> <p>This bit can only be modified if WDT_CON0.ENDINIT is set to 0. WDT_SR.WDTDS is set to this bit's value when ENDINIT is set to 1 again. As long as ENDINIT is left at 0, bit WDT_SR.WDTDS controls the current enable/disable status of the WDT. When ENDINIT is set to 1 again with a Valid Modify Access, WDT_SR.WDTDS is updated with the state of WDTDR.</p>

Field	Bits	Type	Description
0	[1:0], [31:4]	r	Reserved Read as 0; should be written with 0.

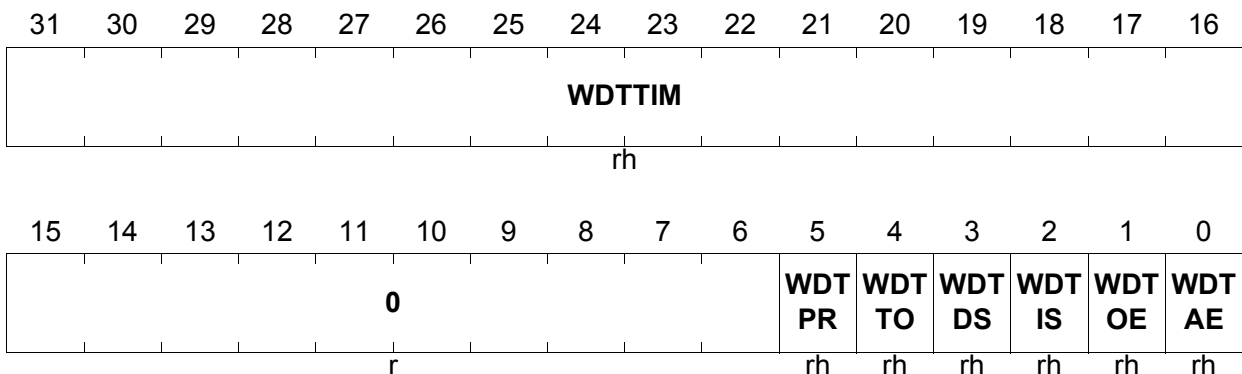
16.6.3 Watchdog Timer Status Register

Register WDT_SR shows the current state of the WDT. Status include bits indicating reset prewarning, Time-out, enable/disable status, input clock status, and access error status.

The reset value for this register depends on the cause of the reset. For any reset other than a Watchdog reset, the reset value is FFFC 001U_H. After a Watchdog reset, bits WDTAE and WDTOE indicate the type of Watchdog error that occurred before the Watchdog reset. Either one or both bits can be set. These bits are not cleared on a Watchdog reset. Bits WDTDS and WDTIS are always 0 after any reset.

WDT_SR

Watchdog Timer Status Register (28_H) **Reset Value: FFFC 0010_H**



Field	Bits	Type	Description
WDTAE	0	rh	<p>Watchdog Access Error Status Flag</p> <p>0_B No Watchdog access error. 1_B An Watchdog access error has occurred.</p> <p>This bit is set by hardware when an illegal Password Access or Modify Access to register WDT_CON0 was attempted. This bit is only cleared when:</p> <ul style="list-style-type: none"> • A power-on, hardware, or software reset occurs. • WDT_CON0.ENDINIT is set to 1 during a Valid Modify Access. <p>However, it is not possible to cleared this bit if the WDT is in Prewarning Mode, indicated by WDT_SR.WDTPR = 1.</p>

Watchdog Timer

Field	Bits	Type	Description
WDTOE	1	rh	<p>Watchdog Overflow Error Status Flag</p> <p>0_B No Watchdog overflow error. 1_B A Watchdog overflow error has occurred.</p> <p>This bit is set by hardware when the WDT overflows from FFFF_H to 0000_H. This bit is only cleared when:</p> <ul style="list-style-type: none"> • A power-on, hardware, or software reset occurs; • WDT_CON0.ENDINIT is set to 1 during a Valid Modify Access. <p>However, it is not possible to cleared this bit if the WDT is in Prewarning Mode, indicated by WDT_SR.WDTPR = 1.</p>
WDTIS	2	rh	<p>Watchdog Input Clock Status Flag</p> <p>0_B WDT input clock is $f_{SYS}/16384$ (default after reset). 1_B WDT input clock is $f_{SYS}/256$.</p> <p>This bit is updated with the state of bit WDT_CON1.WDTIR after WDT_CON0.ENDINIT is written with 1 during a Valid Modify Access to register WDT_CON0.</p>
WDTDS	3	rh	<p>Watchdog Enable/Disable Status Flag</p> <p>0_B WDT is enabled (default after reset). 1_B WDT is disabled.</p> <p>This bit is updated with the state of bit WDT_CON1.WDTDR after WDT_CON0.ENDINIT is written with 1 during a Valid Modify Access to register WDT_CON0.</p>
WDTTO	4	rh	<p>Watchdog Time-Out Mode Flag</p> <p>0_B Normal mode. 1_B The Watchdog is operating in Time-Out Mode (default after reset).</p> <p>This bit is set to 1 when Time-Out Mode is entered, automatically after a reset and after every Password Access to register WDT_CON0. It is automatically cleared by hardware when Time-Out Mode is properly terminated through a Valid Modify Access to WDT_CON0. It is left set when a Watchdog error occurs during Time-Out Mode, and Prewarning Mode is entered.</p>

Watchdog Timer

Field	Bits	Type	Description
WDTPR	5	rh	<p>Watchdog Prewarning Mode Flag</p> <p>0_B Normal mode (default after reset)</p> <p>1_B The Watchdog is operating in Prewarning Mode</p> <p>This bit is set to 1 when a Watchdog error is detected. The WDT has issued an NMI trap and is in Prewarning Mode. A reset of the chip occurs after the prewarning period has expired.</p>
WDTTIM	[31:16]	rh	<p>WDT Value</p> <p>Reflects the current content of the WDT.</p>
0	[15:6]	r	<p>Reserved</p> <p>Read as 0.</p>

17 On-Chip Debug Support

This chapter gives an overview on the debug features of the TC1796 device. This chapter does not describe the TC1796 debug functionality and capabilities in detail. For detailed information about the On-Chip Debug Support (OCDS) functionality (e.g. required for tool suppliers), please contact local Infineon representatives.

17.1 Overview

The TC1796 supports three levels of debug operation:

- OCDS Level 1
- OCDS Level 2
- OCDS Level 3

OCDS Level 1

The OCDS Level 1 is mainly assigned for real-time software debugging purposes which have a demand for low-cost standard debugger hardware.

The OCDS Level 1 is based on a JTAG interface that is used by the external debug hardware to communicate with the system. The on-chip Cerberus module controls the interactions between the JTAG interface and the on-chip modules. The external debug hardware may become master of the internal buses, and read or write the on-chip register/memory resources. The Cerberus also makes it possible to set breakpoint and trigger conditions as well as to control user program execution (run/stop, break, single-step).

OCDS Level 2

The OCDS Level 2 makes it possible to implement program tracing capabilities for enhanced debuggers by extending the OCDS Level 1 debug functionality with an additional 16-bit wide trace output port with trace clock. With the trace extension, the following four trace capabilities are provided (only one of the four trace capabilities can be selected at a time):

- Trace of the CPU program flow
- Trace of the PCP program flow
- Trace of the DMA Controller transaction requests
- Trace of the DMA Controller Move Engine status information

OCDS Level 3

The OCDS Level 3 is based on a Multi-Core Debug Solution (MCDS) using a special TC1796 emulation device, the TC1796ED. This device has additional features required for high-end emulation purposes. The TC1796ED includes the TC1796 product chip and additional emulation extension hardware in a package with the same footprint as the

TC1796. For detailed information about the TC1796ED functionality (e.g. required by high-end emulation manufacturers), please contact local Infineon representatives.

Figure 17-1 is a block diagram of the TC1796 OCDS based system.

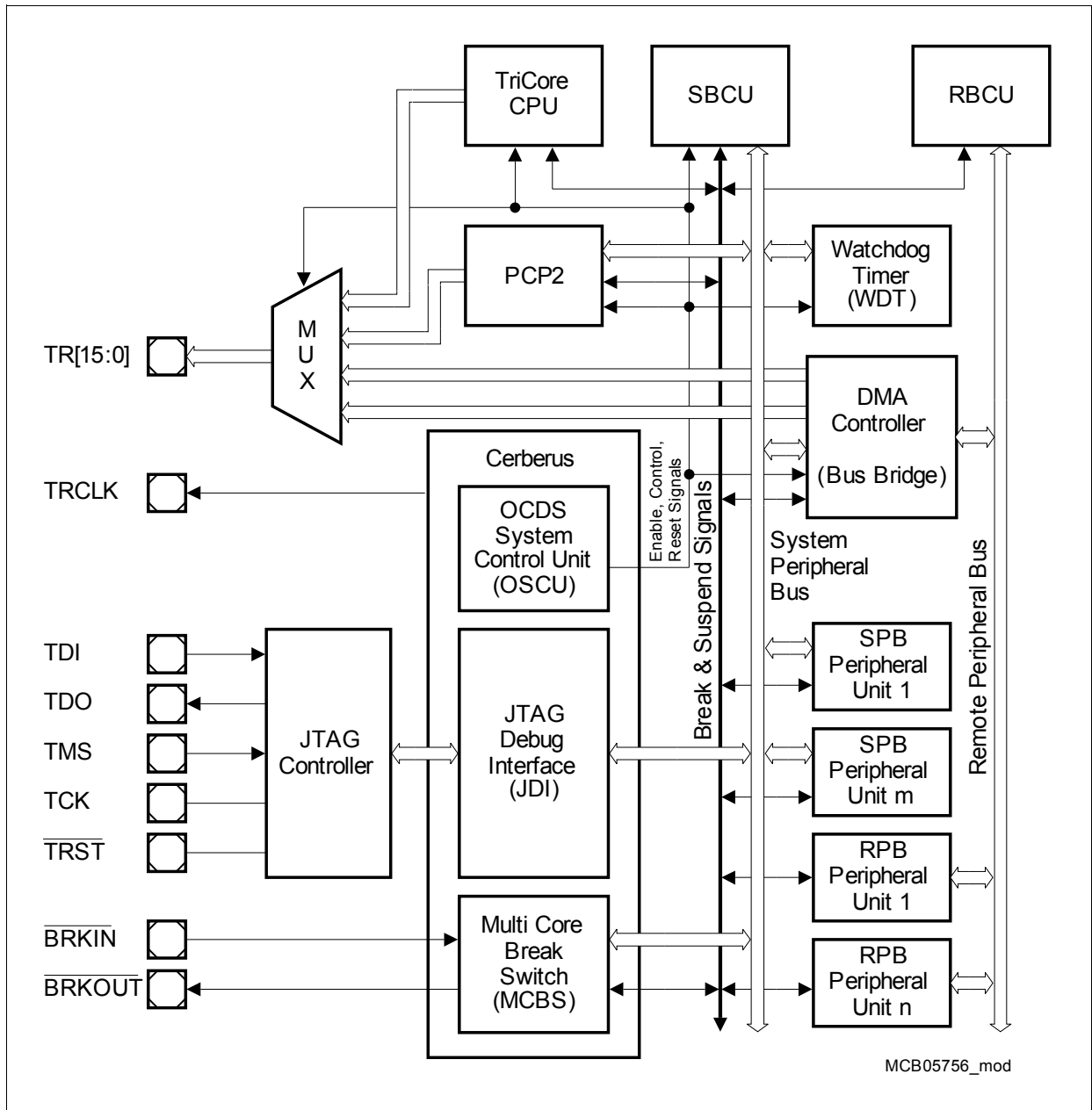


Figure 17-1 OCDS Based System Block Diagram

Components

The OCDS of the TC1796 consists of the following building blocks:

- OCDS Level 1 module of TriCore
- OCDS Level 2 interface of TriCore

- OCDS Level 1 module of PCP
- OCDS Level 2 interface of PCP
- OCDS Level 1 module in the BCU of the System Peripheral Bus (SBCU)
- OCDS Level 1 module in the BCU of the Remote Peripheral Bus (RBCU)
- OCDS Level 1 facilities within the DMA controller
- OCDS Level 2 interface of the DMA controller
- Cerberus - OCDS System Control Unit (OSCU)
- Cerberus - Multi-Core Break Switch (MCBS)
- Cerberus - JTAG Debug Interface (JDI)
- Suspend functionality of the peripherals at the System Peripheral Bus and Remote Peripheral Bus

Summary of OCDS Features

- TriCore Level 1 OCDS:
 - Hardware event generation
 - Break by DEBUG instruction or Break signal from break switch
 - Full hardware supported single step
 - Software Single-Step (code patching) also possible
 - Concurrent access to memory and SFRs via Cerberus possible
- PCP Level 1 OCDS:
 - Break by DEBUG instruction or Break signal from break switch
 - Concurrent access to memory and SFRs via Cerberus possible
- DMA Level 1 OCDS:
 - Break request on error
 - Event generation on specified channel activity
 - Suspending pre-selected channels
- BCU Level 1 OCDS:
 - Event generation on specified transactions
- 16-pin Level 2 trace port; outputs either TriCore, PCP or DMA trace
- OCDS System Control Unit (Cerberus OSCU):
 - Extensive control using only a minimum number of pins (no pins apart from JTAG)
 - Connecting a debugger to a running system allowed (hot attach)
 - Built in System security
 - Optional halt after reset
 - Debug resources not affected by system reset
- Multi-Core Break Switch (Cerberus MCBS):
 - TriCore, PCP, DMA, break pins and BCUs available as break sources
 - TriCore and PCP available as break targets; other parts can be suspended in addition
 - Synchronous stop and restart of the system
 - Break to Suspend converter

17.2 OCDS Level 1

The main philosophy of the TriCore OCDS Level 1 is that the complete architecture and the status of a target system are visible from the FPI Bus. This means that every component of the system can be accessed through its mapping into the FPI address space, including on-chip memories, CPU core registers, registers of the peripheral units, as well as external hardware which is connected and accessible via the External Bus Unit (EBU).

A typical OCDS Level 1 debugging configuration is shown in [Figure 17-2](#). It includes two parts:

1. The debugger software, supporting a standard JTAG protocol via a PC port
2. The debugger hardware interface adapter, connecting the TC1796 JTAG interface in the target system with the PC port (parallel, serial, or USB)

This configuration makes it possible to realize a cheap debugging environment that permits comprehensive real-time debugging tasks to be performed.

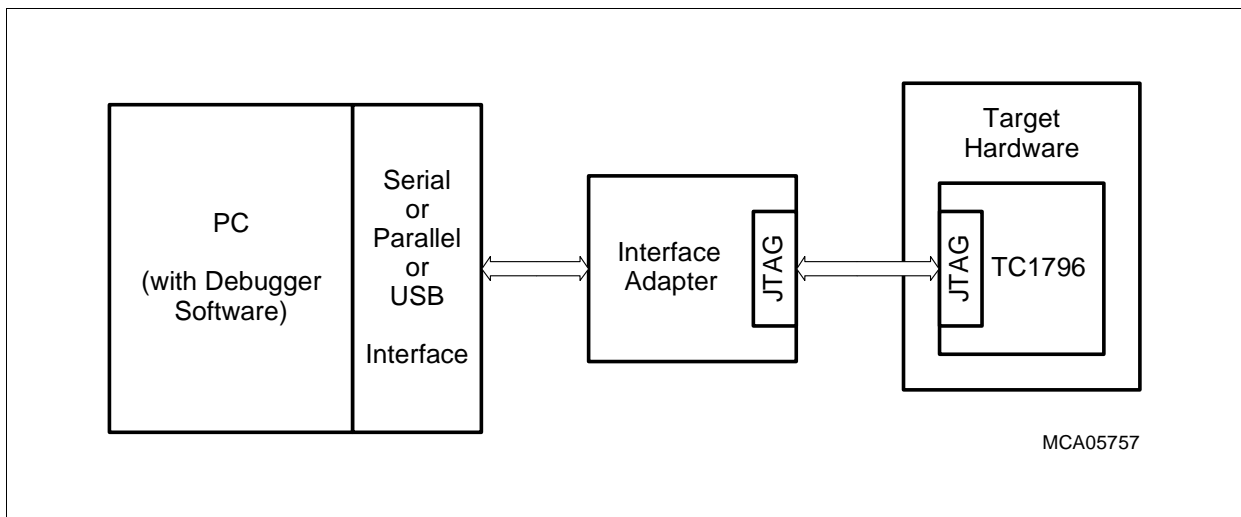


Figure 17-2 Typical OCDS Level 1 Hardware Connections

17.2.1 TriCore CPU Level 1 OCDS

The TriCore CPU provides the following OCDS Level 1 features:

- Full single-step support, as well by hardware as by software (code patching)
- Up to 4 programmable hardware breakpoints:
 - Each one can be defined as a combination of instruction pointer value and memory/SFR address/value:
 - Breaks on program counter (PC) value
 - Two precise PC values or one PC range
 - Break before make (BBM) possible

- Breaks on data/address
 - Two precise data/address values or one data/address range
 - No break before make possible (due to pipelined architecture)
- Combinations of the above break conditions
- Real-time features
 - Read and write of memory/registers quasi-concurrently, with minimum intrusion (stealing bus cycles by Cerberus).
 - High priority requests can still be serviced when the core is in emulation mode, by interrupting the monitor program.

17.2.1.1 Basic Concept

The TriCore CPU in the TC1796 provides OCDS with the following two basic parts:

- Debug Event Trigger Generation
- Debug Event Trigger Processing

The first part controls the generation of debug events, and the second part controls what actions are taken when a debug event is generated.

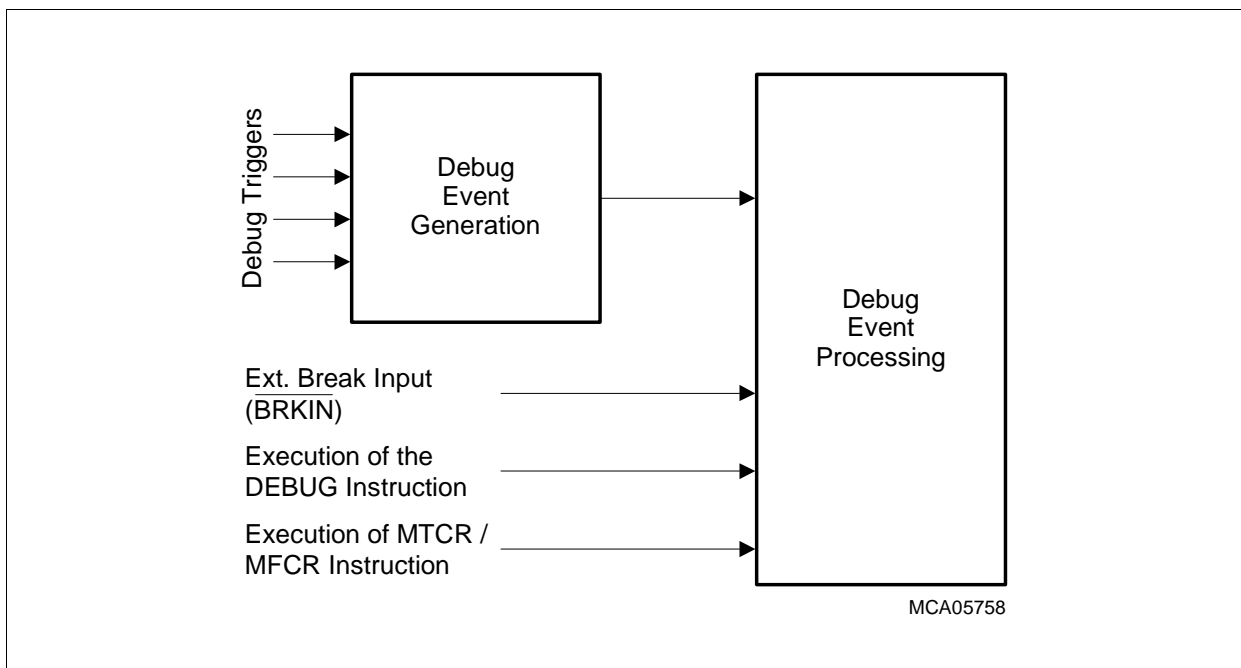


Figure 17-3 Basic TriCore Debug Concept

17.2.1.2 Debug Event Generation

If debug mode is enabled, debug events can be generated by:

- Debug event generation from debug triggers
- Activation of the external break input pin $\overline{\text{BRKIN}}$
- Execution of a DEBUG instruction
- Execution of a MTCR/MFCR instruction

Debug Event Generation from Debug Triggers

The debug event generation unit is responsible for generating debug events when a programmable set of debug triggers is active. Debug triggers can be generated by:

- The code protection logic
- The data protection logic

These debug triggers provide the inputs to a programmable block of combinational logic that outputs debug events. The aim is to be able to specify the breakpoints that use fairly simple criteria purely in the on-chip debug event generation unit, and to rely on help from the external debug system or debug monitor to implement more complex breakpoints.

Activation of the External Break Input Pin $\overline{\text{BRKIN}}$

When activating the TC1796 device pin $\overline{\text{BRKIN}} = 0$, the MCBS unit induces a break event as specified in a External Break Input Event Specifier Register EXEVT.

Execution of a DEBUG Instruction

The TriCore architecture supports a mechanism through which software can explicitly generate a debug event. This can be used, for instance, by a debugger to patch code held in RAM in order to implement breakpoints. A special DEBUG instruction is defined which is a user mode instruction, and its operation depends on whether the debug mode is enabled. 16-bit and 32-bit forms of the DEBUG instruction are provided.

If debug mode is enabled, the DEBUG instruction causes a debug event to be raised and the action defined in the Software Break Event Specifier Register SWEVT is taken. If the debug mode is not enabled, then the DEBUG instruction is treated as a NOP instruction.

Execution of a MTCR/MFCR Instruction

In order to protect the emulator resource, a debug event is raised whenever a MTCR or MFCR instruction is used to read or modify an user core SFR, but an event is not raised when the user reads or modifies one of the dedicated core debug registers:

- DBGSR or
- CREVT or
- SWEVT or
- EXEVT or
- TR0EVT or
- TR1EVT

The action that is performed when a MTCR or MFCR instruction is executed on user core SFRs defined by the content of the Emulator Resource Protection Event Specifier Register CREVT.

17.2.1.3 Debug Actions

Four types of debug actions are available:

- Assert $\overline{\text{BRKOUT}}$ signals by the MCBS unit
- Halt the CPU core
- Cause a breakpoint trap
- Generate an interrupt request

These debug actions are selected by programming the corresponding Event Specifier registers. Their contents determine which action shall be taken when the corresponding debug event occurs.

17.2.1.4 TriCore OCDS Registers

Figure 17-4 shows the TriCore OCDS registers in an overview.

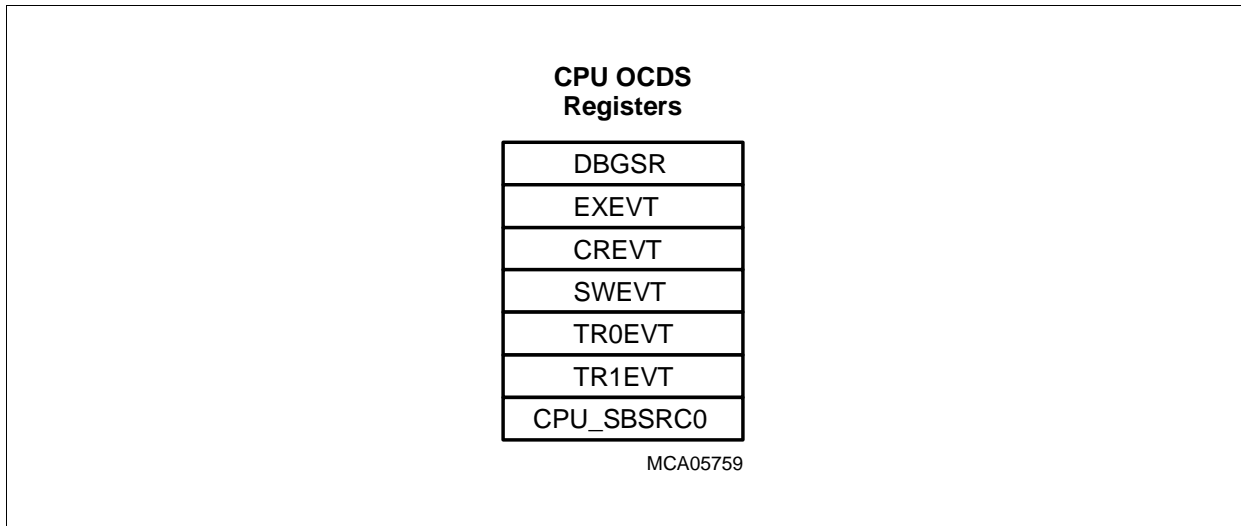


Figure 17-4 TriCore Core Debug Registers

Table 17-1 TriCore OCDS Registers

Register Short Name	Register Long Name	Address
DBGSR	Debug Status Register	F7E1 FD00 _H
EXEVT	External Break Input Event Specifier Register	F7E1 FD08 _H
CREVT	Core SFR Access Break Event Specifier Register	F7E1 FD0C _H
SWEVT	Software Break Event Specifier Register	F7E1 FD10 _H
TR0EVT	Trigger Event 0 Specifier Register	F7E1 FD20 _H
TR1EVT	Trigger Event 1 Specifier Register	F7E1 FD24 _H
CPU_SBSRC0	CPU Software Break Service Request Control Register 0	F7E0 FFBC _H ¹⁾

1) Located in the CPU slave (CPS) interface register area.

17.2.2 PCP OCDS Level 1

The PCP has no means for generating trigger events. To set breakpoints the debugger is expected to patch the code in the PCP code memory (CMEM) with DEBUG instructions.

If a DEBUG instruction is executed, the running channel program is terminated, either with Debug Exit or with Error Exit. Additionally the Break Switch can send an external break request to the PCP, with the same consequence (Debug Exit or Error Exit).

17.2.3 BCU OCDS Level 1

The BCUs of the two FPI buses inside the TC1796 support OCDS Level 1, which offers very comfortable and powerful means for breakpoint generation.

Each BCU (SBCU, RBCU) contains one comparator each for

- the arbitration phase (look for specific bus master)
- the address phase (look for specific address or range)
- the data phase (look for read, write, supervisor mode, etc.)

The results can be combined to generate a break request signal, which is sent to the Break Switch.

The OCDS registers of SBCU and RBCU are described in [Chapter 6](#) starting in section [“SBCU and RBCU Registers” on Page 6-34](#).

17.2.4 DMA OCDS Level 1

The DMA controller in the TC1796 provides the following debugging capabilities:

- Hard suspend mode of the DMA controller (for test purposes only)
- Soft suspend mode of DMA channels
- Break signal generation

In suspend modes, the operations of DMA channels or the complete DMA module are stopped. Under certain condition conditions also a break signal is generated for the on-chip debug support logic.

More details on the OCDS Level 1 debug capabilities of the DMA controller are described in detail in [Chapter 12](#) in section [“On-Chip Debug Capabilities” on Page 12-23](#).

17.3 OCDS Level 2 Debugging via Trace Port

The OCDS Level 2 debug support extends the OCDS Level 1 debug functionalities with an additional 16-bit wide trace port TR[15:0] with the trace clock TRCLK. The trace port extension makes it possible to output one out of four types of trace output signals:

- TriCore CPU OCDS Level 2 trace
- PCP OCDS Level 2 trace
- DMA Controller channel transaction request trace
- DMA Controller move engine trace

The trace output port is controlled by the OSCU. The trace data is always output at CPU clock speed ($f_{\text{TRCLK}} = f_{\text{CPU}}$).

17.3.1 TriCore CPU and PCP OCDS Level 2 Trace

Every trace clock cycle, 16 bits of CPU/PCP trace information are sent out, representing the current state of the CPU/PCP cores. The trace output lines are grouped into three parts:

- 5 bits of pipeline status information
- 8-bit indirect PC bus information
- 3 bits of breakpoint qualification information

With this information, an external emulator can reconstruct a cycle-by-cycle image of the instruction flow through the CPU or PCP. The trace information can be captured by the external debugger hardware and used to rebuild later on (off-line, using the source code) a cycle accurate disassembly of the code that has been executed. It's also possible to follow in real-time the current PC, facilitating advanced tools such as profilers, coverage analysis tools etc.

17.3.2 DMA OCDS Level 2 Trace

The DMA controller provides two sources for OCDS Level 2 tracing:

- DMA Controller channel transaction request trace
- DMA Controller move engine trace

More details on the OCDS Level 2 debug trace capabilities of the DMA controller are described in detail in section [“Trace Signal Generation” on Page 12-25](#).

17.3.3 Concurrent Debugging

A limited concurrent debugging is possible for CPU and DMA.

It is not possible to trace CPU and DMA at the same time. But when the trace port is assigned to the DMA Controller, the break-in and break-out features of the CPU can be used at the same time. Selecting DMA trace and enabling the break-out lines of both units, makes it possible to distinguish between DMA trace events and CPU break-out activation:

- during DMA trace, the $\overline{\text{BRKOUT}}$ output becomes activated only together with one of the trace port pins being active ('1') as well.
- if $\overline{\text{BRKOUT}}$ output becomes activated with all trace pins being zero, this means that the CPU has activated its break-out signal.

17.4 Debug Interface (Cerberus)

The Cerberus module is the on-chip unit that controls all OCDS Level 1 and 2 main debug functions. Generally, the Cerberus should not be used by any application software, since this could disturb the emulation tool behavior.

The Cerberus module is built up by three parts (see also [Figure 17-1](#)):

- OCDS System Control Unit - OSCU
- JTAG Debug Interface - JDI
- Multi Core Break Switch - MCBS

A standard JTAG interface is connected via the JTAG controller with the JDI. Two pins are available to handle an external break condition. An external debug hardware can access the Cerberus registers and arbitrary memory locations across the System Peripheral (FPI) Bus.

Features

- 5-pin standard JTAG interface for OCDS Level 1 control
- Generation of external break condition via pins BRKIN/BRKOUT
- Full access to the complete SPB/RPB (FPI) Bus address space via JTAG
- No user resources (hardware/software) are required
- Minimum run-time impact
- Generic memory read/write functionality
- Write word, half-word and byte
- Block read and write
- Full support for communication between an on-chip monitor program and the external debugger
- Pending reads/writes can be optionally triggered by the OCDS module (memory tracing)
- Download of programs and data via JTAG
- Control of the OCDS blocks
- Data acquisition

17.4.1 RW Mode

As the name implies, the RW mode is used by a JTAG host to read or write arbitrary memory locations via the JTAG interface. The RW mode needs the FPI Bus master interface of the Cerberus to actively request data reads or data writes.

Data Types Supported

- WORD (32-bit): The default data type; used for single word transfers and block transfers.
- HWORD (16-bit): For reading 16-bit registers without getting an FPI Bus error, a dedicated JTAG instruction is provided (IO_READ_HWORD).

- **BYTE (8-bit):** If the host wants to read a byte, it has to read the associated word or half-word. Then the JTAG host has to extract the needed part by itself.

Writing bytes or half-words is supported with the `IO_WRITE_BYTE` and `IO_WRITE_HWORD` JTAG instructions. With these instructions, the JTAG host must again shift in the full 32-bit word, but only the selected byte or half-word is actually written. Its exact position is defined by the two lowest address bits in register `IOADDR`.

17.4.2 Communication Mode

In Communication Mode, the Cerberus has no access to the FPI Bus and communication is established between the external JTAG host and a software monitor (embedded into the application program) via the Cerberus registers. The communication mode is the default mode after reset.

In Communication Mode, the external JTAG host is master of all transactions. He requests the monitor to write or read a value to/from the Cerberus register `COMDATA`. The difference to RW Mode is, that the read or write request is not actively executed by the Cerberus, but it sets request bits in the CPU accessible `IOSR` register to signal the monitor that the debugger wants to send (`IO_WRITE_WORD`) or receive (`IO_READ_WORD`) a value. The software monitor has to poll register `IOSR`. The `IOADDR` register is not used.

17.4.3 Triggered Transfers

Triggered Transfers are an OCDS-specific feature of the Cerberus. They can be used to read from or write to a certain memory location when an OCDS trigger becomes active. Triggered Transfers behave as normal transfers with some exceptions.

The main application for Triggered Transfers is to trace a certain memory location. This can be done, when the OCDS of the CPU activates its break out signal, if this memory location is written by the user program. This event is used as a transfer trigger through the configuration of the MCBS. Cerberus is configured to read the location on this trigger.

17.4.4 Multi Core Break Switch

In the TC1796 there are two main processor units, the CPU and the PCP2. For debugging purposes, the OCDS run control of one processor unit can break (interrupt) the other processor unit or vice versa. This run control tasks are handled by the MCBS unit which is a part of the Cerberus. [Figure 17-5](#) shows the break signal interfaces of this MCBS unit.

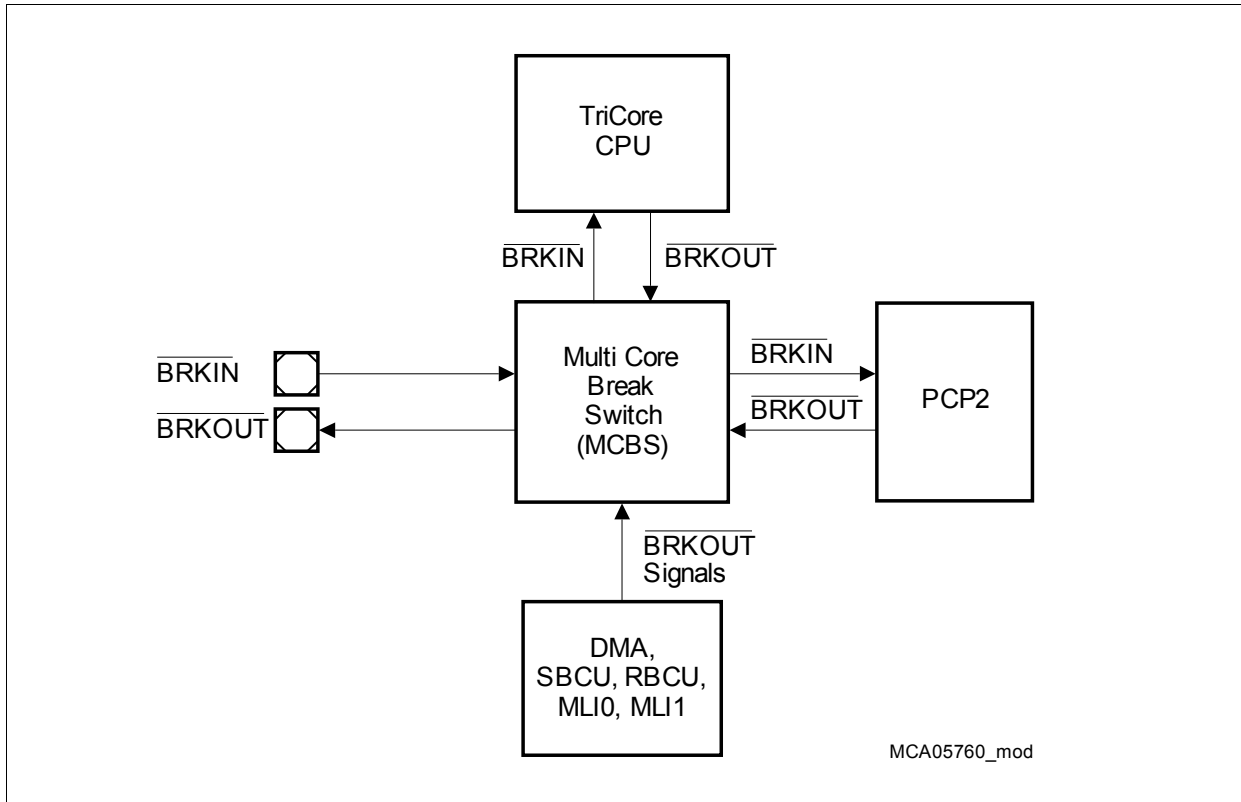


Figure 17-5 Break Switch Interfaces

The MCBS unit supports the following features:

- Two independent break-out master units (TriCore, PCP2)
- Seven break-in sources (TriCore, PCP, DMA, SBCU, RBCU, MLI0, MLI1)
- Two port pins, $\overline{\text{BRKIN}}$ and $\overline{\text{BRKOUT}}$
- Two independent break buses
- Suspend generation supports delayed suspend
- Break-to-suspend converter
- Create interrupt request with a break coming from a source
- Synchronous restart of the system

17.5 JTAG Interface

The JTAG interface is a standardized unit that is typically used for boundary scan and internal device tests. Because both of these applications are not active during normal device operation in a system, the JTAG port can be used during normal device operation as an ideal interface for debugging tasks.

On the other hand, the TC1796 OCDS is designed to support complex multi-core/debugging environments. The challenge here is that several debugger applications may have to share a single resource, i.e. the same JTAG interface. This becomes even more complicated because the JTAG module contains the IEEE 1149.1 JTAG state machine, which needs to be handled in the correct manner.

The solution to this problem is the JTAG driver with its JTAG-API (Application Programming Interface). It allows several debugger applications to share the same JTAG interface. For example, it is possible to run a PCP debugger concurrently with a TriCore debugger on the same TC1796 device. In addition, the tool-specific PC interfaces such as Ethernet, printer-port, or even USB can be hidden from the debugger software by the JTAG-API layer.

The JTAG-API enables the debugger vendor to forget about the complex task of understanding the JTAG module and supporting its functionality at low-level. All required information is provided as specifications and function references, ready for direct implementation into the tool-source, without the need to re-invent the wheel.

The maximum JTAG interface clock frequency (f_{TCK}) is 20 MHz. The following JTAG interface clock frequencies can be achieved:

Table 17-2 Cerberus Performance (Net Data Rates)

Operation	$f_{TCK} = 200 \text{ kHz}$	$f_{TCK} = 10 \text{ MHz}$	$f_{TCK} = 20 \text{ MHz}$
Random Read	48 kbit/s	2.4 Mbit/s	4.6 Mbit/s
Random Write	50 kbit/s	2.5 Mbit/s	4.9 Mbit/s
Block Read	104 kbit/s	5.2 Mbit/s	10.0 Mbit/s
Block Write	114 kbit/s	5.7 Mbit/s	11.2 Mbit/s

17.6 Cerberus and JTAG Registers

This section summarizes all Cerberus and JTAG registers for reference purposes. Details on these registers are contained in OCDS documents that are available for tool suppliers on request (please contact local Infineon representatives).

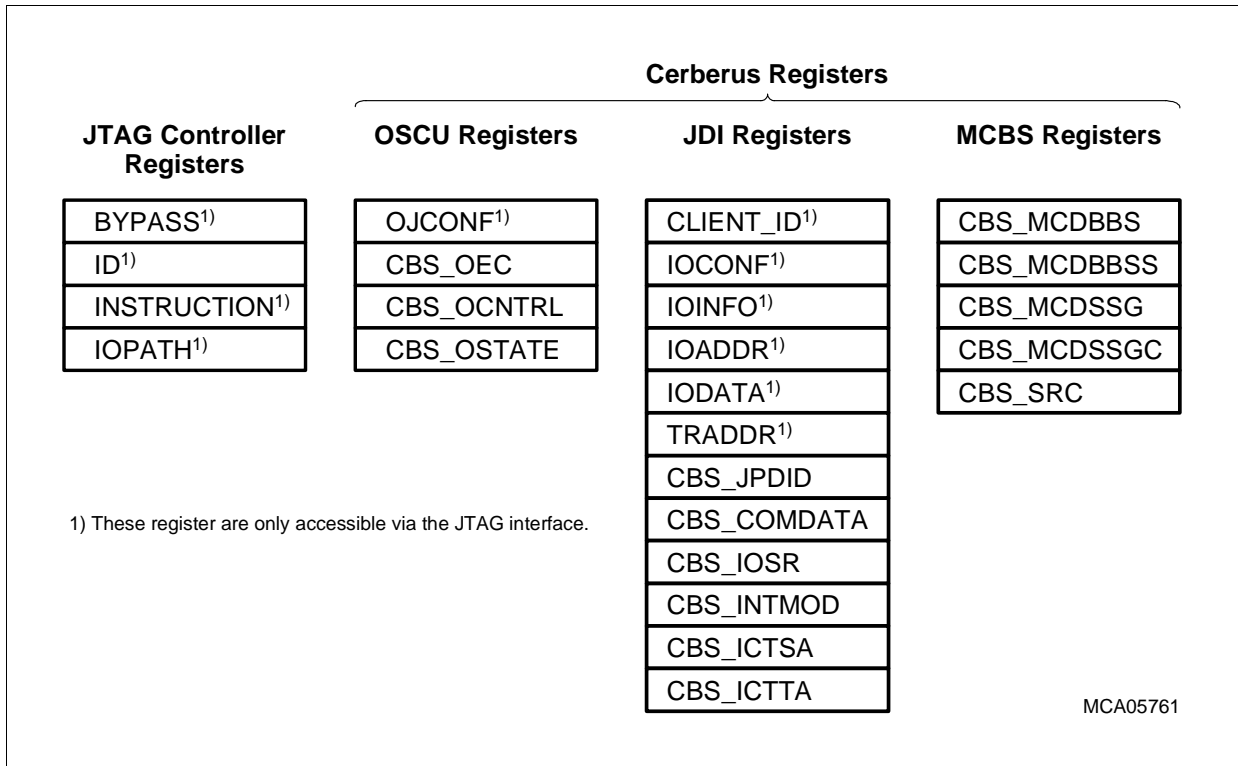


Figure 17-6 JTAG/Cerberus Register Overview

Table 17-3 JTAG/Cerberus Register Overview

Register Short Name	Register Long Name	Address
JTAG Controller Registers		
BYPASS	JTAG Bypass Register (1-bit)	1)
ID	JTAG Module Identification Register (32-bit)	1)
INSTRUCTION	JTAG Instruction Register (8-bit)	1)
IOPATH	IO Client Selection Register (2-bit)	1)
Cerberus Registers		
OJCONF	OSCU Configuration by JTAG Register	1)
CBS_OEC	Cerberus OCDS Enable Control Register	F000 0478 _H
CBS_OCNTRL	Cerberus OSCU Configuration and Control Register	F000 047C _H

Table 17-3 JTAG/Cerberus Register Overview (cont'd)

Register Short Name	Register Long Name	Address
CBS_OSTATE	Cerberus OSCU Status Register	F000 0480 _H
CLIENT_ID	Cerberus JTAG Client Identification Register (32-bit)	1)
IOCONF	Configuration Register (12-bit)	1)
IOINFO	State Information for Error Analysis Register (16-bit)	1)
IOADDR	Address for Data Access Register (32-bit)	1)
IODATA	RW Mode Data Register (32-bit)	1)
TRADDR	External Bus Trace Mode Address Register (8-bit)	1)
CBS_JDPID	Cerberus Module Identification Register	F000 0408 _H
CBS_COMDATA	Cerberus Communication Mode Data Register	F000 0468 _H
CBS_IOSR	Cerberus Status Register	F000 046C _H
CBS_INTMOD	Cerberus Internal Mode Status and Control Register	F000 0484 _H
CBS_ICTSA	Cerberus Internal Controlled Trace Source Address Register	F000 0488 _H
CBS_ICTTA	Cerberus Internal Controlled Trace Target Address Register	F000 048C _H
CBS_MCDBBS	Cerberus Break Bus Switch Configuration Register	F000 0470 _H
CBS_MCDBSS	Cerberus Break Bus Switch Status Register	F000 0490 _H
CBS_MCDSSG	Cerberus Suspend Signal Generation Status and Control Register	F000 0474 _H
CBS_MCDSSGC	Cerberus Suspend Signal Generation Configuration Register	F000 0494 _H
CBS_SRC	Cerberus Service Request Control Register	F000 04FC _H

1) These register are only accessible via the JTAG interface.

18 Register Overview

This chapter describes all registers of the TC1796 that are located in segment 15. It also describes the read/write access rights of the specific address ranges/registers.

Throughout the tables in this chapter, the “Access Mode” “Read” and “Write”, and “Reset Values” columns indicate access rights and values using symbols listed in [Section 18-1](#).

Table 18-1 Address Map Symbols

Symbol	Description
U	Access Mode: Access permitted in User Mode 0 or 1.
	Reset Value: Value or bit is not changed by a reset operation.
SV	Access permitted in Supervisor Mode.
R	Read-only register.
32	Only 32-bit word accesses are permitted to this register/address range.
E	Endinit-protected register/address.
PW	Password protected register/address.
NC	No change, indicated register is not changed.
BE	Indicates that an access to this address range generates a Bus Error.
nBE	Indicates that no Bus Error is generated when accessing this address range, even though it is either an access to an undefined address or the access does not follow the given rules.
nE	Indicates that no Error is generated when accessing this address or address range, even though the access is to an undefined address or address range. True for CPU accesses (MTCR/MFCR) to undefined addresses in the CSFR range.
X	Undefined value or bit.

18.1 Address Map of Segment 15

Table 18-2 shows the block address map of Segment 15.

Table 18-2 Block Address Map of Segment 15

Unit	Address Range	Access Mode		Size
		Read	Write	
System Control Unit (SCU) and Watchdog Timer (WDT)	F000 0000 _H - F000 00FF _H	see Page 18-7		256 byte
System Peripheral Bus Control Unit (SBCU)	F000 0100 _H - F000 01FF _H	see Page 18-11		256 byte
System Timer (STM)	F000 0200 _H - F000 02FF _H	see Page 18-13		256 byte
Reserved	F000 0300 _H - F000 03FF _H	BE	BE	–
On-Chip Debug Support (Cerberus)	F000 0400 _H - F000 04FF _H	see Page 18-14		256 byte
Reserved	F000 0500 _H - F000 07FF _H	BE	BE	–
MicroSecond Channel 0 (MSC0)	F000 0800 _H - F000 08FF _H	see Page 18-16		256 byte
MicroSecond Channel 1 (MSC1)	F000 0900 _H - F000 09FF _H	see Page 18-17		256 byte
Async./Sync. Serial Interface 0 (ASC0)	F000 0A00 _H - F000 0AFF _H	see Page 18-20		256 byte
Async./Sync. Serial Interface 1 (ASC1)	F000 0B00 _H - F000 0BFF _H	see Page 18-21		256 byte
Port 0	F000 0C00 _H - F000 0CFF _H	see Page 18-22		256 byte
Port 1	F000 0D00 _H - F000 0DFF _H	see Page 18-23		256 byte
Port 2	F000 0E00 _H - F000 0EFF _H	see Page 18-24		256 byte
Port 3	F000 0F00 _H - F000 0FFF _H	see Page 18-25		256 byte
Port 4	F000 1000 _H - F000 10FF _H	see Page 18-26		256 byte

Register Overview

Table 18-2 Block Address Map of Segment 15 (cont'd)

Unit		Address Range	Access Mode		Size
			Read	Write	
Port 5		F000 1100 _H - F000 11FF _H	see Page 18-27		256 byte
Port 6		F000 1200 _H - F000 12FF _H	see Page 18-28		256 byte
Port 7		F000 1300 _H - F000 13FF _H	see Page 18-29		256 byte
Port 8		F000 1400 _H - F000 14FF _H	see Page 18-30		256 byte
Port 9		F000 1500 _H - F000 15FF _H	see Page 18-31		256 byte
Port 10		F000 1600 _H - F000 16FF _H	see Page 18-32		256 byte
Reserved		F000 1700 _H - F000 17FF _H	BE	BE	–
General Purpose Timer Array 0 (GPTA0)		F000 1800 _H - F000 1FFF _H	see Page 18-33		8 × 256 byte
General Purpose Timer Array 1 (GPTA1)		F000 2000 _H - F000 27FF _H	see Page 18-42		8 × 256 byte
Local Timer Cell Array 2 (LTCA2)		F000 2800 _H - F000 2FFF _H	see Page 18-50		8 × 256 byte
Reserved		F000 3000 _H - F000 3BFF _H	BE	BE	–
Direct Memory Access Controller (DMA)		F000 3C00 _H - F000 3EFF _H	see Page 18-53		3 × 256 byte
Reserved		F000 3F00 _H - F000 3FFF _H	BE	BE	–
MultiCAN Controller (CAN)		F000 4000 _H - F000 5FFF _H	see Page 18-65		8 Kbyte
Reserved		F000 6000 _H - F003 FFFF _H	BE	BE	–
PCP	Reserved	F004 0000 _H - F004 3EFF _H	BE	BE	–

Register Overview

Table 18-2 Block Address Map of Segment 15 (cont'd)

Unit		Address Range	Access Mode		Size
			Read	Write	
PCP	PCP2 Registers	F004 3F00 _H - F004 3FFF _H	see Page 18-77		256 byte
	Reserved	F004 4000 _H - F004 FFFF _H	BE	BE	–
	PCP Data Memory (PRAM)	F005 0000 _H - F005 3FFF _H	nE, 32	nE, 32	16 Kbyte
	Reserved	F005 4000 _H - F005 FFFF _H	BE	BE	–
	PCP Code Memory (CMEM)	F006 0000 _H - F006 7FFF _H	nE, 32	nE, 32	32 Kbyte
	Reserved	F006 8000 _H - F007 FFFF _H	BE	BE	–
Reserved		F008 0000 _H - F00F FFFF _H	BE	BE	–
Remote Peripheral Bus Control Unit (RBCU)		F010 0000 _H - F010 00FF _H	see Page 18-79		256 byte
Synchronous Serial Interface 0 (SSC0)		F010 0100 _H - F010 01FF _H	see Page 18-81		256 byte
Synchronous Serial Interface 1 (SSC1)		F010 0200 _H - F010 02FF _H	see Page 18-82		256 byte
Fast Analog-to-Digital Converter (FADC)		F010 0300 _H - F010 03FF _H	see Page 18-84		256 byte
Analog-to-Digital Converter 0 (ADC0)		F010 0400 _H - F010 05FF _H	see Page 18-87		2 × 256 byte
Analog-to-Digital Converter 1 (ADC1)		F010 0600 _H - F010 07FF _H	see Page 18-92		2 × 256 byte
Reserved		F010 0800 _H - F010 9FFF _H	BE	BE	–
Dual-Port RAM		F010 A000 _H - F010 BFFF _H	nE	nE	8 Kbyte
Micro Link Interface 0 (MLI0)		F010 C000 _H - F010 C0FF _H	see Page 18-97		256 byte

Register Overview

Table 18-2 Block Address Map of Segment 15 (cont'd)

Unit		Address Range	Access Mode		Size
			Read	Write	
Micro Link Interface 1 (MLI1)		F010 C100 _H - F010 C1FF _H	see Page 18-100		256 byte
Memory Checker (MCHK)		F010 C200 _H - F010 C2FF _H	see Page 18-103		256 byte
Reserved		F010 C300 _H - F01D FFFF _H	BE	BE	–
MLI0 Small Transfer Windows		F01E 0000 _H - F01E 7FFF _H	nE	nE	4 × 8 Kbyte
MLI1 Small Transfer Windows		F01E 8000 _H - F01E FFFF _H	nE	nE	4 × 8 Kbyte
Reserved		F01F 0000 _H - F01F FFFF _H	BE	BE	–
MLI0 Large Transfer Windows		F020 0000 _H - F023FFFF _H	nE	nE	4 × 64 Kbyte
MLI1 Large Transfer Windows		F024 0000 _H - F027 FFFF _H	nE	nE	4 × 64 Kbyte
Reserved		F028 0000 _H - F7E0 FEFF _H	BE	BE	–
CPU	CPU Slave Interface Registers (CPS)	F7E0 FF00 _H - F7E0 FFFF _H	see Page 18-104		256 byte
	CPU Core SFRs & GPRs	F7E1 0000 _H - F7E1 FFFF _H	see Page 18-105		64 Kbyte
Reserved		F7E2 0000 _H - F7FF FFFF _H	BE	BE	–
External Bus Interface Unit (EBU)		F800 0000 _H - F800 03FF _H	see Page 18-112		4 × 256 byte
Reserved		F800 0400 _H - F800 04FF _H	BE	BE	–
Program Memory Unit (PMU)		F800 0500 _H - F800 05FF _H	see Page 18-116		256 byte
Reserved		F800 0600 _H - F800 0FFF _H	BE	BE	–

Register Overview

Table 18-2 Block Address Map of Segment 15 (cont'd)

Unit		Address Range	Access Mode		Size
			Read	Write	
Flash Register		F800 1000 _H - F800 23FF _H	see Page 18-117		5 Kbyte
Reserved		F800 2400 _H - F801 00FF _H	BE	BE	–
Data Memory Unit (DMU)		F801 0100 _H - F801 01FF _H	see Page 18-118		256 byte
Reserved		F801 0200 _H - F87F F9FF _H	BE	BE	–
Data Local Memory Bus Control Unit (DBCUC)		F87F FA00 _H - F87F FAFF _H	see Page 18-122		256 byte
Reserved		F87F FB00 _H - F87F FBFF _H	BE	BE	–
CPU	DMI Registers	F87F FC00 _H - F87F FCFF _H	see Page 18-123		256 byte
	PMI Registers	F87F FD00 _H - F87F FDFF _H	see Page 18-124		256 byte
Program Local Memory Bus Control Unit (PBCUC)		F87F FE00 _H - F87F FEFF _H	see Page 18-125		256 byte
DLMB to SPB Bus Bridge (LFI)		F87F FF00 _H - F87F FFFF _H	see Page 18-126		256 byte
Reserved		F880 0000 _H - FFFF FFFF _H	BE	BE	–

18.2 Registers Tables

Table 18-3 up to **Table 18-42** show the address maps with all register of segment 15.

Note: Addresses listed in columns "Address" are always word (32-bit) addresses.

Table 18-3 Address Map of SCU and WDT

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
System Control Unit (SCU) with Watchdog Timer (WDT)					
–	Reserved	F000 0000 _H - F000 0004 _H	BE	BE	–
SCU_ID	SCU Module Identification Register	F000 0008 _H	U, SV	BE	002C C0XX _H
SCU_SCLKFDR	SCU System Clock Fractional Divider Register	F000 000C _H	U, SV	SV, E	0000 0000 _H
RST_REQ	Reset Request Register	F000 0010 _H	U, SV	U, SV, E	0000 0000 _H
RST_SR	Reset Status Register	F000 0014 _H	U, SV	BE	see Page 4-4
OSC_CON	Oscillator Control Register	F000 0018 _H	U, SV	SV, E	see Page 3-9
–	Reserved	F000 001C _H	BE	BE	–
WDT_CON0	Watchdog Timer Control Register 0	F000 0020 _H	U, SV	U, SV, PW	FFFC 0002 _H
WDT_CON1	Watchdog Timer Control Register 1	F000 0024 _H	U, SV	U, SV, E	0000 0000 _H
WDT_SR	Watchdog Timer Status Register	F000 0028 _H	U, SV	U, SV, NC	FFFC 0010 _H
NMISR	NMI Status Register	F000 002C _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 0030 _H	BE	BE	–
PMG_CSR	Power Management Control and Status Register	F000 0034 _H	U, SV	U, SV	0000 0100 _H

Register Overview

Table 18-3 Address Map of SCU and WDT (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
SCU_SCLR	SCU Software Configuration Latched Inputs Register	F000 0038 _H	U, SV	BE	0000 XXXX _H
–	Reserved	F000 003C _H	BE	BE	–
PLL_CLC	PLL Clock Control Register	F000 0040 _H	U, SV	U, SV, E	see Page 3-18
SCU_EMSR	SCU Emergency Stop Register	F000 0044 _H	U, SV	U, SV, E	0000 0000 _H
SCU_TCCON	SCU Temperature Compensation Control Register	F000 0048 _H	U, SV	U, SV, E	0003 0003 _H
–	Reserved	F000 004C _H	BE	BE	–
SCU_CON	SCU Control Register	F000 0050 _H	U, SV	U, SV, E	FF00 0002 _H
SCU_STAT	SCU Status Register	F000 0054 _H	U, SV	BE	0000 E000 _H
SCU_TCLR0	SCU Temperature Compensation 0 Level Register	F000 0058 _H	U, SV	U, SV, E	02FF FFFF _H
SCU_TCLR1	SCU Temperature Compensation 1 Level Register	F000 005C _H	U, SV	U, SV, E	02FF FFFF _H
–	Reserved; these locations must not be read and written.	F000 0060 _H - F000 0068 _H	–	–	–
–	Reserved	F000 006C _H	BE	BE	–
MANID	Manufacturer Identification Register	F000 0070 _H	U, SV	BE	0000 1820 _H
CHIPID	Chip Identification Register	F000 0074 _H	U, SV	BE	0000 8AXX _H
RTID	Redesign Tracing Identification Register	F000 0078 _H	U, SV	BE	0000 XXXX _H
EICR0	External Input Channel Register 0	F000 0080 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-3 Address Map of SCU and WDT (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
EICR1	External Input Channel Register 1	F000 0084 _H	U, SV	U, SV	0000 0000 _H
EIFR	External Input Flag Register	F000 0088 _H	U, SV	U, SV	0000 0000 _H
FMR	Flag Modification Register	F000 008C _H	U, SV	U, SV	0000 0000 _H
PDRR	Pattern Detection Result Register	F000 0090 _H	U, SV	U, SV	0000 000F _H
IGCR0	Interrupt Gating Register 0	F000 0094 _H	U, SV	U, SV	0000 0000 _H
IGCR1	Interrupt Gating Register 1	F000 0098 _H	U, SV	U, SV	0000 0000 _H
TGADC0	Trigger Gating ADC0 Register	F000 009C _H	U, SV	U, SV	0000 0000 _H
TGADC1	Trigger Gating ADC1 Register	F000 00A0 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 00A4 _H	BE	BE	–
–	Reserved; these locations should not be written.	F000 00A8 _H F000 00AC _H	–	–	–
SCU_PTCON	SCU Pad Test Control Register	F000 00B0 _H	U, SV	SV, E	0000 0000 _H
SCU_PTDAT0	SCU Pad Test Data Register 0	F000 00B4 _H	U, SV	U, SV	XXXX XXXX _H
SCU_PTDAT1	SCU Pad Test Data Register 1	F000 00B8 _H	U, SV	U, SV	XXXX XXXX _H
SCU_PTDAT2	SCU Pad Test Data Register 2	F000 00BC _H	U, SV	U, SV	XXXX 0XXX _H
SCU_PTDAT3	SCU Pad Test Data Register 3	F000 00C0 _H	U, SV	U, SV	XXXX XXXX _H
–	Reserved	F000 00C4 _H - F000 00C8 _H	BE	BE	–
SCU_PETCR	SCU Parity Error Trap Control Register	F000 00D0 _H	U, SV	U, SV, E	0000 0000 _H

Table 18-3 Address Map of SCU and WDT (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
SCU_ PETSr	SCU Parity Error Trap Status Register	F000 00D4 _H	U, SV	BE	0000 0000 _H
–	Reserved	F000 00D8 _H - F000 00F8 _H	BE	BE	–
–	Reserved; this location should not be written.	F000 00FC _H	nBE	nBE	–

Table 18-4 Address Map of SBCU

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
System Peripheral Bus Control Unit (SBCU)					
–	Reserved	F000 0100 _H - F000 0104 _H	BE	BE	–
SBCU_ID	SBCU Module Identification Register	F000 0108 _H	U, SV	BE	0000 6AXX _H
–	Reserved	F000 010C _H	BE	BE	–
SBCU_CON	SBCU Control Register	F000 0110 _H	U, SV	SV	4009 FFFF _H
–	Reserved	F000 0114 _H - F000 011C _H	BE	BE	–
SBCU_ECON	SBCU Error Control Capture Register	F000 0120 _H	U, SV	SV	0000 0000 _H
SBCU_EADD	SBCU Error Address Capture Register	F000 0124 _H	U, SV	SV	0000 0000 _H
SBCU_EDAT	SBCU Error Data Capture Register	F000 0128 _H	U, SV	SV	0000 0000 _H
–	Reserved	F000 012C _H	BE	BE	–
SBCU_DBCNTL	SBCU Debug Control Register	F000 0130 _H	U, SV	SV	0000 7003 _H
SBCU_DBGRNT	SBCU Debug Grant Mask Register	F000 0134 _H	U, SV	SV	0000 FFFF _H
SBCU_DBADR1	SBCU Debug Address 1 Register	F000 0138 _H	U, SV	SV	0000 0000 _H
SBCU_DBADR2	SBCU Debug Address 2 Register	F000 013C _H	U, SV	SV	0000 0000 _H
SBCU_DBBOS	SBCU Debug Bus Operation Signals Register	F000 0140 _H	U, SV	SV	0000 0000 _H
SBCU_DBGNTT	SBCU Debug Trapped Master Register	F000 0144 _H	U, SV	BE	FFFF FFFF _H
SBCU_DBADRT	SBCU Debug Trapped Address Register	F000 0148 _H	U, SV	BE	0000 0000 _H

Table 18-4 Address Map of SBCU (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
SBCU_DBBOST	SBCU Debug Trapped Bus Operation Signals Register	F000 014C _H	U, SV	BE	0000 3180 _H
–	Reserved	F000 0150 _H - F000 01F8 _H	BE	BE	–
SBCU_SRC	SBCU Service Request Control Register	F000 01FC _H	U, SV	SV	0000 0000 _H

Register Overview

Table 18-5 Address Map of STM

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
System Timer (STM)					
STM_CLC	STM Clock Control Register	F000 0200 _H	U, SV	SV, E	0000 0200 _H
–	Reserved	F000 0204 _H	BE	BE	–
STM_ID	STM Module Identification Register	F000 0208 _H	U, SV	BE	0000 C0XX _H
–	Reserved	F000 020C _H	BE	BE	–
STM_TIM0	STM Timer Register 0	F000 0210 _H	U, SV	nBE	0000 0000 _H
STM_TIM1	STM Timer Register 1	F000 0214 _H	U, SV	nBE	0000 0000 _H
STM_TIM2	STM Timer Register 2	F000 0218 _H	U, SV	nBE	0000 0000 _H
STM_TIM3	STM Timer Register 3	F000 021C _H	U, SV	nBE	0000 0000 _H
STM_TIM4	STM Timer Register 4	F000 0220 _H	U, SV	nBE	0000 0000 _H
STM_TIM5	STM Timer Register 5	F000 0224 _H	U, SV	nBE	0000 0000 _H
STM_TIM6	STM Timer Register 6	F000 0228 _H	U, SV	nBE	0000 0000 _H
STM_CAP	STM Timer Capture Reg.	F000 022C _H	U, SV	nBE	0000 0000 _H
STM_CMP0	STM Compare Register 0	F000 0230 _H	U, SV	U, SV	0000 0000 _H
STM_CMP1	STM Compare Register 1	F000 0234 _H	U, SV	U, SV	0000 0000 _H
STM_CMCON	STM Compare Match Control Register	F000 0238 _H	U, SV	U, SV	0000 0000 _H
STM_ICR	STM Interrupt Control Register	F000 023C _H	U, SV	U, SV	0000 0000 _H
STM_ISRR	STM Interrupt Set/Reset Register	F000 0240 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 0244 _H - F000 02F4 _H	BE	BE	–
STM_SRC1	STM Service Request Control Register 1	F000 02F8 _H	U, SV	U, SV	0000 0000 _H
STM_SRC0	STM Service Request Control Register 0	F000 02FC _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-6 Address Map of Cerberus

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
On-Chip Debug Support (Cerberus)					
–	Reserved	F000 0400 _H - F000 0404 _H	BE	BE	–
CBS_JDPID	Cerberus Module Identification Register	F000 0408 _H	U, SV	BE	0000 63XX _H
–	Reserved	F000 040C _H - F000 0464 _H	BE	BE	–
CBS_COMDATA	Cerberus Communication Mode Data Register	F000 0468 _H	U, SV	SV	0000 0000 _H
CBS_IOSR	Cerberus Status Register	F000 046C _H	U, SV	SV	0000 0000 _H
CBS_MCDBBS	Cerberus Break Bus Switch Configuration Register	F000 0470 _H	U, SV	SV	0000 0000 _H
CBS_MCDSSG	Cerberus Suspend Signal Generation Status and Control Register	F000 0474 _H	U, SV	SV	0000 0000 _H
CBS_OEC	Cerberus OCDS Enable Control Register	F000 0478 _H	U, SV	SV	0000 0000 _H
CBS_OCCTRL	Cerberus OSCU Configuration and Control Register	F000 047C _H	U, SV	SV	0000 0000 _H
CBS_OSTATE	Cerberus OSCU Status Register	F000 0480 _H	U, SV	SV	0000 0000 _H
CBS_INTMOD	Cerberus Internal Mode Status and Control Register	F000 0484 _H	U, SV	SV	0000 0000 _H
CBS_ICTSA	Cerberus Internal Controlled Trace Source Address Register	F000 0488 _H	U, SV	SV	0000 0000 _H
CBS_ICTTA	Cerberus Internal Controlled Trace Target Address Register	F000 048C _H	U, SV	SV	0000 0000 _H
CBS_MCDBBSS	Cerberus Break Bus Switch Status Register	F000 0490 _H	U, SV	SV	0000 0000 _H

Register Overview

Table 18-6 Address Map of Cerberus (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
CBS_MCDSSGC	Cerberus Suspend Signal Generation Configuration Register	F000 0494 _H	U, SV	SV	0000 0000 _H
–	Reserved	F000 0498 _H - F000 04F8 _H	BE	BE	–
CBS_SRC	Cerberus Service Request Control Register	F000 04FC _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-7 Address Map of MSC0/MSC1

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Micro Second Channel 0 (MSC0)					
MSC0_CLC	MSC0 Clock Control Register	F000 0800 _H	U, SV	SV, E	0000 0003 _H
–	Reserved	F000 0804 _H	BE	BE	–
MSC0_ID	MSC0 Module Identification Register	F000 0808 _H	U, SV	nBE	0028 C0XX _H
MSC0_FDR	MSC0 Fractional Divider Register	F000 080C _H	U, SV	SV, E	0000 0000 _H
MSC0_USR	MSC0 Upstream Status Register	F000 0810 _H	U, SV	U, SV	0000 0000 _H
MSC0_DSC	MSC0 Downstream Control Register	F000 0814 _H	U, SV	U, SV	0000 0000 _H
MSC0_DSS	MSC0 Downstream Status Register	F000 0818 _H	U, SV	U, SV	0000 0000 _H
MSC0_DD	MSC0 Downstream Data Register	F000 081C _H	U, SV	U, SV	0000 0000 _H
MSC0_DC	MSC0 Downstream Command Register	F000 0820 _H	U, SV	U, SV	0000 0000 _H
MSC0_DSDSL	MSC0 Downstream Select Data Source Low Register	F000 0824 _H	U, SV	U, SV	0000 0000 _H
MSC0_DSDSH	MSC0 Downstream Select Data Source High Register	F000 0828 _H	U, SV	U, SV	0000 0000 _H
MSC0_ESR	MSC0 Emergency Stop Register	F000 082C _H	U, SV	U, SV	0000 0000 _H
MSC0_UD0	MSC0 Upstream Data Register 0	F000 0830 _H	U, SV	U, SV	0000 0000 _H
MSC0_UD1	MSC0 Upstream Data Register 1	F000 0834 _H	U, SV	U, SV	0000 0000 _H
MSC0_UD2	MSC0 Upstream Data Register 2	F000 0838 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-7 Address Map of MSC0/MSC1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
MSC0_UD3	MSC0 Upstream Data Register 3	F000 083C _H	U, SV	U, SV	0000 0000 _H
MSC0_ICR	MSC0 Interrupt Control Register	F000 0840 _H	U, SV	U, SV	0000 0000 _H
MSC0_ISR	MSC0 Interrupt Status Register	F000 0844 _H	U, SV	U, SV	0000 0000 _H
MSC0_ISC	MSC0 Interrupt Set Clear Register	F000 0848 _H	U, SV	U, SV	0000 0000 _H
MSC0_OCR	MSC0 Output Control Register	F000 084C _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 0850 _H - F000 0854 _H	nBE	nBE	–
–	Reserved	F000 0858 _H - F000 08F4 _H	BE	BE	–
MSC0_SRC1	MSC0 Service Request Control Register 1	F000 08F8 _H	U, SV	U, SV	0000 0000 _H
MSC0_SRC0	MSC0 Service Request Control Register 0	F000 08FC _H	U, SV	U, SV	0000 0000 _H
Micro Second Channel 1 (MSC1)					
MSC1_CLC	MSC1 Clock Control Register	F000 0900 _H	U, SV	SV, E	0000 0003 _H
–	Reserved	F000 0904 _H	BE	BE	–
MSC1_ID	MSC1 Module Identification Register	F000 0908 _H	U, SV	nBE	0028 C0XX _H
MSC1_FDR	MSC1 Fractional Divider Register	F000 090C _H	U, SV	SV, E	0000 0000 _H
MSC1_USR	MSC1 Upstream Status Register	F000 0910 _H	U, SV	U, SV	0000 0000 _H
MSC1_DSC	MSC1 Downstream Control Register	F000 0914 _H	U, SV	U, SV	0000 0000 _H
MSC1_DSS	MSC1 Downstream Status Register	F000 0918 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-7 Address Map of MSC0/MSC1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
MSC1_DD	MSC1 Downstream Data Register	F000 091C _H	U, SV	U, SV	0000 0000 _H
MSC1_DC	MSC1 Downstream Command Register	F000 0920 _H	U, SV	U, SV	0000 0000 _H
MSC1_DSDSL	MSC1 Downstream Select Data Source Low Register	F000 0924 _H	U, SV	U, SV	0000 0000 _H
MSC1_DSDSH	MSC1 Downstream Select Data Source High Register	F000 0928 _H	U, SV	U, SV	0000 0000 _H
MSC1_ESR	MSC1 Emergency Stop Register	F000 092C _H	U, SV	U, SV	0000 0000 _H
MSC1_UD0	MSC1 Upstream Data Register 0	F000 0930 _H	U, SV	U, SV	0000 0000 _H
MSC1_UD1	MSC1 Upstream Data Register 1	F000 0934 _H	U, SV	U, SV	0000 0000 _H
MSC1_UD2	MSC1 Upstream Data Register 2	F000 0938 _H	U, SV	U, SV	0000 0000 _H
MSC1_UD3	MSC1 Upstream Data Register 3	F000 093C _H	U, SV	U, SV	0000 0000 _H
MSC1_ICR	MSC1 Interrupt Control Register	F000 0940 _H	U, SV	U, SV	0000 0000 _H
MSC1_ISR	MSC1 Interrupt Status Register	F000 0944 _H	U, SV	U, SV	0000 0000 _H
MSC1_ISC	MSC1 Interrupt Set Clear Register	F000 0948 _H	U, SV	U, SV	0000 0000 _H
MSC1_OCR	MSC1 Output Control Register	F000 094C _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 0950 _H - F000 0954 _H	nBE	nBE	–
–	Reserved	F000 0958 _H - F000 09F4 _H	BE	BE	–

Register Overview

Table 18-7 Address Map of MSC0/MSC1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
MSC1_ SRC1	MSC1 Service Request Control Register 1	F000 09F8 _H	U, SV	U, SV	0000 0000 _H
MSC1_ SRC0	MSC1 Service Request Control Register 0	F000 09FC _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-8 Address Map of ASC0/ASC1

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Asynchronous/Synchronous Serial Interface 0 (ASC0)					
ASC0_CLC	ASC0 Clock Control Register	F000 0A00 _H	U, SV	SV, E	0000 0003 _H
ASC0_PISEL	ASC0 Peripheral Input Select Register	F000 0A04 _H	U, SV	U, SV	0000 0000 _H
ASC0_ID	ASC0 Module Identification Register	F000 0A08 _H	U, SV	BE	0000 44XX _H
–	Reserved	F000 0A0C _H	BE	BE	–
ASC0_CON	ASC0 Control Register	F000 0A10 _H	U, SV	U, SV	0000 0000 _H
ASC0_BG	ASC0 Baud Rate Timer Reload Register	F000 0A14 _H	U, SV	U, SV	0000 0000 _H
ASC0_FDV	ASC0 Fractional Divider Register	F000 0A18 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 0A1C _H	BE	BE	–
ASC0_TBUF	ASC0 Transmit Buffer Register	F000 0A20 _H	U, SV	U, SV	0000 0000 _H
ASC0_RBUF	ASC0 Receive Buffer Register	F000 0A24 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 0A28 _H - F000 0A4C _H	BE	BE	–
ASC0_WHBCON	ASC0 Write Hardware Bits Control Register	F000 0A50 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 0A54 _H - F000 0AEC _H	BE	BE	–
ASC0_TSRC	ASC0 Transmit Interrupt Service Req. Control Reg.	F000 0AF0 _H	U, SV	U, SV	0000 0000 _H
ASC0_RSRC	ASC0 Receive Interrupt Service Req. Control Reg.	F000 0AF4 _H	U, SV	U, SV	0000 0000 _H
ASC0_ESRC	ASC0 Error Interrupt Service Req. Control Reg.	F000 0AF8 _H	U, SV	U, SV	0000 0000 _H
ASC0_TB SRC	ASC0 Transmit Buffer Interrupt Service Request Control Register	F000 0AFC _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-8 Address Map of ASC0/ASC1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Asynchronous/Synchronous Serial Interface 1 (ASC1)					
–	Reserved	F000 0B00 _H	BE	BE	–
ASC1_PISEL	ASC1 Peripheral Input Select Register	F000 0B04 _H	U, SV	U, SV	0000 0000 _H
ASC1_ID	ASC1 Module Identification Register	F000 0B08 _H	U, SV	BE	0000 44XX _H
–	Reserved	F000 0B0C _H	BE	BE	–
ASC1_CON	ASC1 Control Register	F000 0B10 _H	U, SV	U, SV	0000 0000 _H
ASC1_BG	ASC1 Baud Rate Timer Reload Register	F000 0B14 _H	U, SV	U, SV	0000 0000 _H
ASC1_FD	ASC1 Fractional Divider Register	F000 0B18 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 0B1C _H	BE	BE	–
ASC1_TBUF	ASC1 Transmit Buffer Register	F000 0B20 _H	U, SV	U, SV	0000 0000 _H
ASC1_RBUF	ASC1 Receive Buffer Register	F000 0B24 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 0B28 _H - F000 0B4C _H	BE	BE	–
ASC1_WHBCON	ASC1 Write Hardware Bits Control Register	F000 0B50 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 0B54 _H - F000 0BEC _H	BE	BE	–
ASC1_TSRC	ASC1 Transmit Interrupt Service Req. Control Reg.	F000 0BF0 _H	U, SV	U, SV	0000 0000 _H
ASC1_RSRC	ASC1 Receive Interrupt Service Req. Control Reg.	F000 0BF4 _H	U, SV	U, SV	0000 0000 _H
ASC1_ESRC	ASC1 Error Interrupt Service Req. Control Reg.	F000 0BF8 _H	U, SV	U, SV	0000 0000 _H
ASC1_TBSRC	ASC1 Transmit Buffer Interrupt Service Request Control Register	F000 0BFC _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-9 Address Map of Port 0

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Port 0					
P0_OUT	Port 0 Output Register	F000 0C00 _H	U, SV	U, SV	0000 0000 _H
P0_OMR	Port 0 Output Modification Register	F000 0C04 _H	U, SV	U, SV, 32	0000 XXXX _H
–	Reserved	F000 0C08 _H - F000 0C0C _H	U, SV	U, SV	–
P0_IOCRO	Port 0 Input/Output Control Register 0	F000 0C10 _H	U, SV	U, SV	2020 2020 _H
P0_IOCRA	Port 0 Input/Output Control Register 4	F000 0C14 _H	U, SV	U, SV	2020 2020 _H
P0_IOCRA8	Port 0 Input/Output Control Register 8	F000 0C18 _H	U, SV	U, SV	2020 2020 _H
P0_IOCRA12	Port 0 Input/Output Control Register 12	F000 0C1C _H	U, SV	U, SV	2020 2020 _H
–	Reserved	F000 0C20 _H	U, SV	U, SV	–
P0_IN	Port 0 Input Register	F000 0C24 _H	U, SV	U, SV	0000 XXXX _H
–	Reserved	F000 0C28 _H - F000 0C3C _H	U, SV	U, SV	–
P0_PDR	Port 0 Pad Driver Mode Register	F000 0C40 _H	U, SV	SV, E	0000 0000 _H
–	Reserved	F000 0C44 _H - F000 0CFC _H	U, SV	U, SV	–

Register Overview

Table 18-10 Address Map of Port 1

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Port 1					
P1_OUT	Port 1 Output Register	F000 0D00 _H	U, SV	U, SV	0000 0000 _H
P1_OMR	Port 1 Output Modification Register	F000 0D04 _H	U, SV	U, SV, 32	0000 XXXX _H
–	Reserved	F000 0D08 _H - F000 0D0C _H	U, SV	U, SV	–
P1_IOCRO	Port 1 Input/Output Control Register 0	F000 0D10 _H	U, SV	U, SV	2020 2020 _H
P1_IOCRA	Port 1 Input/Output Control Register 4	F000 0D14 _H	U, SV	U, SV	2020 2020 _H
P1_IOCRA8	Port 1 Input/Output Control Register 8	F000 0D18 _H	U, SV	U, SV	2020 2020 _H
P1_IOCRA12	Port 1 Input/Output Control Register 12	F000 0D1C _H	U, SV	U, SV	2020 2020 _H
–	Reserved	F000 0D20 _H	U, SV	U, SV	–
P1_IN	Port 1 Input Register	F000 0D24 _H	U, SV	U, SV	0000 XXXX _H
–	Reserved	F000 0D28 _H - F000 0D3C _H	U, SV	U, SV	–
P1_PDR	Port 1 Pad Driver Mode Register	F000 0D40 _H	U, SV	SV, E	0000 0000 _H
–	Reserved	F000 0D44 _H - F000 0DFC _H	U, SV	U, SV	–

Register Overview

Table 18-11 Address Map of Port 2

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Port 2					
P2_OUT	Port 2 Output Register	F000 0E00 _H	U, SV	U, SV	0000 0000 _H
P2_OMR	Port 2 Output Modification Register	F000 0E04 _H	U, SV	U, SV, 32	0000 XXXX _H
–	Reserved	F000 0E08 _H - F000 0E0C _H	U, SV	U, SV	–
P2_IOCRO	Port 2 Input/Output Control Register 0	F000 0E10 _H	U, SV	U, SV	2020 2020 _H
P2_IOCRA	Port 2 Input/Output Control Register 4	F000 0E14 _H	U, SV	U, SV	2020 2020 _H
P2_IOCRC	Port 2 Input/Output Control Register 8	F000 0E18 _H	U, SV	U, SV	2020 2020 _H
P2_IOCR12	Port 2 Input/Output Control Register 12	F000 0E1C _H	U, SV	U, SV	2020 2020 _H
–	Reserved	F000 0E20 _H	U, SV	U, SV	–
P2_IN	Port 2 Input Register	F000 0E24 _H	U, SV	U, SV	0000 XXXX _H
–	Reserved	F000 0E28 _H - F000 0E3C _H	U, SV	U, SV	–
P2_PDR	Port 2 Pad Driver Mode Register	F000 0E40 _H	U, SV	SV, E	0000 0000 _H
–	Reserved	F000 0E44 _H - F000 0E4C _H	U, SV	U, SV	–
P2_ESR	Port 2 Emergency Stop Register	F000 0E50 _H	U, SV	SV, E	0000 0000 _H
–	Reserved	F000 0E54 _H - F000 0EFC _H	U, SV	U, SV	–

Register Overview

Table 18-12 Address Map of Port 3

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Port 3					
P3_OUT	Port 3 Output Register	F000 0F00 _H	U, SV	U, SV	0000 0000 _H
P3_OMR	Port 3 Output Modification Register	F000 0F04 _H	U, SV	U, SV, 32	0000 XXXX _H
–	Reserved	F000 0F08 _H - F000 0F0C _H	U, SV	U, SV	–
P3_IOCRO	Port 3 Input/Output Control Register 0	F000 0F10 _H	U, SV	U, SV	2020 2020 _H
P3_IOCRA	Port 3 Input/Output Control Register 4	F000 0F14 _H	U, SV	U, SV	2020 2020 _H
P3_IOCRC	Port 3 Input/Output Control Register 8	F000 0F18 _H	U, SV	U, SV	2020 2020 _H
P3_IOCR12	Port 3 Input/Output Control Register 12	F000 0F1C _H	U, SV	U, SV	2020 2020 _H
–	Reserved	F000 0F20 _H	U, SV	U, SV	–
P3_IN	Port 3 Input Register	F000 0F24 _H	U, SV	U, SV	0000 XXXX _H
–	Reserved	F000 0F28 _H - F000 0F3C _H	U, SV	U, SV	–
P3_PDR	Port 3 Pad Driver Mode Register	F000 0F40 _H	U, SV	SV, E	0000 0000 _H
–	Reserved	F000 0F44 _H - F000 0F4C _H	U, SV	U, SV	–
P3_ESR	Port 3 Emergency Stop Register	F000 0F50 _H	U, SV	SV, E	0000 0000 _H
–	Reserved	F000 0F54 _H - F000 0FFC _H	U, SV	U, SV	–

Register Overview

Table 18-13 Address Map of Port 4

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Port 4					
P4_OUT	Port 4 Output Register	F000 1000 _H	U, SV	U, SV	0000 0000 _H
P4_OMR	Port 4 Output Modification Register	F000 1004 _H	U, SV	U, SV, 32	0000 XXXX _H
–	Reserved	F000 1008 _H - F000 100C _H	U, SV	U, SV	–
P4_IOCRO	Port 4 Input/Output Control Register 0	F000 1010 _H	U, SV	U, SV	2020 2020 _H
P4_IOCRA	Port 4 Input/Output Control Register 4	F000 1014 _H	U, SV	U, SV	2020 2020 _H
P4_IOCRC	Port 4 Input/Output Control Register 8	F000 1018 _H	U, SV	U, SV	2020 2020 _H
P4_IOCR12	Port 4 Input/Output Control Register 12	F000 101C _H	U, SV	U, SV	2020 2020 _H
–	Reserved	F000 1020 _H	U, SV	U, SV	–
P4_IN	Port 4 Input Register	F000 1024 _H	U, SV	U, SV	0000 XXXX _H
–	Reserved	F000 1028 _H - F000 103C _H	U, SV	U, SV	–
P4_PDR	Port 4 Pad Driver Mode Register	F000 1040 _H	U, SV	SV, E	0000 0000 _H
–	Reserved	F000 1044 _H - F000 104C _H	U, SV	U, SV	–
P4_ESR	Port 4 Emergency Stop Register	F000 1050 _H	U, SV	SV, E	0000 0000 _H
–	Reserved	F000 1054 _H - F000 10FC _H	U, SV	U, SV	–

Register Overview

Table 18-14 Address Map of Port 5

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Port 5					
P5_OUT	Port 5 Output Register	F000 1100 _H	U, SV	U, SV	0000 0000 _H
P5_OMR	Port 5 Output Modification Register	F000 1104 _H	U, SV	U, SV, 32	0000 00XX _H
–	Reserved	F000 1108 _H - F000 110C _H	U, SV	U, SV	–
P5_IOCRO	Port 5 Input/Output Control Register 0	F000 1110 _H	U, SV	U, SV	2020 2020 _H
P5_IOCRA	Port 5 Input/Output Control Register 4	F000 1114 _H	U, SV	U, SV	2020 2020 _H
–	Reserved	F000 1118 _H - F000 1120 _H	U, SV	U, SV	–
P5_IN	Port 5 Input Register	F000 1124 _H	U, SV	U, SV	0000 00XX _H
–	Reserved	F000 1128 _H - F000 113C _H	U, SV	U, SV	–
P5_PDR	Port 5 Pad Driver Mode Register	F000 1140 _H	U, SV	SV, E	0000 0000 _H
–	Reserved	F000 1144 _H - F000 11FC _H	U, SV	U, SV	–

Register Overview

Table 18-15 Address Map of Port 6

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Port 6					
P6_OUT	Port 6 Output Register	F000 1200 _H	U, SV	U, SV	0000 0000 _H
P6_OMR	Port 6 Output Modification Register	F000 1204 _H	U, SV	U, SV, 32	0000 XXXX _H
–	Reserved	F000 1208 _H - F000 1210 _H	U, SV	U, SV	–
P6_IOCR4	Port 6 Input/Output Control Register 4	F000 1214 _H	U, SV	U, SV	2020 2020 _H
P6_IOCR8	Port 6 Input/Output Control Register 8	F000 1218 _H	U, SV	U, SV	2020 2020 _H
P6_IOCR12	Port 6 Input/Output Control Register 12	F000 121C _H	U, SV	U, SV	2020 2020 _H
–	Reserved	F000 1220 _H	U, SV	U, SV	–
P6_IN	Port 6 Input Register	F000 1224 _H	U, SV	U, SV	0000 XXXX _H
–	Reserved	F000 1228 _H - F000 123C _H	U, SV	U, SV	–
P6_PDR	Port 6 Pad Driver Mode Register	F000 1240 _H	U, SV	SV, E	0000 0000 _H
–	Reserved	F000 1244 _H - F000 12FC _H	U, SV	U, SV	–

Register Overview

Table 18-16 Address Map of Port 7

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Port 7					
P7_OUT	Port 7 Output Register	F000 1300 _H	U, SV	U, SV	0000 0000 _H
P7_OMR	Port 7 Output Modification Register	F000 1304 _H	U, SV	U, SV, 32	0000 00XX _H
–	Reserved	F000 1308 _H - F000 130C _H	U, SV	U, SV	–
P7_IOCRO	Port 7 Input/Output Control Register 0	F000 1310 _H	U, SV	U, SV	2020 2020 _H
P7_IOCRA	Port 7 Input/Output Control Register 4	F000 1314 _H	U, SV	U, SV	2020 2020 _H
–	Reserved	F000 1318 _H - F000 1320 _H	U, SV	U, SV	–
P7_IN	Port 7 Input Register	F000 1324 _H	U, SV	U, SV	0000 00XX _H
–	Reserved	F000 1328 _H - F000 133C _H	U, SV	U, SV	–
P7_PDR	Port 7 Pad Driver Mode Register	F000 1340 _H	U, SV	SV, E	0000 0000 _H
–	Reserved	F000 1344 _H - F000 13FC _H	U, SV	U, SV	–

Register Overview

Table 18-17 Address Map of Port 8

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Port 8					
P8_OUT	Port 8 Output Register	F000 1400 _H	U, SV	U, SV	0000 0000 _H
P8_OMR	Port 8 Output Modification Register	F000 1404 _H	U, SV	U, SV, 32	0000 00XX _H
–	Reserved	F000 1408 _H - F000 140C _H	U, SV	U, SV	–
P8_IOCRO	Port 8 Input/Output Control Register 0	F000 1410 _H	U, SV	U, SV	2020 2020 _H
P8_IOCRA	Port 8 Input/Output Control Register 4	F000 1414 _H	U, SV	U, SV	2020 2020 _H
–	Reserved	F000 1418 _H - F000 1420 _H	U, SV	U, SV	–
P8_IN	Port 8 Input Register	F000 1424 _H	U, SV	U, SV	0000 00XX _H
–	Reserved	F000 1428 _H - F000 143C _H	U, SV	U, SV	–
P8_PDR	Port 8 Pad Driver Mode Register	F000 1440 _H	U, SV	SV, E	0000 0000 _H
–	Reserved	F000 1444 _H - F000 144C _H	U, SV	U, SV	–
P8_ESR	Port 8 Emergency Stop Register	F000 1450 _H	U, SV	SV, E	0000 0000 _H
–	Reserved	F000 1454 _H - F000 14FC _H	U, SV	U, SV	–

Register Overview

Table 18-18 Address Map of Port 9

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Port 9					
P9_OUT	Port 9 Output Register	F000 1500 _H	U, SV	U, SV	0000 0000 _H
P9_OMR	Port 9 Output Modification Register	F000 1504 _H	U, SV	U, SV, 32	0000 0XXX _H
–	Reserved	F000 1508 _H - F000 150C _H	U, SV	U, SV	–
P9_IOCRO	Port 9 Input/Output Control Register 0	F000 1510 _H	U, SV	U, SV	2020 2020 _H
P9_IOCRA	Port 9 Input/Output Control Register 4	F000 1514 _H	U, SV	U, SV	2020 2020 _H
P9_IOCRA8	Port 9 Input/Output Control Register 8	F000 1518 _H	U, SV	U, SV	0000 0020 _H
–	Reserved	F000 151C _H - F000 1520 _H	U, SV	U, SV	–
P9_IN	Port 9 Input Register	F000 1524 _H	U, SV	U, SV	0000 0XXX _H
–	Reserved	F000 1528 _H - F000 153C _H	U, SV	U, SV	–
P9_PDR	Port 9 Pad Driver Mode Register	F000 1540 _H	U, SV	SV, E	0000 0000 _H
–	Reserved	F000 1544 _H - F000 154C _H	U, SV	U, SV	–
P9_ESR	Port 9 Emergency Stop Register	F000 1550 _H	U, SV	SV, E	0000 0000 _H
–	Reserved	F000 1554 _H - F000 15FC _H	U, SV	U, SV	–

Table 18-19 Address Map of Port 10

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Port 10					
–	Reserved	F000 1600 _H - F000 1620 _H	U, SV	U, SV	–
P10_IN	Port 10 Input Register	F000 1624 _H	U, SV	U, SV	0000 000X _H
–	Reserved	F000 1628 _H - F000 16FC _H	U, SV	U, SV	–

Table 18-20 Address Map of GPTA0

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
General Purpose Timer Array 0 (GPTA0)					
GPTA0_CLC	GPTA Clock Control Register	F000 1800 _H	U, SV	SV, E	0000 0003 _H
GPTA0_DBGCTR	GPTA Debug Clock Control Register	F000 1804 _H	U, SV	U, SV	0000 0000 _H
GPTA0_ID	GPTA0 Module Identification Register	F000 1808 _H	U, SV	nBE	0029 C0XX _H
GPTA0_FDR	GPTA Fractional Divider Register	F000 180C _H	U, SV	SV, E	0000 0000 _H
GPTA0_SRSC0	GPTA0 Service Request State Clear Register 0	F000 1810 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRSS0	GPTA0 Service Request State Set Register 0	F000 1814 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRSC1	GPTA0 Service Request State Clear Register 1	F000 1818 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRSS1	GPTA0 Service Request State Set Register 1	F000 181C _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRSC2	GPTA0 Service Request State Clear Register 2	F000 1820 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRSS2	GPTA0 Service Request State Set Register 2	F000 1824 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRSC3	GPTA0 Service Request State Clear Register 3	F000 1828 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRSS3	GPTA0 Service Request State Set Register 3	F000 182C _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRRR	GPTA0 Service Request Node Redirection Register	F000 1830 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 1834 _H	nBE	nBE	0000 0000 _H
GPTA0_MRACTL	GPTA0 Multiplexer Register Array Control Register	F000 1838 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-20 Address Map of GPTA0 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
GPTA0_MRADIN	GPTA0 Multiplexer Register Array Data In Register	F000 183C _H	U, SV, 32	U, SV, 32	0000 0000 _H
GPTA0_MRADOUT	GPTA0 Multiplexer Register Array Data Out Register	F000 1840 _H	U, SV, 32	U, SV, 32	0000 0000 _H
GPTA0_FPCSTAT	GPTA0 Filter and Prescaler Cell Status Register	F000 1844 _H	U, SV	U, SV	0000 0000 _H
GPTA0_FPCCTR0	GPTA0 Filter and Prescaler Cell Control Register 0	F000 1848 _H	U, SV	U, SV	0000 0000 _H
GPTA0_FPCTIM0	GPTA0 Filter and Prescaler Cell Timer Register 0	F000 184C _H	U, SV	U, SV	0000 0000 _H
GPTA0_FPCCTR1	GPTA0 Filter and Prescaler Cell Control Register 1	F000 1850 _H	U, SV	U, SV	0000 0000 _H
GPTA0_FPCTIM1	GPTA0 Filter and Prescaler Cell Timer Register 1	F000 1854 _H	U, SV	U, SV	0000 0000 _H
GPTA0_FPCCTR2	GPTA0 Filter and Prescaler Cell Control Register 2	F000 1858 _H	U, SV	U, SV	0000 0000 _H
GPTA0_FPCTIM2	GPTA0 Filter and Prescaler Cell Timer Register 2	F000 185C _H	U, SV	U, SV	0000 0000 _H
GPTA0_FPCCTR3	GPTA0 Filter and Prescaler Cell Control Register 3	F000 1860 _H	U, SV	U, SV	0000 0000 _H
GPTA0_FPCTIM3	GPTA0 Filter and Prescaler Cell Timer Register 3	F000 1864 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-20 Address Map of GPTA0 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
GPTA0_FPCCTR4	GPTA0 Filter and Prescaler Cell Control Register 4	F000 1868 _H	U, SV	U, SV	0000 0000 _H
GPTA0_FPCTIM4	GPTA0 Filter and Prescaler Cell Timer Register 4	F000 186C _H	U, SV	U, SV	0000 0000 _H
GPTA0_FPCCTR5	GPTA0 Filter and Prescaler Cell Control Register 5	F000 1870 _H	U, SV	U, SV	0000 0000 _H
GPTA0_FPCTIM5	GPTA0 Filter and Prescaler Cell Timer Register 5	F000 1874 _H	U, SV	U, SV	0000 0000 _H
GPTA0_PDLCTR	GPTA0 Phase Discrimination Logic Control Register	F000 1878 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 187C _H	nBE	nBE	–
GPTA0_DCMCTR0	GPTA0 Duty Cycle Measurement Control Register 0	F000 1880 _H	U, SV	U, SV	0000 0000 _H
GPTA0_DCMTIM0	GPTA0 Duty Cycle Measurement Timer Register 0	F000 1884 _H	U, SV	U, SV	0000 0000 _H
GPTA0_DCMCAV0	GPTA0 Duty Cycle Measurement Capture Register 0	F000 1888 _H	U, SV	U, SV	0000 0000 _H
GPTA0_DCMCOV0	GPTA0 Duty Cycle Measurement Capture/Compare Register 0	F000 188C _H	U, SV	U, SV	0000 0000 _H
GPTA0_DCMCTR1	GPTA0 Duty Cycle Measurement Control Register 1	F000 1890 _H	U, SV	U, SV	0000 0000 _H
GPTA0_DCMTIM1	GPTA0 Duty Cycle Measurement Timer Register 1	F000 1894 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-20 Address Map of GPTA0 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
GPTA0_DCMCAV1	GPTA0 Duty Cycle Measurement Capture Register 1	F000 1898 _H	U, SV	U, SV	0000 0000 _H
GPTA0_DCMCOV1	GPTA0 Duty Cycle Measurement Capture/ Compare Register 1	F000 189C _H	U, SV	U, SV	0000 0000 _H
GPTA0_DCMCTR2	GPTA0 Duty Cycle Measurement Control Register 2	F000 18A0 _H	U, SV	U, SV	0000 0000 _H
GPTA0_DCMTIM2	GPTA0 Duty Cycle Measurement Timer Register 2	F000 18A4 _H	U, SV	U, SV	0000 0000 _H
GPTA0_DCMCAV2	GPTA0 Duty Cycle Measurement Capture Register 2	F000 18A8 _H	U, SV	U, SV	0000 0000 _H
GPTA0_DCMCOV2	GPTA0 Duty Cycle Measurement Capture/ Compare Register 2	F000 18AC _H	U, SV	U, SV	0000 0000 _H
GPTA0_DCMCTR3	GPTA0 Duty Cycle Measurement Control Register 3	F000 18B0 _H	U, SV	U, SV	0000 0000 _H
GPTA0_DCMTIM3	GPTA0 Duty Cycle Measurement Timer Register 3	F000 18B4 _H	U, SV	U, SV	0000 0000 _H
GPTA0_DCMCAV3	GPTA0 Duty Cycle Measurement Capture Register 3	F000 18B8 _H	U, SV	U, SV	0000 0000 _H
GPTA0_DCMCOV3	GPTA0 Duty Cycle Measurement Capture/ Compare Register 3	F000 18BC _H	U, SV	U, SV	0000 0000 _H
GPTA0_PLLCTR	GPTA0 Phase Locked Loop Control Register	F000 18C0 _H	U, SV	U, SV	0000 0000 _H
GPTA0_PLLMTI	GPTA0 Phase Locked Loop Microtick Register	F000 18C4 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-20 Address Map of GPTA0 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
GPTA0_PLLCNT	GPTA0 Phase Locked Loop Counter Register	F000 18C8 _H	U, SV	U, SV	0000 0000 _H
GPTA0_PLLSTP	GPTA0 Phase Locked Loop Step Register	F000 18CC _H	U, SV	U, SV	0000 0000 _H
GPTA0_PLLREV	GPTA0 Phase Locked Loop Reload Register	F000 18D0 _H	U, SV	U, SV	0000 0000 _H
GPTA0_PLLDTR	GPTA0 Phase Locked Loop Delta Register	F000 18D4 _H	U, SV	U, SV	0000 0000 _H
GPTA0_CKBCTR	GPTA0 Clock Bus Control Register	F000 18D8 _H	U, SV	U, SV	0000 FFFF _H
–	Reserved	F000 18DC _H	nBE	nBE	–
GPTA0_GTCTR0	GPTA0 Global Timer Control Register 0	F000 18E0 _H	U, SV	U, SV	0000 0000 _H
GPTA0_GTREV0	GPTA0 Global Timer Reload Value Register 0	F000 18E4 _H	U, SV	U, SV	0000 0000 _H
GPTA0_GTTIM0	GPTA0 Global Timer Register 0	F000 18E8 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 18EC _H	nBE	nBE	–
GPTA0_GTCTR1	GPTA0 Global Timer Control Register 1	F000 18F0 _H	U, SV	U, SV	0000 0000 _H
GPTA0_GTREV1	GPTA0 Global Timer Reload Value Register 1	F000 18F4 _H	U, SV	U, SV	0000 0000 _H
GPTA0_GTTIM1	GPTA0 Global Timer Register 1	F000 18F8 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 18FC _H	nBE	nBE	–
GPTA0_GTCCTRn	GPTA0 Global Timer Cell Control Register n (n = 00-31)	F000 1900 _H + n × 08 _H + 00 _H	U, SV	U, SV	0000 0000 _H
GPTA0_GTCXRn	GPTA0 Global Timer Cell X Register n (n = 00-31)	F000 1900 _H + n × 08 _H + 04 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-20 Address Map of GPTA0 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
GPTA0_LTCCTRn	GPTA0 Local Timer Cell Control Register n (n = 00-63)	F000 1A00 _H + n × 08 _H + 00 _H	U, SV	U, SV	0000 0000 _H
GPTA0_LTCXRn	GPTA0 Local Timer Cell X Register n (n = 00-63)	F000 1A00 _H + n × 08 _H + 04 _H	U, SV	U, SV	0000 0000 _H
GPTA0_EDCTR	GPTA Clock Enable/Disable Control Register	F000 1C00 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 1C04 _H - F000 1EFC _H	nBE	nBE	–
GPTA0_MMXCTR00	GPTA-to-MSC Multiplexer Control Register 00	F000 1F00 _H	U, SV	U, SV	0000 0000 _H
GPTA0_MMXCTR01	GPTA-to-MSC Multiplexer Control Register 01	F000 1F04 _H	U, SV	U, SV	0000 0000 _H
GPTA0_MMXCTR10	GPTA-to-MSC Multiplexer Control Register 10	F000 1F08 _H	U, SV	U, SV	0000 0000 _H
GPTA0_MMXCTR11	GPTA-to-MSC Multiplexer Control Register 11	F000 1F0C _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 1F10 _H - F000 1F64 _H	nBE	nBE	–
GPTA0_SRC37	GPTA0 Service Request Control Register 37	F000 1F68 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC36	GPTA0 Service Request Control Register 36	F000 1F6C _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC35	GPTA0 Service Request Control Register 35	F000 1F70 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC34	GPTA0 Service Request Control Register 34	F000 1F74 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC33	GPTA0 Service Request Control Register 33	F000 1F78 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-20 Address Map of GPTA0 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
GPTA0_SRC32	GPTA0 Service Request Control Register 32	F000 1F7C _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC31	GPTA0 Service Request Control Register 31	F000 1F80 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC30	GPTA0 Service Request Control Register 30	F000 1F84 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC29	GPTA0 Service Request Control Register 29	F000 1F88 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC28	GPTA0 Service Request Control Register 28	F000 1F8C _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC27	GPTA0 Service Request Control Register 27	F000 1F90 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC26	GPTA0 Service Request Control Register 26	F000 1F94 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC25	GPTA0 Service Request Control Register 25	F000 1F98 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC24	GPTA0 Service Request Control Register 24	F000 1F9C _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC23	GPTA0 Service Request Control Register 23	F000 1FA0 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC22	GPTA0 Service Request Control Register 22	F000 1FA4 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC21	GPTA0 Service Request Control Register 21	F000 1FA8 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC20	GPTA0 Service Request Control Register 20	F000 1FAC _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC19	GPTA0 Service Request Control Register 19	F000 1FB0 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC18	GPTA0 Service Request Control Register 18	F000 1FB4 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC17	GPTA0 Service Request Control Register 17	F000 1FB8 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-20 Address Map of GPTA0 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
GPTA0_SRC16	GPTA0 Service Request Control Register 16	F000 1FBC _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC15	GPTA0 Service Request Control Register 15	F000 1FC0 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC14	GPTA0 Service Request Control Register 14	F000 1FC4 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC13	GPTA0 Service Request Control Register 13	F000 1FC8 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC12	GPTA0 Service Request Control Register 12	F000 1FCC _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC11	GPTA0 Service Request Control Register 11	F000 1FD0 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC10	GPTA0 Service Request Control Register 10	F000 1FD4 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC09	GPTA0 Service Request Control Register 09	F000 1FD8 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC08	GPTA0 Service Request Control Register 08	F000 1FDC _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC07	GPTA0 Service Request Control Register 07	F000 1FE0 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC06	GPTA0 Service Request Control Register 06	F000 1FE4 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC05	GPTA0 Service Request Control Register 05	F000 1FE8 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC04	GPTA0 Service Request Control Register 04	F000 1FEC _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC03	GPTA0 Service Request Control Register 03	F000 1FF0 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC02	GPTA0 Service Request Control Register 02	F000 1FF4 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-20 Address Map of GPTA0 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
GPTA0_SRC01	GPTA0 Service Request Control Register 01	F000 1FF8 _H	U, SV	U, SV	0000 0000 _H
GPTA0_SRC00	GPTA0 Service Request Control Register 00	F000 1FFC _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-21 Address Map of GPTA1

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
General Purpose Timer Array 1 (GPTA1)					
–	Reserved	F000 2000 _H - F000 2004 _H	nBE	nBE	–
GPTA1_ID	GPTA1 Module Identification Register	F000 2008 _H	U, SV	nBE	0029 C0XX _H
–	Reserved	F000 200C _H	nBE	nBE	–
GPTA1_SRSC0	GPTA1 Service Request State Clear Register 0	F000 2010 _H	U, SV	U, SV	0000 0000 _H
GPTA1_SRSS0	GPTA1 Service Request State Set Register 0	F000 2014 _H	U, SV	U, SV	0000 0000 _H
GPTA1_SRSC1	GPTA1 Service Request State Clear Register 1	F000 2018 _H	U, SV	U, SV	0000 0000 _H
GPTA1_SRSS1	GPTA1 Service Request State Set Register 1	F000 201C _H	U, SV	U, SV	0000 0000 _H
GPTA1_SRSC2	GPTA1 Service Request State Clear Register 2	F000 2020 _H	U, SV	U, SV	0000 0000 _H
GPTA1_SRSS2	GPTA1 Service Request State Set Register 2	F000 2024 _H	U, SV	U, SV	0000 0000 _H
GPTA1_SRSC3	GPTA1 Service Request State Clear Register 3	F000 2028 _H	U, SV	U, SV	0000 0000 _H
GPTA1_SRSS3	GPTA1 Service Request State Set Register 3	F000 202C _H	U, SV	U, SV	0000 0000 _H
GPTA1_SRNR	GPTA1 Service Request Node Redirection Register	F000 2030 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 2034 _H	nBE	nBE	0000 0000 _H
GPTA1_MRACTL	GPTA1 Multiplexer Register Array Control Register	F000 2038 _H	U, SV	U, SV	0000 0000 _H
GPTA1_MRADIN	GPTA1 Multiplexer Register Array Data In Register	F000 203C _H	U, SV, 32	U, SV, 32	0000 0000 _H

Register Overview

Table 18-21 Address Map of GPTA1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
GPTA1_MRADOUT	GPTA1 Multiplexer Register Array Data Out Register	F000 2040 _H	U, SV, 32	U, SV, 32	0000 0000 _H
GPTA1_FPCSTAT	GPTA1 Filter and Prescaler Cell Status Register	F000 2044 _H	U, SV	U, SV	0000 0000 _H
GPTA1_FPCCTR0	GPTA1 Filter and Prescaler Cell Control Register 0	F000 2048 _H	U, SV	U, SV	0000 0000 _H
GPTA1_FPCTIM0	GPTA1 Filter and Prescaler Cell Timer Register 0	F000 204C _H	U, SV	U, SV	0000 0000 _H
GPTA1_FPCCTR1	GPTA1 Filter and Prescaler Cell Control Register 1	F000 2050 _H	U, SV	U, SV	0000 0000 _H
GPTA1_FPCTIM1	GPTA1 Filter and Prescaler Cell Timer Register 1	F000 2054 _H	U, SV	U, SV	0000 0000 _H
GPTA1_FPCCTR2	GPTA1 Filter and Prescaler Cell Control Register 2	F000 2058 _H	U, SV	U, SV	0000 0000 _H
GPTA1_FPCTIM2	GPTA1 Filter and Prescaler Cell Timer Register 2	F000 205C _H	U, SV	U, SV	0000 0000 _H
GPTA1_FPCCTR3	GPTA1 Filter and Prescaler Cell Control Register 3	F000 2060 _H	U, SV	U, SV	0000 0000 _H
GPTA1_FPCTIM3	GPTA1 Filter and Prescaler Cell Timer Register 3	F000 2064 _H	U, SV	U, SV	0000 0000 _H
GPTA1_FPCCTR4	GPTA1 Filter and Prescaler Cell Control Register 4	F000 2068 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-21 Address Map of GPTA1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
GPTA1_ FPCTIM4	GPTA1 Filter and Prescaler Cell Timer Register 4	F000 206C _H	U, SV	U, SV	0000 0000 _H
GPTA1_ FPCCTR5	GPTA1 Filter and Prescaler Cell Control Register 5	F000 2060 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ FPCTIM5	GPTA1 Filter and Prescaler Cell Timer Register 5	F000 2074 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ PDLCTR	GPTA1 Phase Discrimination Logic Control Register	F000 2078 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 207C _H	nBE	nBE	–
GPTA1_ DCMCTR0	GPTA1 Duty Cycle Measurement Control Register 0	F000 2080 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ DCMTIM0	GPTA1 Duty Cycle Measurement Timer Register 0	F000 2084 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ DCMCAV0	GPTA1 Duty Cycle Measurement Capture Register 0	F000 2088 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ DCMCOV0	GPTA1 Duty Cycle Measurement Capture/Compare Register 0	F000 208C _H	U, SV	U, SV	0000 0000 _H
GPTA1_ DCMCTR1	GPTA1 Duty Cycle Measurement Control Register 1	F000 2090 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ DCMTIM1	GPTA1 Duty Cycle Measurement Timer Register 1	F000 2094 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ DCMCAV1	GPTA1 Duty Cycle Measurement Capture Register 1	F000 2098 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-21 Address Map of GPTA1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
GPTA1_ DCMCOV1	GPTA1 Duty Cycle Measurement Capture/ Compare Register 1	F000 209C _H	U, SV	U, SV	0000 0000 _H
GPTA1_ DCMCTR2	GPTA1 Duty Cycle Measurement Control Register 2	F000 20A0 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ DCMTIM2	GPTA1 Duty Cycle Measurement Timer Register 2	F000 20A4 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ DCMCAV2	GPTA1 Duty Cycle Measurement Capture Register 2	F000 20A8 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ DCMCOV2	GPTA1 Duty Cycle Measurement Capture/ Compare Register 2	F000 20AC _H	U, SV	U, SV	0000 0000 _H
GPTA1_ DCMCTR3	GPTA1 Duty Cycle Measurement Control Register 3	F000 20B0 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ DCMTIM3	GPTA1 Duty Cycle Measurement Timer Register 3	F000 20B4 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ DCMCAV3	GPTA1 Duty Cycle Measurement Capture Register 3	F000 20B8 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ DCMCOV3	GPTA1 Duty Cycle Measurement Capture/ Compare Register 3	F000 20BC _H	U, SV	U, SV	0000 0000 _H
GPTA1_ PLLCTR	GPTA1 Phase Locked Loop Control Register	F000 20C0 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ PLLMTI	GPTA1 Phase Locked Loop Microtick Register	F000 20C4 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ PLLCNT	GPTA1 Phase Locked Loop Counter Register	F000 20C8 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ PLLSTP	GPTA1 Phase Locked Loop Step Register	F000 20CC _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-21 Address Map of GPTA1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
GPTA1_ PLLREV	GPTA1 Phase Locked Loop Reload Register	F000 20D0 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ PLLDTR	GPTA1 Phase Locked Loop Delta Register	F000 20D4 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ CKBCTR	GPTA1 Clock Bus Control Register	F000 20D8 _H	U, SV	U, SV	0000 FFFF _H
–	Reserved	F000 20DC _H	nBE	nBE	–
GPTA1_ GTCTR0	GPTA1 Global Timer Control Register 0	F000 20E0 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ GTREV0	GPTA1 Global Timer Reload Value Register 0	F000 20E4 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ GTTIM0	GPTA1 Global Timer Register 0	F000 20E8 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 20EC _H	nBE	nBE	–
GPTA1_ GTCTR1	GPTA1 Global Timer Control Register 1	F000 20F0 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ GTREV1	GPTA1 Global Timer Reload Value Register 1	F000 20F4 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ GTTIM1	GPTA1 Global Timer Register 1	F000 20F8 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 20FC _H	nBE	nBE	–
GPTA1_ GTCCTRn	GPTA1 Global Timer Cell Control Register n (n = 00-31)	F000 2100 _H + n × 08 _H + 00 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ GTCXRn	GPTA1 Global Timer Cell X Register n (n = 00-31)	F000 2100 _H + n × 08 _H + 04 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ LTCCTRn	GPTA1 Local Timer Cell Control Register n (n = 00-63)	F000 2200 _H + n × 08 _H + 00 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ LTCXRn	GPTA1 Local Timer Cell X Register n (n = 00-63)	F000 2200 _H + n × 08 _H + 04 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-21 Address Map of GPTA1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
–	Reserved	F000 2400 _H - F000 2764 _H	nBE	nBE	–
GPTA1_ SRC37	GPTA1 Service Request Control Register 37	F000 2768 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC36	GPTA1 Service Request Control Register 36	F000 276C _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC35	GPTA1 Service Request Control Register 35	F000 2770 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC34	GPTA1 Service Request Control Register 34	F000 2774 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC33	GPTA1 Service Request Control Register 33	F000 2778 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC32	GPTA1 Service Request Control Register 32	F000 277C _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC31	GPTA1 Service Request Control Register 31	F000 2780 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC30	GPTA1 Service Request Control Register 30	F000 2784 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC29	GPTA1 Service Request Control Register 29	F000 2788 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC28	GPTA1 Service Request Control Register 28	F000 278C _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC27	GPTA1 Service Request Control Register 27	F000 2790 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC26	GPTA1 Service Request Control Register 26	F000 2794 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC25	GPTA1 Service Request Control Register 25	F000 2798 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC24	GPTA1 Service Request Control Register 24	F000 279C _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC23	GPTA1 Service Request Control Register 23	F000 27A0 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-21 Address Map of GPTA1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
GPTA1_ SRC22	GPTA1 Service Request Control Register 22	F000 27A4 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC21	GPTA1 Service Request Control Register 21	F000 27A8 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC20	GPTA1 Service Request Control Register 20	F000 27AC _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC19	GPTA1 Service Request Control Register 19	F000 27B0 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC18	GPTA1 Service Request Control Register 18	F000 27B4 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC17	GPTA1 Service Request Control Register 17	F000 27B8 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC16	GPTA1 Service Request Control Register 16	F000 27BC _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC15	GPTA1 Service Request Control Register 15	F000 27C0 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC14	GPTA1 Service Request Control Register 14	F000 27C4 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC13	GPTA1 Service Request Control Register 13	F000 27C8 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC12	GPTA1 Service Request Control Register 12	F000 27CC _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC11	GPTA1 Service Request Control Register 11	F000 27D0 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC10	GPTA1 Service Request Control Register 10	F000 27D4 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC09	GPTA1 Service Request Control Register 09	F000 27D8 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC08	GPTA1 Service Request Control Register 08	F000 27DC _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC07	GPTA1 Service Request Control Register 07	F000 27E0 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-21 Address Map of GPTA1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
GPTA1_ SRC06	GPTA1 Service Request Control Register 06	F000 27E4 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC05	GPTA1 Service Request Control Register 05	F000 27E8 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC04	GPTA1 Service Request Control Register 04	F000 27EC _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC03	GPTA1 Service Request Control Register 03	F000 27F0 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC02	GPTA1 Service Request Control Register 02	F000 27F4 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC01	GPTA1 Service Request Control Register 01	F000 27F8 _H	U, SV	U, SV	0000 0000 _H
GPTA1_ SRC00	GPTA1 Service Request Control Register 00	F000 27FC _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-22 Address Map of LTCA2

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Local Timer Cell Array 2 (LTCA2)					
–	Reserved	F000 2800 _H - F000 2804 _H	nBE	nBE	–
LTCA2_ID	LTCA2 Module Identification Register	F000 2808 _H	U, SV	nBE	002A C0XX _H
–	Reserved	F000 280C _H - F000 281C _H	nBE	nBE	–
LTCA2_SRSC2	LTCA2 Service Request State Clear Register 2	F000 2820 _H	U, SV	U, SV	0000 0000 _H
LTCA2_SRSS2	LTCA2 Service Request State Set Register 2	F000 2824 _H	U, SV	U, SV	0000 0000 _H
LTCA2_SRSC3	LTCA2 Service Request State Clear Register 3	F000 2828 _H	U, SV	U, SV	0000 0000 _H
LTCA2_SRSS3	LTCA2 Service Request State Set Register 3	F000 282C _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 2830 _H - F000 2834 _H	nBE	nBE	0000 0000 _H
LTCA2_MRACTL	LTCA2 Multiplexer Register Array Control Register	F000 2838 _H	U, SV	U, SV	0000 0000 _H
LTCA2_MRADIN	LTCA2 Multiplexer Register Array Data In Register	F000 283C _H	U, SV, 32	U, SV, 32	0000 0000 _H
LTCA2_MRADOUT	LTCA2 Multiplexer Register Array Data Out Register	F000 2840 _H	U, SV, 32	U, SV, 32	0000 0000 _H
–	Reserved	F000 2844 _H - F000 2878 _H	nBE	nBE	–
–	Reserved	F000 287C _H	BE	BE	–
–	Reserved	F000 2880 _H - F000 28D8 _H	nBE	nBE	–
–	Reserved	F000 28DC _H	BE	BE	–

Register Overview

Table 18-22 Address Map of LTCA2 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
–	Reserved	F000 28E0 _H - F000 28E8 _H	nBE	nBE	–
–	Reserved	F000 28EC _H	BE	BE	–
–	Reserved	F000 28F0 _H - F000 28F8 _H	nBE	nBE	–
–	Reserved	F000 28FC _H	BE	BE	–
–	Reserved	F000 2900 _H - F000 29FC _H	nBE	nBE	–
LTCA2_ LTCCTRn	LTCA2 Local Timer Cell Control Register n (n = 00-63)	F000 2A00 _H + n × 08 _H + 00 _H	U, SV	U, SV	0000 0000 _H
LTCA2_ LTCXRn	LTCA2 Local Timer Cell X Register n (n = 00-63)	F000 2A00 _H + n × 08 _H + 04 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 2C00 _H - F000 2FBC _H	nBE	nBE	–
LTCA2_ SRC15	LTCA2 Service Request Control Register 15	F000 2FC0 _H	U, SV	U, SV	0000 0000 _H
LTCA2_ SRC14	LTCA2 Service Request Control Register 14	F000 2FC4 _H	U, SV	U, SV	0000 0000 _H
LTCA2_ SRC13	LTCA2 Service Request Control Register 13	F000 2FC8 _H	U, SV	U, SV	0000 0000 _H
LTCA2_ SRC12	LTCA2 Service Request Control Register 12	F000 2FCC _H	U, SV	U, SV	0000 0000 _H
LTCA2_ SRC11	LTCA2 Service Request Control Register 11	F000 2FD0 _H	U, SV	U, SV	0000 0000 _H
LTCA2_ SRC10	LTCA2 Service Request Control Register 10	F000 2FD4 _H	U, SV	U, SV	0000 0000 _H
LTCA2_ SRC09	LTCA2 Service Request Control Register 09	F000 2FD8 _H	U, SV	U, SV	0000 0000 _H
LTCA2_ SRC08	LTCA2 Service Request Control Register 08	F000 2FDC _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-22 Address Map of LTCA2 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
LTCA2_SRC07	LTCA2 Service Request Control Register 07	F000 2FE0 _H	U, SV	U, SV	0000 0000 _H
LTCA2_SRC06	LTCA2 Service Request Control Register 06	F000 2FE4 _H	U, SV	U, SV	0000 0000 _H
LTCA2_SRC05	LTCA2 Service Request Control Register 05	F000 2FE8 _H	U, SV	U, SV	0000 0000 _H
LTCA2_SRC04	LTCA2 Service Request Control Register 04	F000 2FEC _H	U, SV	U, SV	0000 0000 _H
LTCA2_SRC03	LTCA2 Service Request Control Register 03	F000 2FF0 _H	U, SV	U, SV	0000 0000 _H
LTCA2_SRC02	LTCA2 Service Request Control Register 02	F000 2FF4 _H	U, SV	U, SV	0000 0000 _H
LTCA2_SRC01	LTCA2 Service Request Control Register 01	F000 2FF8 _H	U, SV	U, SV	0000 0000 _H
LTCA2_SRC00	LTCA2 Service Request Control Register 00	F000 2FFC _H	U, SV	U, SV	0000 0000 _H

Table 18-23 Address Map of DMA

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Direct Memory Access Controller (DMA)					
DMA_CLC	DMA Clock Control Register	F000 3C00 _H	U, SV	SV, E	0000 0000 _H
–	Reserved	F000 3C04 _H	nBE	SV	–
DMA_ID	DMA Module Identification Register	F000 3C08 _H	U, SV	BE	001A C0XX _H
–	Reserved	F000 3C0C _H	BE	BE	–
DMA_CHRSTR	DMA Channel Reset Request Register	F000 3C10 _H	U, SV	SV	0000 0000 _H
DMA_TRSR	DMA Transaction Request State Register	F000 3C14 _H	U, SV	BE	0000 0000 _H
DMA_STREQ	DMA Software Transaction Request Register	F000 3C18 _H	U, SV	SV	0000 0000 _H
DMA_HTREQ	DMA Hardware Transaction Request Register	F000 3C1C _H	U, SV	SV	0000 0000 _H
DMA_EER	DMA Enable Error Register	F000 3C20 _H	U, SV	SV	0000 0000 _H
DMA_ERRSR	DMA Error Status Register	F000 3C24 _H	U, SV	BE	0000 0000 _H
DMA_CLRE	DMA Clear Error Register	F000 3C28 _H	U, SV	SV	0000 0000 _H
DMA_GINTR	DMA Global Interrupt Set Register	F000 3C2C _H	U, SV	SV	0000 0000 _H
DMA_MESR	DMA Move Engine Status Register	F000 3C30 _H	U, SV	BE	0000 0000 _H
DMA_ME0R	DMA Move Engine 0 Read Register	F000 3C34 _H	U, SV	BE	0000 0000 _H
DMA_ME1R	DMA Move Engine 1 Read Register	F000 3C38 _H	U, SV	BE	0000 0000 _H
DMA_ME0PR	DMA Move Engine 0 Pattern Register	F000 3C3C _H	U, SV	SV	0000 0000 _H

Register Overview

Table 18-23 Address Map of DMA (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
DMA_ME1PR	DMA Move Engine 1 Pattern Register	F000 3C40 _H	U, SV	SV	0000 0000 _H
DMA_ME0AENR	DMA Move Engine 0 Access Enable Register	F000 3C44 _H	U, SV	SV, E	0000 0000 _H
DMA_ME0ARR	DMA Move Engine 0 Access Range Register	F000 3C48 _H	U, SV	SV, E	0000 0000 _H
DMA_ME1AENR	DMA Move Engine 1 Access Enable Register	F000 3C4C _H	U, SV	SV, E	0000 0000 _H
DMA_ME1ARR	DMA Move Engine 1 Access Range Register	F000 3C50 _H	U, SV	SV, E	0000 0000 _H
DMA_INTSR	DMA Interrupt Status Register	F000 3C54 _H	U, SV	BE	0000 0000 _H
DMA_INTCR	DMA Interrupt Clear Register	F000 3C58 _H	U, SV	SV	0000 0000 _H
DMA_WRPSR	DMA Wrap Status Register	F000 3C5C _H	U, SV	BE	0000 0000 _H
–	Reserved	F000 3C60 _H	BE	BE	–
DMA_OCDSR	DMA OCDS Register	F000 3C64 _H	U, SV	SV, E	0000 0000 _H
DMA_SUSPMR	DMA Suspend Mode Register	F000 3C68 _H	U, SV	SV, E	0000 0000 _H
–	Reserved	F000 3C6C _H - F000 3C7C _H	BE	BE	–
DMA_CHSR00	DMA Channel 00 Status Register	F000 3C80 _H	U, SV	BE	0000 0000 _H
DMA_CHCR00	DMA Channel 00 Control Register	F000 3C84 _H	U, SV	SV	0000 0000 _H
DMA_CHICR00	DMA Channel 00 Interrupt Control Register	F000 3C88 _H	U, SV	SV	0000 0000 _H
DMA_ADRCR00	DMA Channel 00 Address Control Register	F000 3C8C _H	U, SV	SV	0000 0000 _H
DMA_SADR00	DMA Channel 00 Source Address Register	F000 3C90 _H	U, SV	SV	0000 0000 _H

Register Overview

Table 18-23 Address Map of DMA (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
DMA_ DADR00	DMA Channel 00 Destination Address Reg.	F000 3C94 _H	U, SV	SV	0000 0000 _H
DMA_ SHADR00	DMA Channel 00 Shadow Address Register	F000 3C98 _H	U, SV	BE	0000 0000 _H
–	Reserved	F000 3C9C _H	BE	BE	–
DMA_ CHSR01	DMA Channel 01 Status Register	F000 3CA0 _H	U, SV	BE	0000 0000 _H
DMA_ CHCR01	DMA Channel 01 Control Register	F000 3CA4 _H	U, SV	SV	0000 0000 _H
DMA_ CHICR01	DMA Channel 01 Interrupt Control Register	F000 3CA8 _H	U, SV	SV	0000 0000 _H
DMA_ ADRCR01	DMA Channel 01 Address Control Register	F000 3CAC _H	U, SV	SV	0000 0000 _H
DMA_ SADR01	DMA Channel 01 Source Address Register	F000 3CB0 _H	U, SV	SV	0000 0000 _H
DMA_ DADR01	DMA Channel 01 Destination Address Reg.	F000 3CB4 _H	U, SV	SV	0000 0000 _H
DMA_ SHADR01	DMA Channel 01 Shadow Address Register	F000 3CB8 _H	U, SV	BE	0000 0000 _H
–	Reserved	F000 3CBC _H	BE	BE	–
DMA_ CHSR02	DMA Channel 02 Status Register	F000 3CC0 _H	U, SV	BE	0000 0000 _H
DMA_ CHCR02	DMA Channel 02 Control Register	F000 3CC4 _H	U, SV	SV	0000 0000 _H
DMA_ CHICR02	DMA Channel 02 Interrupt Control Register	F000 3CC8 _H	U, SV	SV	0000 0000 _H
DMA_ ADRCR02	DMA Channel 02 Address Control Register	F000 3CCC _H	U, SV	SV	0000 0000 _H
DMA_ SADR02	DMA Channel 02 Source Address Register	F000 3CD0 _H	U, SV	SV	0000 0000 _H

Register Overview

Table 18-23 Address Map of DMA (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
DMA_ DADR02	DMA Channel 02 Destination Address Reg.	F000 3CD4 _H	U, SV	SV	0000 0000 _H
DMA_ SHADR02	DMA Channel 02 Shadow Address Register	F000 3CD8 _H	U, SV	BE	0000 0000 _H
–	Reserved	F000 3CDC _H	BE	BE	–
DMA_ CHSR03	DMA Channel 03 Status Register	F000 3CE0 _H	U, SV	BE	0000 0000 _H
DMA_ CHCR03	DMA Channel 03 Control Register	F000 3CE4 _H	U, SV	SV	0000 0000 _H
DMA_ CHICR03	DMA Channel 03 Interrupt Control Register	F000 3CE8 _H	U, SV	SV	0000 0000 _H
DMA_ ADRCR03	DMA Channel 03 Address Control Register	F000 3CEC _H	U, SV	SV	0000 0000 _H
DMA_ SADR03	DMA Channel 03 Source Address Register	F000 3CF0 _H	U, SV	SV	0000 0000 _H
DMA_ DADR03	DMA Channel 03 Destination Address Reg.	F000 3CF4 _H	U, SV	SV	0000 0000 _H
DMA_ SHADR03	DMA Channel 03 Shadow Address Register	F000 3CF8 _H	U, SV	BE	0000 0000 _H
–	Reserved	F000 3CFC _H	BE	BE	–
DMA_ CHSR04	DMA Channel 04 Status Register	F000 3D00 _H	U, SV	BE	0000 0000 _H
DMA_ CHCR04	DMA Channel 04 Control Register	F000 3D04 _H	U, SV	SV	0000 0000 _H
DMA_ CHICR04	DMA Channel 04 Interrupt Control Register	F000 3D08 _H	U, SV	SV	0000 0000 _H
DMA_ ADRCR04	DMA Channel 04 Address Control Register	F000 3D0C _H	U, SV	SV	0000 0000 _H
DMA_ SADR04	DMA Channel 04 Source Address Register	F000 3D10 _H	U, SV	SV	0000 0000 _H

Register Overview

Table 18-23 Address Map of DMA (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
DMA_ DADR04	DMA Channel 04 Destination Address Reg.	F000 3D14 _H	U, SV	SV	0000 0000 _H
DMA_ SHADR04	DMA Channel 04 Shadow Address Register	F000 3D18 _H	U, SV	BE	0000 0000 _H
–	Reserved	F000 3D1C _H	BE	BE	–
DMA_ CHSR05	DMA Channel 05 Status Register	F000 3D20 _H	U, SV	BE	0000 0000 _H
DMA_ CHCR05	DMA Channel 05 Control Register	F000 3D24 _H	U, SV	SV	0000 0000 _H
DMA_ CHICR05	DMA Channel 05 Interrupt Control Register	F000 3D28 _H	U, SV	SV	0000 0000 _H
DMA_ ADRCR05	DMA Channel 05 Address Control Register	F000 3D2C _H	U, SV	SV	0000 0000 _H
DMA_ SADR05	DMA Channel 05 Source Address Register	F000 3D30 _H	U, SV	SV	0000 0000 _H
DMA_ DADR05	DMA Channel 05 Destination Address Reg.	F000 3D34 _H	U, SV	SV	0000 0000 _H
DMA_ SHADR05	DMA Channel 05 Shadow Address Register	F000 3D38 _H	U, SV	BE	0000 0000 _H
–	Reserved	F000 3D3C _H	BE	BE	–
DMA_ CHSR06	DMA Channel 06 Status Register	F000 3D40 _H	U, SV	BE	0000 0000 _H
DMA_ CHCR06	DMA Channel 06 Control Register	F000 3D44 _H	U, SV	SV	0000 0000 _H
DMA_ CHICR06	DMA Channel 06 Interrupt Control Register	F000 3D48 _H	U, SV	SV	0000 0000 _H
DMA_ ADRCR06	DMA Channel 06 Address Control Register	F000 3D4C _H	U, SV	SV	0000 0000 _H
DMA_ SADR06	DMA Channel 06 Source Address Register	F000 3D50 _H	U, SV	SV	0000 0000 _H

Register Overview

Table 18-23 Address Map of DMA (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
DMA_ DADR06	DMA Channel 06 Destination Address Reg.	F000 3D54 _H	U, SV	SV	0000 0000 _H
DMA_ SHADR06	DMA Channel 06 Shadow Address Register	F000 3D58 _H	U, SV	BE	0000 0000 _H
–	Reserved	F000 3D5C _H	BE	BE	–
DMA_ CHSR07	DMA Channel 07 Status Register	F000 3D60 _H	U, SV	BE	0000 0000 _H
DMA_ CHCR07	DMA Channel 07 Control Register	F000 3D64 _H	U, SV	SV	0000 0000 _H
DMA_ CHICR07	DMA Channel 07 Interrupt Control Register	F000 3D68 _H	U, SV	SV	0000 0000 _H
DMA_ ADRCR07	DMA Channel 07 Address Control Register	F000 3D6C _H	U, SV	SV	0000 0000 _H
DMA_ SADR07	DMA Channel 07 Source Address Register	F000 3D70 _H	U, SV	SV	0000 0000 _H
DMA_ DADR07	DMA Channel 07 Destination Address Reg.	F000 3D74 _H	U, SV	SV	0000 0000 _H
DMA_ SHADR07	DMA Channel 07 Shadow Address Register	F000 3D78 _H	U, SV	BE	0000 0000 _H
–	Reserved	F000 3D7C _H	BE	BE	–
DMA_ CHSR10	DMA Channel 10 Status Register	F000 3D80 _H	U, SV	BE	0000 0000 _H
DMA_ CHCR10	DMA Channel 10 Control Register	F000 3D84 _H	U, SV	SV	0000 0000 _H
DMA_ CHICR10	DMA Channel 10 Interrupt Control Register	F000 3D88 _H	U, SV	SV	0000 0000 _H
DMA_ ADRCR10	DMA Channel 10 Address Control Register	F000 3D8C _H	U, SV	SV	0000 0000 _H
DMA_ SADR10	DMA Channel 10 Source Address Register	F000 3D90 _H	U, SV	SV	0000 0000 _H

Register Overview

Table 18-23 Address Map of DMA (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
DMA_ DADR10	DMA Channel 10 Destination Address Reg.	F000 3D94 _H	U, SV	SV	0000 0000 _H
DMA_ SHADR10	DMA Channel 10 Shadow Address Register	F000 3D98 _H	U, SV	BE	0000 0000 _H
–	Reserved	F000 3D9C _H	BE	BE	–
DMA_ CHSR11	DMA Channel 11 Status Register	F000 3DA0 _H	U, SV	BE	0000 0000 _H
DMA_ CHCR11	DMA Channel 11 Control Register	F000 3DA4 _H	U, SV	SV	0000 0000 _H
DMA_ CHICR11	DMA Channel 11 Interrupt Control Register	F000 3DA8 _H	U, SV	SV	0000 0000 _H
DMA_ ADRCR11	DMA Channel 11 Address Control Register	F000 3DAC _H	U, SV	SV	0000 0000 _H
DMA_ SADR11	DMA Channel 11 Source Address Register	F000 3DB0 _H	U, SV	SV	0000 0000 _H
DMA_ DADR11	DMA Channel 11 Destination Address Reg.	F000 3DB4 _H	U, SV	SV	0000 0000 _H
DMA_ SHADR11	DMA Channel 11 Shadow Address Register	F000 3DB8 _H	U, SV	BE	0000 0000 _H
–	Reserved	F000 3DBC _H	BE	BE	–
DMA_ CHSR12	DMA Channel 12 Status Register	F000 3DC0 _H	U, SV	BE	0000 0000 _H
DMA_ CHCR12	DMA Channel 12 Control Register	F000 3DC4 _H	U, SV	SV	0000 0000 _H
DMA_ CHICR12	DMA Channel 12 Interrupt Control Register	F000 3DC8 _H	U, SV	SV	0000 0000 _H
DMA_ ADRCR12	DMA Channel 12 Address Control Register	F000 3DCC _H	U, SV	SV	0000 0000 _H
DMA_ SADR12	DMA Channel 12 Source Address Register	F000 3DD0 _H	U, SV	SV	0000 0000 _H

Register Overview

Table 18-23 Address Map of DMA (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
DMA_ DADR12	DMA Channel 12 Destination Address Reg.	F000 3DD4 _H	U, SV	SV	0000 0000 _H
DMA_ SHADR12	DMA Channel 12 Shadow Address Register	F000 3DD8 _H	U, SV	BE	0000 0000 _H
–	Reserved	F000 3DDC _H	BE	BE	–
DMA_ CHSR13	DMA Channel 13 Status Register	F000 3DE0 _H	U, SV	BE	0000 0000 _H
DMA_ CHCR13	DMA Channel 13 Control Register	F000 3DE4 _H	U, SV	SV	0000 0000 _H
DMA_ CHICR13	DMA Channel 13 Interrupt Control Register	F000 3DE8 _H	U, SV	SV	0000 0000 _H
DMA_ ADRCR13	DMA Channel 13 Address Control Register	F000 3DEC _H	U, SV	SV	0000 0000 _H
DMA_ SADR13	DMA Channel 13 Source Address Register	F000 3DF0 _H	U, SV	SV	0000 0000 _H
DMA_ DADR13	DMA Channel 13 Destination Address Reg.	F000 3DF4 _H	U, SV	SV	0000 0000 _H
DMA_ SHADR13	DMA Channel 13 Shadow Address Register	F000 3DF8 _H	U, SV	BE	0000 0000 _H
–	Reserved	F000 3DFC _H	BE	BE	–
DMA_ CHSR14	DMA Channel 14 Status Register	F000 3E00 _H	U, SV	BE	0000 0000 _H
DMA_ CHCR14	DMA Channel 14 Control Register	F000 3E04 _H	U, SV	SV	0000 0000 _H
DMA_ CHICR14	DMA Channel 14 Interrupt Control Register	F000 3E08 _H	U, SV	SV	0000 0000 _H
DMA_ ADRCR14	DMA Channel 14 Address Control Register	F000 3E0C _H	U, SV	SV	0000 0000 _H
DMA_ SADR14	DMA Channel 14 Source Address Register	F000 3E10 _H	U, SV	SV	0000 0000 _H

Register Overview

Table 18-23 Address Map of DMA (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
DMA_ DADR14	DMA Channel 14 Destination Address Reg.	F000 3E14 _H	U, SV	SV	0000 0000 _H
DMA_ SHADR14	DMA Channel 14 Shadow Address Register	F000 3E18 _H	U, SV	BE	0000 0000 _H
–	Reserved	F000 3E1C _H	BE	BE	–
DMA_ CHSR15	DMA Channel 15 Status Register	F000 3E20 _H	U, SV	BE	0000 0000 _H
DMA_ CHCR15	DMA Channel 15 Control Register	F000 3E24 _H	U, SV	SV	0000 0000 _H
DMA_ CHICR15	DMA Channel 15 Interrupt Control Register	F000 3E28 _H	U, SV	SV	0000 0000 _H
DMA_ ADRCR15	DMA Channel 15 Address Control Register	F000 3E2C _H	U, SV	SV	0000 0000 _H
DMA_ SADR15	DMA Channel 15 Source Address Register	F000 3E30 _H	U, SV	SV	0000 0000 _H
DMA_ DADR15	DMA Channel 15 Destination Address Reg.	F000 3E34 _H	U, SV	SV	0000 0000 _H
DMA_ SHADR15	DMA Channel 15 Shadow Address Register	F000 3E38 _H	U, SV	BE	0000 0000 _H
–	Reserved	F000 3E3C _H	BE	BE	–
DMA_ CHSR16	DMA Channel 16 Status Register	F000 3E40 _H	U, SV	BE	0000 0000 _H
DMA_ CHCR16	DMA Channel 16 Control Register	F000 3E44 _H	U, SV	SV	0000 0000 _H
DMA_ CHICR16	DMA Channel 16 Interrupt Control Register	F000 3E48 _H	U, SV	SV	0000 0000 _H
DMA_ ADRCR16	DMA Channel 16 Address Control Register	F000 3E4C _H	U, SV	SV	0000 0000 _H
DMA_ SADR16	DMA Channel 16 Source Address Register	F000 3E50 _H	U, SV	SV	0000 0000 _H

Register Overview

Table 18-23 Address Map of DMA (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
DMA_ DADR16	DMA Channel 16 Destination Address Reg.	F000 3E54 _H	U, SV	SV	0000 0000 _H
DMA_ SHADR16	DMA Channel 16 Shadow Address Register	F000 3E58 _H	U, SV	BE	0000 0000 _H
–	Reserved	F000 3E5C _H	BE	BE	–
DMA_ CHSR17	DMA Channel 17 Status Register	F000 3E60 _H	U, SV	BE	0000 0000 _H
DMA_ CHCR17	DMA Channel 17 Control Register	F000 3E64 _H	U, SV	SV	0000 0000 _H
DMA_ CHICR17	DMA Channel 17 Interrupt Control Register	F000 3E68 _H	U, SV	SV	0000 0000 _H
DMA_ ADRCR17	DMA Channel 17 Address Control Register	F000 3E6C _H	U, SV	SV	0000 0000 _H
DMA_ SADR17	DMA Channel 17 Source Address Register	F000 3E70 _H	U, SV	SV	0000 0000 _H
DMA_ DADR17	DMA Channel 17 Destination Address Reg.	F000 3E74 _H	U, SV	SV	0000 0000 _H
DMA_ SHADR17	DMA Channel 17 Shadow Address Register	F000 3E78 _H	U, SV	BE	0000 0000 _H
–	Reserved	F000 3E7C _H - F000 3E88 _H	BE	BE	–
DMA_ SYSSRC4	DMA System Interrupt Service Request Control Register 4	F000 3E8C _H	U, SV	SV	0000 0000 _H
DMA_ SYSSRC3	DMA System Interrupt Service Request Control Register 3	F000 3E90 _H	U, SV	SV	0000 0000 _H
DMA_ SYSSRC2	DMA System Interrupt Service Request Control Register 2	F000 3E94 _H	U, SV	SV	0000 0000 _H

Register Overview

Table 18-23 Address Map of DMA (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
DMA_ SYSSRC1	DMA System Interrupt Service Request Control Register 1	F000 3E98 _H	U, SV	SV	0000 0000 _H
DMA_ SYSSRC0	DMA System Interrupt Service Request Control Register 0	F000 3E9C _H	U, SV	SV	0000 0000 _H
DMA_ MLI0SRC3	DMA MLI0 Service Request Control Reg. 3	F000 3EA0 _H	U, SV	SV	0000 0000 _H
DMA_ MLI0SRC2	DMA MLI0 Service Request Control Reg. 2	F000 3EA4 _H	U, SV	SV	0000 0000 _H
DMA_ MLI0SRC1	DMA MLI0 Service Request Control Reg. 1	F000 3EA8 _H	U, SV	SV	0000 0000 _H
DMA_ MLI0SRC0	DMA MLI0 Service Request Control Reg. 0	F000 3EAC _H	U, SV	SV	0000 0000 _H
–	Reserved	F000 3EB0 _H - F000 3EB4 _H	BE	BE	–
DMA_ MLI1SRC1	DMA MLI1 Service Request Control Reg. 1	F000 3EB8 _H	U, SV	SV	0000 0000 _H
DMA_ MLI1SRC0	DMA MLI1 Service Request Control Reg. 0	F000 3EBC _H	U, SV	SV	0000 0000 _H
–	Reserved	F000 3EC0 _H - F000 3EDC _H	BE	BE	–
DMA_ SRC7	DMA Service Request Control Register 7	F000 3EE0 _H	U, SV	SV	0000 0000 _H
DMA_ SRC6	DMA Service Request Control Register 6	F000 3EE4 _H	U, SV	SV	0000 0000 _H
DMA_ SRC5	DMA Service Request Control Register 5	F000 3EE8 _H	U, SV	SV	0000 0000 _H
DMA_ SRC4	DMA Service Request Control Register 4	F000 3EEC _H	U, SV	SV	0000 0000 _H
DMA_ SRC3	DMA Service Request Control Register 3	F000 3EF0 _H	U, SV	SV	0000 0000 _H
DMA_ SRC2	DMA Service Request Control Register 2	F000 3EF4 _H	U, SV	SV	0000 0000 _H

Register Overview

Table 18-23 Address Map of DMA (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
DMA_ SRC1	DMA Service Request Control Register 1	F000 3EF8 _H	U, SV	SV	0000 0000 _H
DMA_ SRC0	DMA Service Request Control Register 0	F000 3EFC _H	U, SV	SV	0000 0000 _H

Table 18-24 Address Map of CAN

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
MultiCAN Controller (CAN)					
CAN_CLC	CAN Clock Control Register	F000 4000 _H	U, SV	SV, E	0000 0003 _H
–	Reserved	F000 4004 _H	nBE	nBE	–
CAN_ID	CAN Module Identification Register	F000 4008 _H	U, SV	U, SV	002B C0XX _H
CAN_FDR	CAN Fractional Divider Register	F000 400C _H	U, SV	SV, E	0000 0000 _H
–	Reserved	F000 4010 _H - F000 40BC _H	nBE	nBE	–
CAN_SRC15	CAN Service Request Control Register 15	F000 40C0 _H	U, SV	U, SV	0000 0000 _H
CAN_SRC14	CAN Service Request Control Register 14	F000 40C4 _H	U, SV	U, SV	0000 0000 _H
CAN_SRC13	CAN Service Request Control Register 13	F000 40C8 _H	U, SV	U, SV	0000 0000 _H
CAN_SRC12	CAN Service Request Control Register 12	F000 40CC _H	U, SV	U, SV	0000 0000 _H
CAN_SRC11	CAN Service Request Control Register 11	F000 40D0 _H	U, SV	U, SV	0000 0000 _H
CAN_SRC10	CAN Service Request Control Register 10	F000 40D4 _H	U, SV	U, SV	0000 0000 _H
CAN_SRC9	CAN Service Request Control Register 9	F000 40D8 _H	U, SV	U, SV	0000 0000 _H
CAN_SRC8	CAN Service Request Control Register 8	F000 40DC _H	U, SV	U, SV	0000 0000 _H
CAN_SRC7	CAN Service Request Control Register 7	F000 40E0 _H	U, SV	U, SV	0000 0000 _H
CAN_SRC6	CAN Service Request Control Register 6	F000 40E4 _H	U, SV	U, SV	0000 0000 _H
CAN_SRC5	CAN Service Request Control Register 5	F000 40E8 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-24 Address Map of CAN (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
CAN_SRC4	CAN Service Request Control Register 4	F000 40EC _H	U, SV	U, SV	0000 0000 _H
CAN_SRC3	CAN Service Request Control Register 3	F000 40F0 _H	U, SV	U, SV	0000 0000 _H
CAN_SRC2	CAN Service Request Control Register 2	F000 40F4 _H	U, SV	U, SV	0000 0000 _H
CAN_SRC1	CAN Service Request Control Register 1	F000 40F8 _H	U, SV	U, SV	0000 0000 _H
CAN_SRC0	CAN Service Request Control Register 0	F000 40FC _H	U, SV	U, SV	0000 0000 _H
Global Module Control Registers					
CAN_LIST0	CAN List Register 0	F000 4100 _H	U, SV	U, SV	007F 7F00 _H
CAN_LIST1	CAN List Register 1	F000 4104 _H	U, SV	U, SV	0100 0000 _H
CAN_LIST2	CAN List Register 2	F000 4108 _H	U, SV	U, SV	0100 0000 _H
CAN_LIST3	CAN List Register 3	F000 410C _H	U, SV	U, SV	0100 0000 _H
CAN_LIST4	CAN List Register 4	F000 4110 _H	U, SV	U, SV	0100 0000 _H
CAN_LIST5	CAN List Register 5	F000 4114 _H	U, SV	U, SV	0100 0000 _H
CAN_LIST6	CAN List Register 6	F000 4118 _H	U, SV	U, SV	0100 0000 _H
CAN_LIST7	CAN List Register 7	F000 411C _H	U, SV	U, SV	0100 0000 _H
CAN_MSPND0	CAN Message Pending Register 0	F000 4120 _H	U, SV	U, SV	0000 0000 _H
CAN_MSPND1	CAN Message Pending Register 1	F000 4124 _H	U, SV	U, SV	0000 0000 _H
CAN_MSPND2	CAN Message Pending Register 2	F000 4128 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-24 Address Map of CAN (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
CAN_MSPND3	CAN Message Pending Register 3	F000 412C _H	U, SV	U, SV	0000 0000 _H
CAN_MSPND4	CAN Message Pending Register 4	F000 4130 _H	U, SV	U, SV	0000 0000 _H
CAN_MSPND5	CAN Message Pending Register 5	F000 4134 _H	U, SV	U, SV	0000 0000 _H
CAN_MSPND6	CAN Message Pending Register 6	F000 4138 _H	U, SV	U, SV	0000 0000 _H
CAN_MSPND7	CAN Message Pending Register 7	F000 413C _H	U, SV	U, SV	0000 0000 _H
CAN_MSID0	CAN Message Index Register 0	F000 4140 _H	U, SV	U, SV	0000 0020 _H
CAN_MSID1	CAN Message Index Register 1	F000 4144 _H	U, SV	U, SV	0000 0020 _H
CAN_MSID2	CAN Message Index Register 2	F000 4148 _H	U, SV	U, SV	0000 0020 _H
CAN_MSID3	CAN Message Index Register 3	F000 414C _H	U, SV	U, SV	0000 0020 _H
CAN_MSID4	CAN Message Index Register 4	F000 4150 _H	U, SV	U, SV	0000 0020 _H
CAN_MSID5	CAN Message Index Register 5	F000 4154 _H	U, SV	U, SV	0000 0020 _H
CAN_MSID6	CAN Message Index Register 6	F000 4158 _H	U, SV	U, SV	0000 0020 _H
CAN_MSID7	CAN Message Index Register 7	F000 415C _H	U, SV	U, SV	0000 0020 _H
–	Reserved	F000 4160 _H - F000 41BC _H	nBE	nBE	–
CAN_MSIMASK	CAN Message Index Mask Register	F000 41C0 _H	U, SV	U, SV	0000 0000 _H
CAN_PANCTR	CAN Panel Control Register	F000 41C4 _H	U, SV	U, SV	0000 0301 _H

Register Overview

Table 18-24 Address Map of CAN (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
CAN_MCR	CAN Module Control Register	F000 41C8 _H	U, SV	U, SV	0000 0000 _H
CAN_MITR	CAN Module Interrupt Trigger Register	F000 41CC _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 41D0 _H - F000 41FC _H	nBE	nBE	–

CAN Node 0 Registers

CAN_NCR0	CAN Node 0 Control Register	F000 4200 _H	U, SV	U, SV	0000 0001 _H
CAN_NSR0	CAN Node 0 Status Register	F000 4204 _H	U, SV	U, SV	0000 0000 _H
CAN_NIPR0	CAN Node 0 Interrupt Pointer Register	F000 4208 _H	U, SV	U, SV	0000 0000 _H
CAN_NPCR0	CAN Node 0 Port Control Register	F000 420C _H	U, SV	U, SV	0000 0000 _H
CAN_NBTR0	CAN Node 0 Bit Timing Register	F000 4210 _H	U, SV	U, SV	0000 0000 _H
CAN_NECNT0	CAN Node 0 Error Counter Register	F000 4214 _H	U, SV	U, SV	0060 0000 _H
CAN_NFCR0	CAN Node 0 Frame Counter Register	F000 4218 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 421C _H - F000 427C _H	nBE	nBE	–

CAN Node 0 TTCAN Registers

CAN_LTR	CAN Local Time Register	F000 4280 _H	U, SV	U, SV	0000 0000 _H
CAN_SYNMR	CAN Synchronization Mark Register	F000 4284 _H	U, SV	U, SV	0000 0000 _H
CAN_REFMR	CAN Reference Mark Register	F000 4288 _H	U, SV	U, SV	0000 0000 _H
CAN_LREFMR	CAN Last Reference Mark Register	F000 428C _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-24 Address Map of CAN (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
CAN_TURR	CAN Time Unit Ratio Register	F000 4290 _H	U, SV	U, SV	0000 0000 _H
CAN_CYCTMR	CAN Cycle Time Register	F000 4294 _H	U, SV	U, SV	0000 0000 _H
CAN_LOR	CAN Local Offset Register	F000 4298 _H	U, SV	U, SV	0000 0000 _H
CAN_GMR	CAN Global Mark Register	F000 429C _H	U, SV	U, SV	0000 0000 _H
CAN_LGMR	CAN Last Global Mark Register	F000 42A0 _H	U, SV	U, SV	0000 0000 _H
CAN_AWDR	CAN Application Watchdog Register	F000 42A4 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 42A8 _H - F000 42BC _H	nBE	nBE	–
CAN_TTCR	CAN Time Trigger Control Register	F000 42C0 _H	U, SV	U, SV	0000 0000 _H
CAN_TTCFGR	CAN Time Trigger Configuration Register	F000 42C4 _H	U, SV	U, SV	0000 0000 _H
CAN_TTSR	CAN Time Trigger Status Register	F000 42C8 _H	U, SV	U, SV	0000 1000 _H
CAN_TTFMR	CAN Time Trigger Flag Modification Register	F000 42CC _H	U, SV	U, SV	0000 0000 _H
CAN_TTIRR	CAN Time Trigger Interrupt Request Register	F000 42D0 _H	U, SV	U, SV	0000 0000 _H
CAN_TTIER	CAN Time Trigger Interrupt Enable Register	F000 42D4 _H	U, SV	U, SV	0000 0000 _H
CAN_TTINPR	CAN Time Trigger Int. Node Pointer Register	F000 42D8 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 42DC _H - F000 42EC _H	nBE	nBE	–
CAN_ST SRL	CAN Scheduler Timing Status Register Low	F000 42F0 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-24 Address Map of CAN (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
CAN_STSRH	CAN Scheduler Timing Status Register High	F000 42F4 _H	U, SV	U, SV	0000 0000 _H
CAN_SISR	CAN Scheduler Instruction Status Register	F000 42F8 _H	U, SV	U, SV	0000 0000 _H

CAN Node 1 Registers

CAN_NCR1	CAN Node 1 Control Register	F000 4300 _H	U, SV	U, SV	0000 0001 _H
CAN_NSR1	CAN Node 1 Status Register	F000 4304 _H	U, SV	U, SV	0000 0000 _H
CAN_NIPR1	CAN Node 1 Interrupt Pointer Register	F000 4308 _H	U, SV	U, SV	0000 0000 _H
CAN_NPCR1	CAN Node 1 Port Control Register	F000 430C _H	U, SV	U, SV	0000 0000 _H
CAN_NBTR1	CAN Node 1 Bit Timing Register	F000 4310 _H	U, SV	U, SV	0000 0000 _H
CAN_NECNT1	CAN Node 1 Error Counter Register	F000 4314 _H	U, SV	U, SV	0060 0000 _H
CAN_NFCR1	CAN Node 1 Frame Counter Register	F000 4318 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 431C _H - F000 43FC _H	nBE	nBE	–

CAN Node 2 Registers

CAN_NCR2	CAN Node 2 Control Register	F000 4400 _H	U, SV	U, SV	0000 0001 _H
CAN_NSR2	CAN Node 2 Status Register	F000 4404 _H	U, SV	U, SV	0000 0000 _H
CAN_NIPR2	CAN Node 2 Interrupt Pointer Register	F000 4408 _H	U, SV	U, SV	0000 0000 _H
CAN_NPCR2	CAN Node 2 Port Control Register	F000 440C _H	U, SV	U, SV	0000 0000 _H
CAN_NBTR2	CAN Node 2 Bit Timing Register	F000 4410 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-24 Address Map of CAN (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
CAN_NECNT2	CAN Node 2 Error Counter Register	F000 4414 _H	U, SV	U, SV	0060 0000 _H
CAN_NFCR2	CAN Node 2 Frame Counter Register	F000 4418 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 441C _H - F000 44FC _H	nBE	nBE	–

CAN Node 3 Registers

CAN_NCR3	CAN Node 3 Control Register	F000 4500 _H	U, SV	U, SV	0000 0001 _H
CAN_NSR3	CAN Node 3 Status Register	F000 4504 _H	U, SV	U, SV	0000 0000 _H
CAN_NIPR3	CAN Node 3 Interrupt Pointer Register	F000 4508 _H	U, SV	U, SV	0000 0000 _H
CAN_NPCR3	CAN Node 3 Port Control Register	F000 450C _H	U, SV	U, SV	0000 0000 _H
CAN_NBTR3	CAN Node 3 Bit Timing Register	F000 4510 _H	U, SV	U, SV	0000 0000 _H
CAN_NECNT3	CAN Node 3 Error Counter Register	F000 4514 _H	U, SV	U, SV	0060 0000 _H
CAN_NFCR3	CAN Node 3 Frame Counter Register	F000 4518 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 451C _H - F000 45FC _H	nBE	nBE	–

CAN Message Object 0

CAN_MOFCR0	CAN Message Object 0 Function Control Register	F000 4600 _H	U, SV	U, SV	0000 0000 _H
CAN_MOFGPR0	CAN Message Object 0 FIFO Gateway Pointer Register	F000 4604 _H	U, SV	U, SV	0000 0000 _H
CAN_MOIPR0	CAN Message Object 0 Interrupt Pointer Register	F000 4608 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-24 Address Map of CAN (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
CAN_ MOAMR0	CAN Message Object 0 Acceptance Mask Register	F000 460C _H	U, SV	U, SV	3FFF FFFF _H
CAN_ MODATAL0	CAN Message Object 0 Data Register Low	F000 4610 _H	U, SV	U, SV	0000 0000 _H
CAN_ MODATAH0	CAN Message Object 0 Data Register High	F000 4614 _H	U, SV	U, SV	0000 0000 _H
CAN_ MOAR0	CAN Message Object 0 Arbitration Register	F000 4618 _H	U, SV	U, SV	0000 0000 _H
CAN_ MOSTAT0	CAN Message Object 0 Status Register (Read)	F000 461C _H	U, SV	–	0100 0000 _H
CAN_ MOCTR0	CAN Message Object 0 Control Register (Write)		–	U, SV	–

CAN Message Object 1

CAN_ MOFCR1	CAN Message Object 1 Function Control Register	F000 4620 _H	U, SV	U, SV	0000 0000 _H
CAN_ MOFGPR1	CAN Message Object 1 FIFO Gateway Pointer Register	F000 4624 _H	U, SV	U, SV	0000 0000 _H
CAN_ MOIPR1	CAN Message Object 1 Interrupt Pointer Register	F000 4628 _H	U, SV	U, SV	0000 0000 _H
CAN_ MOAMR1	CAN Message Object 1 Acceptance Mask Register	F000 462C _H	U, SV	U, SV	3FFF FFFF _H
CAN_ MODATAL1	CAN Message Object 1 Data Register Low	F000 4630 _H	U, SV	U, SV	0000 0000 _H
CAN_ MODATAH1	CAN Message Object 1 Data Register High	F000 4634 _H	U, SV	U, SV	0000 0000 _H
CAN_ MOAR1	CAN Message Object 1 Arbitration Register	F000 4638 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-24 Address Map of CAN (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
CAN_ MOSTAT1	CAN Message Object 1 Status Register (Read)	F000 463C _H	U, SV	–	0200 0000 _H
CAN_ MOCTR1	CAN Message Object 1 Control Register (Write)		–	U, SV	–

CAN Message Object 2

CAN_ MOFCR2	CAN Message Object 2 Function Control Register	F000 4640 _H	U, SV	U, SV	0000 0000 _H
CAN_ MOFGPR2	CAN Message Object 2 FIFO Gateway Pointer Register	F000 4644 _H	U, SV	U, SV	0000 0000 _H
CAN_ MOIPR2	CAN Message Object 2 Interrupt Pointer Register	F000 4648 _H	U, SV	U, SV	0000 0000 _H
CAN_ MOAMR2	CAN Message Object 2 Acceptance Mask Register	F000 464C _H	U, SV	U, SV	3FFF FFFF _H
CAN_ MODATAL2	CAN Message Object 2 Data Register Low	F000 4650 _H	U, SV	U, SV	0000 0000 _H
CAN_ MODATAH2	CAN Message Object 2 Data Register High	F000 4654 _H	U, SV	U, SV	0000 0000 _H
CAN_ MOAR2	CAN Message Object 2 Arbitration Register	F000 4658 _H	U, SV	U, SV	0000 0000 _H
CAN_ MOSTAT2	CAN Message Object 2 Status Register (Read)	F000 465C _H	U, SV	–	0301 0000 _H
CAN_ MOCTR2	CAN Message Object 2 Control Register (Write)		–	U, SV	–

CAN Message Object n (n = 3-125)

CAN_ MOFCRn	CAN Message Object n Function Control Register	F000 4600 _H + n × 20 _H + 00 _H	U, SV	U, SV	0000 0000 _H
CAN_ MOFGPRn	CAN Message Object n FIFO Gateway Pointer Register	F000 4600 _H + n × 20 _H + 04 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-24 Address Map of CAN (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
CAN_ MOIPRn	CAN Message Object n Interrupt Pointer Register	F000 4600 _H + n × 20 _H + 08 _H	U, SV	U, SV	0000 0000 _H
CAN_ MOAMRn	CAN Message Object n Acceptance Mask Register	F000 4600 _H + n × 20 _H + 0C _H	U, SV	U, SV	3FFF FFFF _H
CAN_ MODATALn	CAN Message Object n Data Register Low	F000 4600 _H + n × 20 _H + 10 _H	U, SV	U, SV	0000 0000 _H
CAN_ MODATAHn	CAN Message Object n Data Register High	F000 4600 _H + n × 20 _H + 14 _H	U, SV	U, SV	0000 0000 _H
CAN_ MOARn	CAN Message Object n Arbitration Register	F000 4600 _H + n × 20 _H + 18 _H	U, SV	U, SV	0000 0000 _H
CAN_ MOSTATn	CAN Message Object n Status Register (Read)	F000 4600 _H + n × 20 _H	U, SV	–	(n+1) << 24 + (n-1) << 16
CAN_ MOCTRn	CAN Message Object n Control Register (Write)	+ 1C _H	–	U, SV	–

CAN Message Object 126

CAN_ MOFCR126	CAN Message Object 126 Function Control Register	F000 55C0 _H	U, SV	U, SV	0000 0000 _H
CAN_ MOFGPR 126	CAN Message Object 126 FIFO Gateway Pointer Register	F000 55C4 _H	U, SV	U, SV	0000 0000 _H
CAN_ MOIPR126	CAN Message Object 126 Interrupt Pointer Register	F000 55C8 _H	U, SV	U, SV	0000 0000 _H
CAN_ MOAMR126	CAN Message Object 126 Acceptance Mask Register	F000 55CC _H	U, SV	U, SV	3FFF FFFF _H
CAN_ MODATAL 126	CAN Message Object 126 Data Register Low	F000 55D0 _H	U, SV	U, SV	0000 0000 _H

Table 18-24 Address Map of CAN (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
CAN_ MODATAH 126	CAN Message Object 126 Data Register High	F000 55D4 _H	U, SV	U, SV	0000 0000 _H
CAN_ MOAR126	CAN Message Object 126 Arbitration Register	F000 55D8 _H	U, SV	U, SV	0000 0000 _H
CAN_ MOSTAT 126	CAN Message Object 126 Status Register (Read)	F000 55DC _H	U, SV	–	7F7D 0000 _H
CAN_ MOCTR126	CAN Message Object 126 Control Register (Write)		–	U, SV	–

CAN Message Object 127

CAN_ MOFCR127	CAN Message Object 127 Function Control Register	F000 55E0 _H	U, SV	U, SV	0000 0000 _H
CAN_ MOFGPR 127	CAN Message Object 127 FIFO Gateway Pointer Register	F000 55E4 _H	U, SV	U, SV	0000 0000 _H
CAN_ MOIPR127	CAN Message Object 127 Interrupt Pointer Register	F000 55E8 _H	U, SV	U, SV	0000 0000 _H
CAN_ MOAMR127	CAN Message Object 127 Acceptance Mask Register	F000 55EC _H	U, SV	U, SV	3FFF FFFF _H
CAN_ MOAR127	CAN Message Object 127 Arbitration Register	F000 55F0 _H	U, SV	U, SV	0000 0000 _H
CAN_ MODATAL 127	CAN Message Object 127 Data Register Low	F000 55F4 _H	U, SV	U, SV	0000 0000 _H
CAN_ MODATAH 127	CAN Message Object 127 Data Register High	F000 55F8 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-24 Address Map of CAN (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
CAN_ MOSTAT 127	CAN Message Object 127 Status Register (Read)	F000 55FC _H	U, SV	–	7F7E 0000 _H
CAN_ MOCTR127	CAN Message Object 127 Control Register (Write)		–	U, SV	–
TTCAN Scheduler Memory					
–	TTCAN Scheduler Memory	F000 5600 _H - F000 57F8 _H	U, SV	U, SV	0000 0000 _H
CAN_ STPTR0	CAN Scheduler Start Pointer Node 0 Register	F000 57FC _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F000 5800 _H - F000 5FFC _H	nBE	nBE	–

Register Overview

Table 18-25 Address Map of PCP

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Peripheral Control Processor (PCP)					
PCP_CLC	PCP Clock Control Register	F004 3F00 _H	U, SV	SV, E, 32	0000 0000 _H
–	Reserved	F004 3F04 _H	BE	BE	–
PCP_ID	PCP Module Identification Register	F004 3F08 _H	U, SV	SV, 32	0020 C0XX _H
–	Reserved	F004 3F0C _H	BE	BE	–
PCP_CS	PCP Control/Status Register	F004 3F10 _H	U, SV, 32	SV, E, 32	0000 0000 _H
PCP_ES	PCP Error/Debug Status Register	F004 3F14 _H	U, SV, 32	SV, 32	0000 0000 _H
–	Reserved	F004 3F18 _H - F004 3F1C _H	BE	BE	–
PCP_ICR	PCP Interrupt Control Register	F004 3F20 _H	U, SV, 32	SV, 32	0000 0000 _H
PCP_ITR	PCP Interrupt Threshold Control Register	F004 3F24 _H	U, SV, 32	SV, 32	0000 0000 _H
PCP_ICON	PCP Interrupt Configuration Register	F004 3F28 _H	U, SV, 32	BE	0000 03E4 _H
PCP_SSR	PCP Stall Status Register	F004 3F2C _H	U, SV, 32	SV, 32	0000 0000 _H
–	Reserved; this location must not be written.	F004 3F30 _H	U, SV, 32	SV, 32	–
–	Reserved	F004 3F34 _H - F004 3FCC _H	BE	BE	–
PCP_SRC11	PCP Service Request Control Register 11	F004 3FD0 _H	U, SV, 32	SV, 32	0000 1400 _H
PCP_SRC10	PCP Service Request Control Register 10	F004 3FD4 _H	U, SV, 32	SV, 32	0000 1400 _H
PCP_SRC9	PCP Service Request Control Register 9	F004 3FD8 _H	U, SV, 32	SV, 32	0000 1400 _H
PCP_SRC8	PCP Service Request Control Register 8	F004 3FDC _H	U, SV, 32	SV, 32	0000 1000 _H

Register Overview

Table 18-25 Address Map of PCP (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
PCP_SRC7	PCP Service Request Control Register 7	F004 3FE0 _H	U, SV, 32	SV, 32	0000 1000 _H
PCP_SRC6	PCP Service Request Control Register 6	F004 3FE4 _H	U, SV, 32	SV, 32	0000 1000 _H
PCP_SRC5	PCP Service Request Control Register 5	F004 3FE8 _H	U, SV, 32	SV, 32	0000 1000 _H
PCP_SRC4	PCP Service Request Control Register 4	F004 3FEC _H	U, SV, 32	SV, 32	0000 1000 _H
PCP_SRC3	PCP Service Request Control Register 3	F004 3FF0 _H	U, SV, 32	SV, 32	0000 1400 _H
PCP_SRC2	PCP Service Request Control Register 2	F004 3FF4 _H	U, SV, 32	SV, 32	0000 1400 _H
PCP_SRC1	PCP Service Request Control Register 1	F004 3FF8 _H	U, SV, 32	SV, 32	0000 1000 _H
PCP_SRC0	PCP Service Request Control Register 0	F004 3FFC _H	U, SV, 32	SV, 32	0000 1000 _H

Table 18-26 Address Map of RBCU

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Remote Peripheral Bus Control Unit (RBCU)					
–	Reserved	F010 0000 _H - F010 0004 _H	BE	BE	–
RBCU_ID	RBCU Module Identification Register	F010 0008 _H	U, SV	BE	0000 6AXX _H
–	Reserved	F010 000C _H	BE	BE	–
RBCU_CON	RBCU Control Register	F010 0010 _H	U, SV	SV	4009 FFFF _H
–	Reserved	F010 0014 _H - F010 001C _H	BE	BE	–
RBCU_ECON	RBCU Error Control Capture Register	F010 0020 _H	U, SV	SV	0000 0000 _H
RBCU_EADD	RBCU Error Address Capture Register	F010 0024 _H	U, SV	SV	0000 0000 _H
RBCU_EDAT	RBCU Error Data Capture Register	F010 0028 _H	U, SV	SV	0000 0000 _H
–	Reserved	F010 002C _H	BE	BE	–
RBCU_DBCNTL	RBCU Debug Control Register	F010 0030 _H	U, SV	SV	0000 7003 _H
RBCU_DBGRNT	RBCU Debug Grant Mask Register	F010 0034 _H	U, SV	SV	0000 FFFF _H
RBCU_DBADR1	RBCU Debug Address 1 Register	F010 0038 _H	U, SV	SV	0000 0000 _H
RBCU_DBADR2	RBCU Debug Address 2 Register	F010 003C _H	U, SV	SV	0000 0000 _H
RBCU_DBBOS	RBCU Debug Bus Operation Signals Register	F010 0040 _H	U, SV	SV	0000 0000 _H
RBCU_DBGNTT	RBCU Debug Trapped Master Register	F010 0044 _H	U, SV	BE	FFFF FFFF _H
RBCU_DBADRT	RBCU Debug Trapped Address Register	F010 0048 _H	U, SV	BE	0000 0000 _H

Register Overview

Table 18-26 Address Map of RBCU (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
RBCU_ DBBOST	RBCU Debug Trapped Bus Operation Signals Register	F010 004C _H	U, SV	BE	0000 3180 _H
–	Reserved	F010 0050 _H - F010 00F8 _H	BE	BE	–
RBCU_ SRC	RBCU Service Request Control Register	F010 00FC _H	U, SV	SV	0000 0000 _H

Table 18-27 Address Map of SSC0/SSC1

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Synchronous Serial Interface 0 (SSC0)					
SSC0_CLC	SSC0 Clock Control Register	F010 0100 _H	U, SV	SV, E	0000 0003 _H
SSC0_PISEL	SSC0 Port Input Select Register	F010 0104 _H	U, SV	U, SV	0000 0000 _H
SSC0_ID	SSC0 Module Identification Register	F010 0108 _H	U, SV	BE	0000 45XX _H
SSC0_FDR	SSC0 Fractional Divider Register	F010 010C _H	U, SV	SV, E	0000 0000 _H
SSC0_CON	SSC0 Control Register	F010 0110 _H	U, SV	U, SV	0000 0000 _H
SSC0_BR	SSC0 Baud Rate Timer Reload Register	F010 0114 _H	U, SV	U, SV	0000 0000 _H
SSC0_SSOC	SSC0 Slave Select Output Control Register	F010 0118 _H	U, SV	U, SV	0000 0000 _H
SSC0_SSOTC	SSC0 Slave Select Output Timing Control Register	F010 011C _H	U, SV	U, SV	0000 0000 _H
SSC0_TB	SSC0 Transmit Buffer Register	F010 0120 _H	U, SV	U, SV	0000 0000 _H
SSC0_RB	SSC0 Receive Buffer Register	F010 0124 _H	U, SV	U, SV	0000 0000 _H
SSC0_STAT	SSC0 Status Register	F010 0128 _H	U, SV	U, SV	0000 0000 _H
SSC0_EFM	SSC0 Error Flag Modification Register	F010 012C _H	U, SV	U, SV	0000 0000 _H
SSC0_RXFCN	SSC0 Receive FIFO Control Register	F010 0130 _H	U, SV	U, SV	0000 0100 _H
SSC0_TXFCN	SSC0 Transmit FIFO Control Register	F010 0134 _H	U, SV	U, SV	0000 0100 _H
SSC0_FSTAT	SSC0 FIFO Status Register	F010 0138 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 013C _H - F010 01F0 _H	BE	BE	–

Register Overview

Table 18-27 Address Map of SSC0/SSC1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
SSC0_TSRC	SSC0 Transmit Interrupt Service Req. Control Reg.	F010 01F4 _H	U, SV	SV	0000 0000 _H
SSC0_RSRC	SSC0 Receive Interrupt Service Req. Control Reg.	F010 01F8 _H	U, SV	SV	0000 0000 _H
SSC0_ESRC	SSC0 Error Interrupt Service Req. Control Reg.	F010 01FC _H	U, SV	SV	0000 0000 _H

Synchronous Serial Interface 1 (SSC1)

SSC1_CLC	SSC1 Clock Control Register	F010 0200 _H	U, SV	SV, E	0000 0003 _H
SSC1_PISEL	SSC1 Port Input Select Register	F010 0204 _H	U, SV	U, SV	0000 0000 _H
SSC1_ID	SSC1 Module Identification Register	F010 0208 _H	U, SV	BE	0000 45XX _H
SSC1_FDR	SSC1 Fractional Divider Register	F010 020C _H	U, SV	SV, E	0000 0000 _H
SSC1_CON	SSC1 Control Register	F010 0210 _H	U, SV	U, SV	0000 0000 _H
SSC1_BR	SSC1 Baud Rate Timer Reload Register	F010 0214 _H	U, SV	U, SV	0000 0000 _H
SSC1_SSOC	SSC1 Slave Select Output Control Register	F010 0218 _H	U, SV	U, SV	0000 0000 _H
SSC1_SSOTC	SSC1 Slave Select Output Timing Control Register	F010 021C _H	U, SV	U, SV	0000 0000 _H
SSC1_TB	SSC1 Transmit Buffer Register	F010 0220 _H	U, SV	U, SV	0000 0000 _H
SSC1_RB	SSC1 Receive Buffer Register	F010 0224 _H	U, SV	U, SV	0000 0000 _H
SSC1_STAT	SSC1 Status Register	F010 0228 _H	U, SV	U, SV	0000 0000 _H
SSC1_EFM	SSC1 Error Flag Modification Register	F010 022C _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-27 Address Map of SSC0/SSC1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
–	Reserved	F010 0230 _H - F010 02F0 _H	BE	BE	–
SSC1_ TSRC	SSC1 Transmit Interrupt Service Req. Control Reg.	F010 02F4 _H	U, SV	SV	0000 0000 _H
SSC1_ RSRC	SSC1 Receive Interrupt Service Req. Control Reg.	F010 02F8 _H	U, SV	SV	0000 0000 _H
SSC1_ ESRC	SSC1 Error Interrupt Service Req. Control Reg.	F010 02FC _H	U, SV	SV	0000 0000 _H

Register Overview

Table 18-28 Address Map of FADC

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Fast Analog-to-Digital Converter (FADC)					
FADC_CLC	FADC Clock Control Register	F010 0300 _H	U, SV	SV, E	0000 0003 _H
–	Reserved	F010 0304 _H	BE	BE	–
FADC_ID	FADC Module Identification Register	F010 0308 _H	U, SV	nBE	0027 C0XX _H
FADC_FDR	FADC Fractional Divider Register	F010 030C _H	U, SV	SV, E	0000 0000 _H
FADC_CRSR	FADC Conversion Request Status Register	F010 0310 _H	U, SV	U, SV	0000 0000 _H
FADC_FMR	FADC Flag Modification Register	F010 0314 _H	U, SV	U, SV	0000 0000 _H
FADC_NCTR	FADC Neighbor Channel Trigger Register	F010 0318 _H	U, SV	U, SV	0000 0000 _H
FADC_GCR	FADC Global Control Register	F010 031C _H	U, SV	U, SV	0000 0000 _H
FADC_CFGR0	FADC Channel 0 Configuration Register	F010 0320 _H	U, SV	U, SV	0000 0000 _H
FADC_CFGR1	FADC Channel 1 Configuration Register	F010 0324 _H	U, SV	U, SV	0000 0000 _H
FADC_CFGR2	FADC Channel 2 Configuration Register	F010 0328 _H	U, SV	U, SV	0000 0000 _H
FADC_CFGR3	FADC Channel 3 Configuration Register	F010 032C _H	U, SV	U, SV	0000 0000 _H
FADC_ACR0	FADC Channel 0 Analog Control Register	F010 0330 _H	U, SV	U, SV	0000 0000 _H
FADC_ACR1	FADC Channel 1 Analog Control Register	F010 0334 _H	U, SV	U, SV	0000 0000 _H
FADC_ACR2	FADC Channel 2 Analog Control Register	F010 0338 _H	U, SV	U, SV	0000 0000 _H
FADC_ACR3	FADC Channel 3 Analog Control Register	F010 033C _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-28 Address Map of FADC (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
FADC_RCH0	FADC Channel 0 Conversion Result Reg.	F010 0340 _H	U, SV	U, SV	0000 0000 _H
FADC_RCH1	FADC Channel 1 Conversion Result Reg.	F010 0344 _H	U, SV	U, SV	0000 0000 _H
FADC_RCH2	FADC Channel 2 Conversion Result Reg.	F010 0348 _H	U, SV	U, SV	0000 0000 _H
FADC_RCH3	FADC Channel 3 Conversion Result Reg.	F010 034C _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 0350 _H - F010 035C _H	BE	BE	–
FADC_FCR0	FADC Filter 0 Control Register	F010 0360 _H	U, SV	U, SV	0000 0000 _H
FADC_CRR0	FADC Filter 0 Current Result Register	F010 0364 _H	U, SV	U, SV	0000 0000 _H
FADC_IRR10	FADC Filter 0 Intermediate Result Register 1	F010 0368 _H	U, SV	U, SV	0000 0000 _H
FADC_IRR20	FADC Filter 0 Intermediate Result Register 2	F010 036C _H	U, SV	U, SV	0000 0000 _H
FADC_IRR30	FADC Filter 0 Intermediate Result Register 3	F010 0370 _H	U, SV	U, SV	0000 0000 _H
FADC_FRR0	FADC Filter 0 Final Result Register	F010 0374 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 0378 _H - F010 037C _H	BE	BE	–
FADC_FCR1	FADC Filter 1 Control Register	F010 0380 _H	U, SV	U, SV	0000 0000 _H
FADC_CRR1	FADC Filter 1 Current Result Register	F010 0384 _H	U, SV	U, SV	0000 0000 _H
FADC_IRR11	FADC Filter 1 Intermediate Result Register 1	F010 0388 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-28 Address Map of FADC (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
–	Reserved	F010 038C _H - F010 0390 _H	nBE	nBE	–
FADC_ FRR1	FADC Filter 1 Final Result Register	F010 0394 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 0398 _H - F010 03EC _H	BE	BE	–
FADC_ SRC3	FADC Service Request Control Register 3	F010 03F0 _H	U, SV	U, SV	0000 0000 _H
FADC_ SRC2	FADC Service Request Control Register 2	F010 03F4 _H	U, SV	U, SV	0000 0000 _H
FADC_ SRC1	FADC Service Request Control Register 1	F010 03F8 _H	U, SV	U, SV	0000 0000 _H
FADC_ SRC0	FADC Service Request Control Register 0	F010 03FC _H	U, SV	U, SV	0000 0000 _H

Table 18-29 Address Map of ADC0/ADC1

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Analog-to-Digital Converter 0 (ADC0)					
ADC0_CLC	ADC Clock Control Register	F010 0400 _{HF}	U, SV	SV, E	0000 0003 _H
–	Reserved	F010 0404 _H	BE	BE	–
ADC0_ID	ADC Module Identification Register	F010 0408 _H	U, SV	BE	0030 C0XX _H
ADC0_FDR	ADC Fractional Divider Register	F010 040C _H	U, SV	SV, E	0000 0000 _H
ADC0_CHCON0	ADC0 Channel Control Register 0	F010 0410 _H	U, SV	U, SV	0000 0000 _H
ADC0_CHCON1	ADC0 Channel Control Register 1	F010 0414 _H	U, SV	U, SV	0000 0000 _H
ADC0_CHCON2	ADC0 Channel Control Register 2	F010 0418 _H	U, SV	U, SV	0000 0000 _H
ADC0_CHCON3	ADC0 Channel Control Register 3	F010 041C _H	U, SV	U, SV	0000 0000 _H
ADC0_CHCON4	ADC0 Channel Control Register 4	F010 0420 _H	U, SV	U, SV	0000 0000 _H
ADC0_CHCON5	ADC0 Channel Control Register 5	F010 0424 _H	U, SV	U, SV	0000 0000 _H
ADC0_CHCON6	ADC0 Channel Control Register 6	F010 0428 _H	U, SV	U, SV	0000 0000 _H
ADC0_CHCON7	ADC0 Channel Control Register 7	F010 042C _H	U, SV	U, SV	0000 0000 _H
ADC0_CHCON8	ADC0 Channel Control Register 8	F010 0430 _H	U, SV	U, SV	0000 0000 _H
ADC0_CHCON9	ADC0 Channel Control Register 9	F010 0434 _H	U, SV	U, SV	0000 0000 _H
ADC0_CHCON10	ADC0 Channel Control Register 10	F010 0438 _H	U, SV	U, SV	0000 0000 _H
ADC0_CHCON11	ADC0 Channel Control Register 11	F010 043C _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-29 Address Map of ADC0/ADC1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
ADC0_CHCON12	ADC0 Channel Control Register 12	F010 0440 _H	U, SV	U, SV	0000 0000 _H
ADC0_CHCON13	ADC0 Channel Control Register 13	F010 0444 _H	U, SV	U, SV	0000 0000 _H
ADC0_CHCON14	ADC0 Channel Control Register 14	F010 0448 _H	U, SV	U, SV	0000 0000 _H
ADC0_CHCON15	ADC0 Channel Control Register 15	F010 044C _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 0450 _H - F010 0480 _H	BE	BE	–
ADC0_AP	ADC0 Arbitration Participation Register	F010 0484 _H	U, SV	U, SV	0000 0000 _H
ADC0_SAL	ADC0 Source Arbitration Level Register	F010 0488 _H	U, SV	U, SV	0103 4067 _H
ADC0_TTC	ADC0 Timer Trigger Control Register	F010 048C _H	U, SV	U, SV	0000 0000 _H
ADC0_EXTC	ADC0 External Trigger Control Register	F010 0490 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 0494 _H	BE	BE	–
ADC0_SCON	ADC0 Source Control Register	F010 0498 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 049C _H	BE	BE	–
–	Reserved; these locations must not be written.	F010 04A0 _H - F010 04DC _H	nBE	nBE	–
–	Reserved	F010 04E0 _H - F010 04FC _H	BE	BE	–
ADC0_LCCON0	ADC0 Limit Check Control Register 0	F010 0500 _H	U, SV	U, SV	0000 0000 _H
ADC0_LCCON1	ADC0 Limit Check Control Register 1	F010 0504 _H	U, SV	U, SV	0000 0000 _H
ADC0_LCCON2	ADC0 Limit Check Control Register 2	F010 0508 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-29 Address Map of ADC0/ADC1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
ADC0_LCCON3	ADC0 Limit Check Control Register 3	F010 050C _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 0510 _H	BE	BE	–
ADC0_TCON	ADC0 Timer Control Register	F010 0514 _H	U, SV	U, SV	0000 0000 _H
ADC0_CHIN	ADC0 Channel Injection Control Register	F010 0518 _H	U, SV	U, SV	0000 0000 _H
ADC0_QR	ADC0 Queue Register	F010 051C _H	U, SV	U, SV	0000 0000 _H
ADC0_CON	ADC0 Converter Control Register	F010 0520 _H	U, SV	U, SV	0000 0001 _H
ADC0_SCN	ADC0 Auto Scan Control Register	F010 0524 _H	U, SV	U, SV	0000 0000 _H
ADC0_REQ0	ADC0 Conversion Request Register SW0	F010 0528 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 052C _H	BE	BE	–
ADC0_CHSTAT0	ADC0 Channel Status Register 0	F010 0530 _H	U, SV	U, SV	0000 0000 _H
ADC0_CHSTAT1	ADC0 Channel Status Register 1	F010 0534 _H	U, SV	U, SV	0000 0000 _H
ADC0_CHSTAT2	ADC0 Channel Status Register 2	F010 0538 _H	U, SV	U, SV	0000 0000 _H
ADC0_CHSTAT3	ADC0 Channel Status Register 3	F010 053C _H	U, SV	U, SV	0000 0000 _H
ADC0_CHSTAT4	ADC0 Channel Status Register 4	F010 0540 _H	U, SV	U, SV	0000 0000 _H
ADC0_CHSTAT5	ADC0 Channel Status Register 5	F010 0544 _H	U, SV	U, SV	0000 0000 _H
ADC0_CHSTAT6	ADC0 Channel Status Register 6	F010 0548 _H	U, SV	U, SV	0000 0000 _H
ADC0_CHSTAT7	ADC0 Channel Status Register 7	F010 054C _H	U, SV	U, SV	0000 0000 _H
ADC0_CHSTAT8	ADC0 Channel Status Register 8	F010 0550 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-29 Address Map of ADC0/ADC1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
ADC0_ CHSTAT9	ADC0 Channel Status Register 9	F010 0554 _H	U, SV	U, SV	0000 0000 _H
ADC0_ CHSTAT10	ADC0 Channel Status Register 10	F010 0558 _H	U, SV	U, SV	0000 0000 _H
ADC0_ CHSTAT11	ADC0 Channel Status Register 11	F010 055C _H	U, SV	U, SV	0000 0000 _H
ADC0_ CHSTAT12	ADC0 Channel Status Register 12	F010 0560 _H	U, SV	U, SV	0000 0000 _H
ADC0_ CHSTAT13	ADC0 Channel Status Register 13	F010 0564 _H	U, SV	U, SV	0000 0000 _H
ADC0_ CHSTAT14	ADC0 Channel Status Register 14	F010 0568 _H	U, SV	U, SV	0000 0000 _H
ADC0_ CHSTAT15	ADC0 Channel Status Register 15	F010 056C _H	U, SV	U, SV	0000 0000 _H
ADC0_ QUEUE0	ADC0 Queue Status Register	F010 0570 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 0574 _H - F010 057C _H	BE	BE	–
ADC0_ SW0CRP	ADC0 Software SW0 Conversion Request Pending Register	F010 0580 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 0584 _H	BE	BE	–
ADC0_ ASCRP	ADC0 Auto Scan Conversion Request Pending Register	F010 0588 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 058C _H	BE	BE	–
ADC0_ SYSTAT	ADC0 Synchronization Status Register	F010 0590 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 0594 _H - F010 059C _H	BE	BE	–
–	Reserved; these locations must not be written.	F010 05A0 _H - F010 05A4 _H	nBE	nBE	–

Register Overview

Table 18-29 Address Map of ADC0/ADC1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
–	Reserved	F010 05A8 _H - F010 05AC _H	BE	BE	–
ADC0_ TSTAT	ADC0 Timer Status Register	F010 05B0 _H	U, SV	U, SV	0000 0000 _H
ADC0_ STAT	ADC0 Converter Status Register	F010 05B4 _H	U, SV	U, SV	0000 0000 _H
ADC0_ TCRP	ADC0 Timer Conversion Request Pending Register	F010 05B8 _H	U, SV	U, SV	0000 0000 _H
ADC0_ EXCRP	ADC0 External Conversion Request Pending Register	F010 05BC _H	U, SV	U, SV	0000 0000 _H
–	Reserved; this location must not be written.	F010 05C0 _H	nBE	nBE	–
–	Reserved	F010 05C4 _H F010 05CC _H	BE	BE	–
ADC0_ MSS0	ADC0 Module Service Request Status Register 0	F010 05D0 _H	U, SV	U, SV	0000 0000 _H
ADC0_ MSS1	ADC0 Module Service Request Status Register 1	F010 05D4 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 05D8 _H	BE	BE	–
ADC0_ SRNP	ADC0 Service Request Node Pointer Register	F010 05DC _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 05E0 _H - F010 05EC _H	BE	BE	–
ADC0_ SRC3	ADC0 Service Request Control Register 3	F010 05F0 _H	U, SV	U, SV	0000 0000 _H
ADC0_ SRC2	ADC0 Service Request Control Register 2	F010 05F4 _H	U, SV	U, SV	0000 0000 _H
ADC0_ SRC1	ADC0 Service Request Control Register 1	F010 05F8 _H	U, SV	U, SV	0000 0000 _H
ADC0_ SRC0	ADC0 Service Request Control Register 0	F010 05FC _H	U, SV	U, SV	0000 0000 _H

Table 18-29 Address Map of ADC0/ADC1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Analog-to-Digital Converter 1 (ADC1)					
–	Reserved	F010 0600 _H	nBE	nBE	–
–	Reserved	F010 0604 _H	BE	BE	–
–	Reserved	F010 0608 _H	BE	BE	–
–	Reserved	F010 060C _H	nBE	nBE	–
ADC1_ CHCON0	ADC1 Channel Control Register 0	F010 0610 _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHCON1	ADC1 Channel Control Register 1	F010 0614 _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHCON2	ADC1 Channel Control Register 2	F010 0618 _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHCON3	ADC1 Channel Control Register 3	F010 061C _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHCON4	ADC1 Channel Control Register 4	F010 0620 _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHCON5	ADC1 Channel Control Register 5	F010 0624 _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHCON6	ADC1 Channel Control Register 6	F010 0628 _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHCON7	ADC1 Channel Control Register 7	F010 062C _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHCON8	ADC1 Channel Control Register 8	F010 0630 _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHCON9	ADC1 Channel Control Register 9	F010 0634 _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHCON10	ADC1 Channel Control Register 10	F010 0638 _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHCON11	ADC1 Channel Control Register 11	F010 063C _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHCON12	ADC1 Channel Control Register 12	F010 0640 _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHCON13	ADC1 Channel Control Register 13	F010 0644 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-29 Address Map of ADC0/ADC1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
ADC1_ CHCON14	ADC1 Channel Control Register 14	F010 0648 _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHCON15	ADC1 Channel Control Register 15	F010 064C _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 0650 _H - F010 0680 _H	BE	BE	–
ADC1_ AP	ADC1 Arbitration Participation Register	F010 0684 _H	U, SV	U, SV	0000 0000 _H
ADC1_ SAL	ADC1 Source Arbitration Level Register	F010 0688 _H	U, SV	U, SV	0103 4067 _H
ADC1_ TTC	ADC1 Timer Trigger Control Register	F010 068C _H	U, SV	U, SV	0000 0000 _H
ADC1_ EXTC	ADC1 External Trigger Control Register	F010 0690 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 0694 _H	BE	BE	–
ADC1_ SCON	ADC1 Source Control Register	F010 0698 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 069C _H	BE	BE	–
–	Reserved; these locations must not be written.	F010 06A0 _H - F010 06DC _H	nBE	nBE	–
–	Reserved	F010 06E0 _H - F010 06FC _H	BE	BE	–
ADC1_ LCCON0	ADC1 Limit Check Control Register 0	F010 0700 _H	U, SV	U, SV	0000 0000 _H
ADC1_ LCCON1	ADC1 Limit Check Control Register 1	F010 0704 _H	U, SV	U, SV	0000 0000 _H
ADC1_ LCCON2	ADC1 Limit Check Control Register 2	F010 0708 _H	U, SV	U, SV	0000 0000 _H
ADC1_ LCCON3	ADC1 Limit Check Control Register 3	F010 070C _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 0710 _H	BE	BE	–
ADC1_ TCON	ADC1 Timer Control Register	F010 0714 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-29 Address Map of ADC0/ADC1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
ADC1_ CHIN	ADC1 Channel Injection Control Register	F010 0718 _H	U, SV	U, SV	0000 0000 _H
ADC1_QR	ADC1 Queue Register	F010 071C _H	U, SV	U, SV	0000 0000 _H
ADC1_CON	ADC1 Converter Control Register	F010 0720 _H	U, SV	U, SV	0000 0001 _H
ADC1_SCN	ADC1 Auto Scan Control Register	F010 0724 _H	U, SV	U, SV	0000 0000 _H
ADC1_ REQ0	ADC1 Conversion Request Register SW0	F010 0728 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 072C _H	BE	BE	–
ADC1_ CHSTAT0	ADC1 Channel Status Register 0	F010 0730 _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHSTAT1	ADC1 Channel Status Register 1	F010 0734 _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHSTAT2	ADC1 Channel Status Register 2	F010 0738 _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHSTAT3	ADC1 Channel Status Register 3	F010 073C _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHSTAT4	ADC1 Channel Status Register 4	F010 0740 _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHSTAT5	ADC1 Channel Status Register 5	F010 0744 _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHSTAT6	ADC1 Channel Status Register 6	F010 0748 _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHSTAT7	ADC1 Channel Status Register 7	F010 074C _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHSTAT8	ADC1 Channel Status Register 8	F010 0750 _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHSTAT9	ADC1 Channel Status Register 9	F010 0754 _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHSTAT10	ADC1 Channel Status Register 10	F010 0758 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-29 Address Map of ADC0/ADC1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
ADC1_ CHSTAT11	ADC1 Channel Status Register 11	F010 075C _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHSTAT12	ADC1 Channel Status Register 12	F010 0760 _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHSTAT13	ADC1 Channel Status Register 13	F010 0764 _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHSTAT14	ADC1 Channel Status Register 14	F010 0768 _H	U, SV	U, SV	0000 0000 _H
ADC1_ CHSTAT15	ADC1 Channel Status Register 15	F010 076C _H	U, SV	U, SV	0000 0000 _H
ADC1_ QUEUE0	ADC1 Queue Status Register	F010 0770 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 0774 _H - F010 077C _H	BE	BE	–
ADC1_ SW0CRP	ADC1 Software SW0 Conversion Request Pending Register	F010 0780 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 0784 _H	BE	BE	–
ADC1_ ASCRP	ADC1 Auto Scan Conversion Request Pending Register	F010 0788 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 078C _H	BE	BE	–
ADC1_ SYSTAT	ADC1 Synchronization Status Register	F010 0790 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 0794 _H - F010 079C _H	BE	BE	–
–	Reserved; these locations must not be written.	F010 07A0 _H - F010 07A4 _H	nBE	nBE	–
–	Reserved	F010 07A8 _H - F010 07AC _H	BE	BE	–
ADC1_ TSTAT	ADC1 Timer Status Register	F010 07B0 _H	U, SV	U, SV	0000 0000 _H

Register Overview

Table 18-29 Address Map of ADC0/ADC1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
ADC1_STAT	ADC1 Converter Status Register	F010 07B4 _H	U, SV	U, SV	0000 0000 _H
ADC1_TCRP	ADC1 Timer Conversion Request Pending Register	F010 07B8 _H	U, SV	U, SV	0000 0000 _H
ADC1_EXCRP	ADC1 External Conversion Request Pending Register	F010 07BC _H	U, SV	U, SV	0000 0000 _H
–	Reserved; this location must not be written.	F010 07C0 _H	nBE	nBE	–
–	Reserved	F010 07C4 _H F010 07CC _H	BE	BE	–
ADC1_MSS0	ADC1 Module Service Request Status Register 0	F010 07D0 _H	U, SV	U, SV	0000 0000 _H
ADC1_MSS1	ADC1 Module Service Request Status Register 1	F010 07D4 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 07D8 _H	BE	BE	–
ADC1_SRNP	ADC1 Service Request Node Pointer Register	F010 07DC _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 07E0 _H - F010 07EC _H	BE	BE	–
ADC1_SRC3	ADC1 Service Request Control Register 3	F010 07F0 _H	U, SV	U, SV	0000 0000 _H
ADC1_SRC2	ADC1 Service Request Control Register 2	F010 07F4 _H	U, SV	U, SV	0000 0000 _H
ADC1_SRC1	ADC1 Service Request Control Register 1	F010 07F8 _H	U, SV	U, SV	0000 0000 _H
ADC1_SRC0	ADC1 Service Request Control Register 0	F010 07FC _H	U, SV	U, SV	0000 0000 _H

Table 18-30 Address Map of MLI0/MLI1

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Micro Link Interface 0 (MLI0)					
–	Reserved	F010 C000 _H	nBE	SV, E	–
–	Reserved	F010 C004 _H	nBE	nBE	–
MLI0_ID	MLI0 Module Identification Register	F010 C008 _H	U, SV	BE	0025 C0XX _H
MLI0_FDR	MLI0 Fractional Divider Register	F010 C00C _H	U, SV	SV, E	03FF 43FF _H
MLI0_TCR	MLI0 Transmitter Control Register	F010 C010 _H	U, SV	U, SV	0000 0110 _H
MLI0_TSTATR	MLI0 Transmitter Status Register	F010 C014 _H	U, SV	BE	0000 0000 _H
MLI0_TP0STATR	MLI0 Transmitter Pipe 0 Status Register	F010 C018 _H	U, SV	BE	0000 0000 _H
MLI0_TP1STATR	MLI0 Transmitter Pipe 1 Status Register	F010 C01C _H	U, SV	BE	0000 0000 _H
MLI0_TP2STATR	MLI0 Transmitter Pipe 2 Status Register	F010 C020 _H	U, SV	BE	0000 0000 _H
MLI0_TP3STATR	MLI0 Transmitter Pipe 3 Status Register	F010 C024 _H	U, SV	BE	0000 0000 _H
MLI0_TCMR	MLI0 Transmitter Command Register	F010 C028 _H	U, SV	U, SV	0000 0000 _H
MLI0_TRSTATR	MLI0 Transmitter Registers Status Register	F010 C02C _H	U, SV	BE	0000 0000 _H
MLI0_TP0AOFR	MLI0 Transmitter Pipe 0 Address Offset Register	F010 C030 _H	U, SV	BE	0000 0000 _H
MLI0_TP1AOFR	MLI0 Transmitter Pipe 1 Address Offset Register	F010 C034 _H	U, SV	BE	0000 0000 _H
MLI0_TP2AOFR	MLI0 Transmitter Pipe 2 Address Offset Register	F010 C038 _H	U, SV	BE	0000 0000 _H
MLI0_TP3AOFR	MLI0 Transmitter Pipe 3 Address Offset Register	F010 C03C _H	U, SV	BE	0000 0000 _H

Register Overview

Table 18-30 Address Map of MLI0/MLI1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
MLI0_TP0DATAR	MLI0 Transmitter Pipe 0 Data Register	F010 C040 _H	U, SV	BE	0000 0000 _H
MLI0_TP1DATAR	MLI0 Transmitter Pipe 1 Data Register	F010 C044 _H	U, SV	BE	0000 0000 _H
MLI0_TP2DATAR	MLI0 Transmitter Pipe 2 Data Register	F010 C048 _H	U, SV	BE	0000 0000 _H
MLI0_TP3DATAR	MLI0 Transmitter Pipe 3 Data Register	F010 C04C _H	U, SV	BE	0000 0000 _H
MLI0_TDRAR	MLI0 Transmitter Data Read Answer Register	F010 C050 _H	U, SV	U, SV	0000 0000 _H
MLI0_TP0BAR	MLI0 Transmitter Pipe 0 Base Address Register	F010 C054 _H	U, SV	U, SV	0000 0000 _H
MLI0_TP1BAR	MLI0 Transmitter Pipe 1 Base Address Register	F010 C058 _H	U, SV	U, SV	0000 0000 _H
MLI0_TP2BAR	MLI0 Transmitter Pipe 2 Base Address Register	F010 C05C _H	U, SV	U, SV	0000 0000 _H
MLI0_TP3BAR	MLI0 Transmitter Pipe 3 Base Address Register	F010 C060 _H	U, SV	U, SV	0000 0000 _H
MLI0_TCBAR	MLI0 Transmitter Copy Base Address Register	F010 C064 _H	U, SV	BE	0000 0000 _H
MLI0_RCR	MLI0 Receiver Control Register	F010 C068 _H	U, SV	U, SV	0100 0000 _H
MLI0_RP0BAR	MLI0 Receiver Pipe 0 Base Address Register	F010 C06C _H	U, SV	BE	0000 0000 _H
MLI0_RP1BAR	MLI0 Receiver Pipe 1 Base Address Register	F010 C070 _H	U, SV	BE	0000 0000 _H
MLI0_RP2BAR	MLI0 Receiver Pipe 2 Base Address Register	F010 C074 _H	U, SV	BE	0000 0000 _H
MLI0_RP3BAR	MLI0 Receiver Pipe 3 Base Address Register	F010 C078 _H	U, SV	BE	0000 0000 _H
MLI0_RP0STATR	MLI0 Receiver Pipe 0 Status Register	F010 C07C _H	U, SV	BE	0000 0000 _H

Register Overview

Table 18-30 Address Map of MLI0/MLI1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
MLI0_RP1STATR	MLI0 Receiver Pipe 1 Status Register	F010 C080 _H	U, SV	BE	0000 0000 _H
MLI0_RP2STATR	MLI0 Receiver Pipe 2 Status Register	F010 C084 _H	U, SV	BE	0000 0000 _H
MLI0_RP3STATR	MLI0 Receiver Pipe 3 Status Register	F010 C088 _H	U, SV	BE	0000 0000 _H
MLI0_RADRR	MLI0 Receiver Address Register	F010 C08C _H	U, SV	BE	0000 0000 _H
MLI0_RDATAR	MLI0 Receiver Data Register	F010 C090 _H	U, SV	BE	0000 0000 _H
MLI0_SCR	MLI0 Set Clear Register	F010 C094 _H	U, SV	U, SV	0000 0000 _H
MLI0_TIER	MLI0 Transmitter Interrupt Enable Register	F010 C098 _H	U, SV	U, SV	0000 0000 _H
MLI0_TISR	MLI0 Transmitter Interrupt Status Register	F010 C09C _H	U, SV	BE	0000 0000 _H
MLI0_TINPR	MLI0 Transmitter Interrupt Node Pointer Register	F010 C0A0 _H	U, SV	U, SV	0000 0000 _H
MLI0_RIER	MLI0 Receiver Interrupt Enable Register	F010 C0A4 _H	U, SV	U, SV	0000 0000 _H
MLI0_RISR	MLI0 Receiver Interrupt Status Register	F010 C0A8 _H	U, SV	BE	0000 0000 _H
MLI0_RINPR	MLI0 Receiver Interrupt Node Pointer Register	F010 C0AC _H	U, SV	U, SV	0000 0000 _H
MLI0_GINTR	MLI0 Global Interrupt Set Register	F010 C0B0 _H	U, SV	U, SV	0000 0000 _H
MLI0_OICR	MLI0 Output Input Control Register	F010 C0B4 _H	U, SV	U, SV	1000 8000 _H
MLI0_AER	MLI0 Access Enable Register	F010 C0B8 _H	U, SV	SV, E	0000 0000 _H
MLI0_ARR	MLI0 Access Range Register	F010 C0BC _H	U, SV	SV, E	0000 0000 _H
–	Reserved	F010 C0C0 _H - F010 C0FC _H	BE	BE	–

Register Overview

Table 18-30 Address Map of MLI0/MLI1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Micro Link Interface 1 (MLI1)					
–	Reserved	F010 C100 _H	nBE	SV, E	–
–	Reserved	F010 C104 _H	nBE	nBE	–
MLI1_ID	MLI1 Module Identification Register	F010 C108 _H	U, SV	BE	0025 C0XX _H
MLI1_FDR	MLI1 Fractional Divider Register	F010 C10C _H	U, SV	SV, E	03FF 43FF _H
MLI1_TCR	MLI1 Transmitter Control Register	F010 C110 _H	U, SV	U, SV	0000 0110 _H
MLI1_TSTATR	MLI1 Transmitter Status Register	F010 C114 _H	U, SV	BE	0000 0000 _H
MLI1_TP0STATR	MLI1 Transmitter Pipe 0 Status Register	F010 C118 _H	U, SV	BE	0000 0000 _H
MLI1_TP1STATR	MLI1 Transmitter Pipe 1 Status Register	F010 C11C _H	U, SV	BE	0000 0000 _H
MLI1_TP2STATR	MLI1 Transmitter Pipe 2 Status Register	F010 C120 _H	U, SV	BE	0000 0000 _H
MLI1_TP3STATR	MLI1 Transmitter Pipe 3 Status Register	F010 C124 _H	U, SV	BE	0000 0000 _H
MLI1_TCMDR	MLI1 Transmitter Command Register	F010 C128 _H	U, SV	U, SV	0000 0000 _H
MLI1_TRSTATR	MLI1 Transmitter Registers Status Register	F010 C12C _H	U, SV	BE	0000 0000 _H
MLI1_TP0AOFR	MLI1 Transmitter Pipe 0 Address Offset Register	F010 C130 _H	U, SV	BE	0000 0000 _H
MLI1_TP1AOFR	MLI1 Transmitter Pipe 1 Address Offset Register	F010 C134 _H	U, SV	BE	0000 0000 _H
MLI1_TP2AOFR	MLI1 Transmitter Pipe 2 Address Offset Register	F010 C138 _H	U, SV	BE	0000 0000 _H
MLI1_TP3AOFR	MLI1 Transmitter Pipe 3 Address Offset Register	F010 C13C _H	U, SV	BE	0000 0000 _H
MLI1_TP0DATAR	MLI1 Transmitter Pipe 0 Data Register	F010 C140 _H	U, SV	BE	0000 0000 _H

Register Overview

Table 18-30 Address Map of MLI0/MLI1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
MLI1_TP1DATAR	MLI1 Transmitter Pipe 1 Data Register	F010 C144 _H	U, SV	BE	0000 0000 _H
MLI1_TP2DATAR	MLI1 Transmitter Pipe 2 Data Register	F010 C148 _H	U, SV	BE	0000 0000 _H
MLI1_TP3DATAR	MLI1 Transmitter Pipe 3 Data Register	F010 C14C _H	U, SV	BE	0000 0000 _H
MLI1_TDRAR	MLI1 Transmitter Data Read Answer Register	F010 C150 _H	U, SV	U, SV	0000 0000 _H
MLI1_TP0BAR	MLI1 Transmitter Pipe 0 Base Address Register	F010 C154 _H	U, SV	U, SV	0000 0000 _H
MLI1_TP1BAR	MLI1 Transmitter Pipe 1 Base Address Register	F010 C158 _H	U, SV	U, SV	0000 0000 _H
MLI1_TP2BAR	MLI1 Transmitter Pipe 2 Base Address Register	F010 C15C _H	U, SV	U, SV	0000 0000 _H
MLI1_TP3BAR	MLI1 Transmitter Pipe 3 Base Address Register	F010 C160 _H	U, SV	U, SV	0000 0000 _H
MLI1_TCBAR	MLI1 Transmitter Copy Base Address Register	F010 C164 _H	U, SV	BE	0000 0000 _H
MLI1_RCR	MLI1 Receiver Control Register	F010 C168 _H	U, SV	U, SV	0100 0000 _H
MLI1_RP0BAR	MLI1 Receiver Pipe 0 Base Address Register	F010 C16C _H	U, SV	BE	0000 0000 _H
MLI1_RP1BAR	MLI1 Receiver Pipe 1 Base Address Register	F010 C170 _H	U, SV	BE	0000 0000 _H
MLI1_RP2BAR	MLI1 Receiver Pipe 2 Base Address Register	F010 C174 _H	U, SV	BE	0000 0000 _H
MLI1_RP3BAR	MLI1 Receiver Pipe 3 Base Address Register	F010 C178 _H	U, SV	BE	0000 0000 _H
MLI1_RP0STATR	MLI1 Receiver Pipe 0 Status Register	F010 C17C _H	U, SV	BE	0000 0000 _H
MLI1_RP1STATR	MLI1 Receiver Pipe 1 Status Register	F010 C180 _H	U, SV	BE	0000 0000 _H

Register Overview

Table 18-30 Address Map of MLI0/MLI1 (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
MLI1_RP2STATR	MLI1 Receiver Pipe 2 Status Register	F010 C184 _H	U, SV	BE	0000 0000 _H
MLI1_RP3STATR	MLI1 Receiver Pipe 3 Status Register	F010 C188 _H	U, SV	BE	0000 0000 _H
MLI1_RADRR	MLI1 Receiver Address Register	F010 C18C _H	U, SV	BE	0000 0000 _H
MLI1_RDATAR	MLI1 Receiver Data Register	F010 C190 _H	U, SV	BE	0000 0000 _H
MLI1_SCR	MLI1 Set Clear Register	F010 C194 _H	U, SV	U, SV	0000 0000 _H
MLI1_TIER	MLI1 Transmitter Interrupt Enable Register	F010 C198 _H	U, SV	U, SV	0000 0000 _H
MLI1_TISR	MLI1 Transmitter Interrupt Status Register	F010 C19C _H	U, SV	BE	0000 0000 _H
MLI1_TINPR	MLI1 Transmitter Interrupt Node Pointer Register	F010 C1A0 _H	U, SV	U, SV	0000 0000 _H
MLI1_RIER	MLI1 Receiver Interrupt Enable Register	F010 C1A4 _H	U, SV	U, SV	0000 0000 _H
MLI1_RISR	MLI1 Receiver Interrupt Status Register	F010 C1A8 _H	U, SV	BE	0000 0000 _H
MLI1_RINPR	MLI1 Receiver Interrupt Node Pointer Register	F010 C1AC _H	U, SV	U, SV	0000 0000 _H
MLI1_GINTR	MLI1 Global Interrupt Set Register	F010 C1B0 _H	U, SV	U, SV	0000 0000 _H
MLI1_OICR	MLI1 Output Input Control Register	F010 C1B4 _H	U, SV	U, SV	1000 8000 _H
MLI1_AER	MLI1 Access Enable Register	F010 C1B8 _H	U, SV	SV, E	0000 0000 _H
MLI1_ARR	MLI1 Access Range Register	F010 C1BC _H	U, SV	SV, E	0000 0000 _H
–	Reserved	F010 C1C0 _H - F010 C1FC _H	BE	BE	–

Register Overview

Table 18-31 Address Map of MCHK

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Memory Checker (MCHK)¹⁾					
–	Reserved	F010 C200 _H - F010 C204 _H	BE	BE	–
MCHK_ID	Memory Checker Module Identification Register	F010 C208 _H	U, SV	BE	001B C0XX _H
–	Reserved	F010 C20C _H	BE	BE	–
MCHK_IR	Memory Checker Input Register	F010 C210 _H	U, SV	U, SV	0000 0000 _H
MCHK_RR	Memory Checker Result Register	F010 C214 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 C218 _H - F010 C21C _H	BE	BE	–
MCHK_WR	Memory Checker Write Register	F010 C220 _H	U, SV	U, SV	0000 0000 _H
–	Reserved	F010 C224 _H - F010 C2FC _H	BE	BE	–

1) The reset values of the memory checker registers can be modified into initial values after Boot ROM exit. See [Table 4-4](#) on [Page 4-19](#) for more details.

Register Overview

Table 18-32 Address Map of CPS

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
CPU Slave Interface Registers (CPS)					
–	Reserved	F7E0 FF00 _H - F7E0 FF04 _H	BE	BE	–
CPS_ID	CPS Module Identification Register	F7E0 FF08 _H	U, SV	U, SV, NC	0015 C0XX _H
–	Reserved	F7E0 FF0C _H - FFFE FFB8 _H	BE	BE	–
CPU_SBSRC0	CPU Software Break Service Request Control Register 0	F7E0 FFBC _H	U, SV	SV	0000 0000 _H
–	Reserved	F7E0 FFC0 _H - F7E0 FFEC _H	BE	BE	–
CPU_SRC3	CPU Service Request Control Register 3	F7E0 FFF0 _H	U, SV	SV	0000 0000 _H
CPU_SRC2	CPU Service Request Control Register 2	F7E0 FFF4 _H	U, SV	SV	0000 0000 _H
CPU_SRC1	CPU Service Request Control Register 1	F7E0 FFF8 _H	U, SV	SV	0000 0000 _H
CPU_SRC0	CPU Service Request Control Register 0	F7E0 FFFC _H	U, SV	SV	0000 0000 _H

Register Overview

Table 18-33 Address Map of CPU Core SFRs & GPRs

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
CPU Core SFRs & GPRs					
–	Reserved	F7E1 0000 _H - F7E1 7FFC _H	BE	BE	–
MMU_CON	MMU Control Register	F7E1 8000 _H	U, SV, 32	SV, 32	0000 8000 _H
–	Reserved	F7E1 8004 _H - F7E1 803C _H	U, SV, 32	SV, 32 ¹⁾	–
–	Reserved	F7E1 8400 _H - F7E1 BFFC _H	BE	BE	–
Memory Protection Registers					
DPR0_0L	Data Seg. Prot. Reg. Set 0, Range 0, Lower Boundary	F7E1 C000 _H	U, SV, 32	SV, 32	0000 0000 _H
DPR0_0U	Data Seg. Prot. Reg. Set 0, Range 0, Upper Boundary	F7E1 C004 _H	U, SV, 32	SV, 32	0000 0000 _H
DPR0_1L	Data Seg. Prot. Reg. Set 0, Range 1, Lower Boundary	F7E1 C008 _H	U, SV, 32	SV, 32	0000 0000 _H
DPR0_1U	Data Seg. Prot. Reg. Set 0, Range 1, Upper Boundary	F7E1 C00C _H	U, SV, 32	SV, 32	0000 0000 _H
DPR0_2L	Data Seg. Prot. Reg. Set 0, Range 2, Lower Boundary	F7E1 C010 _H	U, SV, 32	SV, 32	0000 0000 _H
DPR0_2U	Data Seg. Prot. Reg. Set 0, Range 2, Upper Boundary	F7E1 C014 _H	U, SV, 32	SV, 32	0000 0000 _H
DPR0_3L	Data Seg. Prot. Reg. Set 0, Range 3, Lower Boundary	F7E1 C018 _H	U, SV, 32	SV, 32	0000 0000 _H
DPR0_3U	Data Seg. Prot. Reg. Set 0, Range 3, Upper Boundary	F7E1 C01C _H	U, SV, 32	SV, 32	0000 0000 _H

Register Overview

Table 18-33 Address Map of CPU Core SFRs & GPRs (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
–	Reserved	F7E1 C020 _H - F7E1 C3FC _H	nE	nE	–
DPR1_0L	Data Seg. Prot. Reg. Set 1, Range 0, Lower Boundary	F7E1 C400 _H	U, SV, 32	SV, 32	0000 0000 _H
DPR1_0U	Data Seg. Prot. Reg. Set 1, Range 0, Upper Boundary	F7E1 C404 _H	U, SV, 32	SV, 32	0000 0000 _H
DPR1_1L	Data Seg. Prot. Reg. Set 1, Range 1, Lower Boundary	F7E1 C408 _H	U, SV, 32	SV, 32	0000 0000 _H
DPR1_1U	Data Seg. Prot. Reg. Set 1, Range 1, Upper Boundary	F7E1 C40C _H	U, SV, 32	SV, 32	0000 0000 _H
DPR1_2L	Data Seg. Prot. Reg. Set 1, Range 2, Lower Boundary	F7E1 C410 _H	U, SV, 32	SV, 32	0000 0000 _H
DPR1_2U	Data Seg. Prot. Reg. Set 1, Range 2, Upper Boundary	F7E1 C414 _H	U, SV, 32	SV, 32	0000 0000 _H
DPR1_3L	Data Seg. Prot. Reg. Set 1, Range 3, Lower Boundary	F7E1 C418 _H	U, SV, 32	SV, 32	0000 0000 _H
DPR1_3U	Data Seg. Prot. Reg. Set 1, Range 3, Upper Boundary	F7E1 C41C _H	U, SV, 32	SV, 32	0000 0000 _H
–	Reserved	F7E1 C420 _H - F7E1 CFFC _H	nE	nE	–
CPR0_0L	Code Seg. Prot. Reg. Set 0, Rng. 0, Lower Boundary	F7E1 D000 _H	U, SV, 32	SV, 32	0000 0000 _H
CPR0_0U	Code Seg. Prot. Reg. Set 0, Rng. 0, Upper Boundary	F7E1 D004 _H	U, SV, 32	SV, 32	0000 0000 _H

Register Overview

Table 18-33 Address Map of CPU Core SFRs & GPRs (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
CPR0_1L	Code Seg. Prot. Reg. Set 0, Rng. 1, Lower Boundary	F7E1 D008 _H	U, SV, 32	SV, 32	0000 0000 _H
CPR0_1U	Code Seg. Prot. Reg. Set 0, Rng. 1, Upper Boundary	F7E1 D00C _H	U, SV, 32	SV, 32	0000 0000 _H
–	Reserved	F7E1 D010 _H - F7E1 D3FC _H	nE	nE	–
CPR1_0L	Code Seg. Prot. Reg. Set 1, Rng. 0, Lower Boundary	F7E1 D400 _H	U, SV, 32	SV, 32	0000 0000 _H
CPR1_0U	Code Seg. Prot. Reg. Set 1, Rng. 0, Upper Boundary	F7E1 D404 _H	U, SV, 32	SV, 32	0000 0000 _H
CPR1_1L	Code Seg. Prot. Reg. Set 1, Rng. 1, Lower Boundary	F7E1 D408 _H	U, SV, 32	SV, 32	0000 0000 _H
CPR1_1U	Code Seg. Prot. Reg. Set 1, Rng. 1, Upper Boundary	F7E1 D40C _H	U, SV, 32	SV, 32	0000 0000 _H
–	Reserved	F7E1 D410 _H - F7E1 DFFC _H	nE	nE	–
DPM0	Data Protection Mode Register Set 0	F7E1 E000 _H	U, SV, 32	SV, 32	0000 0000 _H
–	Reserved	F7E1 E004 _H - F7E1 E07C _H	nE	nE	–
DPM1	Data Protection Mode Register Set 1	F7E1 E080 _H	U, SV, 32	SV, 32	0000 0000 _H
–	Reserved	F7E1 E084 _H - F7E1 E1FC _H	nE	nE	–
CPM0	Code Protection Mode Register Set 0	F7E1 E200 _H	U, SV, 32	SV, 32	0000 0000 _H
–	Reserved	F7E1 E204 _H - F7E1 E27C _H	nE	nE	–

Register Overview

Table 18-33 Address Map of CPU Core SFRs & GPRs (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
CPM1	Code Protection Mode Register Set 1	F7E1 E280 _H	U, SV, 32	SV, 32	0000 0000 _H
–	Reserved	F7E1 E284 _H - F7E1 EFFC _H	nE	nE	–
Core Debug Register (OCDS)					
DBGSR	Debug Status Register	F7E1 FD00 _H	U, SV, 32	SV, 32	0000 0000 _H
–	Reserved	F7E1 FD04 _H	nE	nE	–
EXEVT	External Break Input Event Specifier Register	F7E1 FD08 _H	U, SV, 32	SV, 32	0000 0000 _H
CREVT	Core SFR Access Break Event Specifier Register	F7E1FD0C _H	U, SV, 32	SV, 32	0000 0000 _H
SWEVT	Software Break Event Specifier Register	F7E1 FD10 _H	U, SV, 32	SV, 32	0000 0000 _H
–	Reserved	F7E1 FD14 _H - F7E1 FD1C _H	nE	nE	–
TR0EVT	Trigger Event 0 Specifier Register	F7E1 FD20 _H	U, SV, 32	SV, 32	0000 0000 _H
TR1EVT	Trigger Event 1 Specifier Register	F7E1 FD24 _H	U, SV, 32	SV, 32	0000 0000 _H
–	Reserved	F7E1 FD28 _H - F7E1 FD3C _H	nE	nE	–
DMS	Debug Monitor Start Address Register	F7E1 FD40 _H	U, SV, 32	U, SV, 32, NC	DE00 0000 _H
DCX	Debug Context Save Area Pointer	F7E1 FD44 _H	U, SV, 32	SV, 32	DE80 0000 _H
–	Reserved	F7E1 FD48 _H - F7E1 FDFC _H	nE	nE	–
Core Special Function Registers (CSFR)					
PCXI	Previous Context Information Register	F7E1 FE00 _H	U, SV, 32	SV, 32	0000 0000 _H

Register Overview

Table 18-33 Address Map of CPU Core SFRs & GPRs (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
PSW	Program Status Word	F7E1 FE04 _H	U, SV, 32	SV, 32	0000 0B80 _H
PC	Program Counter	F7E1 FE08 _H	U, SV, 32	SV, 32	DFFF FFFC _H DE00 0000 _H
–	Reserved	F7E1 FE0C _H - F7E1 FE10 _H	nE	nE	–
SYSCON	System Configuration Register	F7E1 FE14 _H	U, SV, 32	SV, 32	0000 0000 _H
CPU_ID	CPU Identification Register	F7E1 FE18 _H	U, SV, 32	U, SV, NC, 32	000A C0XX _H
–	Reserved	F7E1 FE1C _H	nE	nE	–
BIV	Interrupt Vector Table Pointer	F7E1 FE20 _H	U, SV, 32	SV, E, 32	0000 0000 _H
BTV	Trap Vector Table Pointer	F7E1 FE24 _H	U, SV, 32	SV, E, 32	A000 0100 _H
ISP	Interrupt Stack Pointer	F7E1 FE28 _H	U, SV, 32	SV, E, 32	0000 0100 _H
ICR	ICU Interrupt Control Register	F7E1 FE2C _H	U, SV, 32	SV, 32	0000 0000 _H
–	Reserved	F7E1 FE30 _H - F7E1 FE34 _H	nE	nE	–
FCX	Free Context List Head Pointer	F7E1 FE38 _H	U, SV, 32	SV, 32	0000 0000 _H
LCX	Free Context List Limit Pointer	F7E1 FE3C _H	U, SV, 32	SV, 32	0000 0000 _H
–	–	F7E1 FE40 _H - F7E1 FEFC _H	nE	nE	–

CPU Core General Purpose Register (GPRs)²⁾

D0	Data Register 0	F7E1 FF00 _H	–	–	XXXX XXXX _H
D1	Data Register 1	F7E1 FF04 _H	–	–	XXXX XXXX _H
D2	Data Register 2	F7E1 FF08 _H	–	–	XXXX XXXX _H

Register Overview

Table 18-33 Address Map of CPU Core SFRs & GPRs (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
D3	Data Register 3	F7E1 FF0C _H	–	–	XXXX XXXX _H
D4	Data Register 4	F7E1 FF10 _H	–	–	XXXX XXXX _H
D5	Data Register 5	F7E1 FF14 _H	–	–	XXXX XXXX _H
D6	Data Register 6	F7E1 FF18 _H	–	–	XXXX XXXX _H
D7	Data Register 7	F7E1 FF1C _H	–	–	XXXX XXXX _H
D8	Data Register 8	F7E1 FF20 _H	–	–	XXXX XXXX _H
D9	Data Register 9	F7E1 FF24 _H	–	–	XXXX XXXX _H
D10	Data Register 0	F7E1 FF28 _H	–	–	XXXX XXXX _H
D11	Data Register 11	F7E1 FF2C _H	–	–	XXXX XXXX _H
D12	Data Register 12	F7E1 FF30 _H	–	–	XXXX XXXX _H
D13	Data Register 13	F7E1 FF34 _H	–	–	XXXX XXXX _H
D14	Data Register 14	F7E1 FF38 _H	–	–	XXXX XXXX _H
D15	Data Register 15	F7E1 FF3C _H	–	–	XXXX XXXX _H
–	Reserved	F7E1 FF40 _H - F7E1 FF7C _H	nE	nE	–
A0	Address Register 0 (Global Address Register)	F7E1 FF80 _H	–	–	XXXX XXXX _H
A1	Address Register 1 (Global Address Register)	F7E1 FF84 _H	–	–	XXXX XXXX _H
A2	Address Register 2	F7E1 FF88 _H	–	–	XXXX XXXX _H
A3	Address Register 3	F7E1 FF8C _H	–	–	XXXX XXXX _H
A4	Address Register 4	F7E1 FF90 _H	–	–	XXXX XXXX _H
A5	Address Register 5	F7E1 FF94 _H	–	–	XXXX XXXX _H
A6	Address Register 6	F7E1 FF98 _H	–	–	XXXX XXXX _H
A7	Address Register 7	F7E1 FF9C _H	–	–	XXXX XXXX _H
A8	Address Register 8 (Global Address Register)	F7E1 FFA0 _H	–	–	XXXX XXXX _H
A9	Address Register 9 (Global Address Register)	F7E1 FFA4 _H	–	–	XXXX XXXX _H
A10	Address Register 10 (Stack Pointer)	F7E1 FFA8 _H	–	–	XXXX XXXX _H

Register Overview

Table 18-33 Address Map of CPU Core SFRs & GPRs (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
A11	Address Register 11 (Return Address)	F7E1 FFAC _H	–	–	XXXX XXXX _H
A12	Address Register 12	F7E1 FFB0 _H	–	–	XXXX XXXX _H
A13	Address Register 13	F7E1 FFB4 _H	–	–	XXXX XXXX _H
A14	Address Register 14	F7E1 FFB8 _H	–	–	XXXX XXXX _H
A15	Address Register 15	F7E1 FFBC _H	–	–	XXXX XXXX _H
–	Reserved	F7E1 FFC0 _H - F7E1 FFFC _H	nE	nE	–

- 1) Read/write operations from/to these locations have no effect.
- 2) Note that all GPRs are undefined (XXXX XXXX_B) after a reset operation.

Register Overview

Table 18-34 Address Map of EBU

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
External Bus Interface Unit (EBU)					
EBU_CLC	EBU Clock Control Register	F800 0000 _H	U, SV, 32	SV, E, 32	0000 0000 _H
–	Reserved	F800 0004 _H	BE	BE	–
EBU_ID	EBU Module Identification Register	F800 0008 _H	U, SV, 32	SV, 32	0014 C0XX _H
–	Reserved	F800 000C _H	BE	BE	–
EBU_CON	EBU Configuration Register	F800 0010 _H	U, SV, 32	SV, 32	see Page 13-83
–	Reserved	F800 0014 _H - F800 001C _H	BE	BE	–
EBU_BFCON	EBU Burst Flash Control Register	F800 0020 _H	U, SV, 32	SV, 32	0010 01D0 _H
–	Reserved	F800 0024 _H - F800 003C _H	BE	BE	–
–	Reserved; do not read/write to this location.	F800 0040 _H	U, SV, 32	SV, 32	–
–	Reserved	F800 0044 _H	BE	BE	–
–	Reserved; do not read/write to this location.	F800 0048 _H	U, SV, 32	SV, 32	–
–	Reserved	F800 004C _H	BE	BE	–
–	Reserved; do not read/write to this location.	F800 0050 _H	U, SV, 32	SV, 32	–
–	Reserved	F800 0054 _H	BE	BE	–
–	Reserved; do not read/write to this location.	F800 0058 _H	U, SV, 32	SV, 32	–
–	Reserved	F800 005C _H	BE	BE	–
–	Reserved; do not read/write to this location.	F800 0060 _H	U, SV, 32	SV, 32	–
–	Reserved	F800 0064 _H	BE	BE	–
–	Reserved; do not read/write to this location.	F800 0068 _H	U, SV, 32	SV, 32	–

Register Overview

Table 18-34 Address Map of EBU (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
–	Reserved	F800 006C _H	BE	BE	–
–	Reserved; do not read/write to this location.	F800 0070 _H	U, SV, 32	SV, 32	–
–	Reserved	F800 0074 _H	BE	BE	–
–	Reserved; do not read/write to this location.	F800 0078 _H	U, SV, 32	SV, 32	–
–	Reserved	F800 007C _H	BE	BE	–
EBU_ADDRSEL0	EBU Address Select Register 0	F800 0080 _H	U, SV, 32	SV, 32	see Page 13-90
–	Reserved	F800 0084 _H	BE	BE	–
EBU_ADDRSEL1	EBU Address Select Register 1	F800 0088 _H	U, SV, 32	SV, 32	0000 0000 _H
–	Reserved	F800 008C _H	BE	BE	–
EBU_ADDRSEL2	EBU Address Select Register 2	F800 0090 _H	U, SV, 32	SV, 32	0000 0000 _H
–	Reserved	F800 0094 _H	BE	BE	–
EBU_ADDRSEL3	EBU Address Select Register 3	F800 0098 _H	U, SV, 32	SV, 32	0000 0000 _H
–	Reserved	F800 009C _H	BE	BE	–
–	Reserved; do not read/write to this location.	F800 00A0 _H	U, SV, 32	SV, 32	–
–	Reserved	F800 00A4 _H	BE	BE	–
–	Reserved; do not read/write to this location.	F800 00A8 _H	U, SV, 32	SV, 32	–
–	Reserved	F800 00AC _H	BE	BE	–
–	Reserved; do not read/write to this location.	F800 00B0 _H	U, SV	SV	–
–	Reserved	F800 00B4 _H - F800 00BC _H	BE	BE	–
EBU_BUSCON0	EBU Bus Configuration Register 0	F800 00C0 _H	U, SV, 32	SV, 32	see Page 13-92
–	Reserved	F800 00C4 _H	BE	BE	–

Register Overview

Table 18-34 Address Map of EBU (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
EBU_ BUSCON1	EBU Bus Configuration Register 1	F800 00C8 _H	U, SV, 32	SV, 32	see Page 13-92
–	Reserved	F800 00CC _H	BE	BE	–
EBU_ BUSCON2	EBU Bus Configuration Register 2	F800 00D0 _H	U, SV, 32	SV, 32	see Page 13-92
–	Reserved	F800 00D4 _H	BE	BE	–
EBU_ BUSCON3	EBU Bus Configuration Register 3	F800 00D8 _H	U, SV, 32	SV, 32	see Page 13-92
–	Reserved	F800 00DC _H	BE	BE	–
–	Reserved; do not read/write to this location.	F800 00E0 _H	U, SV, 32	SV, 32	–
–	Reserved	F800 00E4 _H	BE	BE	–
–	Reserved; do not read/write to this location.	F800 00E8 _H	U, SV, 32	SV, 32	–
–	Reserved	F800 00EC _H	BE	BE	–
–	Reserved; do not read/write to this location.	F800 00F0 _H	U, SV, 32	SV, 32	–
–	Reserved	F800 00F4 _H - F800 00FC _H	BE	BE	–
EBU_ BUSAP0	EBU Bus Access Parameter Register 0	F800 0100 _H	U, SV, 32	SV, 32	FFFF FFFF _H
–	Reserved	F800 0104 _H	BE	BE	–
EBU_ BUSAP1	EBU Bus Access Parameter Register 1	F800 0108 _H	U, SV, 32	SV, 32	FFFF FFFF _H
–	Reserved	F800 010C _H	BE	BE	–
EBU_ BUSAP2	EBU Bus Access Parameter Register 2	F800 0110 _H	U, SV, 32	SV, 32	FFFF FFFF _H
–	Reserved	F800 0114 _H	BE	BE	–
EBU_ BUSAP3	EBU Bus Access Parameter Register 3	F800 0118 _H	U, SV, 32	SV, 32	FFFF FFFF _H
–	Reserved	F800 011C _H	BE	BE	–

Register Overview

Table 18-34 Address Map of EBU (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
–	Reserved; do not read/write to this location.	F800 0120 _H	U, SV, 32	SV, 32	–
–	Reserved	F800 0124 _H	BE	BE	–
–	Reserved; do not read/write to this location.	F800 0128 _H	U, SV, 32	SV, 32	–
–	Reserved	F800 012C _H	BE	BE	–
–	Reserved; do not read/write to this location.	F800 0130 _H	U, SV, 32	SV, 32	–
–	Reserved	F800 0134 _H - F800 015C _H	BE	BE	–
EBU_ EMUAS	EBU Emulator Address Select Register	F800 0160 _H	U, SV, 32	SV, 32	DE00 0031 _H
–	Reserved	F800 0164 _H	BE	BE	–
EBU_ EMUBC	EBU Emulator Bus Configuration Register	F800 0168 _H	U, SV, 32	SV, 32	0190 2077 _H
–	Reserved	F800 016C _H	BE	BE	–
EBU_ EMUBAP	EBU Emulator Bus Access Parameter Register	F800 0170 _H	U, SV, 32	SV, 32	5248 4911 _H
–	Reserved	F800 0174 _H	BE	BE	–
EBU_ EMUOVL	EBU Emulator Overlay Register	F800 0178 _H	U, SV, 32	SV, 32	0000 0000 _H
–	Reserved	F800 017C _H - F800 018C _H	BE	BE	–
EBU_ USERCON	EBU Test/Control Configuration Register	F800 0190 _H	U, SV, 32	SV, 32	0000 0000 _H
–	Reserved	F800 0194 _H - F800 03FC _H	BE	BE	–

Register Overview

Table 18-35 Address Map of PMU

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Program Memory Unit (PMU)					
–	Reserved	F800 0500 _H - F800 0504 _H	BE	BE	–
PMU_ID	PMU Module Identification Register	F800 0508 _H	U, SV	BE	002E C0XX _H
–	Reserved	F800 050C _H	BE	BE	–
–	Reserved; must not be read or written; otherwise unpredictable results may occur.	F800 0510 _H	U, SV	SV	–
–	Reserved	F800 0514 _H - F800 05FC _H	BE	BE	–

Register Overview

Table 18-36 Address Map of Flash

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Flash Register					
–	Reserved	F800 1000 _H - F800 2004 _H	BE	BE	–
FLASH_ID	Flash Module Identification Register	F800 2008 _H	U, SV	BE	0031 C0XX _H
–	Reserved	F800 200C _H	BE	BE	–
FLASH_ FSR	Flash Status Register	F800 2010 _H	U, SV	BE	0XXX X0X0 _H
FLASH_ FCON	Flash Configuration Register	F800 2014 _H	U, SV	SV, E	000X 0636 _H
FLASH_ MARP	Flash Margin Control Register PFlash	F800 2018 _H	U, SV	SV, E	0000 8000 _H
FLASH_ MARD	Flash Margin Control Register DFlash	F800 201C _H	U, SV	U, SV	0000 8000 _H
FLASH_ PROCON0	Flash Protection Config. Register User 0	F800 2020 _H	U, SV	BE	0000 XXXX _H
FLASH_ PROCON1	Flash Protection Config. Register User 1	F800 2024 _H	U, SV	BE	0000 XXXX _H
FLASH_ PROCON2	Flash Protection Config. Register User 2	F800 2028 _H	U, SV	BE	0000 XXXX _H
–	Reserved	F800 202C _H - F800 23FC _H	BE	BE	–

Register Overview

Table 18-37 Address Map of DMU

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Data Memory Unit (DMU)					
–	Reserved	F801 0100 _H - F801 0104 _H	BE	BE	–
DMU_ID	DMU Module Identification Register	F801 0108 _H	U, SV	BE	002D C0XX _H
–	Reserved	F801 010C _H	BE	BE	–
–	Reserved ¹⁾	F801 0110 _H	U, SV	U, SV, E	–
–	Reserved	F801 0114 _H	BE	BE	–
–	Reserved ¹⁾	F801 0118 _H	BE	U, SV, E, 32	–
–	Reserved ¹⁾	F801 011C _H	U, SV	BE	–
DMU_RABR0	DMU Redirected Address Base Register 0	F801 0120 _H	U, SV	SV	0000 0000 _H
DMU_OTAR0	DMU Overlay Target Address Register 0	F801 0124 _H	U, SV	SV	0000 0000 _H
DMU_OMASK0	DMU Overlay Mask Register 0	F801 0128 _H	U, SV	SV	0FFF FE00 _H
DMU_RABR1	DMU Redirected Address Base Register 1	F801 012C _H	U, SV	SV	0000 0000 _H
DMU_OTAR1	DMU Overlay Target Address Register 1	F801 0130 _H	U, SV	SV	0000 0000 _H
DMU_OMASK1	DMU Overlay Mask Register 1	F801 0134 _H	U, SV	SV	0FFF FE00 _H
DMU_RABR2	DMU Redirected Address Base Register 2	F801 0138 _H	U, SV	SV	0000 0000 _H
DMU_OTAR2	DMU Overlay Target Address Register 2	F801 013C _H	U, SV	SV	0000 0000 _H
DMU_OMASK2	DMU Overlay Mask Register 2	F801 0140 _H	U, SV	SV	0FFF FE00 _H
DMU_RABR3	DMU Redirected Address Base Register 3	F801 0144 _H	U, SV	SV	0000 0000 _H

Register Overview

Table 18-37 Address Map of DMU (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
DMU_OTAR3	DMU Overlay Target Address Register 3	F801 0148 _H	U, SV	SV	0000 0000 _H
DMU_OMASK3	DMU Overlay Mask Register 3	F801 014C _H	U, SV	SV	0FFF FE00 _H
DMU_RABR4	DMU Redirected Address Base Register 4	F801 0150 _H	U, SV	SV	0000 0000 _H
DMU_OTAR4	DMU Overlay Target Address Register 4	F801 0154 _H	U, SV	SV	0000 0000 _H
DMU_OMASK4	DMU Overlay Mask Register 4	F801 0158 _H	U, SV	SV	0FFF FE00 _H
DMU_RABR5	DMU Redirected Address Base Register 5	F801 015C _H	U, SV	SV	0000 0000 _H
DMU_OTAR5	DMU Overlay Target Address Register 5	F801 0160 _H	U, SV	SV	0000 0000 _H
DMU_OMASK5	DMU Overlay Mask Register 5	F801 0164 _H	U, SV	SV	0FFF FE00 _H
DMU_RABR6	DMU Redirected Address Base Register 6	F801 0168 _H	U, SV	SV	0000 0000 _H
DMU_OTAR6	DMU Overlay Target Address Register 6	F801 016C _H	U, SV	SV	0000 0000 _H
DMU_OMASK6	DMU Overlay Mask Register 6	F801 0170 _H	U, SV	SV	0FFF FE00 _H
DMU_RABR7	DMU Redirected Address Base Register 7	F801 0174 _H	U, SV	SV	0000 0000 _H
DMU_OTAR7	DMU Overlay Target Address Register 7	F801 0178 _H	U, SV	SV	0000 0000 _H
DMU_OMASK7	DMU Overlay Mask Register 7	F801 017C _H	U, SV	SV	0FFF FE00 _H
DMU_RABR8	DMU Redirected Address Base Register 8	F801 0180 _H	U, SV	SV	0000 0000 _H
DMU_OTAR8	DMU Overlay Target Address Register 8	F801 0184 _H	U, SV	SV	0000 0000 _H

Register Overview

Table 18-37 Address Map of DMU (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
DMU_ OMASK8	DMU Overlay Mask Register 8	F801 0188 _H	U, SV	SV	0FFF FE00 _H
DMU_ RABR9	DMU Redirected Address Base Register 9	F801 018C _H	U, SV	SV	0000 0000 _H
DMU_ OTAR9	DMU Overlay Target Address Register 9	F801 0190 _H	U, SV	SV	0000 0000 _H
DMU_ OMASK9	DMU Overlay Mask Register 9	F801 0194 _H	U, SV	SV	0FFF FE00 _H
DMU_ RABR10	DMU Redirected Address Base Register 10	F801 0198 _H	U, SV	SV	0000 0000 _H
DMU_ OTAR10	DMU Overlay Target Address Register 10	F801 019C _H	U, SV	SV	0000 0000 _H
DMU_ OMASK10	DMU Overlay Mask Register 10	F801 01A0 _H	U, SV	SV	0FFF FE00 _H
DMU_ RABR11	DMU Redirected Address Base Register 11	F801 01A4 _H	U, SV	SV	0000 0000 _H
DMU_ OTAR11	DMU Overlay Target Address Register 11	F801 01A8 _H	U, SV	SV	0000 0000 _H
DMU_ OMASK11	DMU Overlay Mask Register 11	F801 01AC _H	U, SV	SV	0FFF FE00 _H
DMU_ RABR12	DMU Redirected Address Base Register 12	F801 01B0 _H	U, SV	SV	0000 0000 _H
DMU_ OTAR12	DMU Overlay Target Address Register 12	F801 01B4 _H	U, SV	SV	0000 0000 _H
DMU_ OMASK12	DMU Overlay Mask Register 12	F801 01B8 _H	U, SV	SV	0FFF FE00 _H
DMU_ RABR13	DMU Redirected Address Base Register 13	F801 01BC _H	U, SV	SV	0000 0000 _H
DMU_ OTAR13	DMU Overlay Target Address Register 13	F801 01C0 _H	U, SV	SV	0000 0000 _H
DMU_ OMASK13	DMU Overlay Mask Register 13	F801 01C4 _H	U, SV	SV	0FFF FE00 _H

Register Overview

Table 18-37 Address Map of DMU (cont'd)

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
DMU_ RABR14	DMU Redirected Address Base Register 14	F801 01C8 _H	U, SV	SV	0000 0000 _H
DMU_ OTAR14	DMU Overlay Target Address Register 14	F801 01CC _H	U, SV	SV	0000 0000 _H
DMU_ OMASK14	DMU Overlay Mask Register 14	F801 01D0 _H	U, SV	SV	0FFF FE00 _H
DMU_ RABR15	DMU Redirected Address Base Register 15	F801 01D4 _H	U, SV	SV	0000 0000 _H
DMU_ OTAR15	DMU Overlay Target Address Register 15	F801 01D8 _H	U, SV	SV	0000 0000 _H
DMU_ OMASK15	DMU Overlay Mask Register 15	F801 01DC _H	U, SV	SV	0FFF FE00 _H
DMU_ SBRCTR	DMU Stand-by SRAM Control Register	F801 01E0 _H	U, SV	SV	0000 0000 _H
–	Reserved	F801 01E4 _H - F801 01FC _H	BE	BE	–

1) Do not read from or write to these address locations. Otherwise unpredictable results may occur.

Table 18-38 Address Map of DBCU

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Data Local Memory Bus Control Unit (DBCU)					
–	Reserved	F87F FA00 _H - F87F FA04 _H	BE	BE	–
DBCU_ID	DBCU Module Identification Register	F87F FA08 _H	U, SV	BE	000F C0XX _H
–	Reserved	F87F FA0C _H - F87F FA18 _H	BE	BE	–
DBCU_LEATT	DBCU LMB Error Attribute Register	F87F FA20 _H	U, SV, 32	SV, 32	XXXX XXX0 _H
DBCU_LEADDR	DBCU LMB Error Address Register	F87F FA24 _H	U, SV, 32	BE	XXXX XXXX _H
DBCU_LEDATL	DBCU LMB Error Data Low Register	F87F FA28 _H	U, SV, 64	BE	XXXX XXXX _H
DBCU_LEDATH	DBCU LMB Error Data High Register	F87F FA2C _H			XXXX XXXX _H
–	Reserved	F87F FA30 _H - F87F FAF8 _H	BE	BE	–
DBCU_SRC	DBCU Service Request Control Register	F87F FAFC _H	U, SV, 32	SV, 32	0000 0000 _H

Register Overview

Table 18-39 Address Map of DMI

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
DMI Registers					
–	Reserved	F87F FC00 _H - F87F FC04 _H	nBE	nBE	–
DMI_ID	DMI Module Identification Register	F87F FC08 _H	U, SV	¹⁾	0008 C0XX _H
–	Reserved	F87F FC0C _H	nBE	nBE	–
DMI_CON	DMI Control Register	F87F FC10 _H	U, SV	SV, E	0000 0070 _H
–	Reserved	F87F FC14 _H	nBE	nBE	–
DMI_STR	DMI Synchronous Trap Flag Register	F87F FC18 _H	U, SV ¹⁾²⁾	¹⁾	0000 0000 _H
–	Reserved	F87F FC1C _H	nBE	nBE	–
DMI_ATR	DMI Asynchronous Trap Flag Register	F87F FC20 _H	U, SV ¹⁾²⁾	¹⁾	0000 0000 _H
–	Reserved	F87F FC24 _H	nBE	nBE	–
DMI_CON1	DMI Control Register 1	F87F FC28 _H	U, SV, 32	SV, E, 32	0000 0000 _H
–	Reserved	F87F FC2C _H - F87F FCFC _H	nBE	nBE	–

1) Access to the DMI registers must only be made with double-word-aligned word accesses. An access not conforming to this rule, or an access that does not follow the specified privilege mode (supervisor mode, Endinit-protection), or a write access to a read-only register, will lead to a bus error if the access was from the FPI Bus, or to a trap, flagged with a DMI Control Register Error Flag (see DMI_STR/DMI_ATR registers) in case of a CPU load/store access.XXX

2) Reading this register in supervisor mode returns the contents and then clears the register. Reading it in user mode only returns the contents of the register; it is not cleared. No error will be reported in this case.

Register Overview

Table 18-40 Address Map of PMI

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
PMI Registers					
–	Reserved	F87F FD00 _H - F87F FD04 _H	BE	BE	–
PMI_ID	PMI Module Identification Register	F87F FD08 _H	U, SV, 32	U, SV, NC, 32	000B C0XX _H
–	Reserved	F87F FD0C _H	BE	BE	–
PMI_CON0	PMI Control Register 0	F87F FD10 _H	U, SV, 32	SV, E, 32	0000 0002 _H
PMI_CON1	PMI Control Register 1	F87F FD14 _H	U, SV, 32	SV, 32	0000 0000 _H
PMI_CON2	PMI Control Register 2	F87F FD18 _H	U, SV, 32	SV, NC, 32	0000 0073 _H
–	Reserved	F87F FD1C _H - F87F FDFC _H	BE	BE	–

Register Overview

Table 18-41 Address Map of PBCU

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Program Local Memory Bus Control Unit (PBCU)					
–	Reserved	F87F FE00 _H - F87F FE04 _H	BE	BE	–
PBCU_ID	PBCU Module Identification Register	F87F FE08 _H	U, SV	BE	000F C0XX _H
–	Reserved	F87F FE0C _H - F87F FE18 _H	BE	BE	–
PBCU_ LEATT	PBCU LMB Error Attribute Register	F87F FE20 _H	U, SV, 32	SV, 32	XXXX XXX0 _H
PBCU_ LEADDR	PBCU LMB Error Address Register	F87F FE24 _H	U, SV, 32	BE	XXXX XXXX _H
PBCU_ LEDATL	PBCU LMB Error Data Low Register	F87F FE28 _H	U, SV, 64	BE	XXXX XXXX _H
PBCU_ LEDATH	PBCU LMB Error Data High Register	F87F FE2C _H			XXXX XXXX _H
–	Reserved	F87F FE30 _H - F87F FEF8 _H	BE	BE	–
PBCU_SRC	PBCU Service Request Control Register	F87F FEFC _H	U, SV, 32	SV, 32	0000 0000 _H

Table 18-42 Address Map of LFI-Bridge

Short Name	Description	Address	Access Mode		Reset Value
			Read	Write	
Local Memory to FPI Bus Bridge (LFI-Bridge)					
–	Reserved	F87F FF00 _H - F87F FF04 _H	BE	BE	–
LFI_ID	LFI Module Identification Register	F87F FF08 _H	U, SV	BE	000C C0XX _H
–	Reserved	F87F FF0C _H	BE	BE	–
LFI_CON	LFI Configuration Register	F87F FF10 _H	U, SV	SV	0000 0B02 _H
–	Reserved	F87F FF14 _H - F87F FFFF _H	BE	BE	–

Asynchronous/Synchronous Serial Interface (ASC)

19 Asynchronous/Synchronous Serial Interface (ASC)

This chapter describes the two Asynchronous/Synchronous Serial Interfaces of the TC1796, ASC0 and ASC1. It contains the following sections:

- Functional description of the ASC kernel, valid for ASC0 and ASC1 (see [Page 19-1](#))
- ASC kernel register description, describes all ASC kernel specific registers, valid for ASC0 and ASC1 (see [Page 19-19](#))
- TC1796 implementation-specific details and registers of the ASC0/ASC1 interfaces (see [Page 19-30](#))

Note: The ASC kernel register names described in [Section 19.2](#) will be referenced in the TC1796 User's Manual by the module name prefix "ASC0_" for the ASC0 interface and by "ASC1_" for the ASC1 interface.

19.1 ASC Kernel Description

[Figure 19-1](#) shows a global view of the ASC interface.

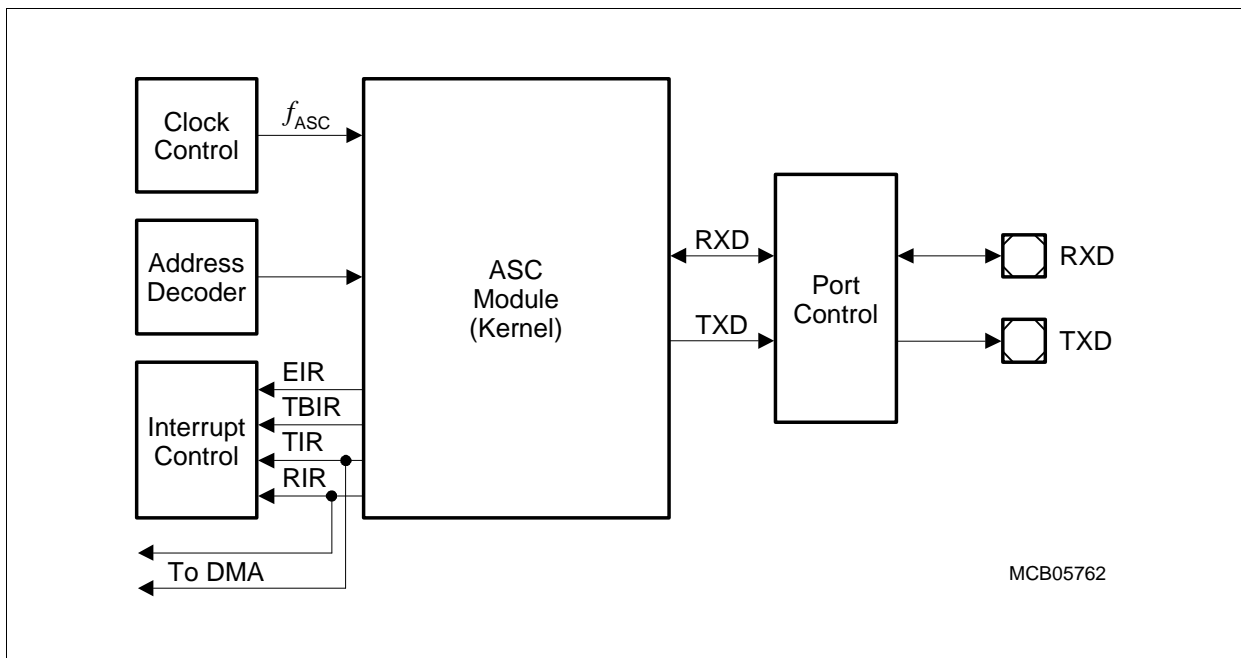


Figure 19-1 General Block Diagram of the ASC Interface

The ASC module communicates with the external world via two I/O lines. The RXD line is the receive data input signal (and also output signal in Synchronous Mode), and TXD is the transmit output signal.

Clock control, address decoding, and interrupt service request control are managed outside the ASC module kernel.

Asynchronous/Synchronous Serial Interface (ASC)**19.1.1 Overview**

The ASC provides serial communication between the TC1796 and other microcontrollers, microprocessors, or external peripherals.

The ASC supports full-duplex asynchronous communication and half-duplex synchronous communication. In Synchronous Mode, data is transmitted or received synchronous to a shift clock that is generated by the ASC internally. In Asynchronous Mode, 8-bit or 9-bit data transfer, parity generation, and the number of stop bits can be selected. Parity, framing, and overrun error detection are provided to increase the reliability of data transfers. Transmission and reception of data is double-buffered. For multiprocessor communication, a mechanism is included to distinguish address bytes from data bytes. Testing is supported by a loop-back option. A 13-bit baud rate generator provides the ASC with a separate serial clock signal, which can be accurately adjusted by a prescaler implemented as fractional divider.

Features

- Full-duplex asynchronous operating modes
 - 8-bit or 9-bit data frames, LSB first
 - Parity-bit generation/checking
 - One or two stop bits
 - Baud rate from 4.69 Mbit/s to 1.12 bit/s (@ 75 MHz module clock)
 - Multiprocessor mode for automatic address/data byte detection
 - Loop-back capability
- Half-duplex 8-bit synchronous operating mode
 - Baud rate from 9.38 Mbit/s to 763 bit/s (@ 75 MHz module clock)
- Double-buffered transmitter/receiver
- Interrupt generation
 - On a transmit buffer empty condition
 - On a transmit last bit of a frame condition
 - On a receive buffer full condition
 - On an error condition (frame, parity, overrun error)

Asynchronous/Synchronous Serial Interface (ASC)**19.1.2 General Operation**

The ASC supports full-duplex asynchronous communication up to 4.69 Mbit/s, and half-duplex synchronous communication up to 9.38 Mbit/s (@ 75 MHz module clock). In Synchronous Mode, data is transmitted or received synchronous to a shift clock generated by the microcontroller. In Asynchronous Mode, 8-bit or 9-bit data transfer, parity generation, and the number of stop bits can be selected. Parity, framing, and overrun error detection are provided to increase the reliability of data transfers. Transmission and reception of data are double-buffered. For multiprocessor communication, a mechanism is included to distinguish address bytes from data bytes. Testing is supported by a loop-back option. A 13-bit baud rate generator provides the ASC with a separate serial clock signal, which can be accurately adjusted by a prescaler implemented as fractional divider.

A transmission is started by writing to the Transmit Buffer Register, TBUF. Only the number of data bits determined by the selected operating mode will actually be transmitted; that is, bits written to positions 9 through 15 of register TBUF are always insignificant. Data transmission is double-buffered, so a new character may be written to TBUF before the transmission of the previous character is complete. This allows a back-to-back transmission of characters to take place without gaps.

Data reception is enabled by the receiver enable bit CON.REN. After a reception has been completed, the received data and, if provided by the selected operating mode, the received parity bit can be read from the (read-only) receive buffer register RBUF. Unused bits in the upper half of RBUF that are not required in the selected operating mode will be read as zeros.

Data reception is double-buffered, so that reception of a second character may already begin before the previously received character has been read out of the receive buffer register. In all modes, receive buffer overrun error detection can be selected through bit CON.OEN. When enabled, the overrun error status flag CON.OE and the error interrupt request line EIR will be activated when the receive buffer register has not been read by the time reception of a second character is complete. In this case, the previously received character in the receive buffer is overwritten.

The loop-back option (selected by bit CON.LB) allows the data currently being transmitted to be received simultaneously in the receive buffer. This may be used to test serial communication routines at an early stage without having to provide an external network. In loop-back mode, the alternate input/output function of port pins is not required.

Asynchronous/Synchronous Serial Interface (ASC)

19.1.3 Asynchronous Operation

Asynchronous Mode supports full-duplex communication, in which both transmitter and receiver use the same data frame format and have the same baud rate. Data is transmitted on pin TXD and received on pin RXD. **Figure 19-2** shows the block diagram of the ASC when operating in Asynchronous Mode.

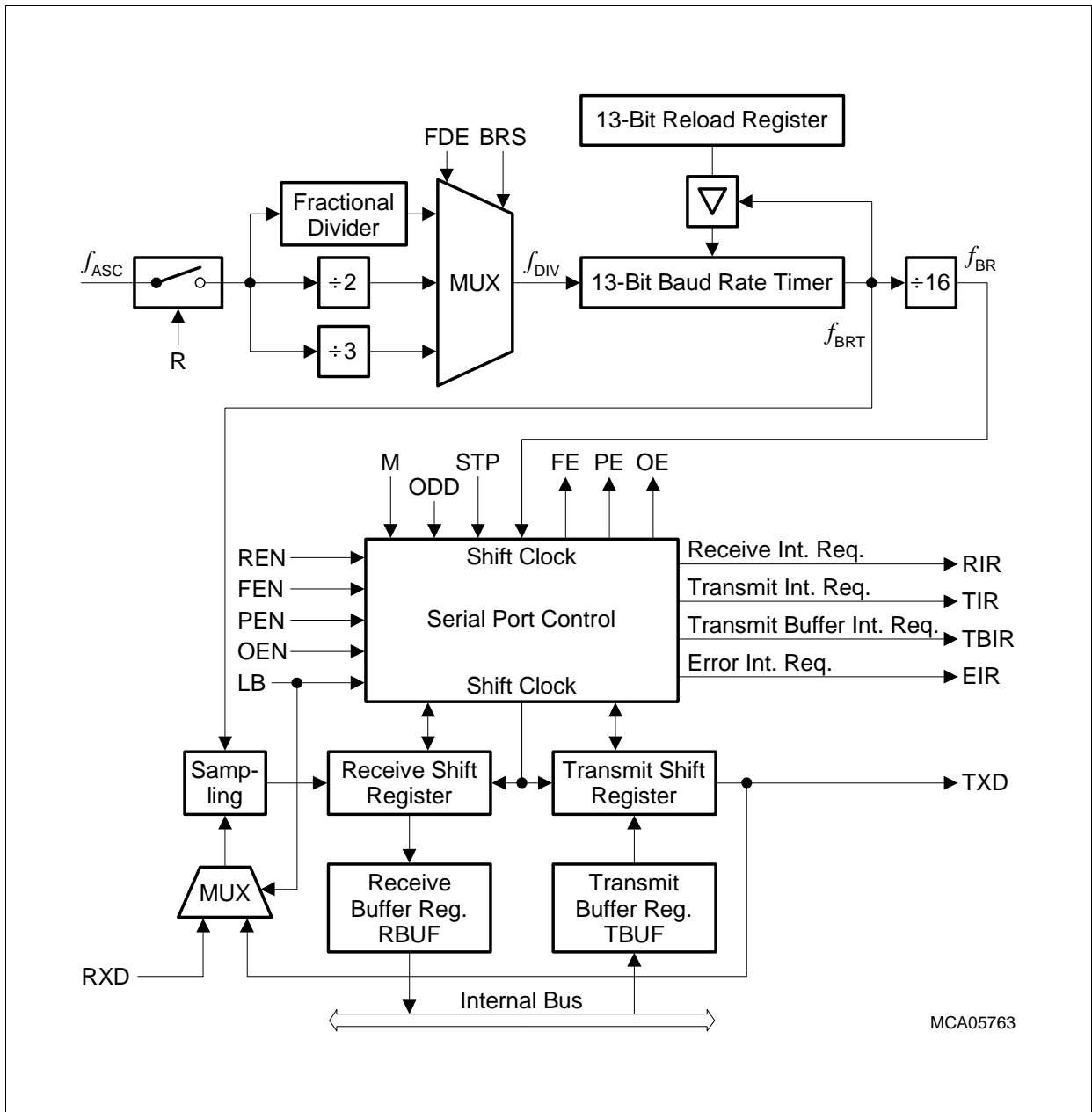


Figure 19-2 Asynchronous Mode of the ASC

Asynchronous/Synchronous Serial Interface (ASC)

19.1.3.1 Asynchronous Data Frames

Asynchronous data frames can consist of 8-bit or 9-bit data frames.

8-Bit Data Frames

The 8-bit data frames consist of either eight data bits D7 ... D0 (CON.M = 001_B), or of seven data bits D6 ... D0 plus an automatically generated parity bit (CON.M = 011_B). Parity may be odd or even, depending on bit CON.ODD. An even parity bit will be set if the modulo-2 sum of the seven data bits is 1. An odd parity bit will be cleared in this case. Parity checking is enabled via bit CON.PEN (always OFF in 8-bit data mode). The parity error flag CON.PE will be set, along with the error interrupt request flag, if a wrong parity bit is received. The received parity bit itself will be stored in RBUF too.

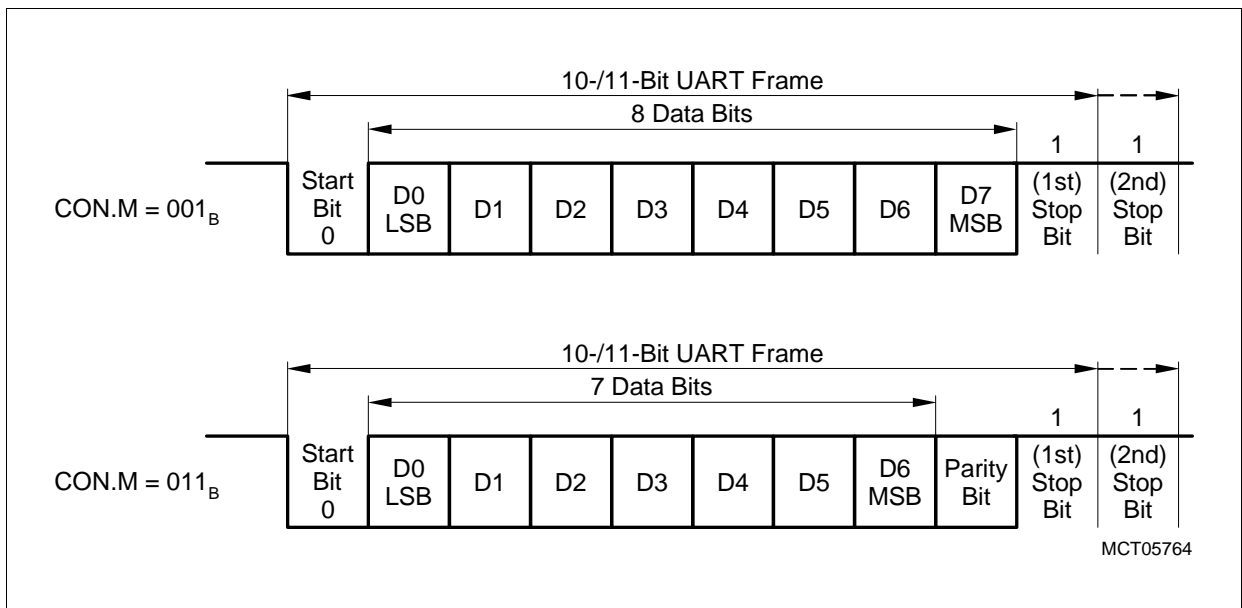


Figure 19-3 Asynchronous 8-Bit Frames

Asynchronous/Synchronous Serial Interface (ASC)

9-Bit Data Frames

The 9-bit data frames consist of nine data bits D8 ... D0 (CON.M = 100_B), or of eight data bits D7 ... D0 plus an automatically generated parity bit (CON.M = 111_B) or of eight data bits D7 ... D0 plus wake-up bit (CON.M = 101_B). Parity may be odd or even, depending on bit CON.ODD. An even parity bit will be set if the modulo-2-sum of the eight data bits is 1. An odd parity bit will be cleared in this case. Parity checking is enabled via bit CON.PEN (always OFF in 9-bit data and wake-up mode). The parity error flag CON.PE will be set along with the error interrupt request flag if a wrong parity bit is received. The received parity bit itself will be stored in RBUF too.

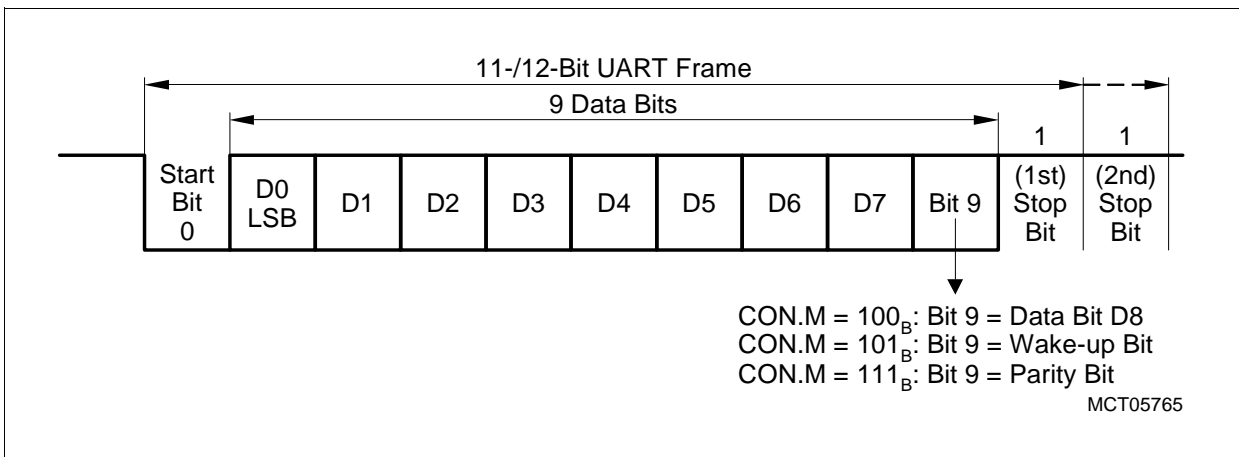


Figure 19-4 Asynchronous 9-Bit Frames

In Wake-up Mode (CON.M = 101_B), received frames are transferred to the receive buffer register only if the 9th bit (the wake-up bit) of the frame is 1. If this bit is 0, no receive interrupt request will be activated and no data will be transferred.

This feature can be used to control communication in multi-processor systems, for example.

When the master processor aims to transmit a block of data to one of several slaves, it first sends out an address 'byte' (in this case, a 'byte' consists of nine bits) that identifies the target slave. An address 'byte' differs from a data 'byte' in that the additional 9th bit is a 1 for an address 'byte' but is a 0 for a data 'byte', so, no slave will be interrupted by a data 'byte'. An address 'byte' will interrupt all slaves (operating in 8-bit data + wake-up bit mode), so each slave can examine the eight LSBs of the received character (the address). The addressed slave will switch to 9-bit data mode (for example, by clearing bit CON.M[0]), which enables it to also receive the data bytes that will be coming (having the wake-up bit cleared). The slaves that were not being addressed remain in 8-bit data + wake-up bit mode, ignoring the following data 'bytes'.

Asynchronous/Synchronous Serial Interface (ASC)**19.1.3.2 Asynchronous Transmission**

Asynchronous transmission begins when the next overflow of the divide-by-16 baud rate timer (transition of the baud rate clock f_{BR}) occurs, if bit CON.R is set and data has been loaded into TBUF. The transmitted data frame consists of three elements:

1. The start bit
2. The data field (8 or 9 bits, LSB first, including a parity bit, if selected)
3. The delimiter (1 or 2 stop bits)

Data transmission is double-buffered. When the transmitter is idle, the transmit data loaded into TBUF is immediately moved to the transmit shift register; thus freeing TBUF for the next transmit data to be loaded. This is indicated by the transmit buffer interrupt request line TBIR being activated. TBUF may then be loaded with the next transmit data while transmission of the previous one continues.

The Transmit Interrupt Request line TIR will be activated before the last bit of a frame is transmitted, that is, before the first or the second stop bit is shifted out of the transmit shift register.

Note: A dedicated GPIO device pin which is connected to the module output pin TXD must be configured by software as alternate data output for asynchronous transmission.

19.1.3.3 Asynchronous Reception

Asynchronous reception is initiated by a falling edge (1-to-0 transition) on pin RXD, on condition that bits CON.R and CON.REN are set. The receive data input pin RXD is sampled at sixteen times the rate of the selected baud rate. A majority decision of the 7th, 8th and 9th sample determines the effective sampled bit value. This avoids erroneous results that may be caused by noise.

If the detected value is not a 0 when the start bit is sampled, the receive circuit is reset and waits for the next 1-to-0 transition at pin RXD. If the start bit proves valid, the receive circuit continues sampling and shifts the incoming data frame into the receive shift register.

When the last stop bit has been received, the contents of the receive shift register are transferred to the Receive Data Buffer Register RBUF. Simultaneously, the receive interrupt request line RIR is activated after the 9th sample in the last stop bit time-slot (as programmed), regardless whether valid stop bits have been received or not. The receive circuit then waits for the next start bit (1-to-0 transition) at the receive data input line.

Note: A dedicated GPIO pin that is connected to the module input pin RXD must be configured by software as input for asynchronous reception.

Asynchronous reception is stopped by clearing bit CON.REN. A currently received frame is completed including generation of the receive interrupt request and an error interrupt request, if appropriate. Start bits that follow this frame will not be recognized.

Asynchronous/Synchronous Serial Interface (ASC)

Note: In wake-up mode, received frames are transferred to the receive buffer register only if the 9th bit (the wake-up bit) is 1. If this bit is 0, no receive interrupt request will be activated and no data will be transferred.

19.1.3.4 RXD/TXD Data Path Selection in Asynchronous Modes

The data paths for the serial input and output data in Asynchronous Modes are affected by control bit CON.LB (loop-back) as shown in [Figure 19-5](#).

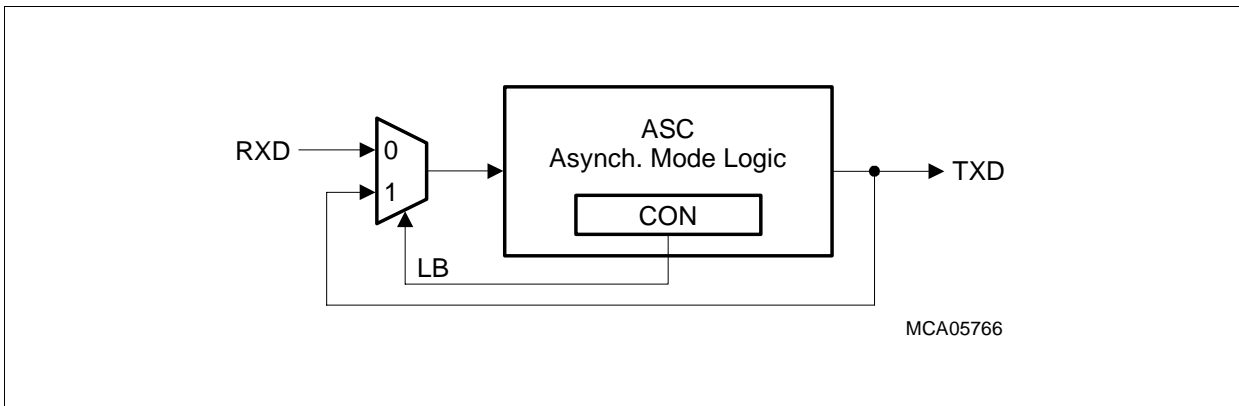


Figure 19-5 RXD/TXD Data Path Selection in Asynchronous Modes

Asynchronous/Synchronous Serial Interface (ASC)

19.1.4 Synchronous Operation

Synchronous Mode supports half-duplex communication, usable for simple I/O expansion via shift registers. Data is transmitted and received via pin RXD while pin TXD outputs the shift clock. These signals are typically connected as alternate functions with GPIO port pins. Synchronous Mode is selected with CON.M = 000_B.

Eight data bits are transmitted or received synchronous to a shift clock generated by the internal baud rate generator. The shift clock is active only as long as data bits are transmitted or received.

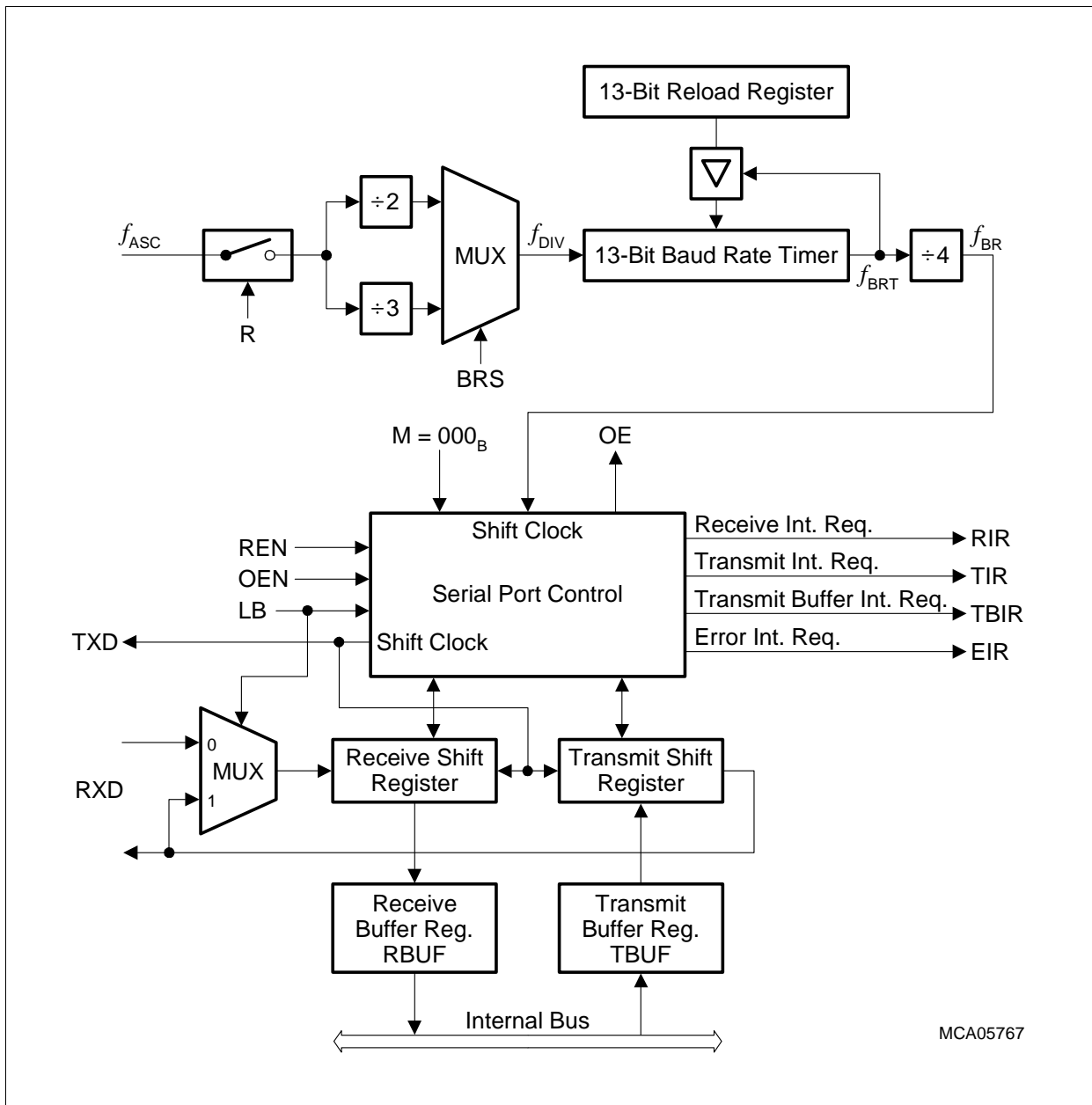


Figure 19-6 Synchronous Mode of Serial Channel ASC

Asynchronous/Synchronous Serial Interface (ASC)**19.1.4.1 Synchronous Transmission**

Synchronous transmission begins within four state times after data has been loaded into TBUF, provided that CON.R is set and CON.REN = 0 (half-duplex, no reception), with one exception: in Loop-back Mode (bit CON.LB set), CON.REN must be set for reception of the transmitted byte. Data transmission is double-buffered. When the transmitter is idle, the transmit data loaded into TBUF is immediately moved to the transmit shift register, thus freeing TBUF for the next data to be sent. This is indicated by the transmit buffer interrupt request line TBIR being activated. TBUF may now be loaded with the next data, while transmission of the previous one continues. The data bits are transmitted synchronously with the shift clock. After the bit time for the 8th data bit, both TXD and RXD will be set to high level, the transmit interrupt request line TIR is activated, and serial data transmission stops.

Note: The dedicated GPIO device pins that are connected to TXD and RXD must be configured by software as alternate data outputs in order to provide the shift clock and the output data during synchronous transmission.

19.1.4.2 Synchronous Reception

Synchronous reception is initiated by setting bit CON.REN = 1. If bit CON.R = 1, the data applied at RXD is clocked into the receive shift register synchronously to the clock which is output at TXD. After the 8th bit has been shifted in, the contents of the receive shift register are transferred to the receive data buffer RBUF, the receive interrupt request line RIR is activated, the receiver enable bit CON.REN is cleared, and serial data reception stops.

Synchronous reception is stopped by clearing bit CON.REN. Any byte that is currently being received is completed, including the generation of the receive interrupt request and an error interrupt request, if appropriate. Writing to the transmit buffer register while a reception is in progress has no effect on reception and will not start a transmission.

If a previously received byte has not been read out of the receive buffer register by the time the reception of the next byte is complete, both the error interrupt request line EIR and the overrun error status flag CON.OE will be activated/set, provided that the overrun check has been enabled by bit CON.OEN.

Note: The dedicated GPIO device pin that is connected to TXD must be configured by software as alternate data output in order to provide the shift clock. The dedicated GPIO device pin that is connected to RXD must be configured by software as input during synchronous reception.

Asynchronous/Synchronous Serial Interface (ASC)

19.1.4.3 Synchronous Timing

Figure 19-7 shows timing diagrams of the ASC Synchronous Mode data reception and data transmission. In idle state, the shift clock is at high level. With the beginning of a synchronous transmission of a data byte, the data is shifted out at RXD with the falling edge of the shift clock. If a data byte is received through RXD, data is latched with the rising edge of the shift clock.

One shift clock cycle (f_{BR}) delay is inserted between two consecutive receive or transmit data bytes.

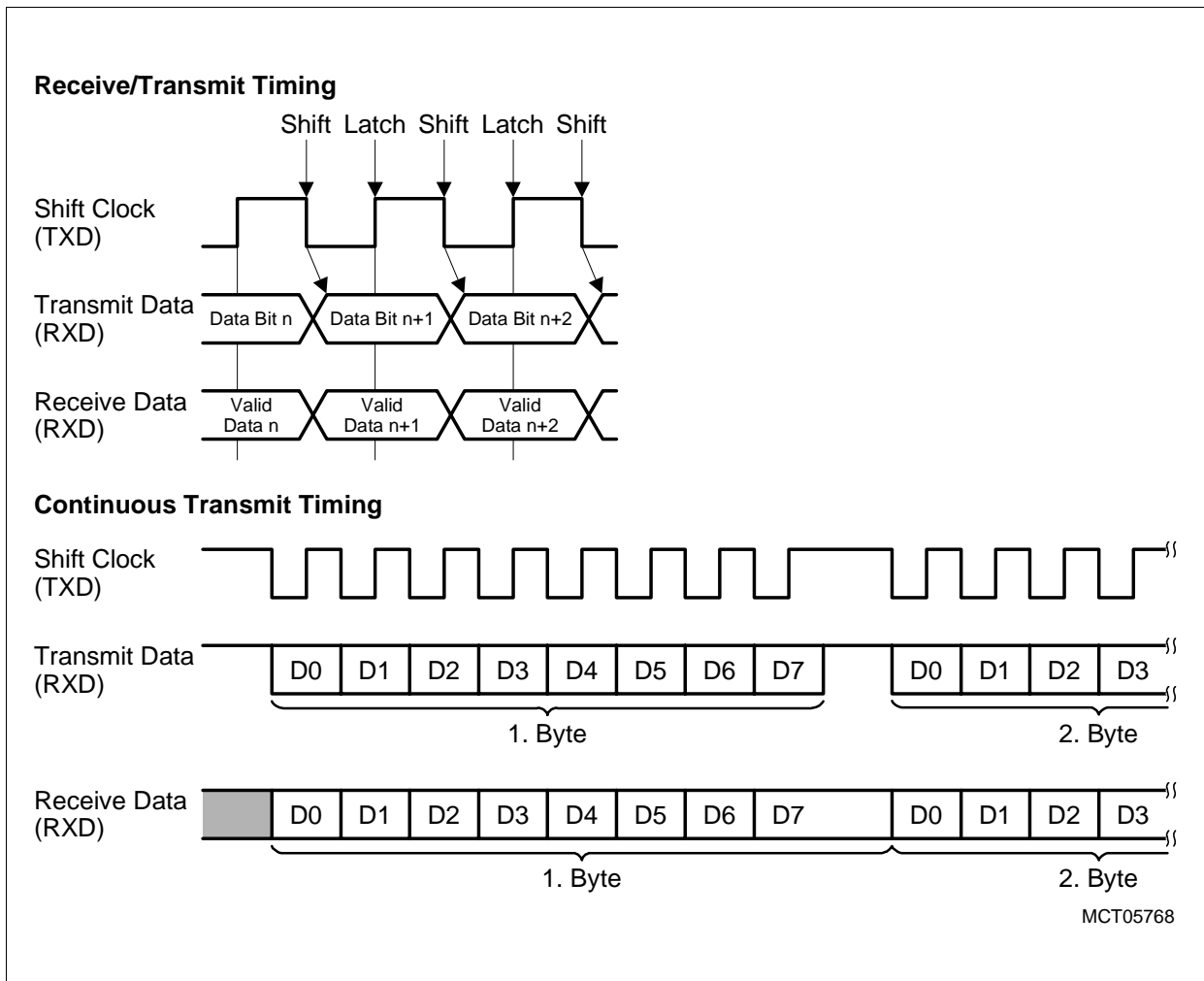


Figure 19-7 ASC Synchronous Mode Waveforms

Asynchronous/Synchronous Serial Interface (ASC)**19.1.5 Baud Rate Generation**

The ASC has its own dedicated 13-bit baud rate generator with 13-bit reload capability, allowing baud rate generation independent of other timers.

The baud rate generator is clocked with a clock (f_{DIV}) which is derived via a prescaler from the ASC module clock f_{ASC} . The baud rate timer is counting downwards and can be started or stopped through the baud rate generator run bit CON.R. Each underflow of the timer generates one clock pulse to the serial channel. The timer is reloaded with the value stored in its 13-bit reload register at each underflow. The resulting clock f_{BRT} is again divided by a factor for the baud rate clock ($\div 16$ in asynchronous operating modes and $\div 4$ in synchronous operating mode). The prescaler is selected by the bits CON.BRS and CON.FDE. In the asynchronous operating modes, a fractional divider prescaler unit is available (in addition to the two fixed dividers) that allows selection of prescaler divider ratios of $n/512$ with $n = 0-511$. Therefore, the baud rate of ASC is determined by the module clock, the content of register FDV, the reload value in register BG, and the operating mode (asynchronous or synchronous).

Register BG is the dual-function baud rate generator/reload register. Reading BG returns the contents of the timer in bit field BR_VALUE (bits 31:13 return zero), while writing to BG always updates the reload register (bits 31:13 are insignificant).

An auto-reload of the timer with the contents of the reload register is performed each time BG is written to. However, if CON.R = 0 at the time the write operation to BG is performed, the timer will not be reloaded until the first instruction cycle after CON.R = 1. For a clean baud rate initialization, BG should only be written if CON.R = 0. If BG is written with CON.R = 1, an unpredicted behavior of the ASC may occur during running transmit or receive operations.

Asynchronous/Synchronous Serial Interface (ASC)

19.1.5.1 Baud Rates in Asynchronous Mode

For asynchronous operation, the baud rate generator provides a clock f_{BRT} with sixteen times the rate of the established baud rate. Every received bit is sampled on the 7th, 8th and 9th cycle of this clock. The clock divider circuitry, which generates the input clock f_{DIV} for the 13-bit baud rate timer, is extended by a fractional divider circuitry that allows the adjustment of more accurate baud rates and the extension of the baud rate range.

The baud rate of the baud rate generator depends on the settings of the following bits and register values:

- Input clock f_{ASC}
- Selection of the baud rate timer input clock f_{DIV} by bits CON.FDE and CON.BRS
- If bit CON.FDE = 1 (fractional divider): value of register FDV
- Value of the 13-bit reload register BG

The output clock of the baud rate timer with the reload register is the sample clock in the asynchronous operating modes of the ASC. For baud rate calculations, this baud rate clock f_{BR} is derived from the sample clock f_{BRT} by a division of sixteen.

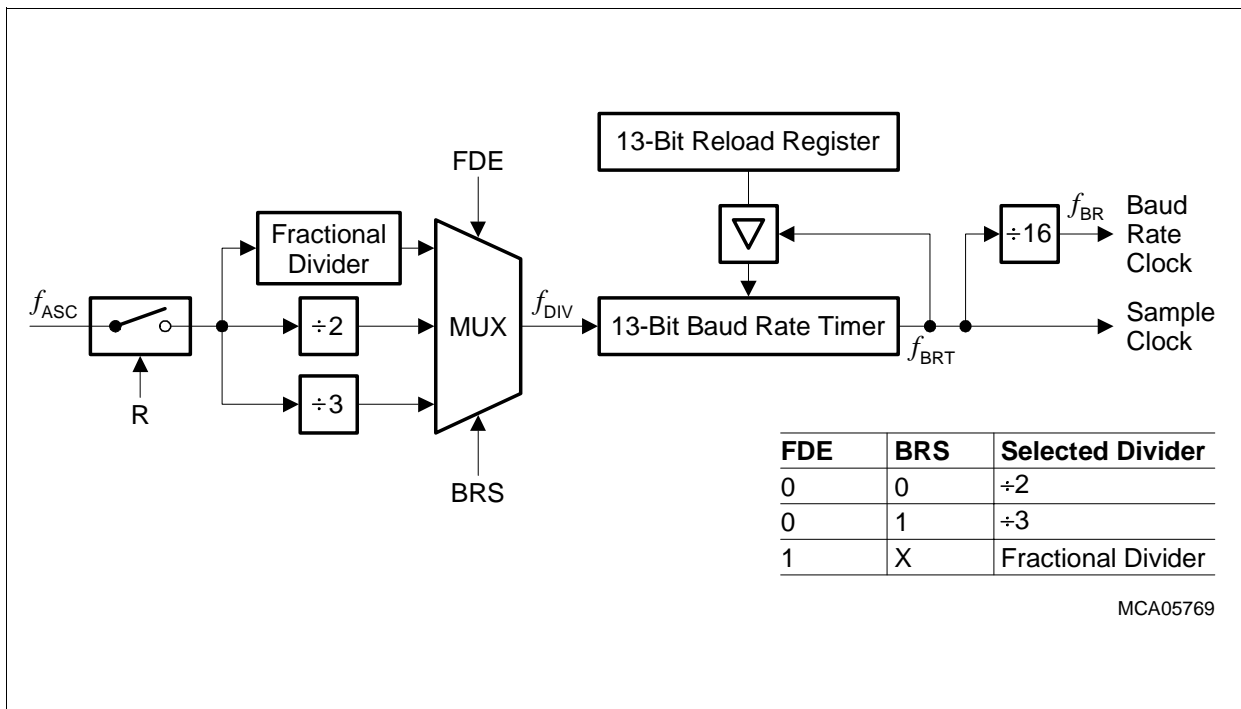


Figure 19-8 ASC Baud Rate Generator Circuitry in Asynchronous Modes

Asynchronous/Synchronous Serial Interface (ASC)

Using the Fixed Input Clock Divider

The baud rate for asynchronous operation of the serial channel ASC when using the fixed input clock divider ratios (CON.FDE = 0) and the required BG reload value for a given baud rate can be determined by the following formulas:

Table 19-1 Asynchronous Baud Rate Formulas using the Fixed Input Clock Dividers

FDE	BRS	BG	Formula
0	0	0 ... 8191	$\text{Baud rate} = \frac{f_{ASC}}{32 \times (BG + 1)}$ $BG = \frac{f_{ASC}}{32 \times \text{Baud rate}} - 1$
	1		$\text{Baud rate} = \frac{f_{ASC}}{48 \times (BG + 1)}$ $BG = \frac{f_{ASC}}{48 \times \text{Baud rate}} - 1$

BG represents the content of the reload register bit field BG.BR_VALUE, taken as an unsigned 13-bit integer.

The maximum baud rate that can be achieved for the asynchronous operating modes when using the two fixed clock dividers and a module clock of 75 MHz is 2.34 Mbit/s. [Table 19-2](#) lists various commonly used baud rates together with the required reload values and the deviation errors compared to the intended baud rate.

Table 19-2 Typical Asynchronous Baud Rates using Fixed Input Clock Dividers

Baud Rate	CON.BRS = 0, $f_{ASC} = 75 \text{ MHz}$		CON.BRS = 1, $f_{ASC} = 75 \text{ MHz}$	
	Deviation Error	Reload Value	Deviation Error	Reload Value
2.344 Mbit/s	–	0000 _H	–	–
19.2 kbit/s	+0.1% / -0.8%	0079 _H / 007A _H	+0.5% / -0.8%	0050 _H / 0051 _H
9600 bit/s	+0.1% / -0.4%	00F3 _H / 00F4 _H	+0.5% / -0.1%	00A1 _H / 00A2 _H
4800 bit/s	+0.1% / -0.2%	01E7 _H / 01E8 _H	+0.2% / -0.1%	0144 _H / 0145 _H

Note: CON.FDE must be 0 to achieve the baud rates in the table above. The deviation errors given in the table are rounded. Using a baud rate crystal will provide correct baud rates without deviation errors.

Asynchronous/Synchronous Serial Interface (ASC)

Using the Fractional Divider

When the fractional divider is selected, the input clock f_{DIV} for the baud rate timer is derived from the module clock f_{ASC} by a programmable fractional divider. If $CON.FDE = 1$, the fractional divider is activated. It divides f_{ASC} by a fraction of $n/512$ for any value of n from 0 to 511. If $n = 0$, the divider ratio is 1, which means that $f_{DIV} = f_{ASC}$. In general, the fractional divider allows the baud rate to be programmed with much better accuracy than with the two fixed prescaler divider stages.

Note: In fractional divider mode, the clock f_{DIV} can have a maximum period jitter of one f_{ASC} clock period.

Table 19-3 Asynchronous Baud Rate Formulas using the Fractional Input Clock Divider

FDE	BRS	BG	FDV	Formula
1	–	0 ... 8191	1 ... 511	Baud rate = $\frac{FDV}{512} \times \frac{f_{ASC}}{16 \times (BG + 1)}$
			0	Baud rate = $\frac{f_{ASC}}{16 \times (BG + 1)}$

BG represents the content of the reload register bit field $BG.BR_VALUE$, taken as an unsigned 13-bit integer. FDV represents the contents of the fractional divider register bit field $FDV.FD_VALUE$, taken as an unsigned 9-bit integer.

Table 19-4 Typical Asynchronous Baud Rates using the Fractional Input Clock Divider

f_{ASC}	Desired Baud Rate	BG	FDV	Resulting Baud Rate	Deviation
75 MHz	115.2 kbit/s	17 _H	12E _H	115.204 kbit/s	< 0.01%
	57.6 kbit/s	2F _H	12E _H	57.602 kbit/s	< 0.01%
	38.4 kbit/s	23 _H	097 _H	38.401 kbit/s	< 0.01%
	19.2 kbit/s	AE _H	16F _H	19.200 kbit/s	< 0.01%

Asynchronous/Synchronous Serial Interface (ASC)

19.1.5.2 Baud Rates in Synchronous Mode

For synchronous operation, the baud rate generator provides a clock f_{BRT} that runs with four times the established baud rate (see Figure 19-9).

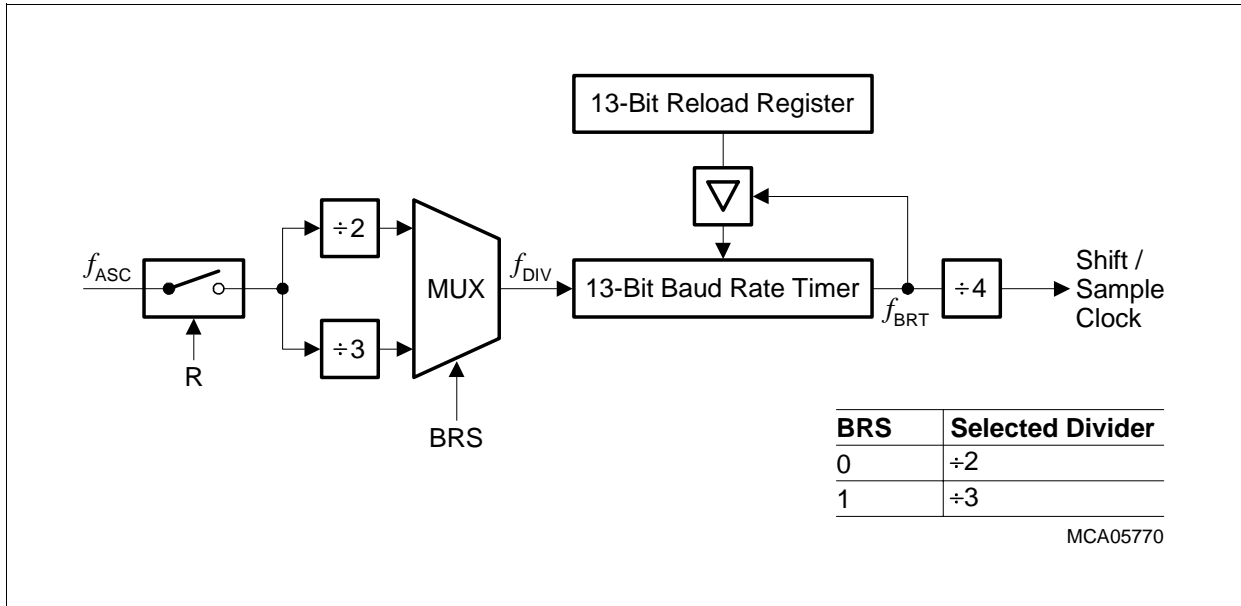


Figure 19-9 ASC Baud Rate Generator Circuitry in Synchronous Mode

The baud rate for synchronous operation of the serial channel ASC can be determined by the formulas as shown in Table 19-5.

Table 19-5 Synchronous Baud Rate Formulas

BRS	BG	Formula
0	0 ... 8191	Baud rate = $\frac{f_{ASC}}{8 \times (BG + 1)}$ BG = $\frac{f_{ASC}}{8 \times \text{Baud rate}} - 1$
1		Baud rate = $\frac{f_{ASC}}{12 \times (BG + 1)}$ BG = $\frac{f_{ASC}}{12 \times \text{Baud rate}} - 1$

BG represents the content of the reload register bit field BG.BR_VALUE, taken as an unsigned 13-bit integer.

The maximum baud rate that can be achieved in Synchronous Mode when using a module clock of 75 MHz is 9.38 Mbit/s.

Asynchronous/Synchronous Serial Interface (ASC)

19.1.6 Hardware Error Detection Capabilities

To improve the reliability of serial data exchange, the serial channel ASC provides an error interrupt request flag that indicates the presence of an error and three (selectable) error status flags in register CON that indicate which error has been detected during reception. Upon completion of a reception, the error interrupt request line EIR will be activated simultaneously with the receive interrupt request line RIR, if one or more of the following conditions are met:

- If the framing error detection enable bit CON.FEN is set and any of the expected stop bits is not high, the framing error flag CON.FE is set, indicating that the error interrupt request is due to a framing error (asynchronous operating modes only).
- If the parity error detection enable bit CON.PEN is set in the modes where a parity bit is received and the parity check on the received data bits proves false, the parity error flag CON.PE is set, indicating that the error interrupt request is due to a parity error (asynchronous operating modes only).
- If the overrun error detection enable bit CON.OEN is set and the last character received was not read out of the receive buffer by software or DMA transfer at the time the reception of a new frame is complete, the overrun error flag CON.OE is set indicating that the error interrupt request is due to an overrun error (Asynchronous Mode and Synchronous Mode).

19.1.7 Interrupts

Four interrupt sources are provided for serial channel ASC. Line TIR indicates a transmit interrupt, TBIR indicates a transmit buffer interrupt, RIR indicates a receive interrupt, and EIR indicates an error interrupt of the serial channel. The interrupt output lines TBIR, TIR, RIR, and EIR are activated (active state) for two periods of the module clock f_{ASC} .

The cause of an error interrupt request EIR (framing, parity, overrun error) can be identified by the error status flags CON.FE, CON.PE, and CON.OE.

Note: By contrast to the error interrupt request line EIR, the error status flags CON.FE/CON.PE/CON.OE are not cleared automatically but must be cleared by software.

For normal operation (that is, other than error interrupt), the ASC provides three interrupt requests to control data exchange via this serial channel:

- TBIR is activated when data is moved from TBUF to the transmit shift register.
- TIR is activated before the last bit of an asynchronous frame is transmitted, or after the last bit of a synchronous frame has been transmitted.
- RIR is activated when the received frame is moved to RBUF.

While the task of the receive interrupt handler is quite clear, the transmitter is serviced by two interrupt handlers. This provides advantages for the servicing software.

Asynchronous/Synchronous Serial Interface (ASC)

For single transfers, it is sufficient to use the transmitter interrupt (TIR), which indicates that the previously loaded data has been transmitted, except for the last bit of an asynchronous frame.

For multiple back-to-back transfers, it is necessary to load the following piece of data at least before the last bit of the previous frame has been transmitted. In Asynchronous Mode, this leaves just one bit-time for the handler to respond to the transmitter interrupt request; in Synchronous Mode, it is entirely impossible.

Using the Transmit Buffer Interrupt (TBIR) to reload transmit data provides the time necessary to transmit a complete frame for the service routine, as TBUF may be reloaded while the previous data is still being transmitted.

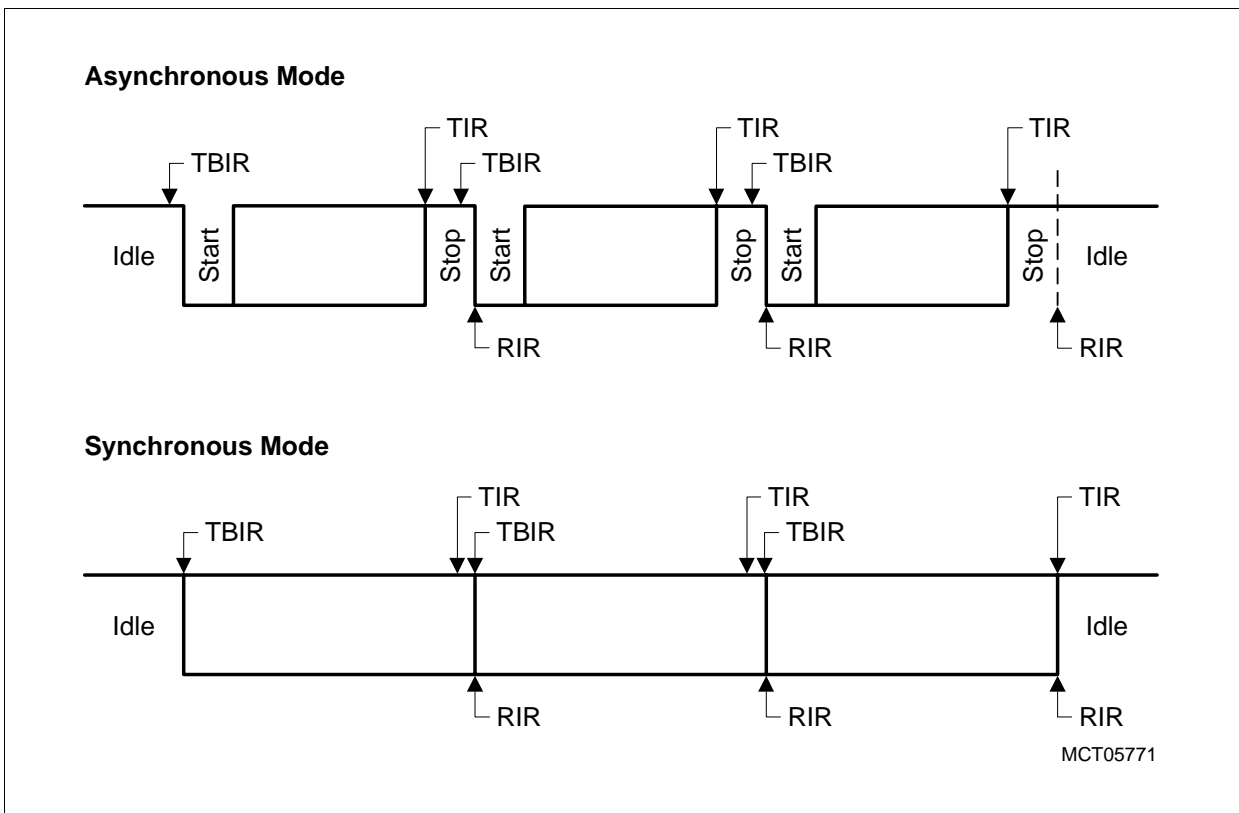


Figure 19-10 ASC Interrupt Generation

As shown in [Figure 19-10](#), TBIR is an early trigger for the reload routine, while TIR indicates the completed transmission. Software using handshake should, therefore, rely on TIR at the end of a data block to ensure that all data has been transmitted.

Asynchronous/Synchronous Serial Interface (ASC)

19.2 ASC Kernel Registers

This section describes the kernel registers of the ASC module. All ASC kernel register names described in this section will be referenced in other parts of the TC1796 User's Manual by the module name prefix "ASC0_" for the ASC0 interface and "ASC1_" for the ASC1 interface.

ASC Kernel Register Overview

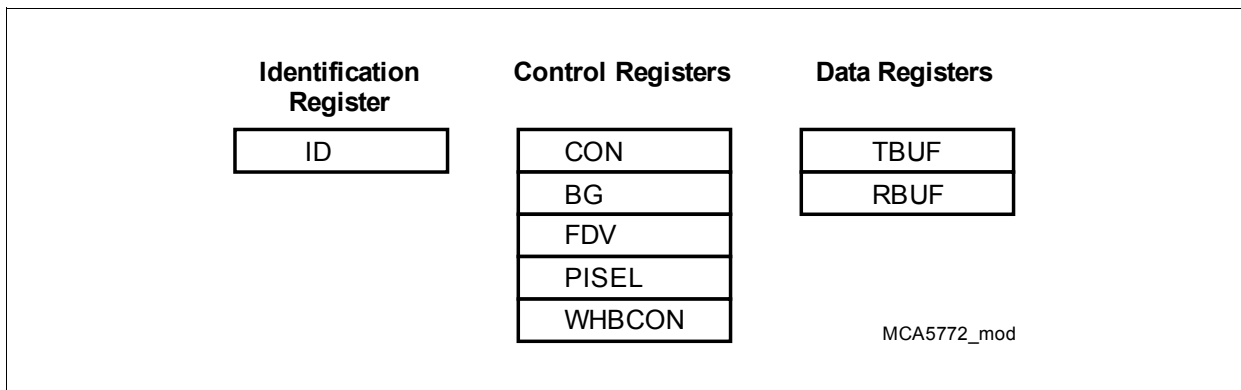


Figure 19-11 ASC Kernel Registers

The complete and detailed address map of the of the ASC modules is described in [Table 18-8](#) on [Page 18-20](#) of the TC1796 User's Manual System Units part (Volume 1).

Table 19-6 Registers Address Space

Module	Base Address	End Address	Note
ASC0	F000 0A00 _H	F000 0AFF _H	–
ASC1	F000 0B00 _H	F000 0BFF _H	–

Table 19-7 Registers Overview - ASC Kernel Registers

Register Short Name	Register Long Name	Offset Address ¹⁾	Description see
PISEL	Peripheral Input Select Register	04 _H	Page 19-21
ID	Module Identification Register	08 _H	Page 19-20
CON	Control Register	10 _H	Page 19-22
BG	Baud Rate Timer Reload Register	14 _H	Page 19-27
FDV	Fractional Divider Register	18 _H	Page 19-27
TBUF	Transmit Buffer Register	20 _H	Page 19-28

Asynchronous/Synchronous Serial Interface (ASC)

19.2.2 Control Registers

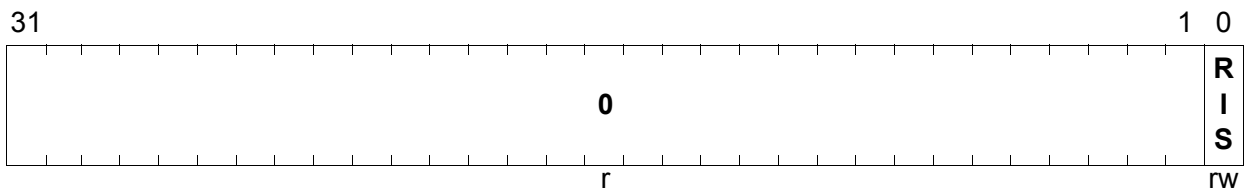
The ASC module kernel provides two receive input lines, RXD_I0 and RXD_I1. Bit RIS in the Peripheral Input Select Register PISEL determines which of these two input lines is taken for RXD receive input purposes.

PISEL

Peripheral Input Select Register

(04_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
RIS	0	rw	Receive Input Select 0 _B ASC receiver input RXD_I0 selected 1 _B ASC receiver input RXD_I1 selected
0	[31:1]	r	Reserved Read as 0; should be written with 0.

Asynchronous/Synchronous Serial Interface (ASC)

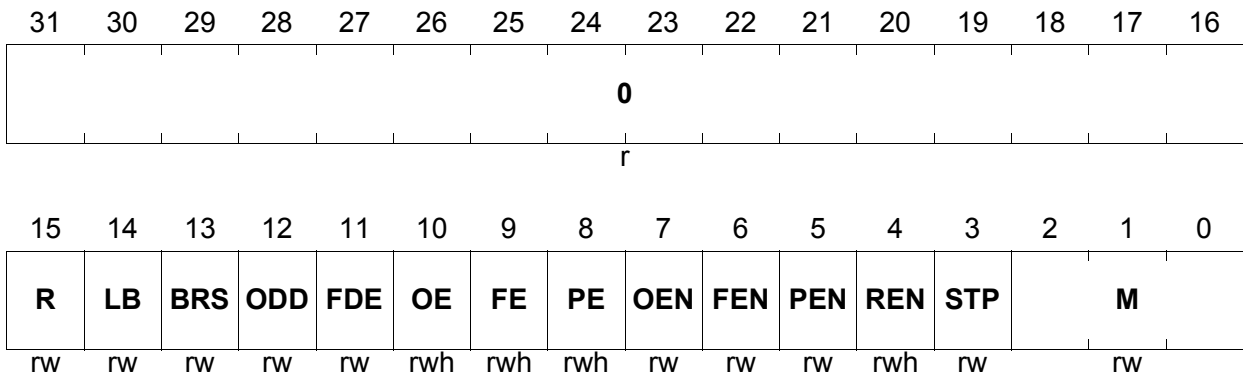
The serial operating modes of the ASC module are controlled by its Control Register CON. This register contains control bits for mode and error check selection, and status flags for error identification.

CON

Control Register

(10_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
M	[2:0]	rw	Mode Selection 000 _B 8-bit data Synchronous Mode 001 _B 8-bit data Asynchronous Mode 010 _B Reserved. Do not use this combination. 011 _B 7-bit data + parity Asynchronous Mode 100 _B 9-bit data Asynchronous Mode 101 _B 8-bit data + wake up bit Asynchronous Mode 110 _B Reserved. Do not use this combination. 111 _B 8-bit data + parity Asynchronous Mode
STP	3	rw	Number of Stop Bit Selection 0 _B One stop bit 1 _B Two stop bits
REN	4	rwh	Receiver Enable Control 0 _B Receiver disabled 1 _B Receiver enabled Bit is cleared by hardware after reception of a byte in Synchronous Mode.
PEN	5	rw	Parity Check Enable (asynchronous modes only) 0 _B Ignore parity 1 _B Check parity

Asynchronous/Synchronous Serial Interface (ASC)

Field	Bits	Type	Description
FEN	6	rw	Framing Check Enable (asynchronous mode only) 0 _B Ignore framing errors 1 _B Check framing errors
OEN	7	rw	Overrun Check Enable 0 _B Ignore overrun errors 1 _B Check overrun errors
PE	8	rwh	Parity Error Flag Set by hardware on a parity error (PEN = 1). Must be cleared by software.
FE	9	rwh	Framing Error Flag Set by hardware on a framing error (FEN = 1). Must be cleared by software.
OE	10	rwh	Overrun Error Flag Set by hardware on an overrun error (OEN = 1). Must be cleared by software.
FDE	11	rw	Fractional Divider Enable 0 _B Fractional divider disabled 1 _B Fractional divider is enabled and used as prescaler for baud rate timer (bit BRS is “don’t care”)
ODD	12	rw	Parity Selection 0 _B Even parity selected (parity bit = 1 on odd number of 1s in data, parity bit = 0 on even number of 1s in data) 1 _B Odd parity selected (parity bit = 1 on even number of 1s in data, parity bit = 0 on odd number of 1s in data)
BRS	13	rw	Baud Rate Selection 0 _B Baud rate timer prescaler divide-by-2 selected 1 _B Baud rate timer prescaler divide-by-3 selected BRS is “don’t care” if FDE = 1 (fractional divider enabled)
LB	14	rw	Loopback Mode Enable 0 _B Loop-Back mode disabled 1 _B Loop-Back mode enabled

Asynchronous/Synchronous Serial Interface (ASC)

Field	Bits	Type	Description
R	15	rw	Baud Rate Generator Run Control 0 _B Baud rate generator disabled (ASC inactive) 1 _B Baud rate generator enabled Register BG should only be written if R = 0.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

Serial data transmission or reception is possible only when the run bit CON.R is set to 1. Otherwise, the serial interface is idle. To avoid unpredictable behavior of the serial interface, do not program the mode control field CON.M to one of the reserved combinations.

Critical “rwh” Bits

Register CON contains three error flags: PE, FE, and OE. If the software modifies only one of these error flags, it uses typically a Read-Modify-Write (RMW) instruction. When one of the other error flags that is not intended to be modified by the RMW instruction is changed by hardware after the read access but before the write back access of the RMW instruction, it is overwritten with the old bit value, and the hardware change of the bit gets lost. This problem does not affect the bits that are intended to be modified by the RMW instruction. It only affects bits that were not intended to be changed with the RMW instruction.

The three error flags in register CON and the REN bit can be additionally set or cleared by software via register WHBCON. This capability avoids the problem with the CON register RMW instruction access to the error flags. WHBCON is a write-only register. Reading WHBCON always returns 0000 0000_H.

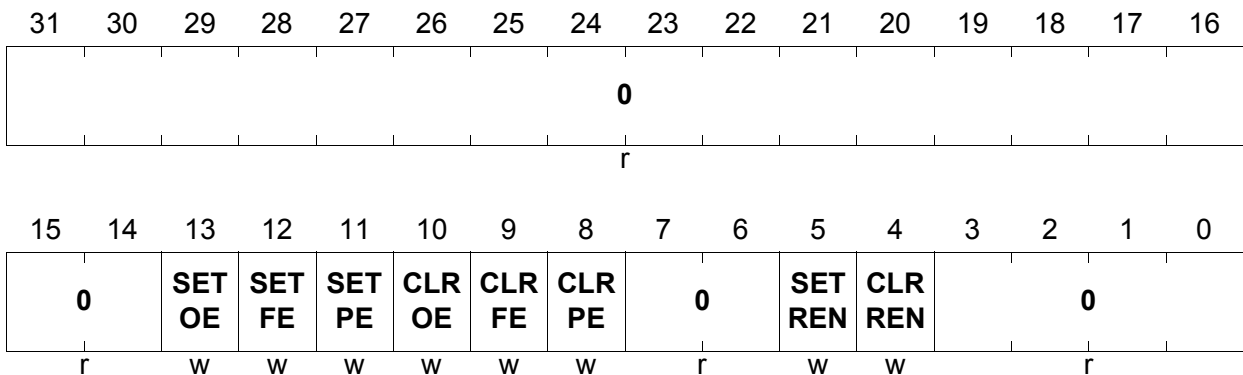
Asynchronous/Synchronous Serial Interface (ASC)

The WHBCON register contains control bits that makes it possible to set/clear flags of the CON register.

WHBCON

Write Hardware Bits Control Register (50_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CLRREN	4	w	Clear Receiver Enable Bit 0 _B No effect. 1 _B Bit CON.REN is cleared. Bit is always read as 0.
SETREN	5	w	Set Receiver Enable Bit 0 _B No effect. 1 _B Bit CON.REN is set. Bit is always read as 0.
CLRPE	8	w	Clear Parity Error Flag 0 _B No effect. 1 _B Bit CON.PE is cleared. Bit is always read as 0.
CLRFE	9	w	Clear Framing Error Flag 0 _B No effect. 1 _B Bit CON.FE is cleared. Bit is always read as 0.
CLROE	10	w	Clear Overrun Error Flag 0 _B No effect. 1 _B Bit CON.OE is cleared. Bit is always read as 0.

Asynchronous/Synchronous Serial Interface (ASC)

Field	Bits	Type	Description
SETPE	11	w	Set Parity Error Flag 0 _B No effect. 1 _B Bit CON.PE is set. Bit is always read as 0.
SETFE	12	w	Set Framing Error Flag 0 _B No effect. 1 _B Bit CON.FE is set. Bit is always read as 0.
SETOE	13	w	Set Overrun Error Flag 0 _B No effect. 1 _B Bit CON.OE is set. Bit is always read as 0.
0	[3:0], [7:6], [31:14]	r	Reserved Read as 0; should be written with 0.

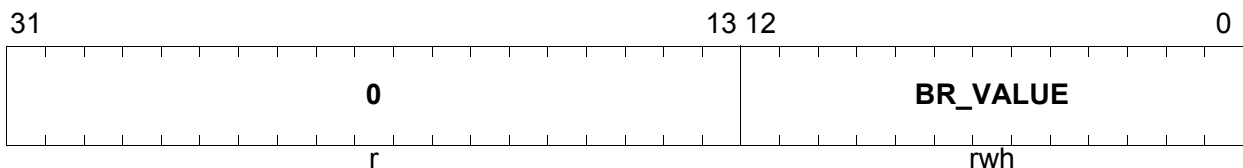
Note: When the set and clear bits for an error flag are set at the same time during a WHBCON write operation (e.g SETPE = CLRPE = 1), the error flag in CON is not affected.

Asynchronous/Synchronous Serial Interface (ASC)

The Baud Rate Timer Reload Register BG of the ASC module contains the 13-bit reload value for the baud rate timer in Asynchronous Mode and Synchronous Mode.

BG

Baud Rate Timer/Reload Register (14_H) **Reset Value: 0000 0000_H**

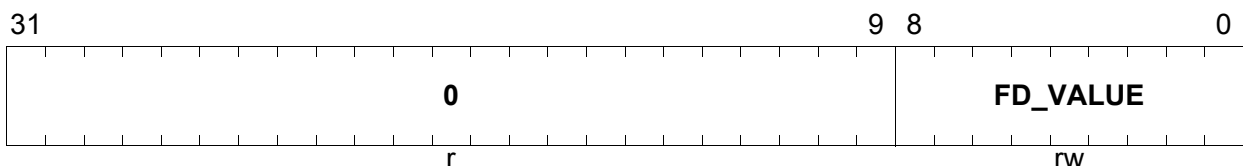


Field	Bits	Type	Description
BR_VALUE	[12:0]	rwh	Baud Rate Timer/Reload Register Value Reading BR_VALUE returns the 13-bit content of the baud rate timer. Writing BR_VALUE loads the baud rate timer reload register. BG should only be written if CON.R = 0.
0	[31:13]	r	Reserved Read as 0; should be written with 0.

The Fractional Divider Register FDV of the ASC module contains the 9-bit divider value for the fractional divider (Asynchronous Mode only).

FDV

Fractional Divider Register (18_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
FD_VALUE	[8:0]	rw	Fractional Divider Register Value FD_VALUE contains the 9-bit value n of the fractional divider which determines the fractional divider ratio n/512 (n = 0-511). With n = 0, the fractional divider is switched off (divider ratio = 1).
0	[31:9]	r	Reserved Read as 0; should be written with 0.

Asynchronous/Synchronous Serial Interface (ASC)

19.2.3 Data Registers

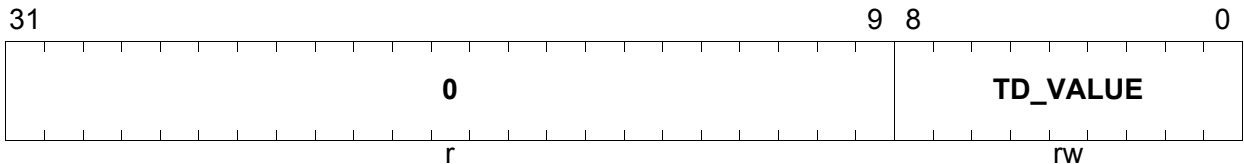
The Transmit Buffer Register TBUF of the ASC module contains the transmit data value in Asynchronous Mode and Synchronous Mode.

TBUF

Transmit Buffer Register

(20_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
TD_VALUE	[8:0]	rw	Transmit Data Register Value TBUF contains the data to be transmitted in the asynchronous and synchronous operating modes of the ASC. Data transmission is double-buffered; therefore, a new value can be written to TBUF before the transmission of the previous value is complete.
0	[31:9]	r	Reserved Read as 0; should be written with 0.

Asynchronous/Synchronous Serial Interface (ASC)

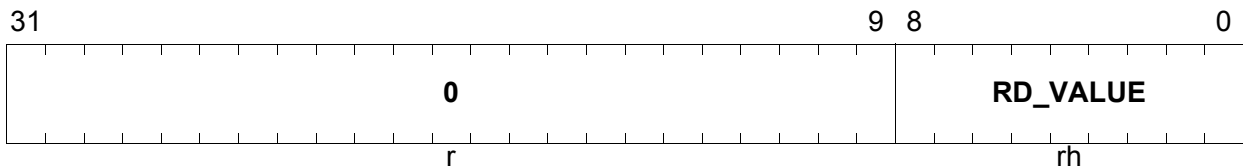
The Receive Buffer Register RBUF of the ASC module contains the receive data value in Asynchronous Mode and Synchronous Mode.

RBUF

Receive Buffer Register

(24_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
RD_VALUE	[8:0]	rh	<p>Receive Data Register Value</p> <p>RBUF contains the received data bits and, depending on the selected mode, the parity bit in the asynchronous and synchronous operating modes of the ASC.</p> <p>In Asynchronous Mode, with CON.M = 011_B (7-bit data + parity), the received parity bit is written into RBUF.7.</p> <p>In Asynchronous Mode, with CON.M = 111_B (8-bit data + parity), the received parity bit is written into RBUF.8.</p>
0	[31:9]	r	<p>Reserved</p> <p>Read as 0.</p>

Asynchronous/Synchronous Serial Interface (ASC)

19.3 ASC0/ASC1 Module Implementation

This section describes ASC0/ASC1 module interfaces with the clock control, port connections, interrupt control, and address decoding.

19.3.1 Interfaces of the ASC Modules

The serial I/O lines of both modules can be connected either to Port 5 or Port 6. Each of the ASC modules is further supplied with interrupt control, address decoding, and port control logic. Two DMA requests can be generated by each ASC module. Both ASC modules are supplied by one common clock control unit.

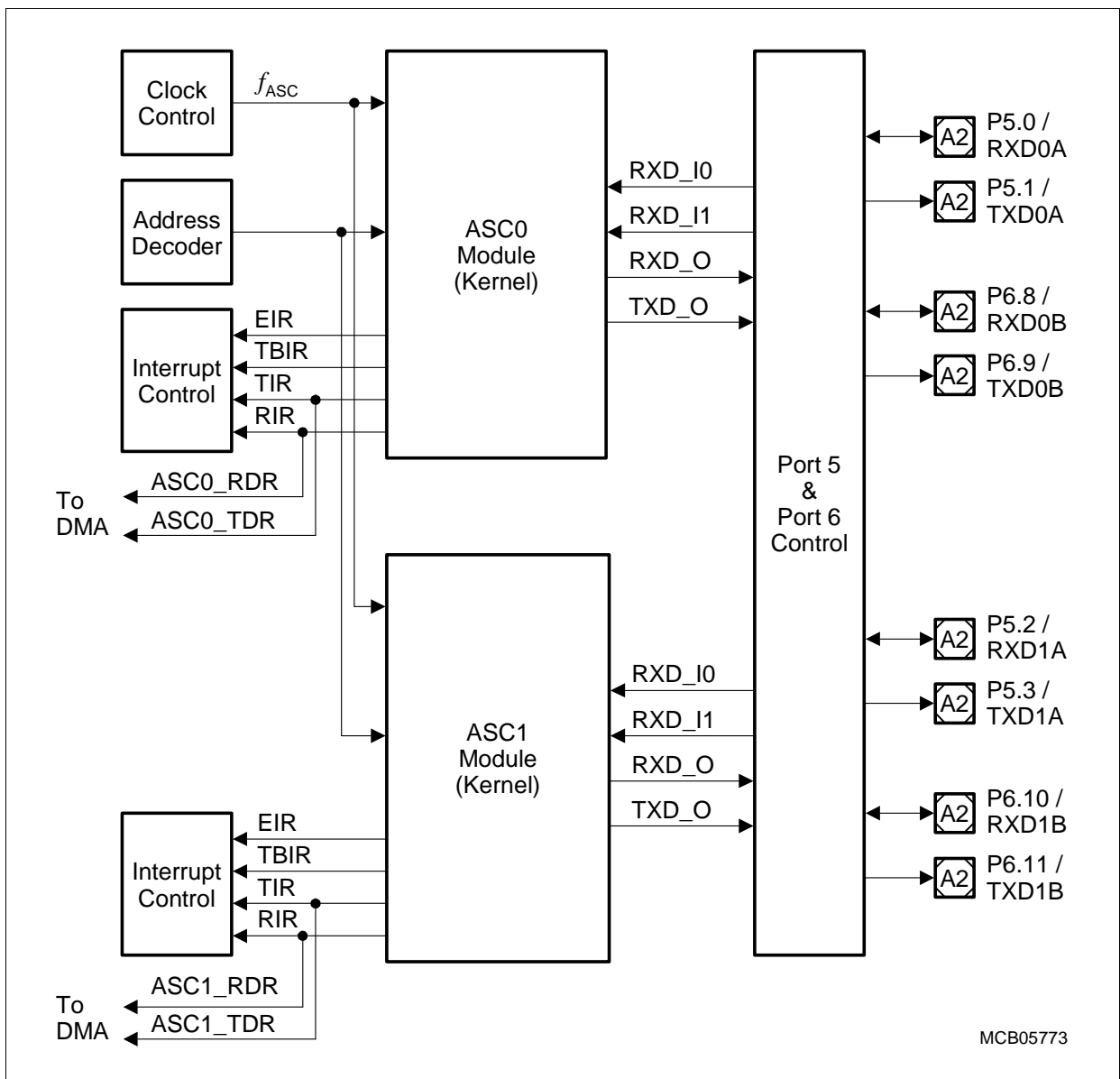


Figure 19-12 ASC0/ASC1 Module Implementation and Interconnections

Asynchronous/Synchronous Serial Interface (ASC)

19.3.2 ASC0/ASC1 Module Related External Registers

Figure 19-13 summarizes the module-related external registers which are required for ASC0/ASC1 programming (see also Figure 19-7 for the module kernel-specific registers).

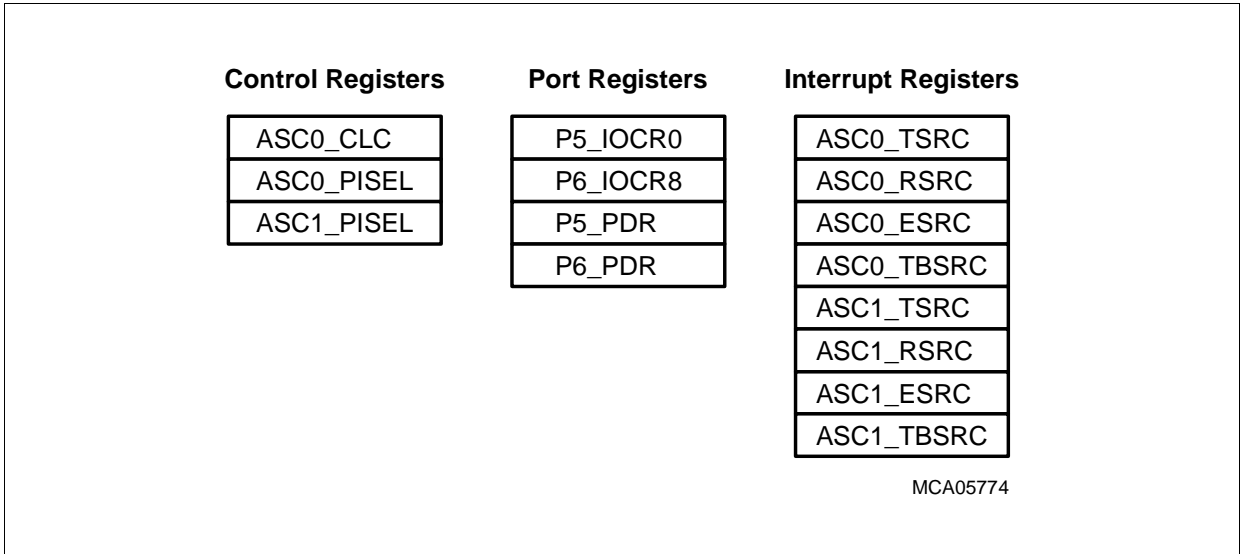


Figure 19-13 ASC0/ASC1 Implementation-specific Special Function Registers

Asynchronous/Synchronous Serial Interface (ASC)

19.3.2.1 ASC0 Clock Control Register

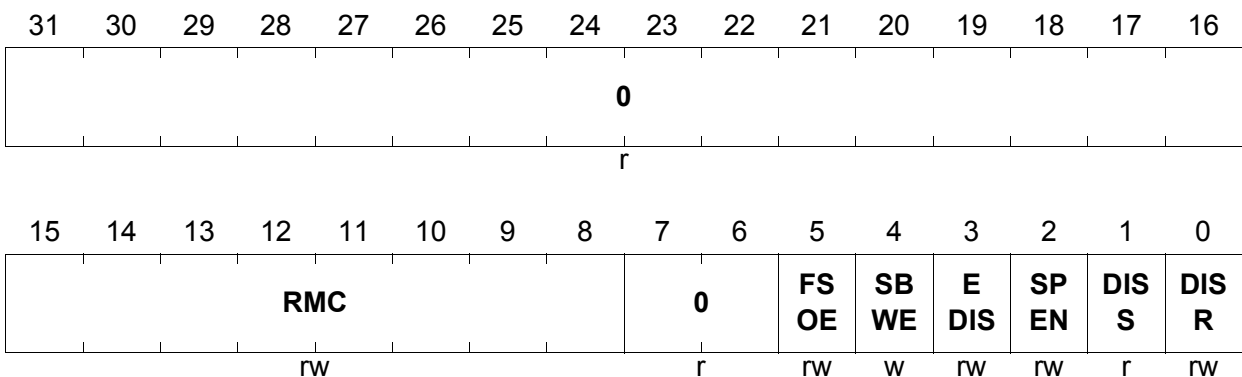
The Clock Control Register ASC0_CLC allows the programmer to adapt the functionality and power consumption of the ASC modules to the requirements of the application. The description below shows the clock control register functionality which is implemented for the ASC modules. Because ASC0 and ASC1 share one common clock control interface, ASC0_CLC controls the f_{ASC} module clock signal, sleep mode, suspend mode and fast shut-off mode for both modules.

ASC0_CLC

ASC0 Clock Control Register

(00_H)

Reset Value: 0000 0003_H



Field	Bits	Type	Description
DISR	0	rw	Module Disable Request Bit Used for enable/disable control of the module.
DISS	1	r	Module Disable Status Bit Bit indicates the current status of the module.
SPEN	2	rw	Module Suspend Enable for OCDS Used to enable the suspend mode.
EDIS	3	rw	Sleep Mode Enable Control Used to control module's sleep mode.
SBWE	4	w	Module Suspend Bit Write Enable for OCDS Determines whether SPEN and FSOE are write-protected.
FSOE	5	rw	Fast Switch Off Enable Used to switch off fast clock in Suspend Mode.
RMC	[15:8]	rw	8-bit Clock Divider Value in RUN Mode
0	[7:6], [31:16]	r	Reserved Read as 0; should be written with 0.

Asynchronous/Synchronous Serial Interface (ASC)

Note: After a hardware reset operation, the two ASC modules are disabled.

Note: The number of module clock cycles (wait states) which are required for a “destructive read” access (means: flags/bits are set/cleared by one read access) to ASC module register depends on the selected CLC clock frequency, which is selected via bit field RMC in the CLC register. Therefore, increasing ASC0_CLC.RMC may result in a longer SPB read cycle access time.

*Note: Further details of the clock control register functionality are described in section **“Clock Control Register CLC”** on **Page 3-24** of the TC1796 User’s Manual System Units part (Volume 1).*

Asynchronous/Synchronous Serial Interface (ASC)

19.3.2.2 Peripheral Input Select Registers

The ASC0/ASC1 modules include a peripheral input select register that is used to switch the RXD input lines of the ASC0/ASC1 module kernels alternatively to Port 5 or Port 6 as shown in **Table 19-14**. Register ASC0_PISEL controls the RXD input selection for ASC0, and ASC1_PISEL controls the RXD input selection for ASC1.

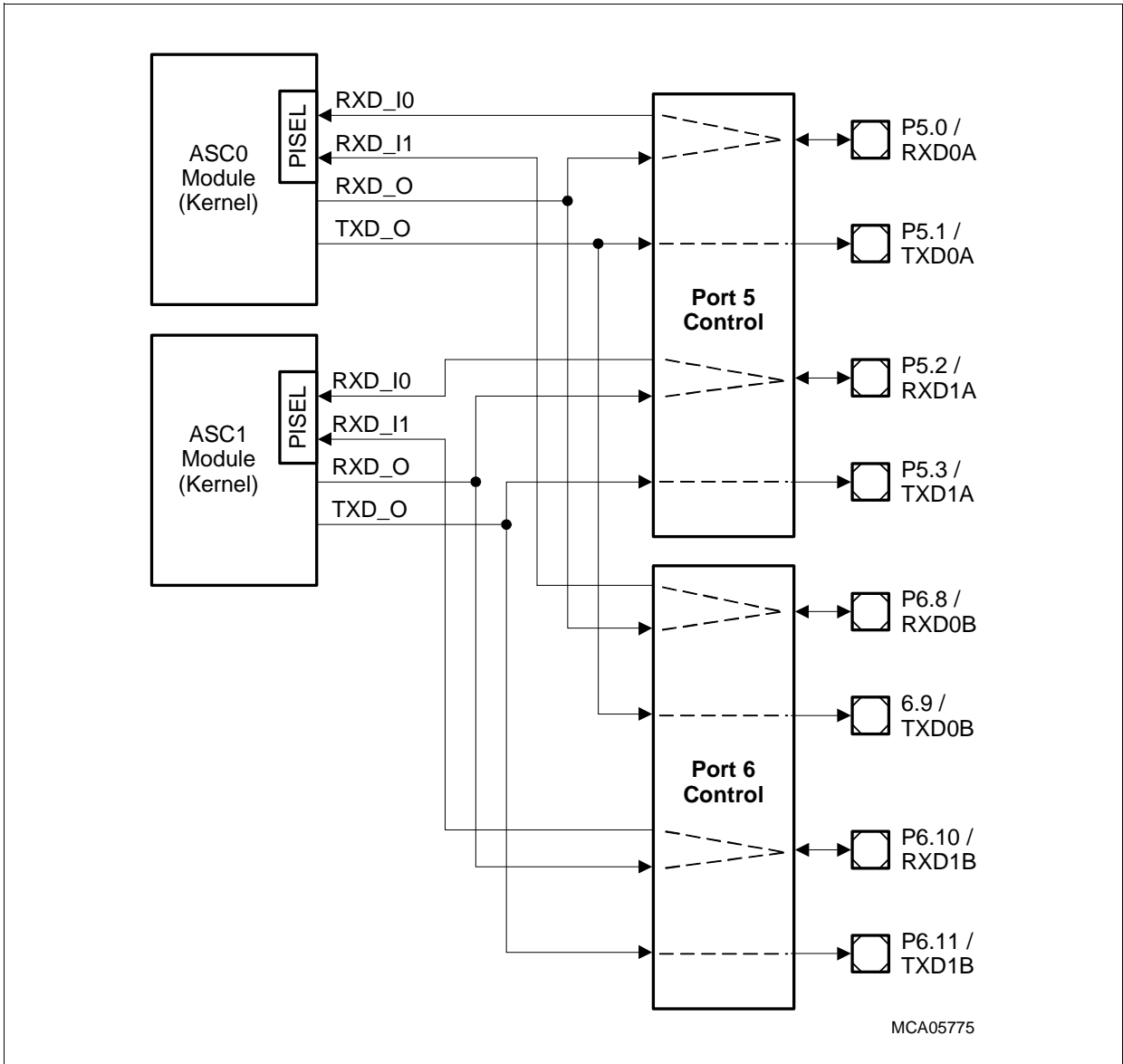


Figure 19-14 RXD Input Line Selection of the ASC Modules

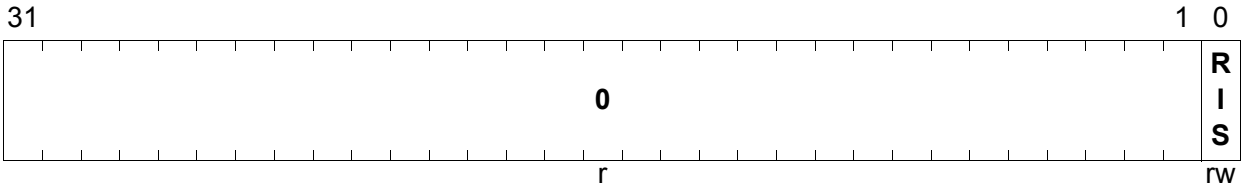
Asynchronous/Synchronous Serial Interface (ASC)

ASC0_PISEL

ASC0 Peripheral Input Select Register

(04_H)

Reset Value: 0000 0000_H



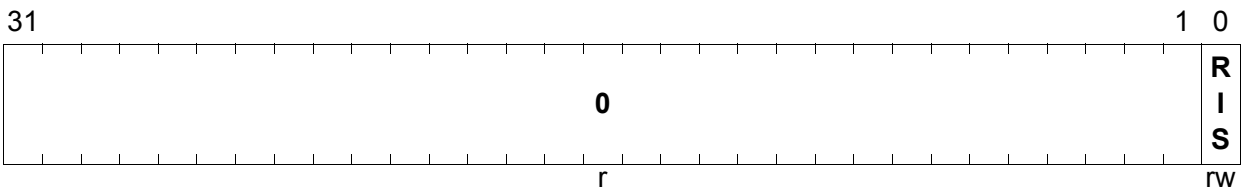
Field	Bits	Type	Description
RIS	0	rw	Receive Input Select 0 _B ASC0 receiver input RXD0A (P5.0) selected 1 _B ASC0 receiver input RXD0B (P6.8) selected
0	[31:1]	0	Reserved Read as 0; should be written with 0.

ASC1_PISEL

ASC1 Peripheral Input Select Register

(04_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
RIS	0	rw	Receive Input Select 0 _B ASC1 receiver input RXD1A (P5.2) selected 1 _B ASC1 receiver input RXD1B (P6.10) selected
0	[31:1]	0	Reserved Read as 0; should be written with 0.

Asynchronous/Synchronous Serial Interface (ASC)

19.3.2.3 Port Control Registers

As shown in [Figure 19-14](#), the I/O lines of the ASC modules are connected to Class A2 port pins of Port 5 and Port 6. Additionally to the PISEL register programming, the required ASC port lines must be programmed by software for the desired ASC input/output functionality. Two selections must be executed:

- Input/output function selection
(controlled by the port input/output control registers IOCR)
- Pad driver characteristics selection for the outputs
(controlled by the port pad driver mode register PDR)

Input/Output Function Selection

The port input/output control registers contain the 4-bit wide bit fields that select the digital output and input driver characteristics such as pull-up/down devices, port direction (input/output), open-drain, and alternate output selections individually for each pin. The I/O lines for the ASC modules are controlled by the port input/output control registers P5_IOCR0 and P6_IOCR8.

[Table 19-8](#) shows how bits and bit fields must be programmed for the required I/O functionality of the ASC I/O lines. This table also shows the values of the peripheral input select registers.

Table 19-8 ASC0/ASC1 I/O Control Selection and Setup

Module	Port Lines	PISEL Register	Input/Output Control Register Bits ¹⁾	I/O
ASC0	P5.0/RXD0A	ASC0_PISEL.RIS = 0	P5_IOCR0.PC0 = 0XXX _B	Input
		–	P5_IOCR0.PC0 = 1X01 _B	Output ²⁾
	P6.8/RXD0B	ASC0_PISEL.RIS = 1	P6_IOCR8.PC8 = 0XXX _B	Input
		–	P6_IOCR8.PC8 = 1X10 _B	Output ²⁾
	P5.1/TXD0A	–	P5_IOCR0.PC1 = 1X01 _B	Output
P6.9/TXD0B	–	P6_IOCR8.PC9 = 1X10 _B	Output	
ASC1	P5.2/RXD1A	ASC1_PISEL.RIS = 0	P5_IOCR0.PC2 = 0XXX _B	Input
		–	P5_IOCR0.PC2 = 1X01 _B	Output ²⁾
	P6.10/RXD1B	ASC1_PISEL.RIS = 1	P6_IOCR8.PC10 = 0XXX _B	Input
		–	P6_IOCR8.PC10 = 1X10 _B	Output ²⁾
	P5.3/TXD1A	–	P5_IOCR0.PC3 = 1X01 _B	Output
P6.11/TXD1B	–	P6_IOCR8.PC11 = 1X10 _B	Output	

1) Valid PCx bit field combinations see [Table 19-9](#).

2) Applicable in Synchronous Mode only.

Asynchronous/Synchronous Serial Interface (ASC)

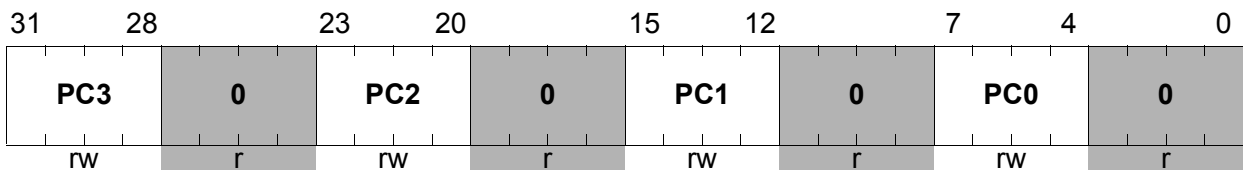
*Note: In synchronous operating mode of the ASC, the type of the selected RXD port pin (input or output) is **not** automatically controlled by the ASC but must be defined by a user program by writing the appropriate bit field in the IOCR registers.*

Input/Output Control Registers

P5_IOCR0

Port 5 Input/Output Control Register 0(10_H)

Reset Value: 2020 2020_H



Field	Bits	Type	Description
PC0, PC1, PC2, PC3	[7:4], [15:12], [23:20], [31:28]	rw	Port Output Control for Port 5 Pins 0-3¹⁾ These bit fields determine the output port functionality: Port input/output control for P5.0/RXD0A Port input/output control for P5.1/TXD0A Port input/output control for P5.2/RXD1A Port input/output control for P5.3/TXD1A

1) Coding of bit field see [Table 19-9](#). Shaded bits and bit fields are “don’t care” for ASC I/O port control.

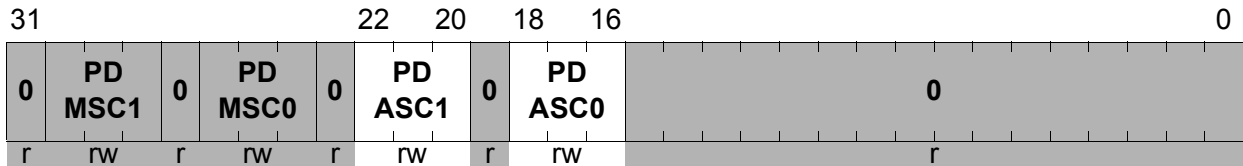
Asynchronous/Synchronous Serial Interface (ASC)

Pad Driver Mode Registers

The Port 5 and Port 6 pad driver mode register contains bit fields that determine the output driver strength and the slew rate of ASC output lines.

P5_PDR

Port 5 Pad Driver Mode Register (40_H) **Reset Value: 0000 0000_H**

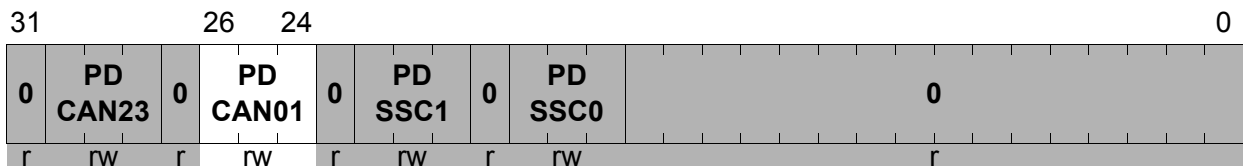


Field	Bits	Type	Description
PDASC0	[18:16]	rw	Pad Driver Mode for P5.0/RXD0A and P5.1/TXD0A ¹⁾
PDASC1	[22:20]	rw	Pad Driver Mode for P5.2/RXD1A and P5.3/TXD1A ¹⁾

1) Coding of bit field see [Table 19-10](#). Shaded bits and bit fields are “don’t care” for ASC I/O port control.

P6_PDR

Port 6 Pad Driver Mode Register (40_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
PDCAN01	[26:24]	rw	Pad Driver Mode for P6.8/RXD0B, P6.9/TXD0B, P6.10/RXD1B, and P6.11/TXD1B ¹⁾

1) Coding of bit field see [Table 19-10](#). Shaded bits and bit fields are “don’t care” for ASC I/O port control.

Asynchronous/Synchronous Serial Interface (ASC)

PDx Selection Table

Table 19-10 Pad Driver Mode Mode Selection (Class A2 Pads)

PDx Bit Field	Driver Strength	Signal Transitions
000 _B	Strong driver	Sharp edge ¹⁾
001 _B		Medium edge ¹⁾
010 _B		Soft edge ¹⁾
011 _B	Weak driver	–
100 _B	Medium driver	–
101 _B		–
110 _B		–
111 _B	Weak driver	–

1) In strong driver mode, the output driver characteristics of class A2 pads can be additionally controlled by the temperature compensation logic.

19.3.2.4 Interrupt Control Registers

The eight interrupts of the ASC0 and ASC1 modules are controlled by the following service request control registers:

- ASC0_TSRC, ASC1_TSRC: control the transmit interrupts
- ASC0_RSRC, ASC1_RSRC: control the receive interrupts
- ASC0_ESRC, ASC1_ESRC: control the error interrupts
- ASC0_TBSRC, ASC1_TBSRC: control the transmit buffer empty interrupts

Asynchronous/Synchronous Serial Interface (ASC)

TSRC

Transmit Interrupt Service Request Control Register
(F0_H)

Reset Value: 0000 0000_H

RSRC

Receive Interrupt Service Request Control Register
(F4_H)

Reset Value: 0000 0000_H

ESRC

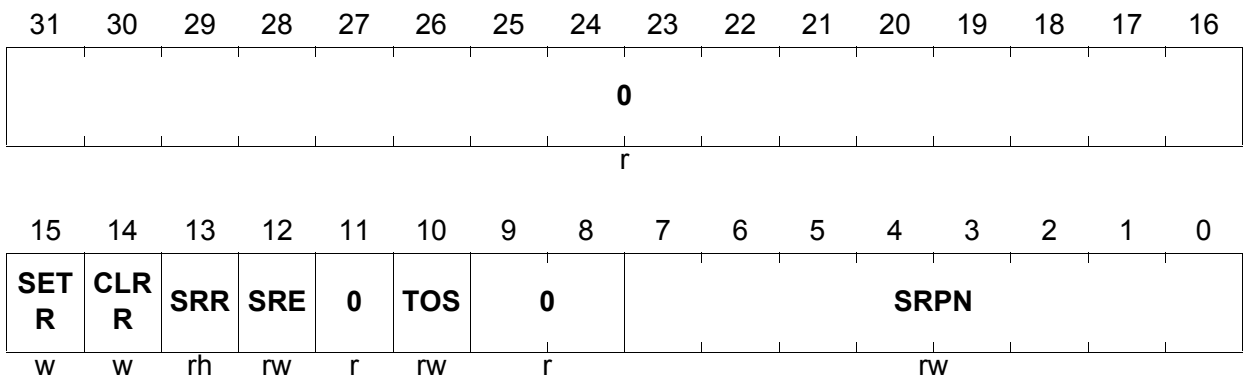
Error Interrupt Service Request Control Register
(F8_H)

Reset Value: 0000 0000_H

TBSRC

Transmit Buffer Interrupt Service Request Control Register
(FC_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SRPN	[7:0]	rw	Service Request Priority Number
TOS	10	rw	Type of Service Control
SRE	12	rw	Service Request Enable
SRR	13	rh	Service Request Flag
CLRR	14	w	Request Clear Bit
SETR	15	w	Request Set Bit
0	[9:8], 11, [31:16]	r	Reserved Read as 0; should be written with 0.

Note: Additional details on service request nodes and the service request control registers are described on [Page 14-3](#) of the TC1796 User's Manual System Units part (Volume 1).

Asynchronous/Synchronous Serial Interface (ASC)

19.3.3 DMA Requests

The DMA request output lines of the ASC0/ASC1 modules become active whenever the related ASC interrupt line is activated. The DMA request lines are connected to the DMA controller as shown in [Table 19-11](#).

Table 19-11 DMA Request Lines of ASC0/ASC1

Module	Related ASC Interrupt	DMA Request Line	Description
ASC0	RIR	ASC0_RDR	ASC0 Receive DMA Request
	TIR	ASC0_TDR	ASC0 Transmit DMA Request
ASC1	RIR	ASC1_RDR	ASC1 Receive DMA Request
	TIR	ASC1_TDR	ASC1 Transmit DMA Request

Note: Further details on DMA handling and processing are described in the chapter “DMA Controller” of the TC1796 System Units User’s Manual.

20 Synchronous Serial Interface (SSC)

This chapter describes the two Synchronous Serial Interfaces, SSC0 and SSC1, of the TC1796. It contains the following sections:

- Functional description of the SSC kernel, valid for SSC0 and SSC1 (see [Page 20-1](#)).
- SSC kernel register description, describes all SSC Kernel specific registers (see [Page 20-27](#)).
- TC1796 implementation-specific details and registers of the SSC0/SSC1 modules (port connections and control, interrupt control, address decoding, clock control, see [Page 20-46](#)).

Note: The SSC kernel register names described in [Section 20.2](#) will be referenced in the TC1796 User's Manual by the module name prefix "SSC0_" for the SSC0 interface and by "SSC1_" for the SSC1 interface.

20.1 SSC Kernel Description

[Figure 20-1](#) shows a global view of the SSC interface.

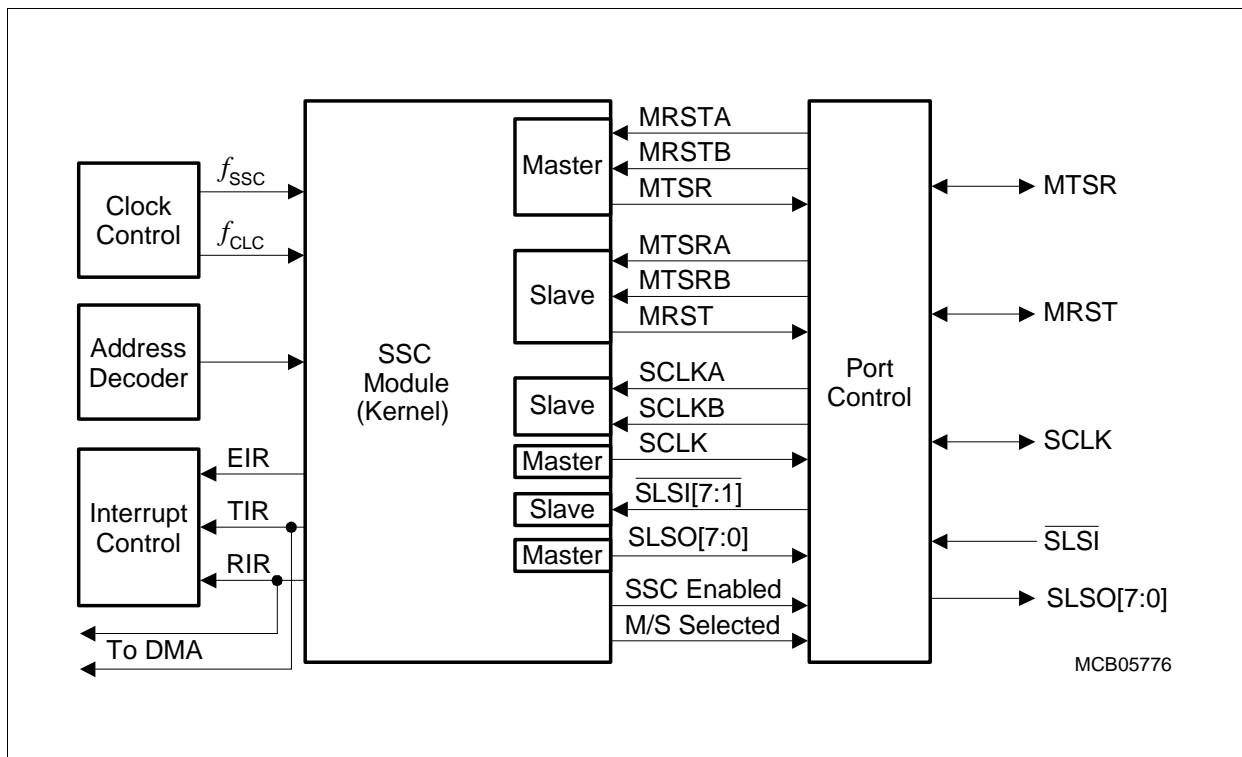


Figure 20-1 General Block Diagram of the SSC Interface

Synchronous Serial Interface (SSC)

20.1.1 Overview

The SSC supports full-duplex and half-duplex serial synchronous communication up to 37.5 Mbit/s (@ 75 MHz module clock) with Receive and Transmit FIFO support. The serial clock signal can be generated by the SSC itself (Master Mode) or can be received from an external master (Slave Mode). Data width, shift direction, clock polarity and phase are programmable. This allows communication with SPI-compatible devices. Transmission and reception of data is double-buffered. A shift clock generator provides the SSC with a separate serial clock signal. Seven slave select inputs are available for Slave Mode operation. Eight programmable slave select outputs (chip selects) are supported in Master Mode.

Note: The SSC0 contains an 8-stage Receive and Transmit FIFO. The SSC1 does not provide any FIFO functionality. Therefore, all FIFO related references in [Section 20.1](#) and [Section 20.2](#) are not valid for the SSC1 module!

Features

- Master and Slave Mode operation
 - Full-duplex or half-duplex operation
 - Automatic pad control possible
- Flexible data format
 - Programmable number of data bits: 2 to 16 bits
 - Programmable shift direction: LSB or MSB shift first
 - Programmable clock polarity: Idle low or idle high state for the shift clock
 - Programmable clock/data phase: Data shift with leading or trailing edge of the shift clock
- Baud rate generation from 37.5 Mbit/s to 572.2 bit/s (@ 75 MHz module clock)
- Interrupt generation
 - On a transmitter empty condition
 - On a receiver full condition
 - On an error condition (receive, phase, baud rate, transmit error)
- Flexible SSC pin configuration
- Seven slave select inputs $\overline{\text{SLSI}}[7:1]$ in Slave Mode
- Eight programmable slave select outputs $\text{SLSO}[7:0]$ in Master Mode
 - Automatic SLSO generation with programmable timing
 - Programmable active level and enable control
- SSC0 only: 8-stage Receive FIFO (RXFIFO) and 8-stage Transmit FIFO (TXFIFO)
 - Independent control of RXFIFO and TXFIFO
 - 2- to 16-bit FIFO data width
 - Programmable receive/transmit interrupt trigger level
 - Receive and Transmit FIFO filling level indication
 - Overrun error generation
 - Underflow error generation

20.1.2 General Operation

The SSC supports full-duplex and half-duplex synchronous communication up to 37.5 Mbit/s (@ 75 MHz module clock). The serial clock signal can be generated by the SSC itself (Master Mode) or be received from an external master (Slave Mode). Data width, shift direction, clock polarity and phase are programmable. This allows communication with SPI-compatible devices. Transmission and reception of data are double-buffered. A shift clock generator provides the SSC with a separate serial clock signal.

Configuration of the high-speed synchronous serial interface is very flexible, so it can work with other synchronous serial interfaces, can serve for master/slave or multi-master interconnections, or can operate compatibly with the popular SPI interface. It can be used to communicate with shift registers (I/O expansion), peripherals (e.g. EEPROMs etc.), or other controllers (networking). The SSC supports half-duplex and full-duplex communication. Data is transmitted or received on pins MTSR (Master Transmit / Slave Receive) and MRST (Master Receive/Slave Transmit). The clock signal is output or input via pin SCLK (Serial Clock). These three pins are typically used for alternate output functions of port pins. If they are implemented as dedicated bi-directional pins, they can be directly controlled by the SSC. In Slave Mode, the SSC can be selected from a master via dedicated slave select input lines (SLSI). In Master Mode, automatic generation of slave select output lines (SLSO) is supported.

Synchronous Serial Interface (SSC)

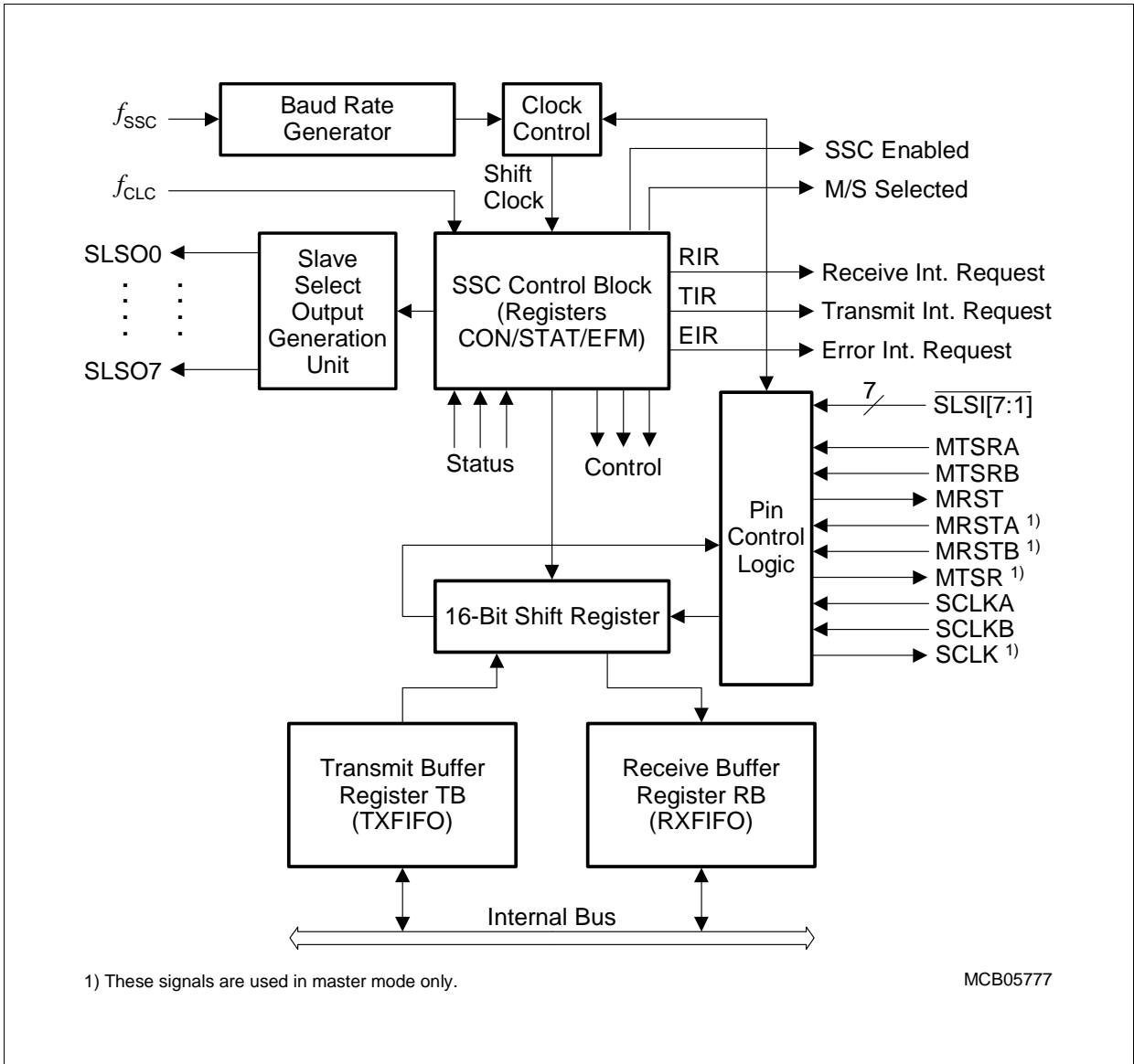


Figure 20-2 Synchronous Serial Channel SSC Block Diagram

20.1.2.1 Operating Mode Selection

The operating mode of the serial channel SSC is controlled by its Control Register, CON. Status information is contained in its Status Register, STAT.

The shift register of the SSC is connected to both the transmit pin and the receive pin via the pin control logic (see block diagram in [Figure 20-2](#)). Transmission and reception of serial data are synchronized and take place at the same time, that is, the same number of transmitted bits is also received. Transmit data is written into the Transmit Buffer TB. It is moved to the shift register as soon as this is empty. An SSC master (CON.MS = 1) immediately begins transmitting, while an SSC slave (CON.MS = 0) will wait for an active shift clock. When the transfer starts, the busy flag STAT.BSY is set and the transmit interrupt request line (TIR) will be activated to indicate that the Transmit Buffer Register (TB) may be reloaded. When the number of bits (2 to 16, as programmed) have been transferred, the contents of the shift register are moved to the Receive Buffer Register (RB), and the receive interrupt request line (RIR) will be activated. If no further transfer is to take place (TB is empty), STAT.BSY will be cleared at the same time. Software should not modify STAT.BSY, as this flag is hardware-controlled.

Note: Only one SSC can be master at a given time.

The following features of the serial data bit transfer can be programmed:

- The data width can be selected from 2 bits to 16 bits
- A transfer may start with the LSB or the MSB
- The shift clock may be idle low or idle high
- The data bits may be shifted with the leading or trailing edge of the clock signal
- The baud rate (shift clock) can be set from 572.2 bit/s up to 37.5 Mbit/s (@ 75 MHz module clock)
- The shift clock can be generated (master) or received (slave)

These features allow the SSC to be adapted to a wide range of applications that require serial data transfer.

The Data Width Selection supports the transfer of frames of any data length from 2-bit “characters” up to 16-bit “characters”. Starting with the LSB (CON.HB = 0) allows communication with such devices as an SSC device in Synchronous Mode, or 8051-like serial interfaces. Starting with the MSB (CON.HB = 1) allows operation compatible with the SPI interface.

Regardless of the data width selected and whether the MSB or the LSB is transmitted first, the transfer data is always right-aligned in registers TB and RB, with the LSB of the transfer data in bit 0 of these registers. The data bits are rearranged for transfer by the internal shift register logic. The unselected bits of TB are ignored, and the unselected bits of RB will not be valid and should be ignored by the receiver service routine.

The Clock Control allows the adaptation of transmit and receive behavior of the SSC to a variety of serial interfaces. A specific clock edge (rising or falling) is used to shift out transmit data, while the other clock edge is used to latch in receive data. Bit CON.PH

Synchronous Serial Interface (SSC)

selects the leading edge or the trailing edge for each function. Bit CON.PO selects the level of the clock line in the idle state. For an idle-high clock, the leading edge is a falling one, a 1-to-0 transition (see [Figure 20-3](#)).

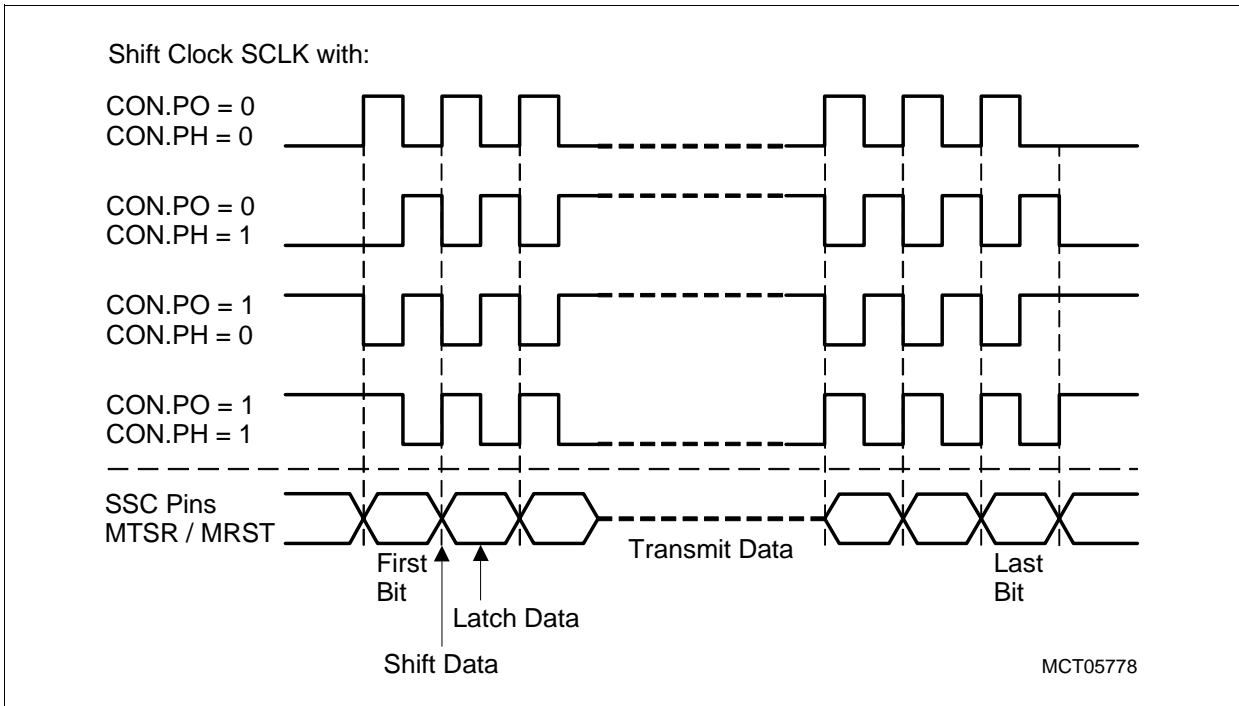


Figure 20-3 Serial Clock SCLK Phase and Polarity Options

20.1.2.2 Full-Duplex Operation

The description in this section assumes that the SSC is used with software controlled bi-directional GPIO port lines that have open-drain capability (see also [Section 20.1.2.5](#)).

The various devices are connected through three lines. The definition of these lines is always determined by the master. The line connected to the master's data output pin MTSR is the transmit line, the receive line is connected to its data input line MRST, and the clock line is connected to pin SCLK. Only the device selected for master operation generates and outputs the serial clock on pin SCLK. All slaves receive this clock, so their pin SCLK must be switched to input mode. The output of the master's shift register is connected to the external transmit line, which in turn is connected to the slaves' shift register input. The output of the slaves' shift register is connected to the external receive line in order to enable the master to receive the data shifted out of the slave. The external connections are hard-wired, with the function and direction of these pins determined by the master or slave operation of the individual device.

Note: The shift direction shown in [Figure 20-4](#) applies to both MSB-first and LSB-first operation.

Synchronous Serial Interface (SSC)

When initializing the devices in this configuration, one device must be selected for master operation while all other devices must be programmed for slave operation. Initialization includes the operating mode of the device's SSC and also the function of the respective port lines.

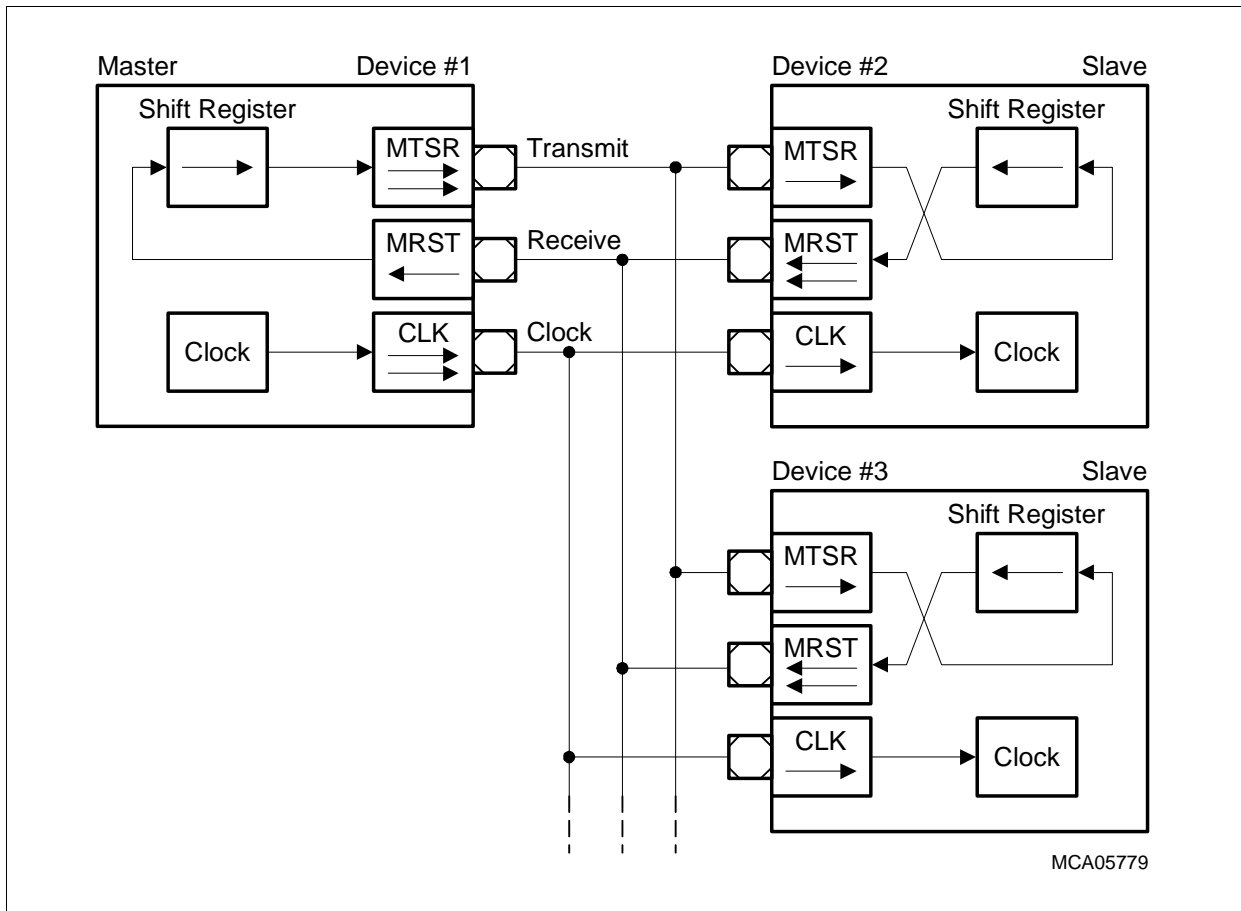


Figure 20-4 SSC Full-Duplex Configuration

The data output pins MRST of all slave devices are connected onto one receive line in this configuration. During a transfer, each slave shifts out data from its shift register. There are two ways to avoid collisions on the receive line due to different slave data:

- **Only one slave drives the line** and enables the driver of its MRST pin. All the other slaves must program their MRST pins to input. Therefore, only one slave can put its data onto the master's receive line. Only reception of data from the master is possible. The master selects the slave device from which it expects data either by separate select lines, or by sending a special command to this slave. The selected slave then switches its MRST line to output until it gets a de-selection signal or command.
- **The slaves use open drain output on MRST.** This forms a wired-AND connection. The receive line needs an external pull-up in this case. Corruption of the data on the receive line sent by the selected slave is avoided when all slaves not selected for

Synchronous Serial Interface (SSC)

transmission to the master send only 1s. Since this high level is not actively driven onto the line, but is only held through the pull-up device, the selected slave can pull this line actively to a low level when transmitting a zero bit. The master selects the slave device from which it expects data either by separate select lines, or by sending a special command to this slave.

After performing all necessary initializations of the SSC, the serial interfaces can be enabled. For a master device, the alternate clock line will now go to its programmed polarity. The alternate data line will go to either 0 or 1, until the first transfer starts. After a transfer, the alternate data line will always remain at the logic level of the last transmitted data bit.

When the serial interfaces are enabled, the master device can initiate the first data transfer by writing the transmit data into register TB. This value is copied into the shift register (assumed to be empty at this time), and the selected first bit of the transmit data will be placed onto the MTSR line on the next clock from the shift clock generator (transmission only starts, if CON.EN = 1). Depending on the selected clock phase, a clock pulse is generated on the SCLK line. With the opposite clock edge, the master simultaneously latches and shifts in the data detected at its input line MRST. This “exchanges” the transmit data with the receive data. Because the clock line is connected to all slaves, their shift registers will be shifted synchronously with the master’s shift register, shifting out the data contained in the registers, and shifting in the data detected at the input line. After the pre-programmed number of clock pulses (via the data width selection), the data transmitted by the master is contained in all slaves’ shift registers, while the master’s shift register holds the data of the selected slave. In the master and all slaves, the content of the shift register is copied into the Receive Buffer (RB) and the receive interrupt request line (RIR) is activated.

A slave device will immediately output the selected first bit (MSB or LSB of the transfer data) at pin MRST when the contents of the transmit buffer are copied into the slave’s shift register. Bit STAT.BSY is not set until the first clock edge at SCLK appears. The slave device will not wait for the next clock from the shift clock generator – as the master does – because the first clock edge generated by the master may be already used to clock in the first data bit, depending on the selected clock phase. So the slave’s first data bit must already be valid at this time.

Note: On the SSC, a transmission and a reception always take place at the same time, regardless whether valid data has been transmitted or received.

20.1.2.3 Half-Duplex Operation

The description in this section assumes that the SSC is used with software controlled bi-directional GPIO port lines that provide open-drain capability (see also [Section 20.1.2.5](#)).

In a half-duplex configuration, only one data line is necessary for both receiving **and** transmitting data. The data exchange line is connected to both pins MTSR and MRST of each device, and the clock line is connected to the SCLK pin.

The master device controls the data transfer by generating the shift clock, while the slave devices receive it. Due to the fact that all transmit and receive pins are connected to the one data exchange line, serial data may be moved between arbitrary stations.

As in full-duplex mode, there are two ways to avoid collisions on the data exchange line:

- Only the transmitting device may enable its transmit pin driver
- The non-transmitting devices use open-drain output and send only 1's

Because the data inputs and outputs are connected together, a transmitting device will clock in its own data at the input pin (MRST for a master device, MTSR for a slave). In this way, any corruption is detected on the common data exchange line when the received data is not equal to the transmitted data.

Synchronous Serial Interface (SSC)

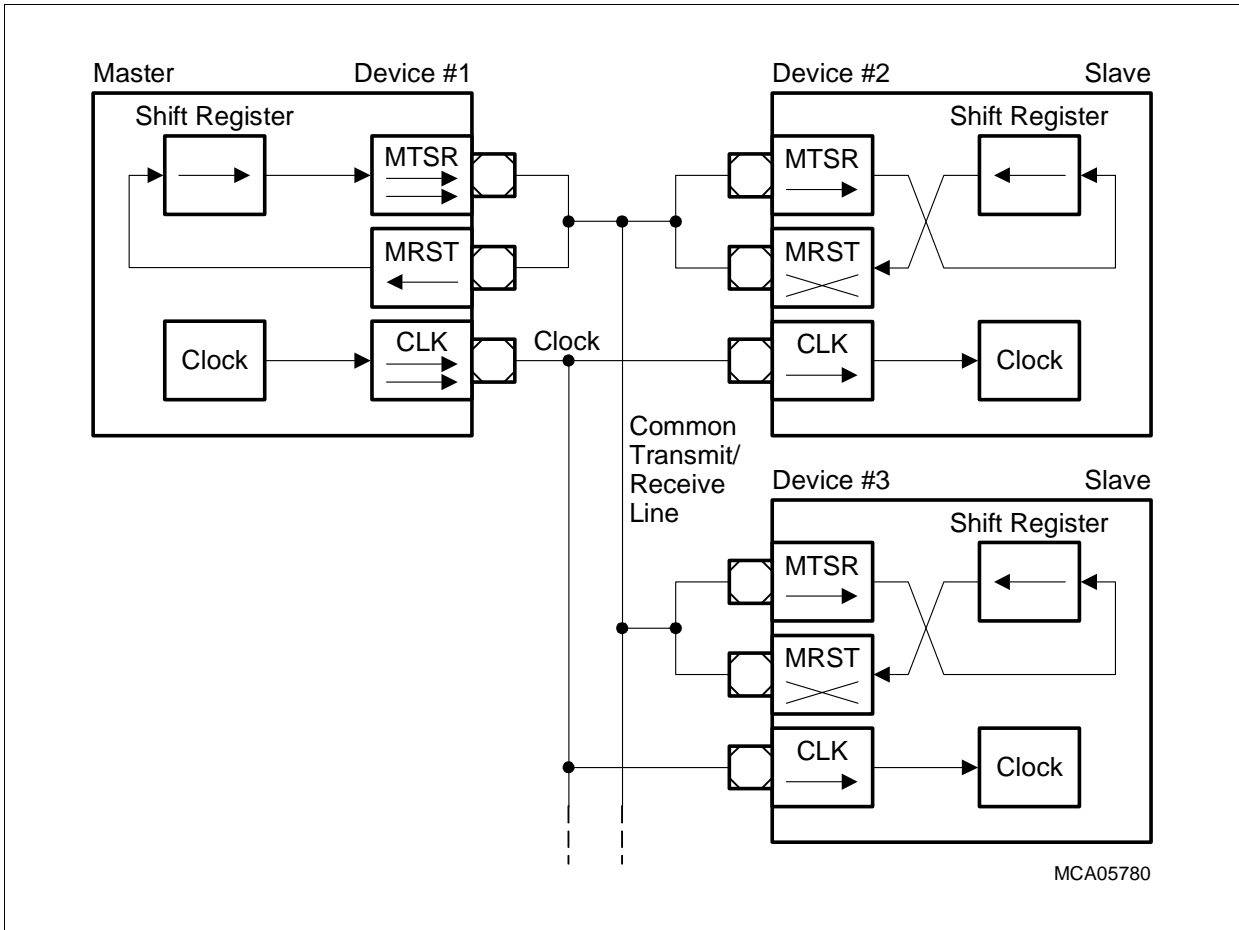


Figure 20-5 SSC Half-Duplex Configuration

20.1.2.4 Continuous Transfers

When the transmit interrupt request flag is set, it indicates that the Transmit Buffer (TB) is empty and is ready to be loaded with the next transmit data. If the TB has been reloaded by the time the current transmission is finished, the data is immediately transferred to the shift register and the next transmission can start without any additional delay (according to the selected SLSO timings). On the data line, there is no gap between the two successive frames if no delays are selected. For example, two byte transfers would look the same as one word transfer. This feature can be used to interface with devices that can operate with (or require more than) 16 data bits per transfer. It is just a matter for software how long a total data frame length can be. This option can also be used, e.g., to interface to byte-wide and word-wide devices on the same serial bus.

Note: This option can only happen in multiples of the selected basic data width, because it would require disabling/enabling of the SSC to reprogram the basic data width on-the-fly.

Synchronous Serial Interface (SSC)

1. If in Master Mode the Transmit Buffer register TB is loaded with new data for the following transmission just at the end of the current transmission and a leading delay > 0 is selected (SSOTC.LEAD not equal 00_B), a slightly enlarged leading delay ($< \text{one SCLK shift clock period}$) is generated for the following transmission.

20.1.2.5 Port Control

The SSC uses three lines to communicate with the external world. Pin SCLK serves as the clock line, while pins MRST (Master Receive/Slave Transmit) and MTSR (Master Transmit/Slave Receive) serve as the serial data input/output lines. As shown in [Figure 20-1](#), these three lines (SCLK as input, Master Receive, Slave Receive) have two inputs each at the SSC Module kernel. Three bits in register PISEL determine which of the two kernel inputs (A or B) are connected. This feature allows for each of the three SSC communication lines to be connected to two inputs coming from different port pins.

Operation of the SSC I/O lines depends on the selected operating mode (master or slave). The direction of the port lines depends on the operating mode. The SSC will automatically use the correct kernel output or kernel input line of the ports when switching modes. Port pins assigned as SSC I/O lines can be controlled either by hardware or by software.

When the SSC I/O lines are connected to dedicated pins, hardware I/O control should typically be used. In this case, two output signals reflect the state of the CON.EN and CON.MS bits directly (the M/S select line is inverted to the CON.MS bit definition).

When the SSC I/O lines are connected with bi-directional lines of general purpose I/O ports, software I/O control should be typically used. In this case, port registers must be programmed for alternate output and input selection. When switching between master and slave mode, port registers must be reprogrammed.

Using the open-drain output feature of port lines helps to avoid bus contention problems and reduces the need for hard-wired hand-shaking or slave select lines. In open-drain output mode, it is not always necessary to switch the direction of a port pin. Note that in hardware-controlled I/O mode, the availability of open-drain outputs depends on the type of the dedicated output pins that are used. The SSC module itself does not provide any control capability for open-drain control.

Note: For details of SSC port connections and configuration, see [Section 20.3.1](#).

20.1.2.6 Transmit FIFO Operation

The Transmit FIFO (TXFIFO) provides the following functionality:

- Enable/disable control
- Programmable filling level for transmit interrupt generation
- Filling level indication
- FIFO clear (flush) operation
- FIFO overflow error generation
- 2 to 16 bit TXFIFO data width

The TXFIFO is controlled by the TXFCON control register. When bit TXFCON.TXFEN is set, the TXFIFO is enabled. The interrupt trigger level defined by TXFCON.TXFITL determines the filling level of the TXFIFO at which a transmit interrupt TIR is generated. This interrupt is always generated when the filling level of the TXFIFO is equal to or less than the value stored in TXFCON.TXFITL.

Bit field TXFFL in the FIFO Status Register FSTAT indicates the number of entries that are actually written (valid) in the TXFIFO. Therefore, the software can verify in the interrupt service routine, for instance, how many bytes can be still written into the TXFIFO via register TB without getting an overrun error.

The TXFIFO cannot be accessed directly. All data write operations to the TXFIFO are executed by writing into the TB register. The data width of one TXFIFO stage can be from 2 to 16 bits (as programmed in CON.BM).

Figure 20-6 shows an example of a TXFIFO operation with a typical data width of 8 bits, representing a byte. In this example, seven bytes are transmitted via the transmit output line. The TXFIFO interrupt trigger level TXFCON.TXFITL is set to 0011_B. The first byte written into the empty TXFIFO via TB is directly transferred into the transmit shift register and is not written into the FIFO. After Byte 1, Bytes 2 to 6 are written into the TXFIFO.

After the transfer of Byte 3 from the TXFIFO into the transmit shift register of the SSC, 3 bytes remain in the TXFIFO. Therefore, the value of TXFCON.TXFITL is reached and a transmit buffer interrupt will be generated at the beginning and a transmit interrupt at the end of the Byte 3 serial transmission. During the serial transmission of Byte 4, another message (Byte 7) is written into the TXFIFO (TB write operation). Finally, after the start of the serial transmission of Byte 7, the TXFIFO is again empty.

If the TXFIFO is full and additional bytes are written into TB, the transmit interrupt will be generated with bit CON.TE set if bit CON.TEN was set. In this case, the data that was last written into the TXFIFO is overwritten and the TXFIFO filling level FSTAT.TXFFL is set to maximum.

The TXFIFO can be flushed or cleared by setting bit TXFCON.TXFFLU. After this TXFIFO flush operation, the TXFIFO is empty and the TXFIFO filling level FSTAT.TXFFL is set to 0000_B. A running serial transmission is not aborted by a RXFIFO flush operation.

Synchronous Serial Interface (SSC)

Note: The TXFIFO is flushed automatically at a reset operation of the SSC module or if the TXFIFO becomes disabled (resetting bit TXFCON.TXFEN) after it was previously enabled.

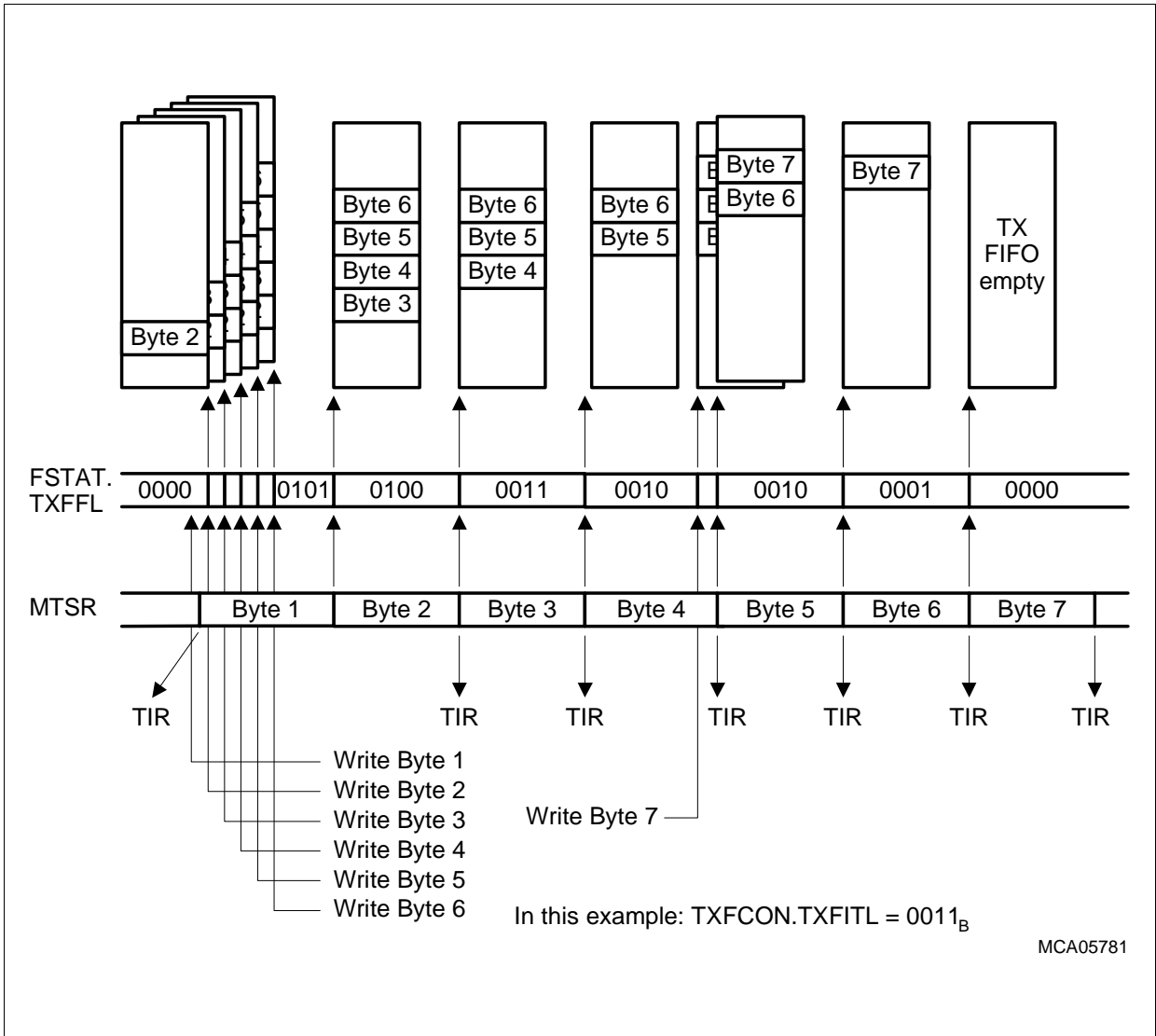


Figure 20-6 Transmit FIFO Operation Example

20.1.2.7 Receive FIFO Operation

The Receive FIFO (RXFIFO) provides the following functionality:

- Enable/disable control
- Programmable filling level for receive interrupt generation
- Filling level indication
- FIFO clear (flush) operation
- FIFO overflow error generation
- 2 to 16 bit RXFIFO data width

The RXFIFO is controlled by the RXFCON control register. When bit RXFCON.RXFEN is set, the RXFIFO is enabled. The interrupt trigger level defined by RXFCON.RXFITL determines the filling level of RXFIFO at which a receive interrupt request RIR is generated. RIR is always generated when the filling level of the RXFIFO is equal to or greater than the value stored in RXFCON.RXFITL.

Bit field RXFFL in the FIFO Status Register FSTAT indicates the number of bytes that have been actually written into the FIFO and can be read out of the FIFO by a user program.

The RXFIFO cannot be accessed directly. All data read operations from the RXFIFO are executed by reading the RB register. The data width of one RXFIFO stage can be from 2 to 16 bits (as programmed in CON.BM).

Figure 20-7 shows an example of a RXFIFO operation with a typical data width of 8 bits, representing a byte. In this example, six bytes are received via the receive input line. The RXFIFO interrupt trigger level RXFCON.RXFITL is set to 0011_B. Therefore, the first receive interrupt request RIR is generated after the reception of Byte 3 (RXFIFO is filled with three messages).

After the reception of Byte 4, three bytes are read out of the RXFIFO. After this read operation, the RXFIFO still contains one message. RIR becomes again active after two more bytes (Bytes 5 and 6) have been received (RXFIFO filled again with 3 bytes). Finally, the FIFO is cleared after three read operations.

If the RXFIFO is full and additional data are received, the receive interrupt request RIR will be generated and bit CON.RE is set, if CON.REN is not cleared. In this case, the data byte last written into the RXFIFO is overwritten. With the overrun condition, the RXFIFO filling level FSTAT.RXFFL is set to maximum. If a RB read operation is executed with the RXFIFO enabled but empty, a RIR will be generated. In this case, the RXFIFO filling level FSTAT.RXFFL is set to 0000_B.

If the RXFIFO is available but disabled (RXFCON.RXFEN = 0) the receive operation is functionally equivalent to the receive operation of the SSC module without FIFO.

The RXFIFO can be flushed or cleared by setting bit RXFCON.RXFFLU in register RXFCON. After this RXFIFO flush operation, the RXFIFO is empty and the RXFIFO filling level FSTAT.RXFFL is set to 0000_B.

Synchronous Serial Interface (SSC)

The RXFIFO is flushed automatically at a reset operation of the SSC module or if the RXFIFO becomes disabled (resetting bit RXFCON.RXFEN) after it was previously enabled. Resetting bit CON.EN without resetting RXFCON.RXFEN does not affect (reset) the RXFIFO state. This means that the receive operation of the SSC is stopped, in this case, without changing the content of the RXFIFO. After setting CON.EN again, the RXFIFO with its content is again available.

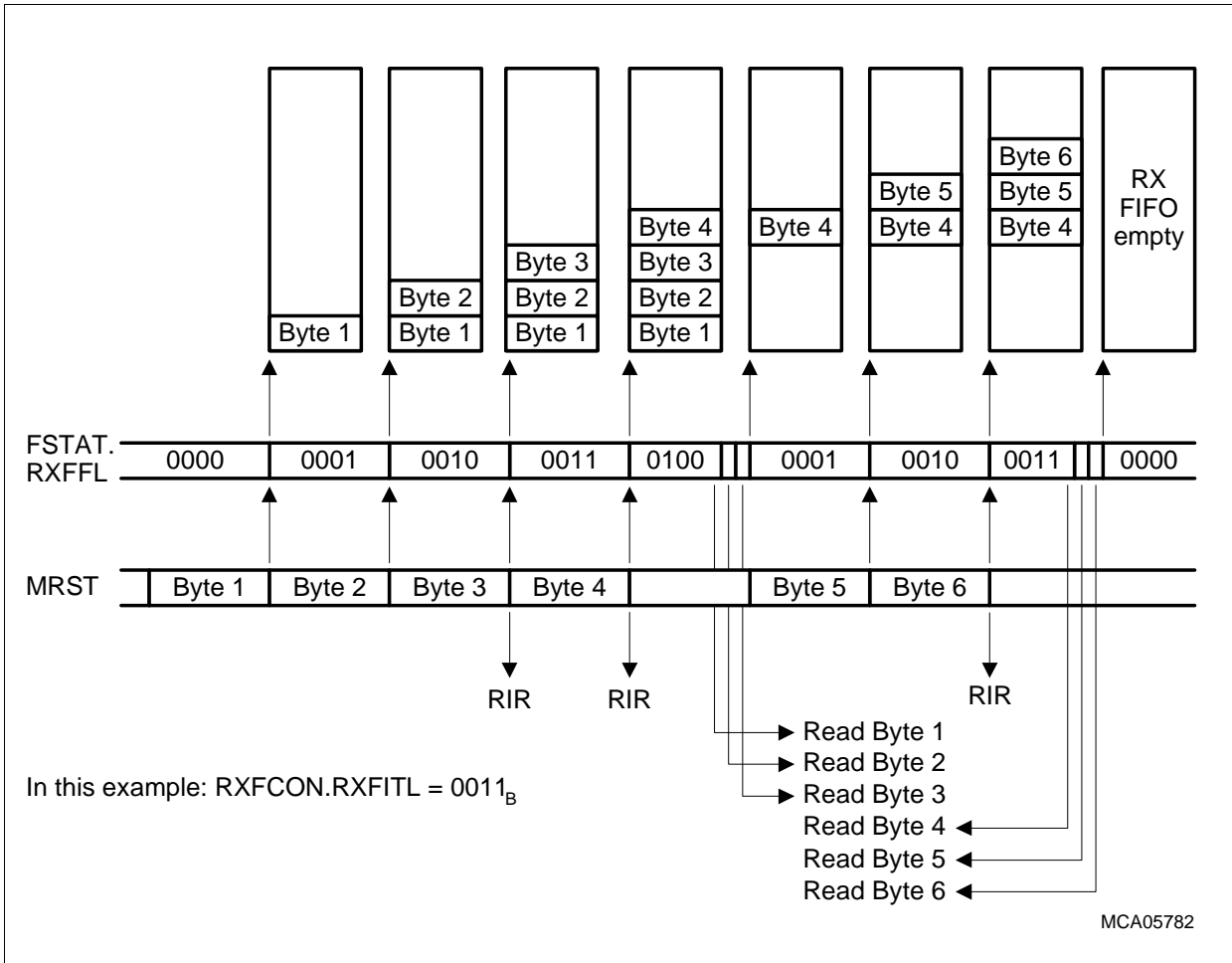


Figure 20-7 Receive FIFO Operation Example

20.1.2.8 FIFO Transparent Mode

In Transparent Mode, a specific interrupt generation mechanism is used for receive and transmit interrupts. In Transparent Mode, receive interrupts are always generated if data bytes are available in the RXFIFO. The relevant conditions for interrupt generation in Transparent Mode are:

- FIFO filling level
- Read/Write operations from/to the RB/TB data registers

Receive Operation

The interrupt generation for the RXFIFO depends on its filling level and the execution of read operations of register RB (see [Figure 20-8](#)). Transparent Mode for the RXFIFO is enabled when bits RXTMEN and RXFEN in register RXFCON are set.

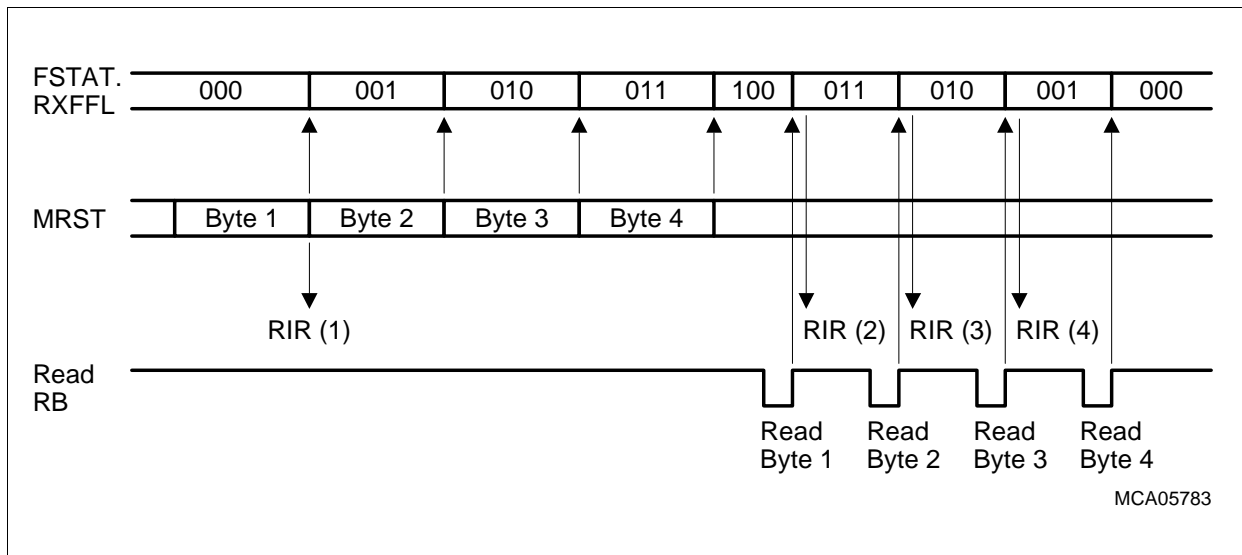


Figure 20-8 Transparent Mode Receive FIFO Operation

If the RXFIFO is empty, a receive interrupt request RIR is always generated when the first message is written into an empty RXFIFO (FSTAT.RXFFL changes from 0000_B to 0001_B). If the RXFIFO is filled with at least one message, the occurrence of further receive interrupts depends on the read operations of register RB. The RIR will always be activated after a RB read operation if the RXFIFO still contains data (FSTAT.RXFFL is not equal to 0000_B). If the RXFIFO is empty after a RB read operation, no further receive interrupt will be generated.

If the RXFIFO is full (FSTAT.RXFFL = 1000_B) and additional messages are received, a receive interrupt request RIR will be generated. In this case, the message last written into the RXFIFO is overwritten. If a RB read operation is executed with the RXFIFO enabled but empty (underflow condition), an RIR will be generated as well, with bit CON.RE set.

Synchronous Serial Interface (SSC)

If the RXFIFO is flushed in Transparent Mode, the software must take care that a previously pending receive interrupt becomes ignored.

Note: The RXFIFO Interrupt Trigger Level bit field `RXFCON.RXFITL` is “don’t care” in Transparent Mode.

Transmit Operation

Interrupt generation for the TXFIFO depends on the TXFIFO filling level and the execution of write operations to the register TB. Transparent Mode for the TXFIFO is enabled when bits `TXFCON.TXTMEN` and `TXFCON.TXFEN` are set.

TIR is also activated after a TXFIFO flush operation or when the TXFIFO becomes enabled (`TXFCON.TXTMEN` and `TXFCON.TXFEN` set) when it was previously disabled. In these cases, the TXFIFO is empty and ready to be filled with data.

If the TXFIFO is full (`FSTAT.TXFFL = 1000B`) and an additional message is written into the TB, a transmit interrupt will be generated after the TB write operation. In this case, the data byte last written into the TXFIFO is overwritten and a TIR will be generated with bit `CON.TE` set.

Note: The TXFIFO Interrupt Trigger Level bit field `TXFCON.TXFITL` is “don’t care” in Transparent Mode.

20.1.2.9 Baud Rate Generation

The serial channel SSC has its own dedicated 16-bit baud rate generator with 16-bit reload capability, allowing baud rate generation independent of timers. In addition to [Figure 20-2](#), [Figure 20-9](#) shows the baud rate generator of the SSC in more detail.

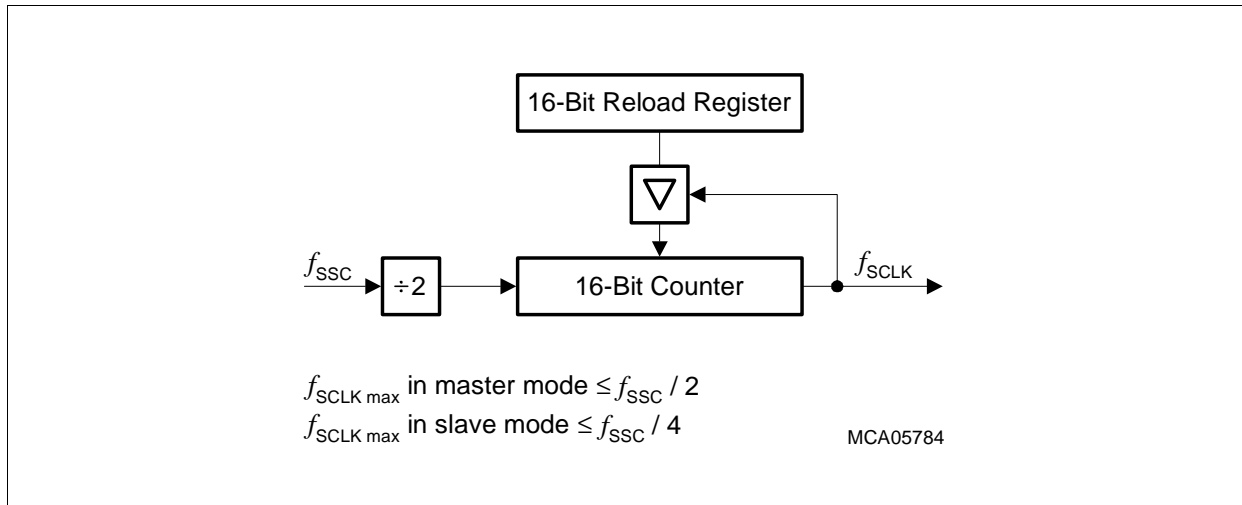


Figure 20-9 SSC Baud Rate Generator

The baud rate generator is clocked with f_{SSC} . The timer counts downwards. Register BR is the dual-function Baud Rate Generator/Reload register. Reading BR while the SSC is enabled returns the contents of the timer. Reading BR while the SSC is disabled returns the programmed reload value. In this mode, the desired reload value can be written to BR.

Note: Never write to BR while the SSC is enabled.

The formulas below calculate either the resulting baud rate for a given reload value, or the required reload value for a given baud rate:

$$\text{Baud rate}_{SSC} = \frac{f_{SSC}}{2 \times (\text{BR_VALUE} + 1)} \quad \text{BR_VALUE} = \frac{f_{SSC}}{2 \times \text{Baud rate}_{SSC}} - 1 \quad (20.1)$$

BR_VALUE represents the content of the reload register, taken as an unsigned 16-bit integer, while Baud rate_{SSC} is equal to f_{SCLK} as shown in [Figure 20-9](#).

The maximum baud rate that can be achieved with $f_{SSC} = 75$ MHz is 37.5 Mbit/s in Master Mode (with $\text{BR_VALUE} = 0000_H$) and 18.75 Mbit/s in Slave Mode (with $\text{BR_VALUE} = 0001_H$).

[Table 20-1](#) lists some possible baud rates together with the required reload values and the resulting bit times, assuming a module clock f_{SSC} of 75 MHz.

Synchronous Serial Interface (SSC)

Table 20-1 Typical Baud Rates of the SSC ($f_{SSC} = 75 \text{ MHz}$)

Reload Value BR_VALUE	Baud Rate ($= f_{SCLK}$)	Deviation
0000 _H	37.5 Mbit/s (only in Master Mode)	0.0%
0001 _H	18.75 Mbit/s	0.0%
0025 _H	1 Mbit/s	-1.3%
0176 _H	100 kbit/s	0.0%
0EA5 _H	10 kbit/s	0.0%
9276 _H	1 kbit/s	0.0%
FFFF _H	572.2 bit/s	0.0%

In the TC1796, the module clock f_{SSC} is generated outside the SSC module kernel. Therefore, for baud rate calculations the dependencies of f_{SSC} from f_{SYS} must be taken into account. [Section 20.3.2.1](#) on [Page 20-48](#) describes these dependencies in detail.

Synchronous Serial Interface (SSC)

20.1.2.10 Slave Select Input Operation

For systems with multiple slaves, the SSC module provides seven $\overline{\text{SLSI}}$ slave select input lines, that permit enabling or disabling of the SCLK, MTSR, and MRST signals in Slave Mode. Slave Mode is selected by CON.MS = 0. The $\overline{\text{SLSI}}$ input logic shown in Figure 20-10 is controlled by register PISEL and CON.

Note: In the following description, only one of the seven $\overline{\text{SLSI}}$ input lines is mentioned. The remaining six $\overline{\text{SLSI}}$ input lines are connected to the other six inputs of the input multiplexer, which is controlled by PISEL.SLSIS.

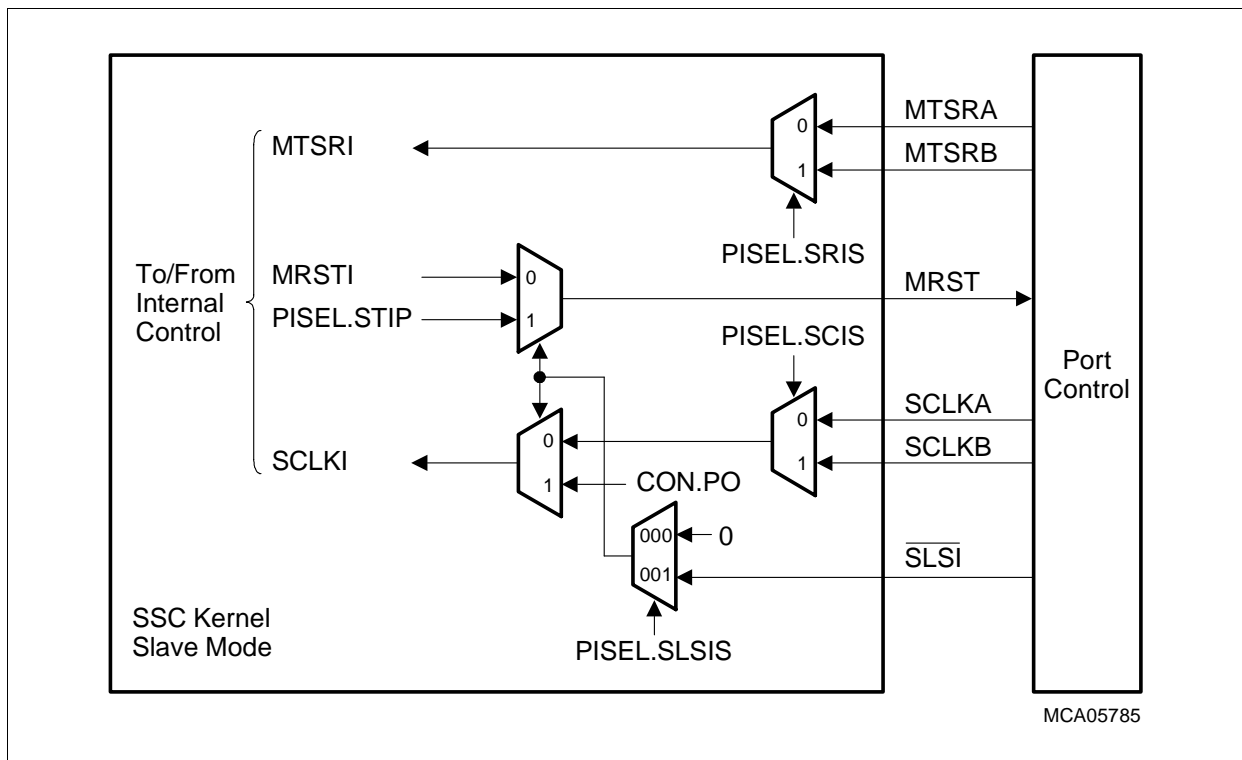


Figure 20-10 Slave Select Input Logic

With PISEL.SLSIS = 000_B and Slave Mode selected, the $\overline{\text{SLSI}}$ input line does not control the SSC I/O lines. The slave receive input signal MTSRA or MTSRB (selected by PISEL.SRIS) and the slave clock input signal SCLKA or SCLKB (selected by PISEL.SCIS) are passed further as MTSRI and SCLKI to the internal SSC control logic. The slave transmit signal MRSTI from the internal SSC control logic MRSTI is passed directly to MRST.

With PISEL.SLSIS = 001_B, input signal $\overline{\text{SLSI}}$ controls the operation of the SSC I/O lines as a slave select signal as follows:

Synchronous Serial Interface (SSC)

- $\overline{\text{SLSI}} = 1$: SSC slave is not selected.
 - The slave receive input signals MTSRA or MTSRB are connected to MTSRI, depending on PISEL.SRIS (Slave Mode receive input select).
 - MRST is driven with the logic level of bit PISEL.STIP (slave transmit idle state).
 - SCLKI is driven with the logic level of CON.PO (clock polarity control).
- $\overline{\text{SLSI}} = 0$: SSC is selected as slave.
 - The slave receive input signals MTSRA or MTSRB are connected to MTSRI, depending on PISEL.SRIS (Slave Mode receive input select).
 - MRST is directly driven with the slave transmit output signal MRSTI.
 - The slave clock input signals SCLKA or SCLKB are connected to SCLKI, depending on PISEL.SCIS (Slave Mode clock input select).

20.1.2.11 Slave Select Output Generation Unit

In Master Mode, the slave select output generation unit of the SSC automatically generates up to eight slave select output lines for serial transmit operations. The slave select output generation unit further makes it possible to adjust the chip select timing parameters. The active/inactive state of a slave select output as well as the enable/disable state can be controlled individually for each slave select output (see [Figure 20-12](#)). The basic slave select output timing is shown in [Figure 20-11](#), assuming a low active level of the SLSON lines.

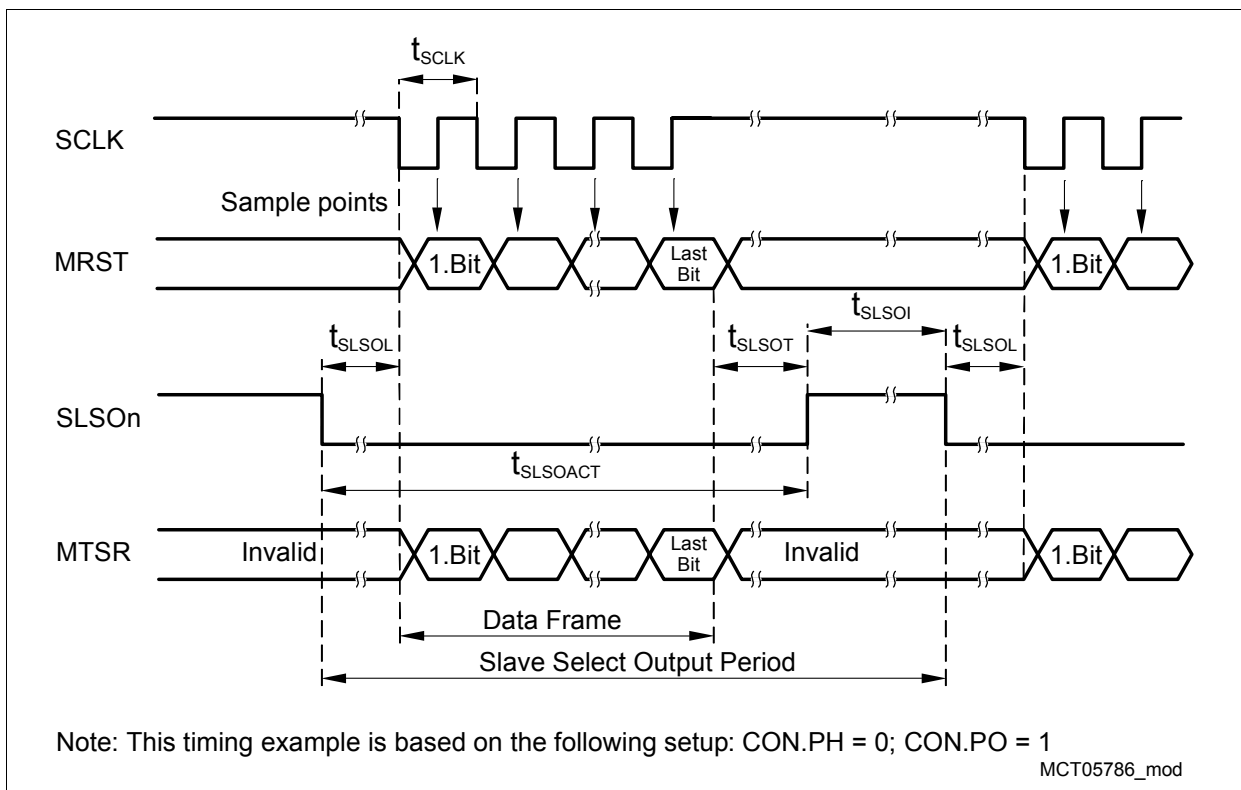


Figure 20-11 SSC Slave Select Output Timing

Synchronous Serial Interface (SSC)

A slave select output period always starts after a serial write operation to register TB. With a TB write operation, all timing parameters stored in register SSOTC as well as the SSOC register are latched and remain valid for the consecutive transmission. Following that, SLSON becomes active (low) for a number of SCLK cycles (leading delay cycles) before the first bit of the serial data stream occurs at MTSR. After the transmission of the data frame, SLSON remains active (low) for a number of SCLK cycles (trailing delay cycles) before it becomes inactive again. This inactive state of SLSON is valid at least for a number of SCLK cycles (inactive delay cycles) before a new chip select period can be started.

Note: When operating in Master Mode with CON.PH = 1 and sampling data from a slave device that becomes enabled by an SLSON output, a leading delay of at least one leading delay clock cycle should be selected. The reason is that with CON.PH = 1, the first SCLK edge already latches the first data bit at MRST.

The three parameters of a chip select period are controlled by bit fields in the Slave Select Output Timing Control Register SSOTC. Each of these bit fields can contain a value from 0 to 3 defining delay cycles of 0 to 3 multiples of the t_{SCLK} shift clock period. The three parameters are:

1. Number of leading delay cycles ($t_{SLSOL} = SSOTC.LEAD \times t_{SCLK}$)
2. Number of trailing delay cycles ($t_{SLSOT} = SSOTC.TRAIL \times t_{SCLK}$)
3. Number of inactive delay cycles ($t_{SLSOI} = SSSOTC.INACT \times t_{SCLK}$)

If SSOTC.INACT = 00_B and register TB has already been loaded with the data for the next data frame, the next chip select period is started with its leading delay phase without SLSON going inactive. If, in this case, TB has not been loaded in time with the data for the next data frame, SLSON becomes inactive again.

Slave Select Output Control

Each slave select output SLSON can be enabled individually. When SSOC.OENn = 1, SLSON is enabled. Furthermore, active and inactive levels of the SLSON outputs are programmable. Bit SSOC.AOLn determines the state of the active level of SLSON.

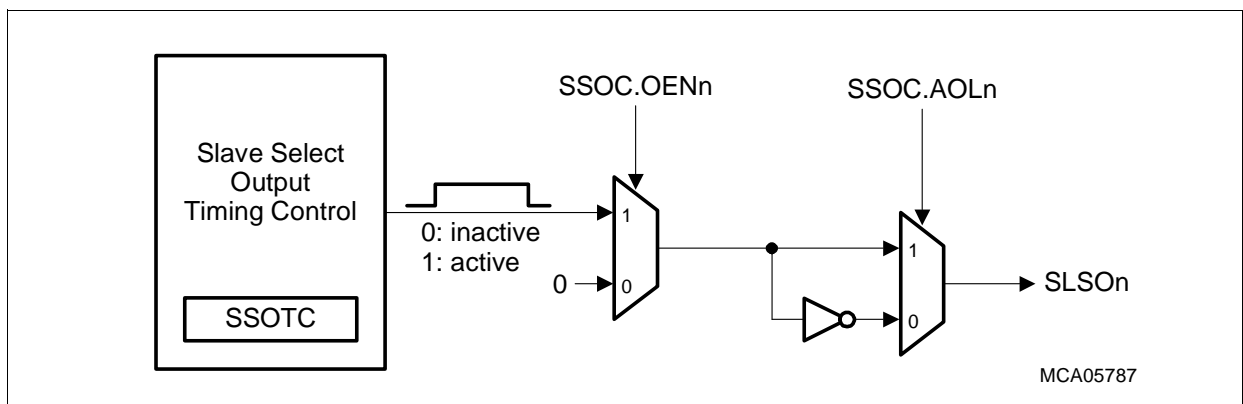


Figure 20-12 Slave Select Output Control Logic

Synchronous Serial Interface (SSC)

Slave Select Output 7 Delayed Mode

In the SLSO7 delayed mode (SSOTC.SLSO7MOD = 1), the timing of the slave select output SLSO7 as programmed by the three parameters in SSOTC (number of trailing, leading, and inactive delay clock cycles) is delayed by one shift clock period for the inactive-to-active edge. The active-to-inactive edge is not delayed. The timing of SLSO7 in the delayed mode is shown in **Figure 20-13**. The bold lines show the timing of SLSO7 in normal operating mode, and the dotted lines show the timing of SLSO7 in delayed mode.

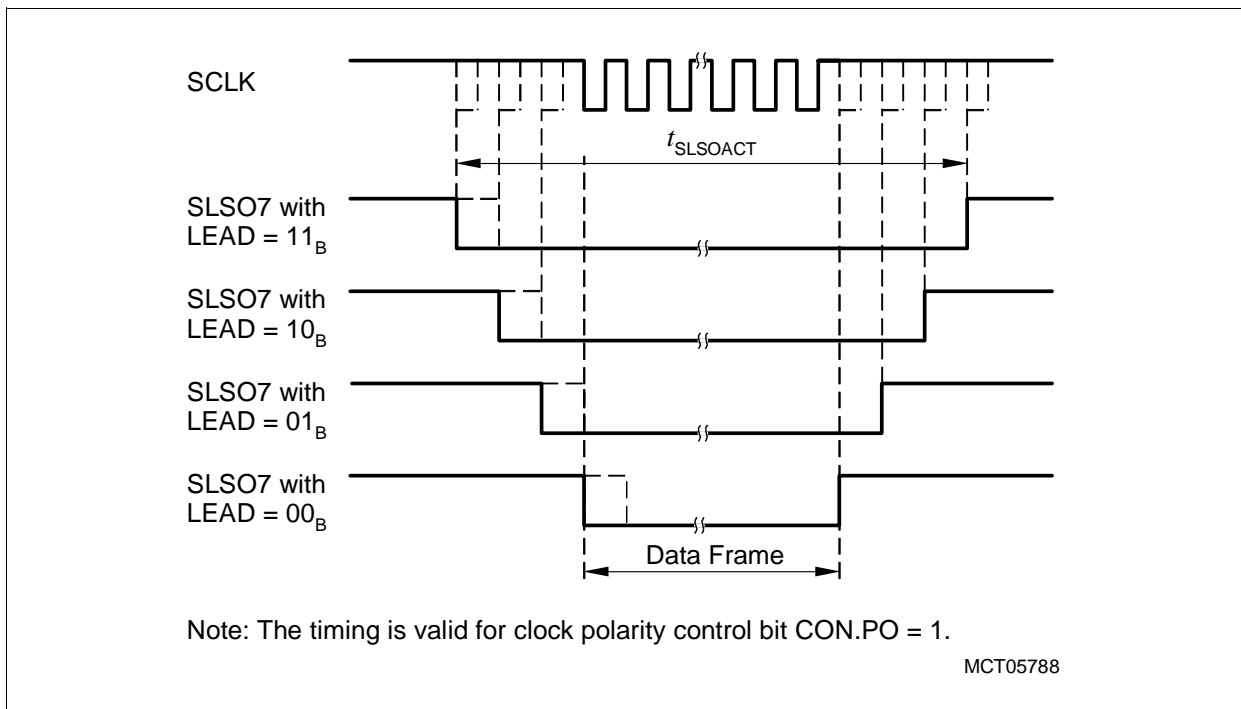


Figure 20-13 SLSO7 Delayed Mode

Slave Select Register Update

At the start of an internal transmit sequence (with the TB register write operation), the parameters in registers SSOC and SSOTC are buffered. This means that they remain stable while a serial transmission is in progress. Therefore, it is always guaranteed that the data of one serial transmission is always transmitted with a constant slave select configuration setup. A configuration change by reprogramming SSOC or SSOTC during a serial transmission will first become valid with the start of the subsequent serial transmission.

Synchronous Serial Interface (SSC)

20.1.2.12 Error Detection Mechanisms

The SSC is able to detect four different error conditions. Receive Error and Phase Error are detected in all modes, while Transmit Error and Baud Rate Error apply to Slave Mode only. When an error is detected, the respective error flag is always set and the error interrupt request will be generated by activating the EIR line if the corresponding error enable bit is set (see Figure 20-14). The error interrupt handler may then check the error flags to determine the cause of the error interrupt. The error flags are not cleared automatically, but must be cleared via register EFM after servicing. This allows servicing of some error conditions via interrupt, while others may be polled by software. The error status flags can be set and cleared by software via the error flag modification register EFM.

Note: The error interrupt handler must clear the associated (enabled) error flag(s) to prevent repeated interrupt requests.

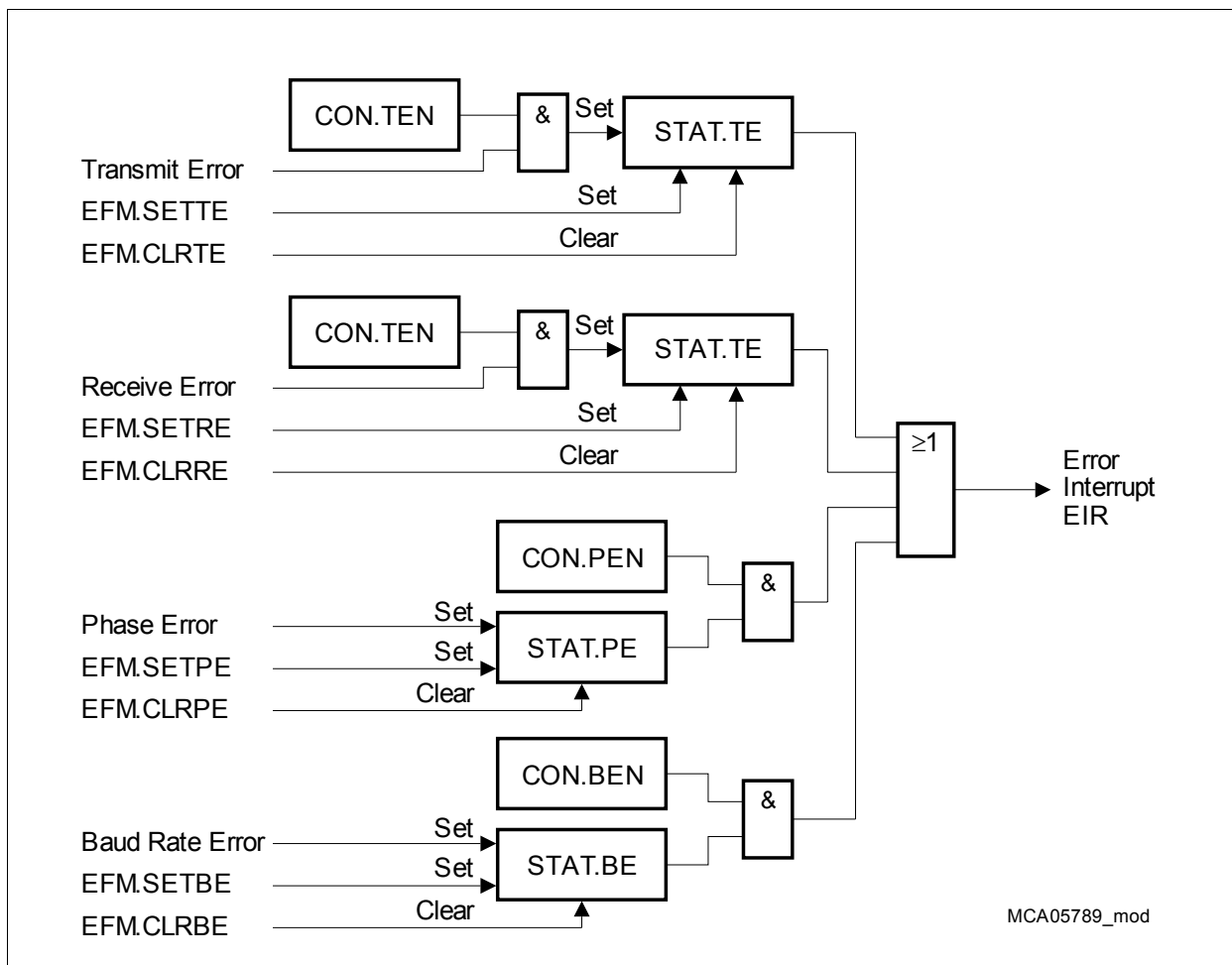


Figure 20-14 SSC Error Interrupt Control

Synchronous Serial Interface (SSC)

A **Receive Error** (Master or Slave Mode) is detected when a new data frame is completely received, but the previous data was not read out of the receive buffer register RB. This condition sets the error flag STAT.RE and, if enabled via CON.REN, sets the error interrupt request line EIR. The old data in the receive buffer RB will be overwritten with the new value and is irretrievably lost.

A **Phase Error** (Master or Slave Mode) is detected when the incoming data at pin MRST (Master Mode) or MTSR (Slave Mode), sampled with the same frequency as the module clock, changes between one cycle before and two cycles after the latching edge of the shift clock signal SCLK. This condition sets the error status flag STAT.PE and, if enabled via CON.PEN, the error interrupt request line EIR.

Note: When CON.PH = 1, the data output signal may be disturbed shortly when the slave select input signal is changed after a serial transmission, resulting in a phase error.

A **Baud Rate Error** (Slave Mode) is detected when the incoming clock signal deviates from the programmed baud rate (shift clock) by more than 100%, meaning it is either more than double or less than half the expected baud rate. This condition sets the error status flag STAT.BE and, if enabled via CON.BEN, the EIR line. Using this error detection capability requires that the slave's shift clock generator is programmed to the same baud rate as the master device. This feature detects false additional pulses or missing pulses on the clock line (within a certain frame).

Note: If this error condition occurs and bit CON.AREN = 1, an automatic reset of the SSC will be performed. This is done to re-initialize the SSC, if too few or too many clock pulses have been detected.

Note: This error can occur after any transfer if the communication is stopped. This is due to the fact that SSC supports back-to-back transfers for multiple transfers. In order to handle this, the baud rate detection logic expects a next clock cycle immediately for a new transfer after a finished transfer.

If baud rate error is enabled and the transmit buffer of the slave SSC is loaded with a new value for the next data frame while the current data frame is not yet finished, the slave SSC expects continuation of the clock pulses for the next data frame transmission immediately after finishing the current data frame. Therefore, if the master (shift) clock is not continued, the slave SSC will detect a baud rate error. Note that the master SSC does not necessarily send out a continuous shift clock in the case that it's transmit buffer is not yet filled with new data or transmission delays occur.

A **Transmit Error** (Slave Mode) is detected when a transfer was initiated by the master (shift clock becomes active), but the transmit buffer TB of the slave was not updated since the last transfer. This condition sets the error status flag STAT.TE and, if enabled via CON.TEN, the EIR line. If a transfer starts while the transmit buffer is not updated, the slave will shift out the 'old' contents of the shift register, which is normally the data received during the last transfer. This may lead to the corruption of the data on the

Synchronous Serial Interface (SSC)

transmit/receive line in half-duplex mode (open-drain configuration) if this slave is not selected for transmission. This mode requires that slaves not selected for transmission only shift out ones; thus, their transmit buffers must be loaded with $FFFF_H$ prior to any transfer.

Note: A slave with push/pull output drivers not selected for transmission will normally have its output drivers switched off. However, to avoid possible conflicts or misinterpretations, it is recommended to always load the slave's transmit buffer prior to any transfer.

The cause of an error interrupt request (receive, phase, baud rate, transmit error) can be identified by the error status flags in control register CON.

Note: In contrast to the EIR line, the error status flags STAT.TE, STAT.RE, STAT.PE, and STAT.BE, are not cleared automatically upon entry into the error interrupt service routine, but must be cleared by software.

Synchronous Serial Interface (SSC)

20.2 SSC Kernel Registers

This section describes the kernel registers of the SSC module. All SSC kernel register names described in this section will be referenced in other parts of the TC1796 User's Manual by the module name prefix "SSC0_" for the SSC0 interface and "SSC1_" for the SSC1 interface.

SSC Kernel Register Overview

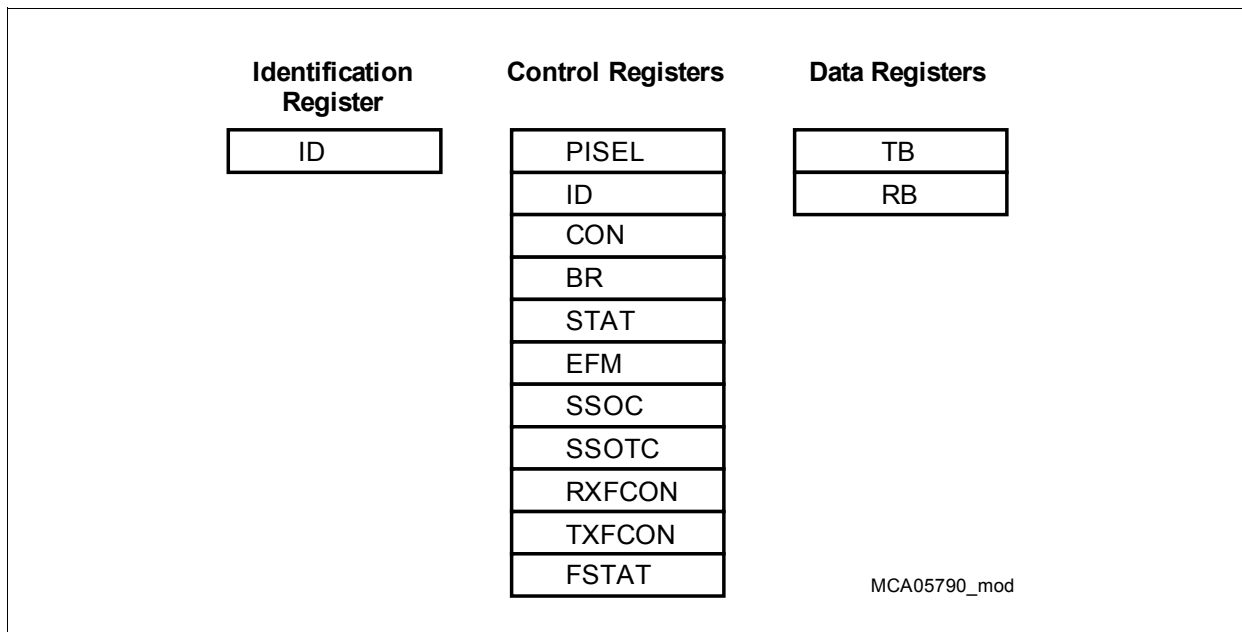


Figure 20-15 SSC Kernel Registers

The complete and detailed address map of the SSC modules is described in [Table 18-27](#) on [Page 18-81](#) of the TC1796 User's Manual System Units part (Volume 1).

Table 20-2 Registers Address Space - SSC Kernel Registers

Module	Base Address	End Address	Note
SSC0	F010 0100 _H	F010 01FF _H	—
SSC1	F010 0200 _H	F010 02FF _H	—

Table 20-3 Registers Overview - SSC Kernel Registers

Register Short Name	Register Long Name	Offset Address ¹⁾	Description see
PISEL	Port Input Select Register	04 _H	Page 20-30
ID	Module Identification Register	08 _H	Page 20-29

Synchronous Serial Interface (SSC)

Table 20-3 Registers Overview - SSC Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Address ¹⁾	Description see
CON	Control Register	10 _H	Page 20-32
BR	Baud Rate Timer Reload Register	14 _H	Page 20-39
STAT	Status Register	28 _H	Page 20-34
EFM	Error Flag Modification Register	2C _H	Page 20-35
SSOC	Slave Select Output Control Register	18 _H	Page 20-37
SSOTC	Slave Select Output Timing Control Register	1C _H	Page 20-38
TB	Transmit Buffer Register	20 _H	Page 20-45
RB	Receive Buffer Register	24 _H	Page 20-45
RXFCON	Receive FIFO Control Register	30 _H	Page 20-40
TXFCON	Transmit FIFO Control Register	34 _H	Page 20-42
FSTAT	FIFO Status Register	38 _H	Page 20-44

1) The absolute register address is calculated as follows:
Module Base Address ([Table 20-2](#)) + Offset Address (shown in this column)

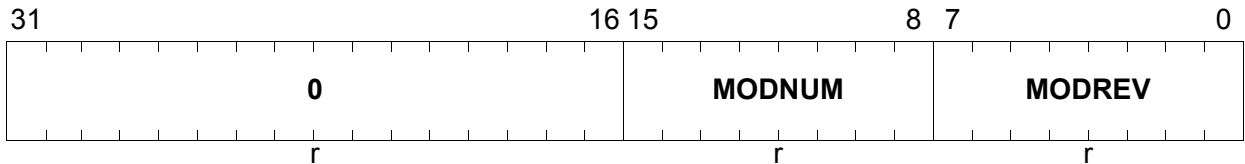
Synchronous Serial Interface (SSC)

20.2.1 Identification Register

The Module Identification Register ID contains read-only information about the SSC module version.

ID

Module Identification Register (08_H) **Reset Value: 0000 45XX_H**



Field	Bits	Type	Description
MODREV	[7:0]	r	Module Revision Number MODREV defines the module revision number. The value of a module revision starts with 01 _H (first revision).
MODNUM	[15:8]	r	Module Number Value This bit field defines the module identification number for the SSC: 45 _H
0	[31:16]	r	Reserved Read as 0.

Synchronous Serial Interface (SSC)

20.2.2 Control Registers

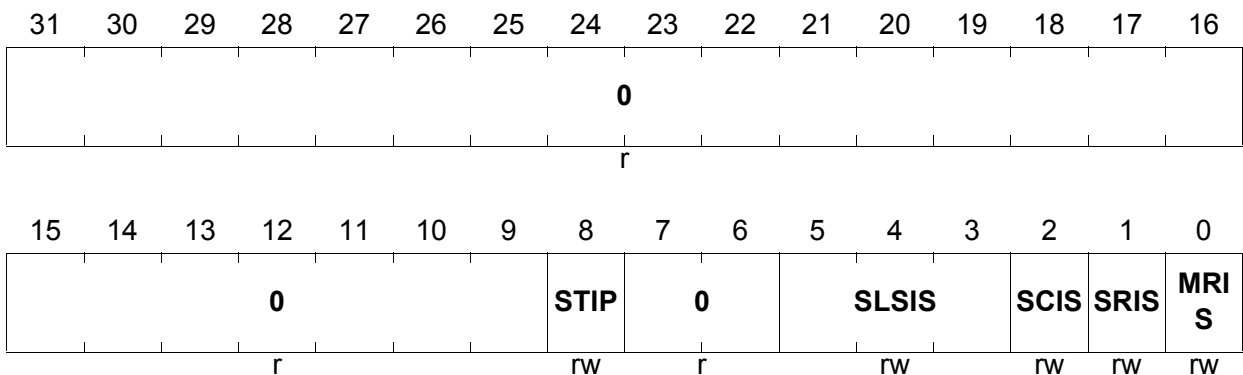
The PISEL register controls the input signal selection of the SSC module. Each input of the module kernel receive, transmit and clock signals has associated two input lines (port A and port B).

PISEL

Port Input Select Register

(04_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
MRIS	0	rw	Master Mode Receive Input Select MRIS selects the receive input line in Master Mode. 0 _B Receive input line MRSTA is selected 1 _B Receive input line MRSTB is selected
SRIS	1	rw	Slave Mode Receive Input Select SRIS selects the receive input line in Slave Mode. 0 _B Receive input line MTSRA is selected 1 _B Receive input line MTSRB is selected
SCIS	2	rw	Slave Mode Clock Input Select SCIS selects the module kernel SCLK input line that is used as clock input line in Slave Mode. 0 _B Slave Mode clock input line SCLKA is selected 1 _B 1Slave Mode clock input line SCLKB is selected

Synchronous Serial Interface (SSC)

Field	Bits	Type	Description
SLSIS	[5:3]	rw	<p>Slave Mode Slave Select Input Selection</p> <p>000_B Slave select input lines are deselected; SSC is operating without slave select input functionality.</p> <p>001_B <u>SLSI</u> input line 1 is selected for operation.</p> <p>010_B <u>SLSI</u> input line 2 is selected for operation.</p> <p>011_B <u>SLSI</u> input line 3 is selected for operation.</p> <p>100_B <u>SLSI</u> input line 4 is selected for operation.</p> <p>101_B <u>SLSI</u> input line 5 is selected for operation.</p> <p>110_B <u>SLSI</u> input line 6 is selected for operation.</p> <p>111_B <u>SLSI</u> input line 7 is selected for operation.</p> <p>In the TC1796, other combinations of SLSIS except 000_B and 001_B are reserved and must not be used.</p>
STIP	8	rw	<p>Slave Transmit Idle State Polarity</p> <p>This bit determines the logic level of the Slave Mode transmit signal when the SSC slave select input signals are inactive (PISEL.SLSIS ≠ 000_B).</p> <p>0_B MRST = 0 when slave select input signals are inactive in Slave Mode.</p> <p>1_B MRST = 1 when slave select input signals are inactive in Slave Mode.</p>
0	[7:6], [31:9]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Synchronous Serial Interface (SSC)

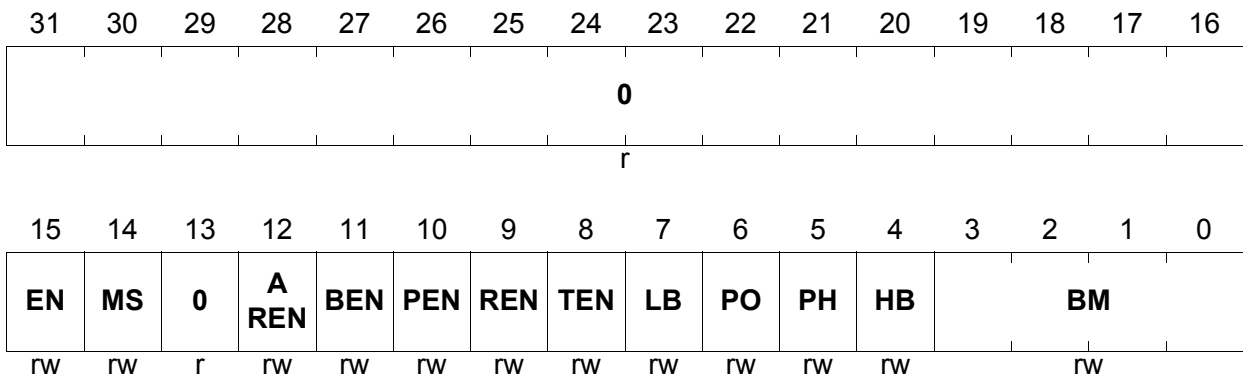
The operating modes of the SSC are controlled by the Control Register CON. This register contains control bits for mode and error check selection.

CON

Control Register

(10_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
BM	[3:0]	rw	Data Width Selection BM determines the number of data bits of the serial frame. 0000 _B Reserved; do not use this combination. 0001 _B Transfer Data Width is 2 bit. 0010 _B Transfer Data Width is 3 bit. ... _B ... 1110 _B Transfer Data Width is 15 bit. 1111 _B Transfer Data Width is 16 bit.
HB	4	rw	Heading Bit Control 0 _B Transmit/Receive LSB First 1 _B Transmit/Receive MSB First
PH	5	rw	Clock Phase Control 0 _B Shift transmit data on the leading clock edge, latch on trailing edge 1 _B Latch receive data on leading clock edge, shift on trailing edge
PO	6	rw	Clock Polarity Control 0 _B Idle clock line is low, the leading clock edge is low-to-high transition 1 _B Idle clock line is high, the leading clock edge is high-to-low transition

Synchronous Serial Interface (SSC)

Field	Bits	Type	Description
LB	7	rw	Loop-Back Control 0 _B Normal output 1 _B Receive input is connected to transmit output (Half-duplex Mode)
TEN	8	rw	Transmit Error Enable 0 _B Ignore transmit errors 1 _B Check transmit errors
REN	9	rw	Receive Error Enable 0 _B Ignore receive errors 1 _B Check receive errors
PEN	10	rw	Phase Error Enable 0 _B Ignore phase errors 1 _B Check phase errors
BEN	11	rw	Baud Rate Error Enable 0 _B Ignore baud rate errors 1 _B Check baud rate errors
AREN	12	rw	Automatic Reset Enable 0 _B No additional action upon a baud rate error 1 _B SSC is automatically reset on a baud rate error
MS	14	rw	Master Select 0 _B Slave Mode. Operate on shift clock received via SCLK 1 _B Master Mode. Generate shift clock and output it via SCLK The inverted state of this bit is available on module output line "M/S selected" (see Figure 20-2).
EN	15	rw	Enable Bit 0 _B Transmission and reception are disabled. 1 _B Transmission and reception are enabled. This bit is available as module output line "SSC enabled" (see Figure 20-2). Note that EN should only be cleared by software while no transfer is in progress (STAT.BSY = 0).
0	13, [31:16]	r	Reserved Read as 0; should be written with 0.

Synchronous Serial Interface (SSC)

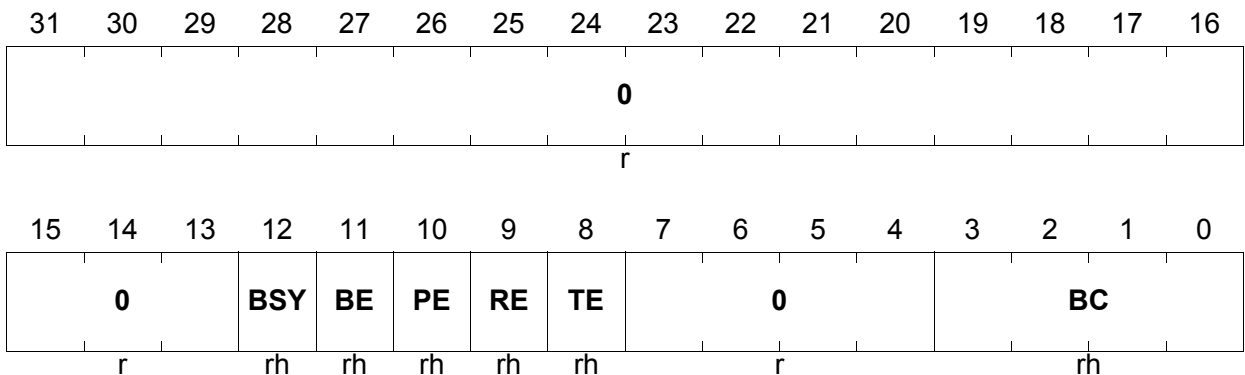
The Status Register STAT contains status flags for error identification, the busy flag, and a bit field that indicates the current shift counter status.

STAT

Status Register

(28_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
BC	[3:0]	rh	Bit Count Status BC indicates the current status of the shift counter. The shift counter is updated with every shifted bit.
TE	8	rh	Transmit Error Flag 0 _B No error 1 _B Transfer starts with the slave's transmit buffer not being updated
RE	9	rh	Receive Error Flag 0 _B No error 1 _B Reception completed before the receive buffer was read
PE	10	rh	Phase Error Flag 0 _B No error 1 _B Received data changes during the sampling clock edge
BE	11	rh	Baud Rate Error Flag 0 _B No error 1 _B There's more than factor 2 or less than factor 0.5 between the slave's actual and the expected baud rate.
BSY	12	rh	Busy Flag BSY is set while a transfer is in progress.

Synchronous Serial Interface (SSC)

Field	Bits	Type	Description
SETTE	12	w	Set Transmit Error Flag 0 _B No effect. 1 _B Bit STAT.TE is set. Bit is always read as 0.
SETRE	13	w	Set Receive Error Flag 0 _B No effect. 1 _B Bit STAT.RE is set. Bit is always read as 0.
SETPE	14	w	Set Phase Error Flag 0 _B No effect. 1 _B Bit STAT.PE is set. Bit is always read as 0.
SETBE	15	w	Set Baud Rate Error Flag 0 _B No effect. 1 _B Bit STAT.BE is set. Bit is always read as 0.
0	[7:0], [31:16]	r	Reserved Read as 0; should be written with 0.

Note: When the set and clear bits for an error flag are set at the same time during an EFM write operation (e.g. SETPE = CLRPE = 1), the error flag in STAT is not affected.

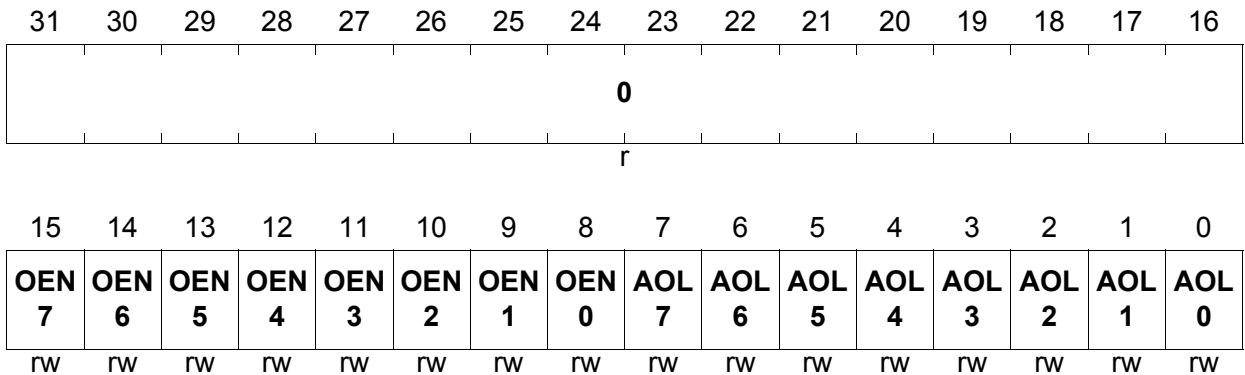
Synchronous Serial Interface (SSC)

The Slave Select Output Control Register controls the operation of the Slave Select Output Generation Unit.

SSOC

Slave Select Output Control Register (18_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
AOLn (n = 0-7)	n	rw	<p>Active Output Level</p> <p>0_B SLSON is at low level during the chip select active time $t_{SLSOACT}$. The high level is the inactive level of SLSON.</p> <p>1_B SLSON line n is at high level during the chip select active time $t_{SLSOACT}$. The low level is the inactive level of SLSON.</p>
OENn (n = 0-7)	8 + n	rw	<p>Output n Enable Control</p> <p>0_B SLSON output is disabled; SLSON is always at inactive level as defined by AOLn.</p> <p>1_B SLSON output is enabled.</p>
0	[31:16]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Note: The SSOC register content is latched by each TB register write operation and remains latched during the consecutive serial transmission.

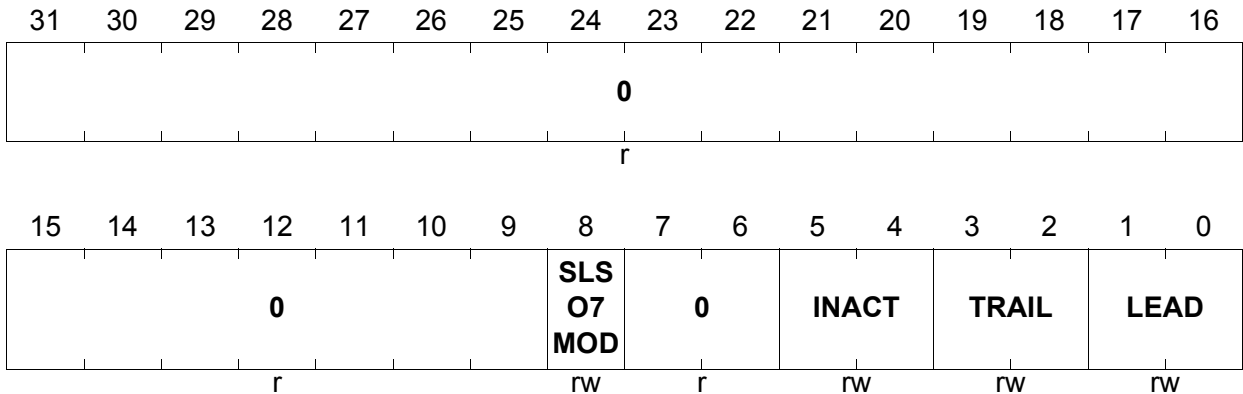
Synchronous Serial Interface (SSC)

The Slave Select Output Timing Control Register controls the timing parameters of the Slave Select Output Generation Unit.

SSOTC

**Slave Select Output Timing Control Register
(1C_H)**

Reset Value: 0000 0000_H



Field	Bits	Type	Description
LEAD	[1:0]	rw	<p>Slave Output Select Leading Delay</p> <p>This bit field determines the number of leading delay clock cycles. A leading delay clock cycle is always a multiple of an SCLK shift clock period.</p> <p>00_B Zero leading delay clock cycle selected¹⁾</p> <p>01_B One leading delay clock cycle selected</p> <p>10_B Two leading delay clock cycles selected</p> <p>11_B Three leading delay clock cycles selected</p>
TRAIL	[3:2]	rw	<p>Slave Output Select Trailing Delay</p> <p>This bit field determines the number of trailing delay clock cycles. A trailing delay clock cycle is always a multiple of an SCLK shift clock period.</p> <p>00_B Zero trailing delay clock cycle selected¹⁾</p> <p>01_B One trailing delay clock cycle selected</p> <p>10_B Two trailing delay clock cycles selected</p> <p>11_B Three trailing delay clock cycles selected</p>

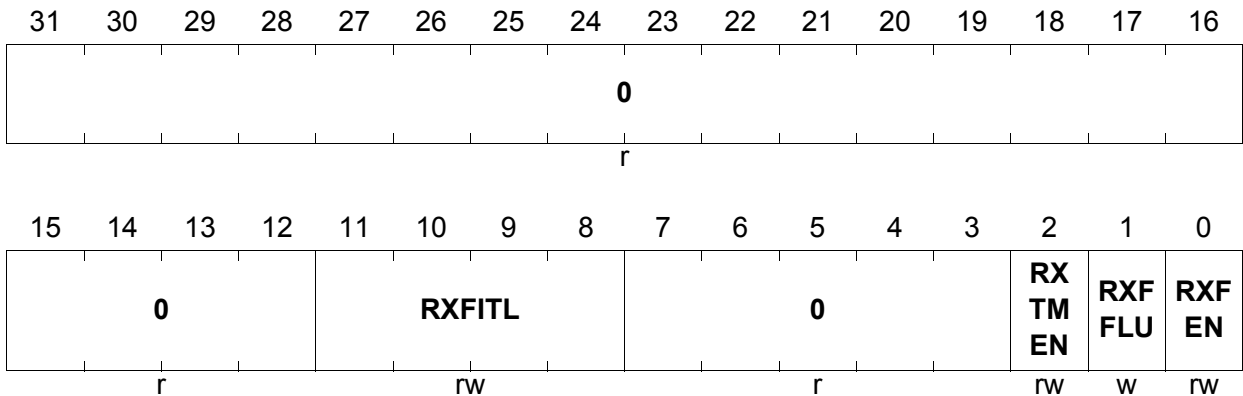
Synchronous Serial Interface (SSC)

The Receive FIFO Control Register RXFCN (only available in SSC0) contains control bits and bit fields that determine the operating mode of the RXFIFO.

SSC0_RXFCN

SSC0 Receive FIFO Control Register (30_H)

Reset Value: 0000 0100_H



Field	Bits	Type	Description
RXFEN	0	rw	<p>RXFIFO Enable</p> <p>0_B RXFIFO is disabled 1_B RXFIFO is enabled</p> <p><i>Note: Resetting RXFEN automatically flushes the RXFIFO.</i></p>
RXFFLU	1	w	<p>RXFIFO Flush</p> <p>0_B No operation 1_B RXFIFO is flushed</p> <p><i>Note: Setting RXFFLU clears bit field RXFFL in register FSTAT. RXFFLU is always read as 0.</i></p>
RXTMEN	2	rw	<p>RXFIFO Transparent Mode Enable</p> <p>0_B RXFIFO Transparent Mode is disabled 1_B RXFIFO Transparent Mode is enabled</p> <p><i>Note: This bit is “don’t care” if the RXFIFO is disabled (RXFEN = 0).</i></p>

Synchronous Serial Interface (SSC)

Field	Bits	Type	Description
RXFITL	[11:8]	rw	<p>RXFIFO Interrupt Trigger Level¹⁾ Determines the RXFIFO interrupt trigger level. A receive interrupt request (RIR) is always generated after the reception of a byte when the filling level of the RXFIFO is equal to or greater than RXFITL.</p> <p>0000_B Reserved. Do not use this combination 0001_B Interrupt trigger level is set to 1 0010_B Interrupt trigger level is set to 2 ..._B ... 0111_B Interrupt trigger level is set to 7 1000_B Interrupt trigger level is set to 8 Other combinations of RXFITL are reserved and should not be used.</p> <p><i>Note: In Transparent Mode this bit field is "don't care".</i></p>
0	[7:3], [31:12]	r	<p>Reserved Read as 0; should be written with 0.</p>

1) In the SSC0 module with the 8-stage RXFIFO, the most significant bit of RXFITL (= RXFCON.12) is always read as 0 and should be written with 0.

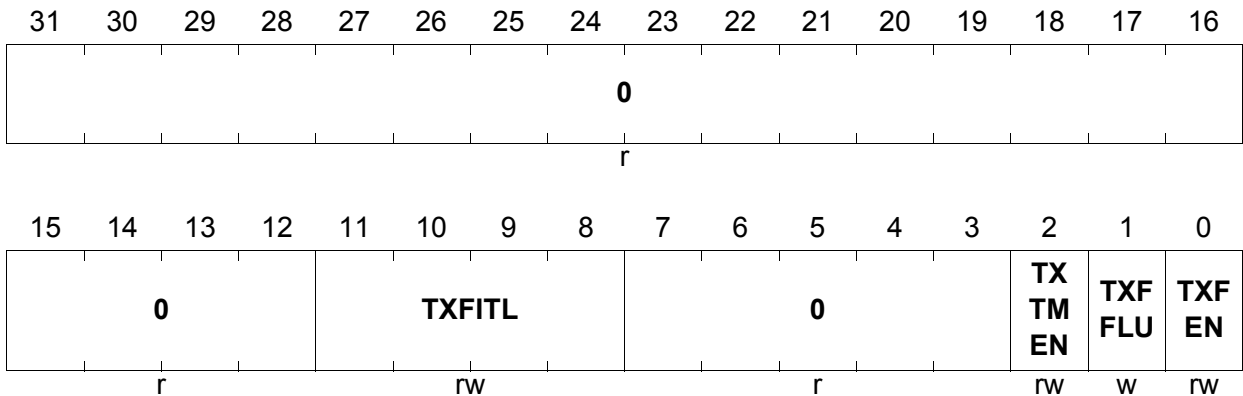
Synchronous Serial Interface (SSC)

The Transmit FIFO Control Register TXFCON (only available in SSC0) contains control bits and bit fields that determine the operating mode of the RXFIFO.

SSC0_TXFCON

SSC0 Transmit FIFO Control Register (34_H)

Reset Value: 0000 0100_H



Field	Bits	Type	Description
TXFEN	0	rw	TXFIFO Enable 0 _B TXFIFO is disabled 1 _B TXFIFO is enabled <i>Note: Resetting TXFEN automatically flushes the TXFIFO.</i>
TXFFLU	1	w	TXFIFO Flush 0 _B No operation 1 _B TXFIFO is flushed <i>Note: Setting TXFFLU clears bit field TXFFL in register FSTAT. TXFFLU is always read as 0.</i>
TXTMEN	2	rw	TXFIFO Transparent Mode Enable 0 _B TXFIFO Transparent Mode is disabled 1 _B TXFIFO Transparent Mode is enabled <i>Note: This bit is “don’t care” if the TXFIFO is disabled (TXFEN = 0).</i>

Synchronous Serial Interface (SSC)

Field	Bits	Type	Description
TXFITL	[11:8]	rw	<p>TXFIFO Interrupt Trigger Level¹⁾ Determines a TXFIFO interrupt trigger level. A transmit interrupt request (TIR) is always generated after the transfer of a byte when the filling level of the TXFIFO is equal to or less than TXFITL.</p> <p>0000_B Reserved. Do not use this combination 0001_B Interrupt trigger level is set to 1 0010_B Interrupt trigger level is set to 2 ..._B ... 0111_B Interrupt trigger level is set to 7 1000_B Interrupt trigger level is set to 8 Other combinations of TXFITL are reserved and should not be used.</p> <p><i>Note: In Transparent Mode this bit field is "don't care".</i></p>
0	[7:3], [31:12]	r	<p>Reserved Read as 0; should be written with 0.</p>

1) In the SSC0 module with the 8-stage TXFIFO, the most significant bit of TXFITL (= TXFCON.12) is always read as 0 and should be written with 0.

Synchronous Serial Interface (SSC)

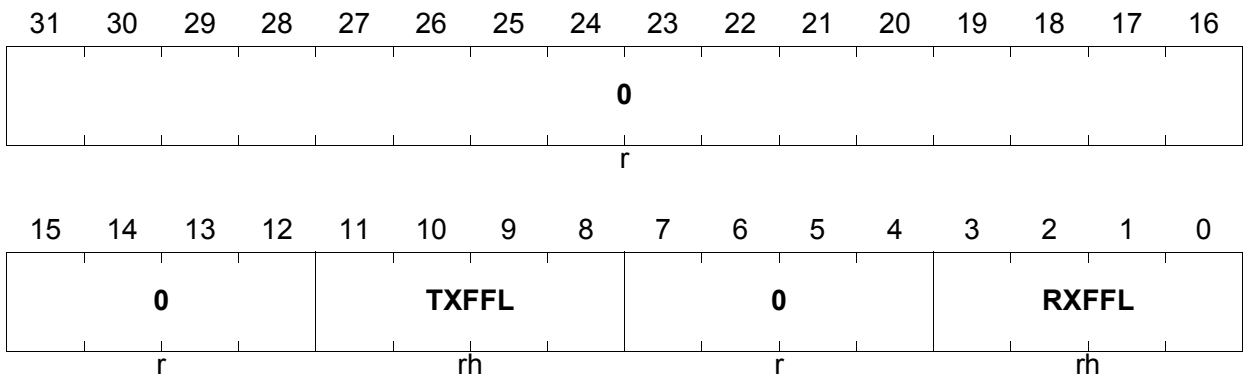
The FIFO Status Register FSTAT (only available in SSC0) indicates the filling levels of the Receive and Transmit FIFOs.

SSC0_FSTAT

SSC0 FIFO Status Register

(38_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
RXFFL	[3:0]	rh	RXFIFO Filling Level¹⁾²⁾ This bit field indicates the filling level of the RXFIFO. 0000 _B RXFIFO is filled with 0 bytes 0001 _B RXFIFO is filled with 1 byte ... 0111 _B RXFIFO is filled with 7 bytes 1000 _B RXFIFO is filled with 8 bytes RXFFL is cleared after a RXFIFO flush operation.
TXFFL	[11:8]	rh	TXFIFO Filling Level¹⁾²⁾ This bit field indicates the filling level of the TXFIFO. 0000 _B TXFIFO is filled with 0 bytes 0001 _B TXFIFO is filled with 1 byte ... 0111 _B TXFIFO is filled with 7 bytes 1000 _B TXFIFO is filled with 8 bytes TXFFL is cleared after a TXFIFO flush operation.
0	[7:4], [31:12]	r	Reserved Read as 0; should be written with 0.

- 1) The data width of a RXFIFO and TXFIFO stage can be programmed from 2 to 15 bits. The data width "byte" mentioned in this description represents a data width of 8 bits.
- 2) In the SSC0 module with 8-stage RXFIFO and TXFIFO, the most significant bits of RXFFL and TXFFL (FSTAT.4 and FSTAT.12) are always read as 0 and should be written with 0.

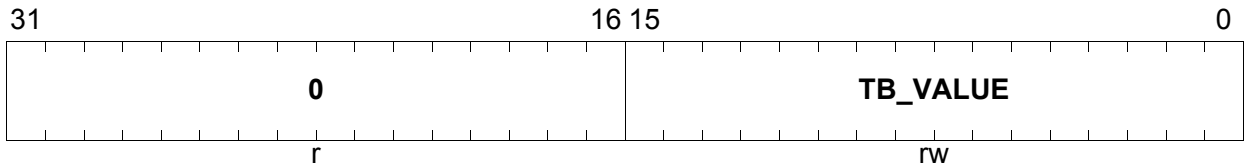
Synchronous Serial Interface (SSC)

20.2.3 Data Registers

The Transmit Buffer Register TB contains the transmit data value. A TB write operation latches all timing parameters stored in register SSOTC.

TB

Transmit Buffer Register (20_H) **Reset Value: 0000 0000_H**

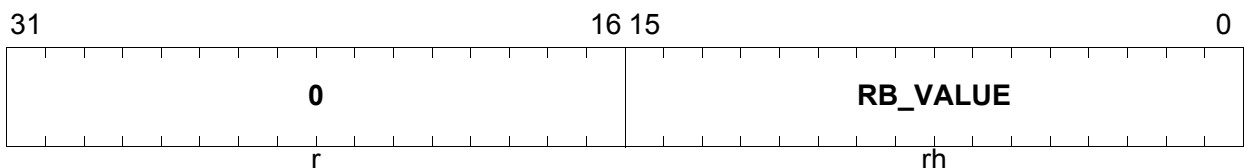


Field	Bits	Type	Description
TB_VALUE	[15:0]	rw	Transmit Data Register Value Register TB_VALUE stores the data value to be transmitted TB_VALUE. Unused bits of TB_VALUE (as defined by CON.BM) are ignored during transmission.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

The Receive Buffer Register RB contains the receive data value.

RB

Receive Buffer Register (24_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
RB_VALUE	[15:0]	rh	Receive Data Register Value Register RB contains the received data value RB_VALUE. Unused bits of RB_VALUE (as defined by CON.BM) will be not valid and should be ignored.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

Synchronous Serial Interface (SSC)

20.3 SSC0/SSC1 Module Implementation

This section describes SSC0/SSC1 module interfaces with the clock control, port connections, interrupt control, and address decoding.

20.3.1 Interfaces of the SSC Modules

Figure 20-16 shows the TC1796-specific implementation details and interconnections of the SSC0/SSC1 modules.

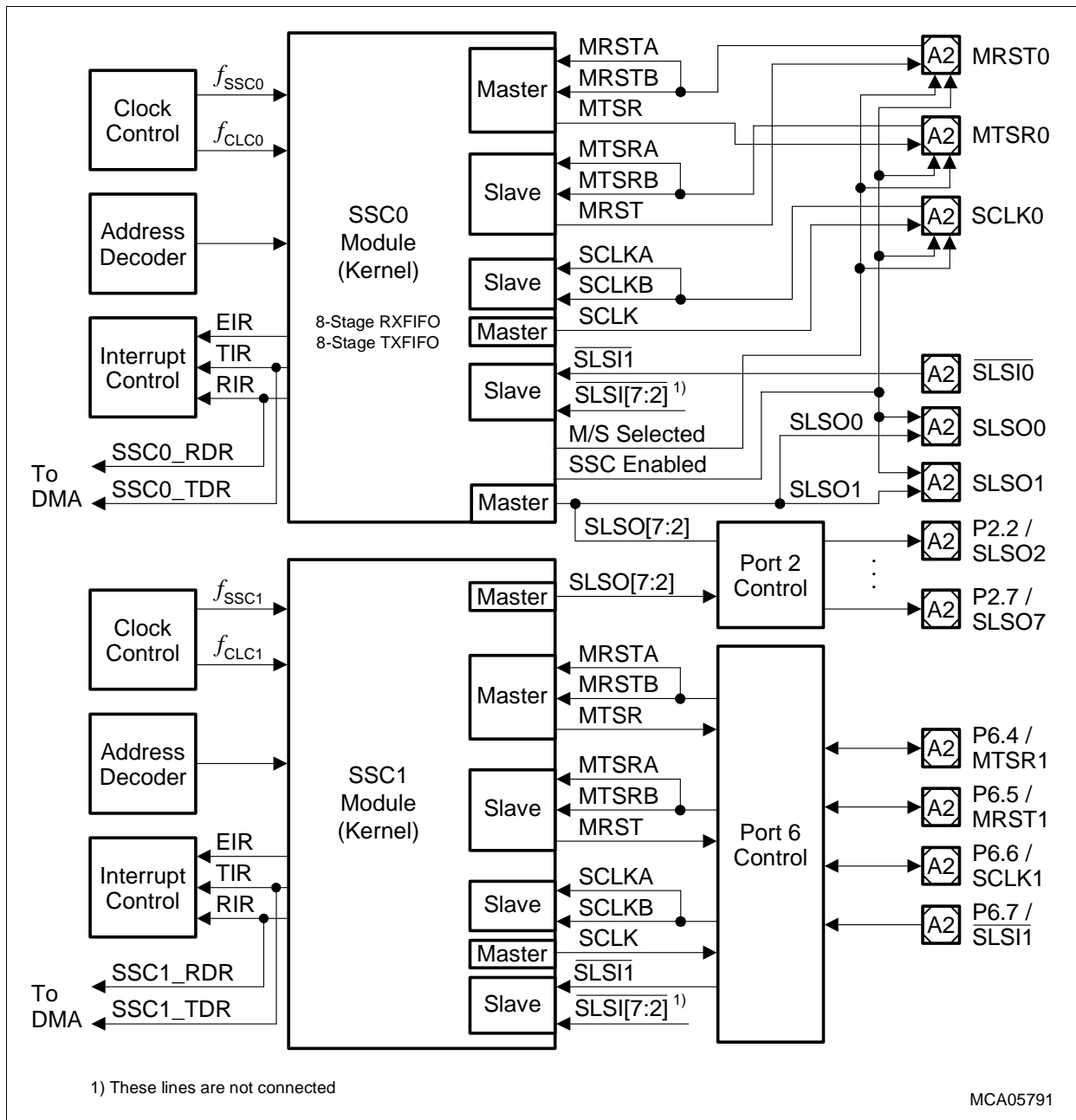


Figure 20-16 SSC0/SSC1 Module Implementation and Interconnections

Synchronous Serial Interface (SSC)

Each of the SSC modules is supplied by a separate clock control, interrupt control, and address decoding logic. Two interrupt outputs can be used to generate DMA requests. The SSC0 I/O lines are connected to dedicated pins. The SSC1 I/O lines are connected to Port 2 and Port 6. The SLSOx outputs of SSC0 and SSC1 are wired as alternate function to six I/O lines of Port 2. Two SLSOx outputs of SSC0 are connected to dedicated pins. Note that only SSC0 contains an 8-stage Receive and Transmit FIFO. SSC1 does not provide any FIFO functionality.

20.3.2 SSC0/SSC1 Module Related External Registers

Figure 20-17 summarizes the module-related external registers which are required for SSC0/SSC1 programming (see also **Figure 20-15** for the module kernel-specific registers).

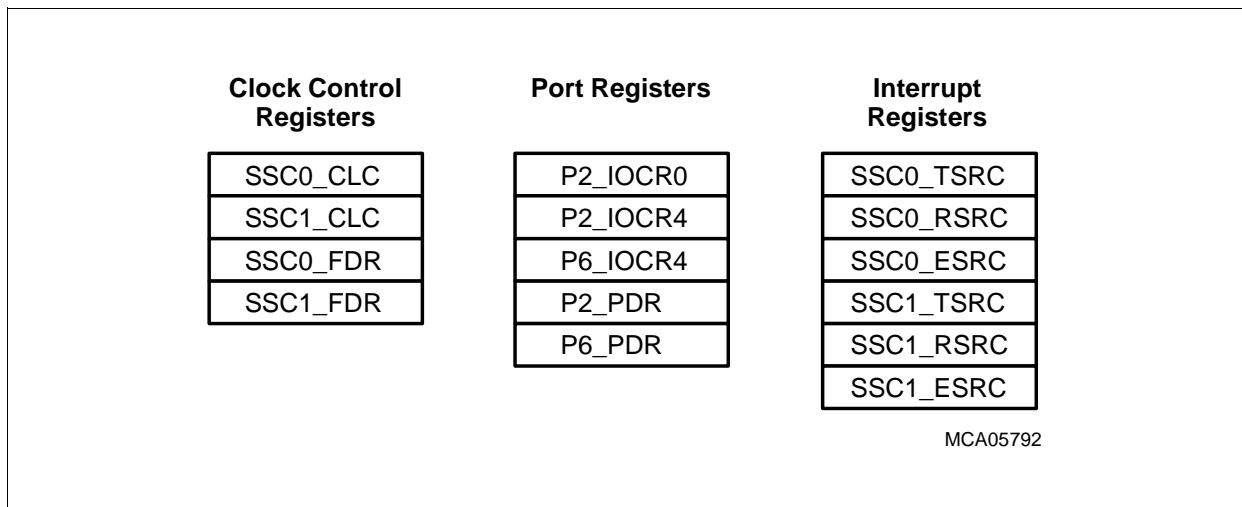


Figure 20-17 SSC0/SSC1 Implementation-Specific Special Function Registers

20.3.2.1 Clock Control

The SSC modules each have two clock signals:

- f_{CLC0} and f_{CLC1}
This is the module clock that is used inside the SSC kernel for control purposes such as, e.g. clocking of control logic and register operations. The frequency of f_{CLC0} and f_{CLC1} is always identical to the system clock frequency f_{SYS} . The clock control registers SSC0_CLC and SSC1_CLC make it possible to enable/disable f_{CLC0} and f_{CLC1} under certain conditions.
- f_{SSC0} and f_{SSC1}
This clock is the module clock that is used in the SSC as input clock of the baud rate generator, which finally determines the baud rate of the serial data. The fractional divider registers SSC0_FDR and SSC1_FDR control the frequency of f_{SSC0} and f_{SSC1} and make it possible to enable/disable it independently of f_{CLC0} and f_{CLC1} . The Baud Rate Timer Reload Register SSC0_BR and SSC1_BR define serial data baud rate dependent from the frequency of f_{SSC0} and f_{SSC1} .

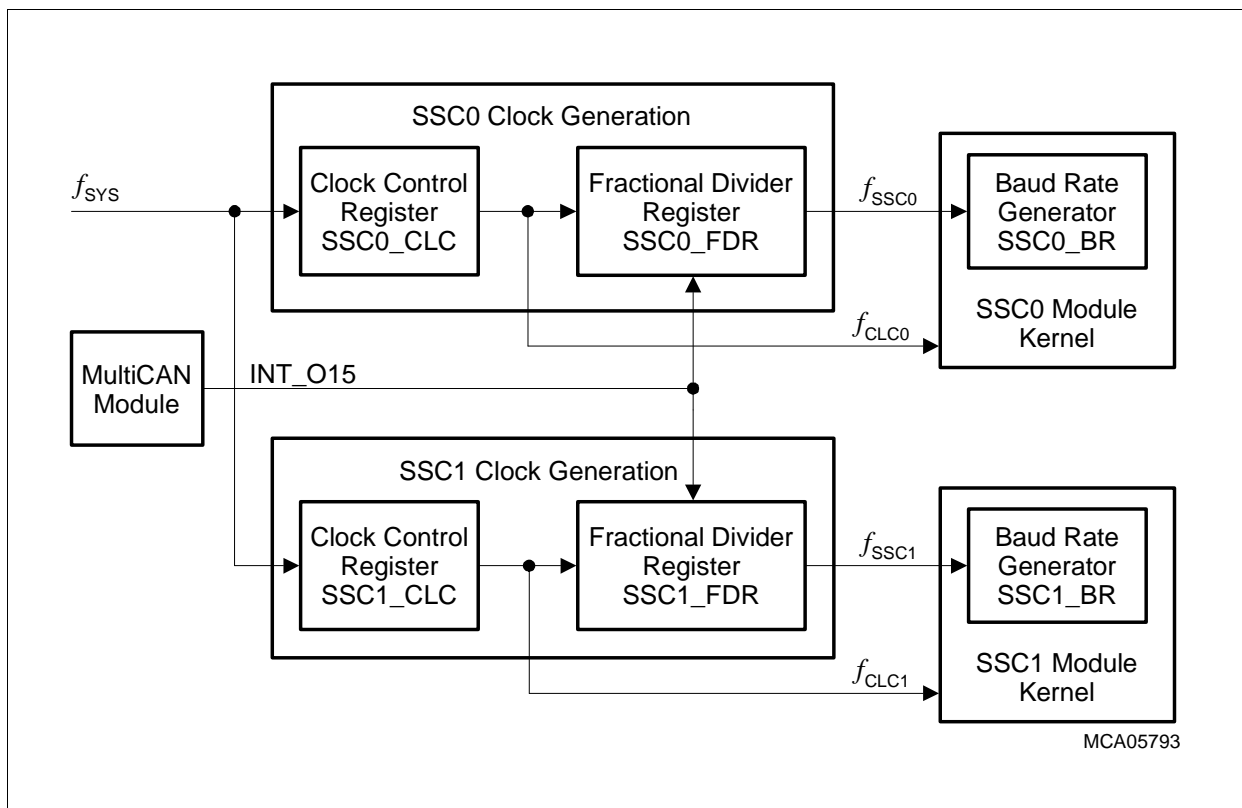


Figure 20-18 SSC Clock Generation

Output signal CAN_INT_O15 of the MultiCAN module can be used for external clock enable control of the fractional divider.

Synchronous Serial Interface (SSC)

The following formulas define the frequency of f_{SSC0} or f_{SSC1} :

$$f_{SSCx} = f_{SYS} \times \frac{1}{n} \text{ with } n = 1024 - \text{FDR.STEP} \quad (20.2)$$

$$f_{SSCx} = f_{SYS} \times \frac{n}{1024} \text{ with } n = 0-1023 \quad (20.3)$$

Note: In SSC Master Mode, the maximum shift clock frequency is $f_{SSCx}/2$. In SSC Slave Mode, the maximum shift clock frequency is $f_{SSCx}/4$.

Combined with the formulas of the baud rate generator (see [Page 20-18](#)) and the fractional divider (see chapter “System Control Unit” of the TC1796 System Units User’s Manual), the resulting serial data baud rate is defined by:

$$\text{Baud rate}_{SSC} = \frac{f_{SYS}}{2 \times (\text{BR.BR_VALUE} + 1) \times (1024 - \text{FDR.STEP})} \quad (20.4)$$

$$\text{Baud rate}_{SSC} = \frac{f_{SYS} \times \text{FDR.STEP}}{2 \times (\text{BR.BR_VALUE} + 1) \times 1024} \text{ with } \text{FDR.STEP} = 0-1023 \quad (20.5)$$

Note: [Equation \(20.2\)](#) and [Equation \(20.4\)](#) apply to normal divider mode of the fractional divider ($\text{FDR.DM} = 01_B$). [Equation \(20.3\)](#) and [Equation \(20.5\)](#) apply to fractional divider mode ($\text{FDR.DM} = 10_B$).

Synchronous Serial Interface (SSC)

Note: Additional details of the clock control register functionality are described in section **“Clock Control Register CLC” on Page 3-24** of the TC1796 User’s Manual System Units part (Volume 1).

Fractional Divider Register

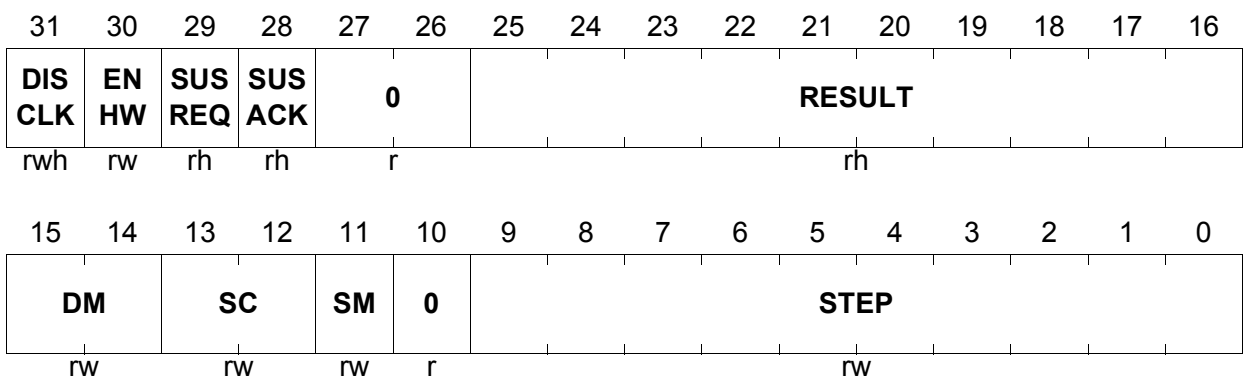
The Fractional Divider Registers SSC0_FDR and SSC1_FDR control the clock rate of the shift clock f_{SSC0} and f_{SSC1} . Each SSC has its own fractional divider register.

SSC0_FDR

Fractional Divider Register (0C_H) Reset Value: 0000 0000_H

SSC1_FDR

SSC1 Fractional Divider Register (0C_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
STEP	[9:0]	rw	Step Value Reload or addition value for RESULT.
SM	11	rw	Suspend Mode SM selects between granted or immediate suspend mode.
SC	[13:12]	rw	Suspend Control This bit field determines the behavior of the fractional divider in suspend mode.
DM	[15:14]	rw	Divider Mode This bit field selects normal divider mode, fractional divider mode, and off-state.
RESULT	[25:16]	rh	Result Value Bit field for the addition result.
SUSACK	28	rh	Suspend Mode Acknowledge Indicates state of SPNDACK signal.

Synchronous Serial Interface (SSC)

Field	Bits	Type	Description
SUSREQ	29	rh	Suspend Mode Request Indicates state of SPND signal.
ENHW	30	rw	Enable Hardware Clock Control Controls operation of ECEN input and DISCLK bit.
DISCLK	31	rwh	Disable Clock Hardware-controlled disable for f_{OUT} signal.
0	10, [27:26]	rw	Reserved Read as 0; should be written with 0.

Note: Further details of the fractional divider register functionality are described in section “[Fractional Divider Operation](#)” on [Page 3-29](#) of the TC1796 User’s Manual System Units part (Volume 1).

20.3.2.2 Port Control

The interconnections between the SSC modules and the I/O lines/pins are controlled by:

- Hardware signals of the module kernel for SSC0
- Software in the port logic for SSC1

The **SSC0** is directly connected to dedicated class A2 pins. Bit SSC0_CON.EN enables/disables the dedicated pins (signal “SSC Enabled”). After a reset, SSC0_CON.EN is 0 and the dedicated SSC0 I/O pins are tri-stated. With SSC0_CON.EN = 1, the dedicated SSC0 I/O pins are enabled and the state of SSC0_CON.MS (signal “M/S Enabled”) determines whether they act as input or output. When the SSC0 is enabled (SSC0_CON.EN = 1), changing of SSC0_CON.MS directly affects the I/O direction of the three SSC0 communication I/O pins.

Note: The dedicated SSC0 output pins do not provide open-drain capability.

The **SSC1** I/O functionality must be selected by the following port control operations (additionally to the PISEL programming):

- Input/output function selection (IOCR registers)
- Pad driver characteristics selection for the outputs (PDR registers)

The SSC1 port input/output control registers contain the bit fields that select the digital output and input driver characteristics such as pull-up/down devices, port direction (input/output), open-drain, and alternate output selections. The master/slave I/O lines and the slave select input of the SSC1 module are class A2 GPIO pins that are controlled by a port input/output control register of Port 6.

Each of the six **SLSOx** outputs of SSC0 and SSC1 is wired as alternate function to six class A2 GPIO pins of Port 2 (P2.[7:2]). Two SLSOx outputs with identical index “x” (not the case for P2.0 and P2.1) are always connected to one I/O line of Port 2, and can be

Synchronous Serial Interface (SSC)

selected as alternate function via the P2_IOCR0 and P2_IOCR4 registers. Additionally, the logical AND combination of two SLSOx outputs with identical index “x” can be connected as output to Port 2 (see [Figure 20-19](#)). Two SLSOx outputs of SSC0, SLSO0 and SLSO1, are connected to dedicated class A2 pins. These dedicated pins are always outputs that are tri-stated when SSC0 is disabled (SSC0_CON.EN = 0).

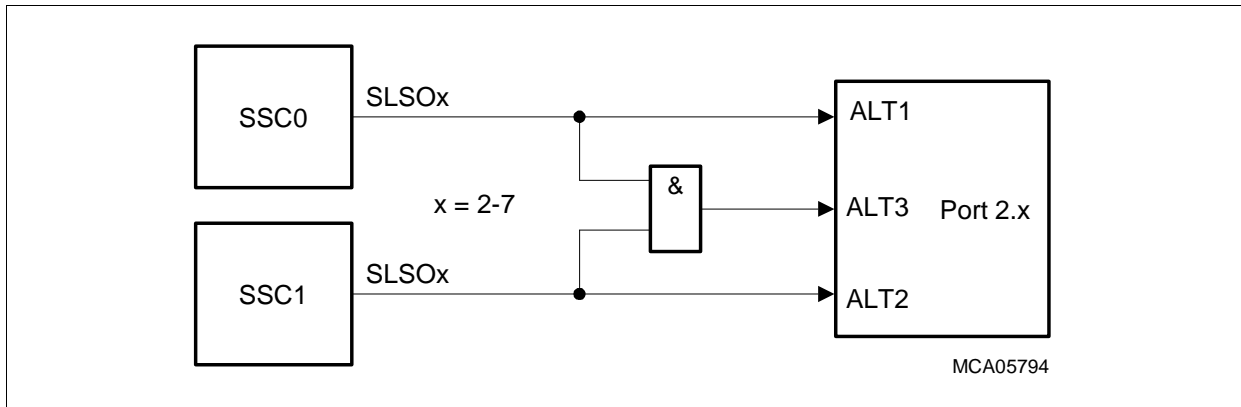


Figure 20-19 SLSO Output Selection

[Table 20-4](#) shows how bits and bit fields must be programmed for the required I/O functionality of the SSC I/O lines.

Table 20-4 SSC0 and SSC1 I/O Line Selection and Setup

Module	Port Lines	Input/Output Control Register Bits ¹⁾		I/O	
SSC0	MTR0	SSC0_CON.MS = 0	SSC0_CON.EN = 1	Input	
		SSC0_CON.MS = 1		Output	
	MRST0	SSC0_CON.MS = 1		Input	
		SSC0_CON.MS = 0		Output	
	SCLK0	SSC0_CON.MS = 0		Input	
		SSC0_CON.MS = 1		Output	
	MTR0 MRST0 SCLK0	–		SSC0_CON.EN = 0	Tri-States
	SLSI0	–		–	Input

Synchronous Serial Interface (SSC)

Table 20-4 SSC0 and SSC1 I/O Line Selection and Setup (cont'd)

Module	Port Lines	Input/Output Control Register Bits ¹⁾	I/O
SSC1	P6.4/MTSR1	P6_IOCR4.PC4 = 0XXX _B	Input
		P6_IOCR4.PC4 = 1X01 _B	Output
	P6.5/MRST1	P6_IOCR4.PC5 = 0XXX _B	Input
		P6_IOCR4.PC5 = 1X01 _B	Output
	P6.6/SCLK1	P6_IOCR4.PC6 = 0XXX _B	Input
		P6_IOCR4.PC6 = 1X01 _B	Output
P6.7/SLSI1	P6_IOCR4.PC7 = 0XXX _B	Input	

Slave Select Outputs

SSC0	SLSO0 SLSO1	SSC0_CON.EN = 1	Output
	SLSO0 SLSO1	SSC0_CON.EN = 0	Tri-States
SSC0	P2.2/SLSO2	P2_IOCR0.PC2 = 1X01 _B	Output
SSC1		P2_IOCR0.PC2 = 1X10 _B	
SSC0 & SSC1		P2_IOCR0.PC2 = 1X11 _B	
SSC0	P2.3/SLSO3	P2_IOCR0.PC3 = 1X01 _B	Output
SSC1		P2_IOCR0.PC3 = 1X10 _B	
SSC0 & SSC1		P2_IOCR0.PC3 = 1X11 _B	
SSC0	P2.4/SLSO4	P2_IOCR4.PC4 = 1X01 _B	Output
SSC1		P2_IOCR4.PC4 = 1X10 _B	
SSC0 & SSC1		P2_IOCR4.PC4 = 1X11 _B	
SSC0	P2.5/SLSO5	P2_IOCR4.PC5 = 1X01 _B	Output
SSC1		P2_IOCR4.PC5 = 1X10 _B	
SSC0 & SSC1		P2_IOCR4.PC5 = 1X11 _B	
SSC0	P2.6/SLSO6	P2_IOCR4.PC6 = 1X01 _B	Output
SSC1		P2_IOCR4.PC6 = 1X10 _B	
SSC0 & SSC1		P2_IOCR4.PC6 = 1X11 _B	
SSC0	P2.7/SLSO7	P2_IOCR4.PC7 = 1X01 _B	Output
SSC1		P2_IOCR4.PC7 = 1X10 _B	
SSC0 & SSC1		P2_IOCR4.PC7 = 1X11 _B	

1) Possible PCx bit field combinations see [Table 20-5](#).

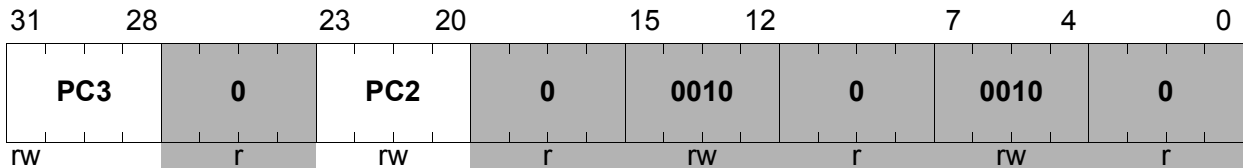
Synchronous Serial Interface (SSC)

Input/Output Control Registers

P2_IOCRO

Port 2 Input/Output Control Register 0(10_H)

Reset Value: 2020 2020_H



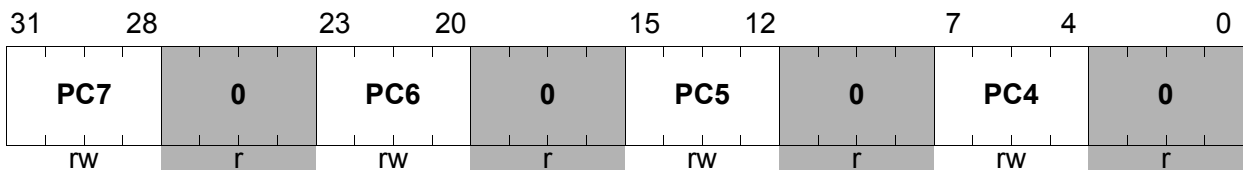
Field	Bits	Type	Description
PC2, PC3	[23:20], [31:28]	rw	Port Output Control for Port 2.2 and P2.3¹⁾ These bit fields determine the output port functionality: Port input/output control for P2.2/SLSO2 Port input/output control for P2.3/SLSO3

1) Coding of bit field see [Table 20-5](#). Shaded bits and bit fields are “don’t care” for SSC I/O port control.

P2_IOCRA

Port 2 Input/Output Control Register 4(14_H)

Reset Value: 2020 2020_H



Field	Bits	Type	Description
PC4, PC5, PC6, PC7	[7:4], [15:12], [23:20], [31:28]	rw	Port Output Control for Port 2.[7:4]¹⁾ These bit fields determine the output port functionality: Port input/output control for P2.4/SLSO4 Port input/output control for P2.5/SLSO5 Port input/output control for P2.6/SLSO6 Port input/output control for P2.7/SLSO7

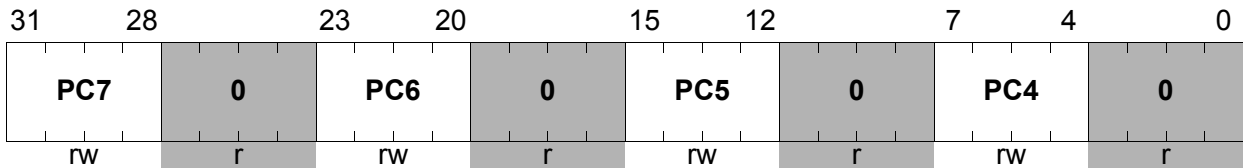
1) Coding of bit field see [Table 20-5](#). Shaded bits and bit fields are “don’t care” for SSC I/O port control.

Synchronous Serial Interface (SSC)

P6_IOCRA4

Port 6 Input/Output Control Register 4(14_H)

Reset Value: 2020 2020_H



Field	Bits	Type	Description
PC4, PC5, PC6, PC7	[7:4], [15:12], [23:20], [31:28]	rw	Port Output Control for Port 6.[7:4]¹⁾ These bit fields determine the output port functionality:
			Port input/output control for P6.4/MTSR1
			Port input/output control for P6.5/MRST1
			Port input/output control for P6.6/SCLK1 Port input/output control for P6.7/SLSI1

1) Coding of bit field see [Table 20-5](#). Shaded bits and bit fields are “don’t care” for SSC I/O port control.

PCX Coding Table

Table 20-5 PCx Coding

PCx[3:0]	I/O	Output Characteristics	Selected Pull-up/Pull-down/ Selected Output Function
0X00 _B	Input	–	No pull device connected
0X01 _B			Pull-down device connected
0X10 _B ¹⁾			Pull-up device connected
0X11 _B			No pull device connected
1001 _B	Output	Push-pull	Output function ALT1
1101 _B		Open-drain	
1010 _B		Push-pull	Output function ALT2
1110 _B		Open-drain	
1011 _B		Push-pull	Output function ALT3
1111 _B		Open-drain	

1) This bit field value is default after reset.

Synchronous Serial Interface (SSC)

PDx Selection Table

Table 20-6 Pad Driver Mode Mode Selection (Class A2 Pads)

PDx Bit Field	Driver Strength	Signal Transitions
000 _B	Strong driver	Sharp edge ¹⁾
001 _B		Medium edge ¹⁾
010 _B		Soft edge ¹⁾
011 _B	Weak driver	–
100 _B	Medium driver	Sharp edge
101 _B		Medium edge
110 _B		Soft edge
111 _B	Weak driver	–

1) In strong driver mode, the output driver characteristics of class A2 pads can be additionally controlled by the temperature compensation logic.

Synchronous Serial Interface (SSC)

20.3.2.3 Interrupt Control Registers

The 2 × 3 interrupts of the SSC0 and SSC1 modules are controlled by the following service request control registers:

- SSC0_TSRC, SSC1_TSRC controls the transmit interrupts
- SSC0_RSRC, SSC1_RSRC controls the receive interrupts
- SSC0_ESRC, SSC1_ESRC controls the error interrupts

TSRC

Transmit Interrupt Service Request Control Register

(F4_H)

Reset Value: 0000 0000_H

RSRC

Receive Interrupt Service Request Control Register

(F8_H)

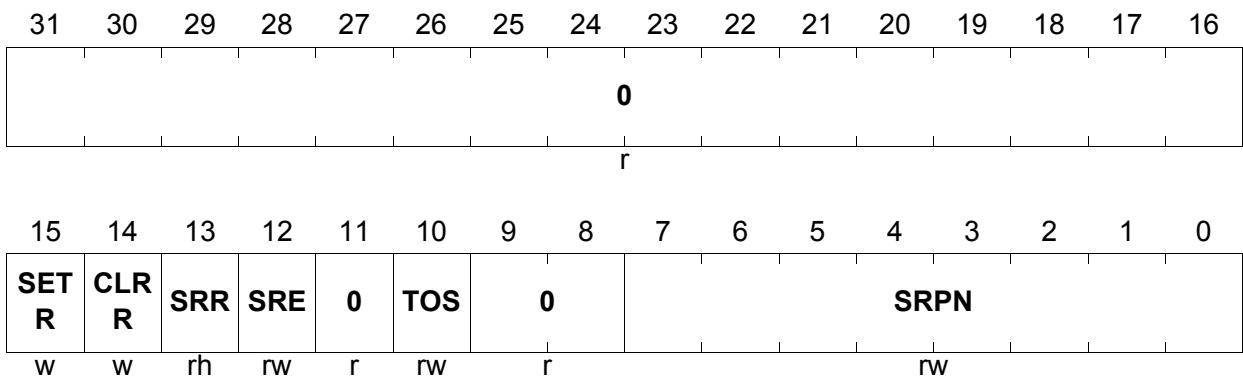
Reset Value: 0000 0000_H

ESRC

Error Interrupt Service Request Control Register

(FC_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SRPN	[7:0]	rw	Service Request Priority Number
TOS	10	rw	Type of Service Control
SRE	12	rw	Service Request Enable
SRR	13	rh	Service Request Flag
CLRR	14	w	Request Clear Bit
SETR	15	w	Request Set Bit
0	[9:8], 11, [31:16]	r	Reserved Read as 0; should be written with 0.

Synchronous Serial Interface (SSC)

Note: Additional details on service request nodes and the service request control registers are described on [Page 14-3](#) of the TC1796 User's Manual System Units part (Volume 1).

20.3.3 DMA Requests

The DMA request lines of the SSC0/SSC1 modules become active whenever the corresponding interrupt request line becomes activated. The DMA request lines are connected to the DMA controller as shown in [Table 20-7](#).

Table 20-7 DMA Request Lines of SSC0/SSC1

Module	SSC Interrupt Request Line	DMA Request Line	Description
SSC0	RIR	SSC0_RDR	SSC0 Receive DMA Request
	TIR	SSC0_TDR	SSC0 Transmit DMA Request
SSC1	RIR	SSC1_RDR	SSC1 Receive DMA Request
	TIR	SSC1_TDR	SSC1 Transmit DMA Request

Note: Further details on DMA handling and processing are described in [Chapter 12](#) of the TC1796 System Units User's Manual (Volume 1).

21 Micro Second Channel (MSC)

This chapter describes the two Micro Second Channel Interfaces, MSC0 and MSC1, of the TC1796. It contains the following sections:

- Functional description of the MSC kernel (see [Page 21-3](#))
- MSC kernel register descriptions (see [Page 21-36](#))
- TC1796 implementation-specific details and registers of the MSC module (port connections and control, interrupt control, address decoding, and clock control, see [Page 21-62](#))

Note: The MSC kernel register names described in [Section 21.2](#) will be referenced in the TC1796 User's Manual by the module name prefix "MSC0_" for the MSC0 module, and by "MSC1_" for the MSC1 module.

Micro Second Channel (MSC)

MSC Applications

The MSC is a serial interface that is especially designed to connect external power devices to the TC1796. The serial data transmission capability minimizes the number of pins required to connect such external power devices. Parallel data information (coming from the timer units) or command information is sent out to the power device via a high-speed synchronous serial data stream (downstream channel). The MSC receives data and status back from the power device via a low-speed asynchronous serial data stream (upstream channel).

Figure 21-1 shows a typical TC1796 application in which an MSC interfaces controls two power devices. Output data is provided by the GPTA modules.

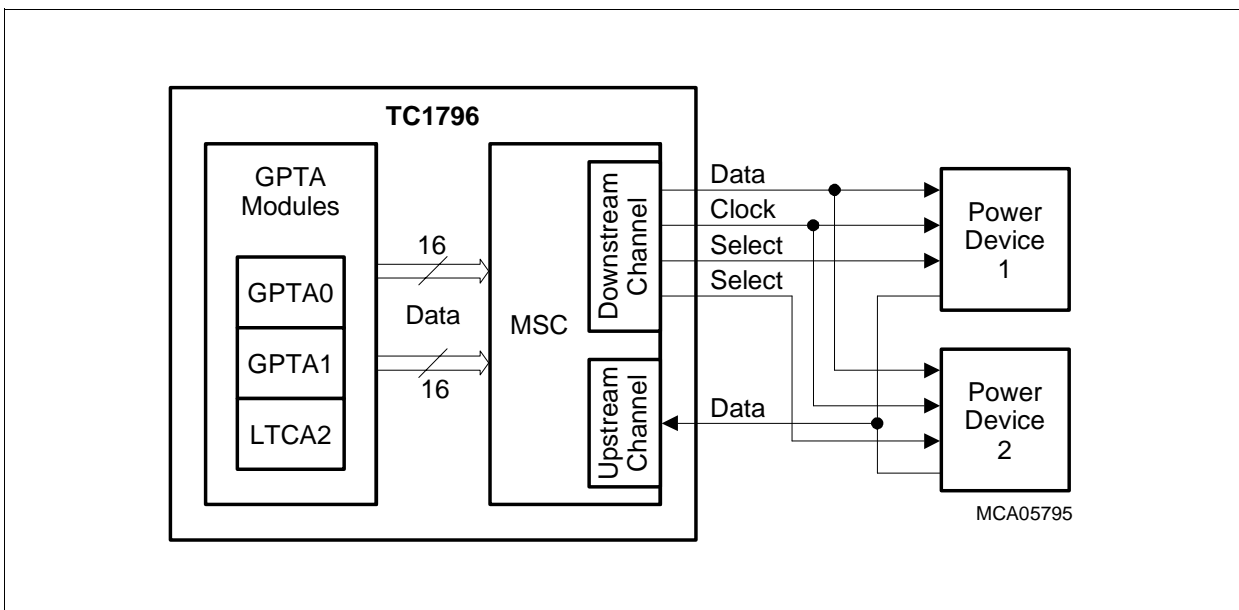


Figure 21-1 MSC to External Power Device Connection

Some applications are:

- Control of the external power switching unit via the downstream channel
- Receiving informations back from power switching unit
- Serial connections of the TC1796 to other peripheral devices

21.1 MSC Kernel Description

This section describes the functionality of the MSC kernel.

21.1.1 Overview

The MSC interface provides a serial communication link typically used to connect power switches or other peripheral devices. The serial communication link includes a fast synchronous downstream channel and a slow asynchronous upstream channel.

Figure 21-2 shows a global view of the MSC interface signals.

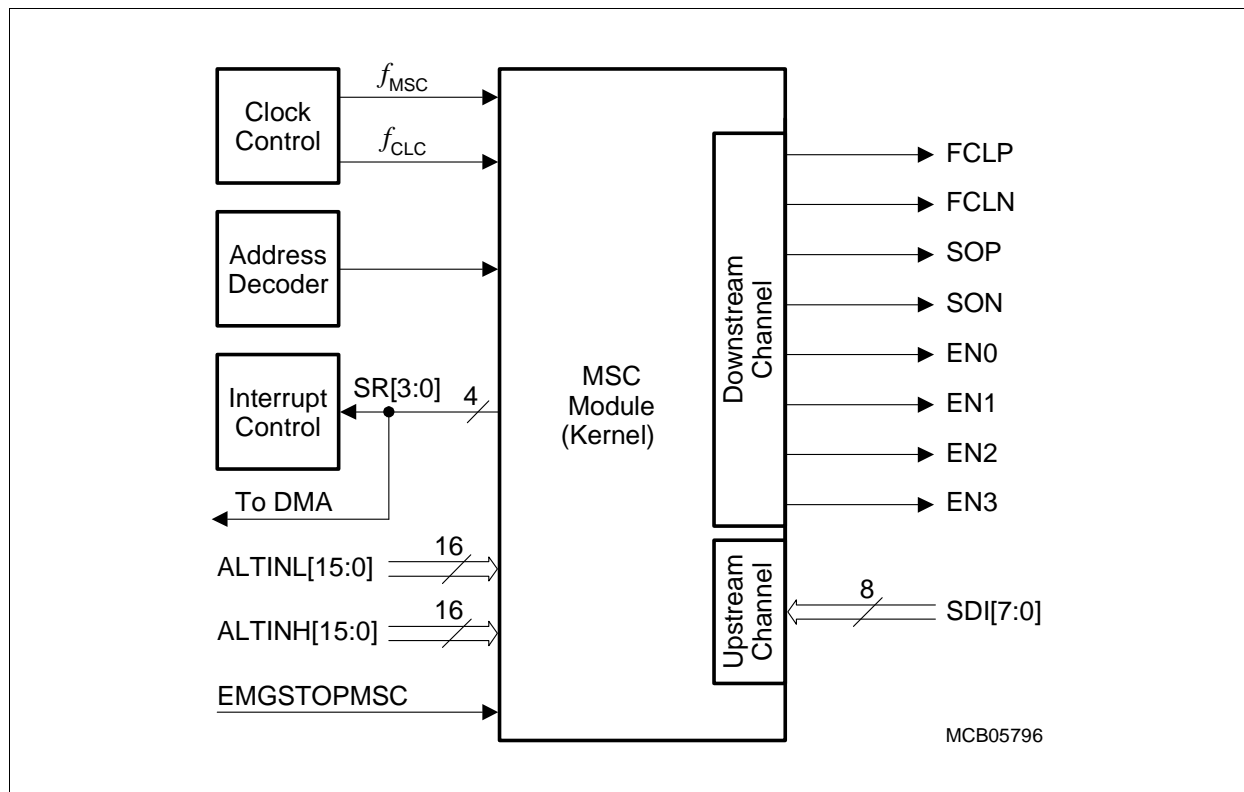


Figure 21-2 General Block Diagram of the MSC Interface

The downstream and upstream channels of the MSC module communicate with the external world via nine I/O lines. Eight output lines are required for the serial communication of the downstream channel (clock, data, and enable signals). One out of eight input lines $SDI[7:0]$ is used as serial data input signal for the upstream channel. The source of the serial data to be transmitted by the downstream channel can be MSC register contents or data that is provided at the ALTINL/ALTINH input lines. These input lines are typically connected to other on-chip peripheral units (for example with a timer unit such as the GPTA). An emergency stop input signal makes it possible to set bits of the serial data stream to dedicated values in emergency case.

Micro Second Channel (MSC)

Clock control, address decoding, and interrupt service request control are managed outside the MSC module kernel. Service request outputs are able to trigger an interrupt or a DMA request.

Features

- Fast synchronous serial interface to connect power switches in particular, or other peripheral devices via serial buses
- High-speed synchronous serial transmission on downstream channel
 - Serial output clock frequency: $f_{FCL} = f_{MSC}/2$
 - Fractional clock divider for precise frequency control of serial clock f_{MSC}
 - Command, data, and passive frame types
 - Start of serial frame: Software-controlled, timer-controlled, or free-running
 - Programmable upstream data frame length (16 or 12 bits)
 - Transmission with or without SEL bit
 - Flexible chip select generation indicates status during serial frame transmission
 - Emergency stop without CPU intervention
- Low-speed asynchronous serial reception on upstream channel
 - Baud rate: f_{MSC} divided by 4, 8, 16, 32, 64, 128, or 256
 - Standard asynchronous serial frames
 - Parity error checker
 - 8-to-1 input multiplexer for SDI lines
 - Built-in spike filter on SDI lines

21.1.2 Downstream Channel

The downstream channel performs a high-speed synchronous serial transmission of data to external devices. Its 32-bit shift register is divided into two 16-bit parts, SRH and SRL. Each bit of SRL and SRH can be selected to be delivered by the downstream data register DD, by the Downstream Command Register DC, or by two 16-bit wide input signal buses ALTINL and ALTINH.

Figure 21-3 is a diagram of the MSC downstream channel.

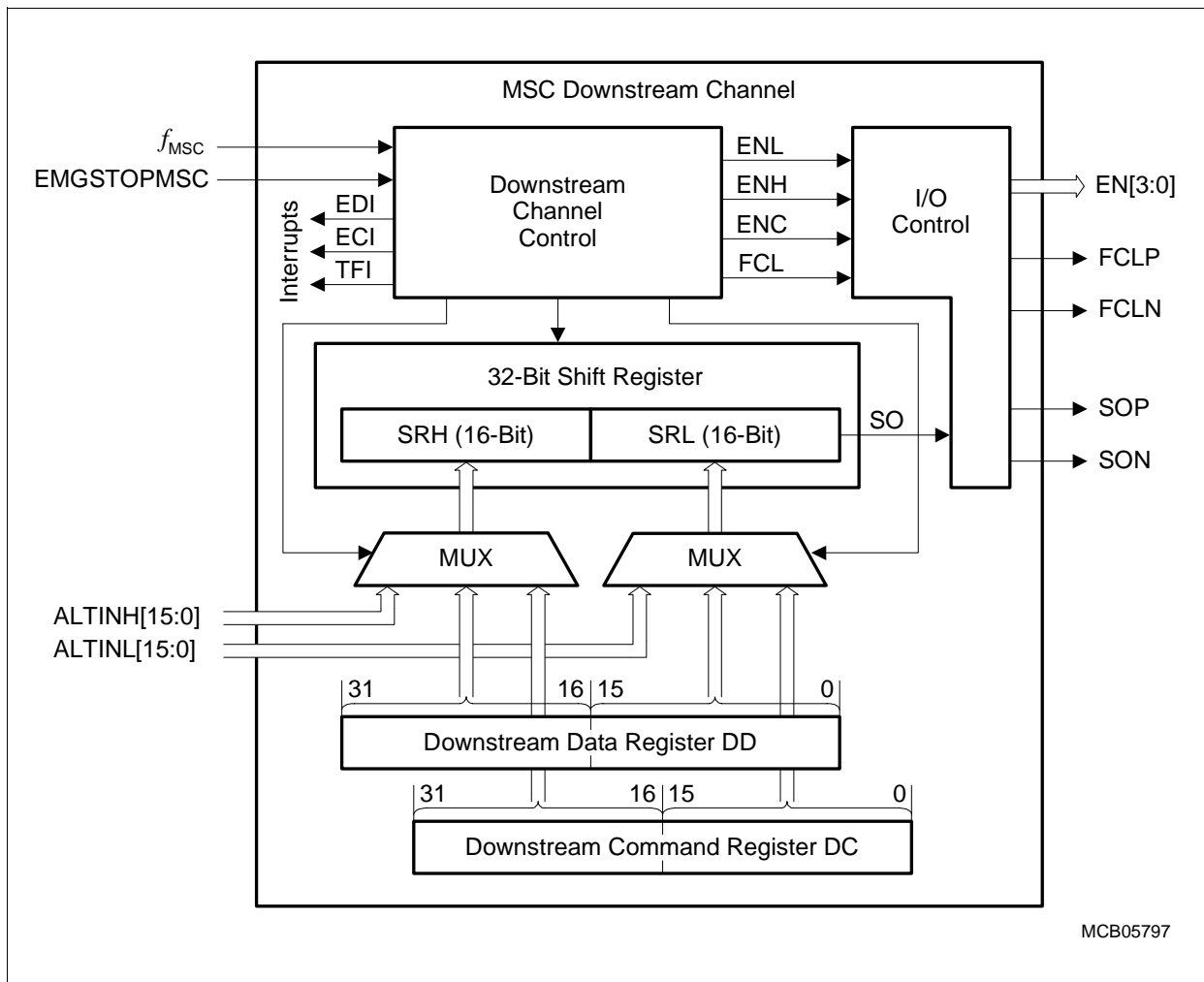


Figure 21-3 Downstream Channel Block Diagram

The enable signals ENL, ENH, and ENC indicate certain phases of the serial transmission in relation to the serial clock FCL. In the I/O control logic, these signals can be combined to four enable/select outputs EN[3:0]. For supporting differential output drivers, the serial clock output FCL and the serial data output SO are available in both polarities, indicated by the signal name suffix "P" and "N".

Micro Second Channel (MSC)

The emergency stop input line EMGSTOPMSC is used to indicate an emergency stop condition of a power device. In emergency case, shift register bits can be loaded bit-wise from the downstream data register instead from the ALTINL and ALTINH buses.

21.1.2.1 Frame Formats and Definitions

This section describes the frame formats and definitions of the MSC.

Basic Definitions

Figure 21-4 shows the layout and definitions of a downstream frame. A downstream frame is composed of an active phase and a passive phase. During the active phase, data transmission takes place and during the passive phase no data is transmitted at SO. The active phase is split into two parts: The SRL active phase in which the content of the shift register low part SRL is transmitted, and the SRH active phase in which the content of the shift register high part SRH is transmitted. At the beginning of the SRL and SRH active phase, a selection bit (SELL) can be optionally inserted into the serial data stream. In the frame shown in Figure 21-4, SELL is generated at the beginning of the SRL active phase (not for the SRH active phase). The least significant bits of SRL and SRH are sent out first.

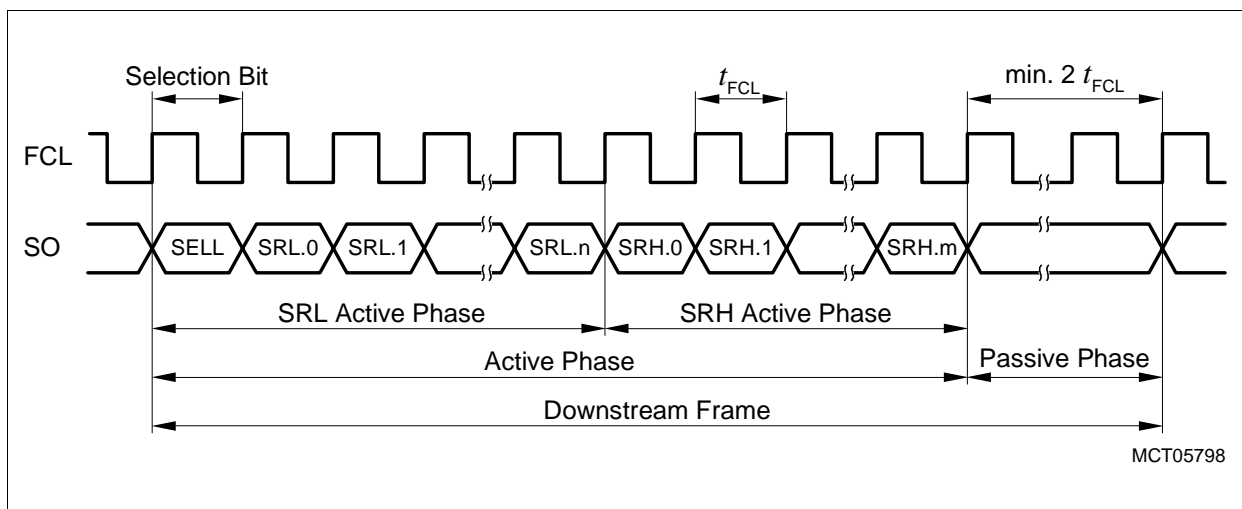


Figure 21-4 Downstream Channel Frame

The MSC downstream channel uses three types of frame formats for operation:

- Command frames, indicated by SELL = 1
- Data frames, indicated by SELL = 0 or SELL bit insertion disabled
- Passive time frame, indicated by ENL = ENH = 0

Command Frames

A command frame has two active phase parts, SRL active phase and SRH active phase. The command frame always starts with a high-level selection bit, independently whether the selection bit insertion (as defined by bit DSC.ENSELL) is enabled or not. The number of the bits transmitted during SRL and SRH active phases (except the selection bit) is defined by bit field DSC.NBC. SRL and SRH are combined to a 32-bit value whose length can be selected from 0 up to 32 bits. In other words, whenever bits of SRH are transmitted, they are always preceded by the transmission of the complete SRL content.

During the active phase of a command frame, the enable output signal ENC becomes active. The enable output signals ENL and ENH remain inactive.

The passive phase of a command frame always has a fixed length of $2 \times t_{FCL}$. The diagram shown in [Figure 21-4](#) assumes that the FCL clock is only generated during the active phase of the command frame ($OCR.CLKCTRL = 0$).

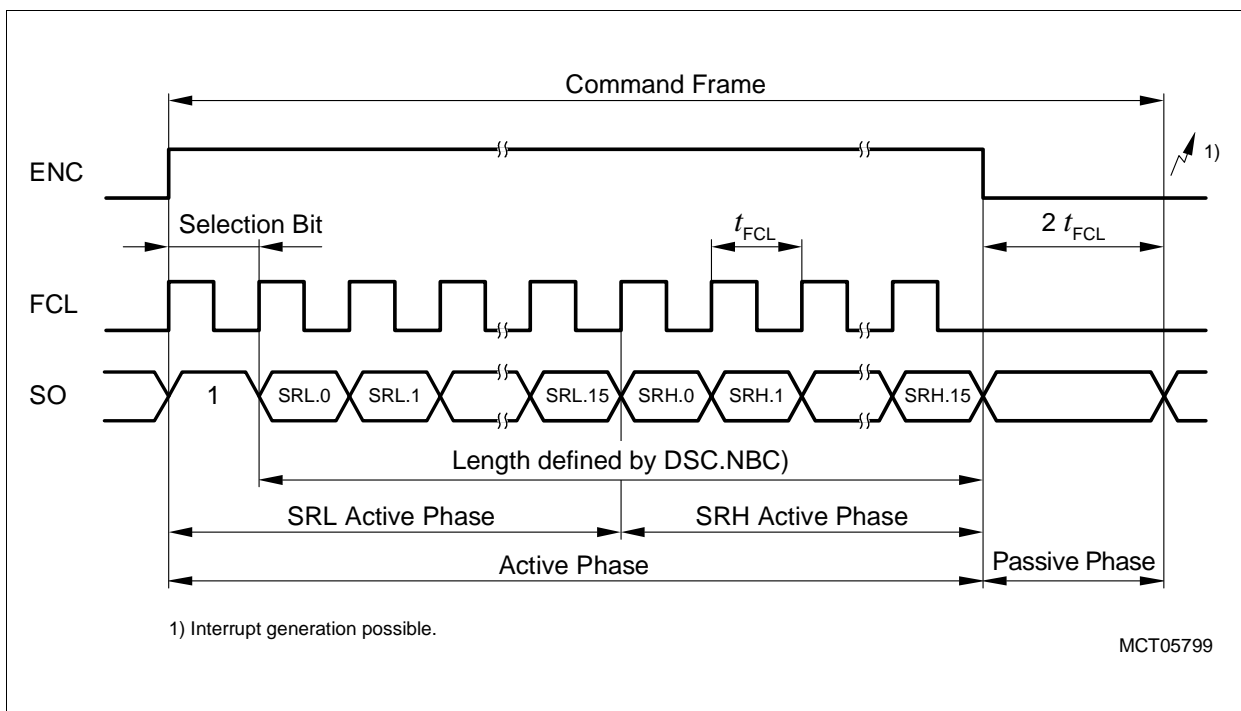


Figure 21-5 Command Frame Layout

Micro Second Channel (MSC)

Table 21-1 shows the programming of the bits to be transmitted and the resulting length of the complete command frame.

Table 21-1 Command Frame Length

DSC.NBC	SRL/SRH Bits that are Transmitted in Active Phase	Command Frame Length in t_{FCL} Periods
000000 _B	No bit shifted out	$1 + 0 + 2 = 3$
000001 _B	SRL[0] shifted out	$1 + 1 + 2 = 4$
000010 _B	SRL[1:0] shifted out	$1 + 2 + 2 = 5$
000011 _B	SRL[2:0] shifted out	$1 + 3 + 2 = 6$
...
001111 _B	SRL[14:0] shifted out	$1 + 15 + 2 = 18$
010000 _B	SRL[15:0] shifted out	$1 + 16 + 2 = 19$
010001 _B	SRL[15:0] and SRH[0] shifted out	$1 + 17 + 2 = 20$
010010 _B	SRL[15:0] and SRH[1:0] shifted out	$1 + 18 + 2 = 21$
010011 _B	SRL[15:0] and SRH[2:0] shifted out	$1 + 19 + 2 = 22$
...
011111 _B	SRL[15:0] and SRH[14:0] shifted out	$1 + 31 + 2 = 34$
100000 _B	SRL[15:0] and SRH[15:0] shifted out	$1 + 19 + 2 = 35$
Other NBC combinations	Reserved; do not use these bit combinations.	

Data Frames

A data frame has two active phase parts, SRL active phase and SRH active phase. The number of bits that are transmitted can be programmed separately for each of these two phases. Bit field DSC.NDBL determines the number of SRL bits that are transmitted during the SRL active phase and DSC.NDBH determines the number of SRH bits that are transmitted during the SRH active phase.

SRL and SRH active phases can start with a low-level selection bit when enabled by bits DSC.ENSELL or DSC.ENSELH.

During the SRL active phase of a data frame, the enable output signal ENL becomes active and during the SRH active phase of a data frame, the enable output signal ENH becomes active. The enable output signal ENC remains inactive.

The length of the data frame's passive phase is variable and is defined by bit field DSC.PPD. It can be within a range of $2 \times t_{FCL}$ up to $31 \times t_{FCL}$. The diagram shown in **Figure 21-5** assumes that the FCL clock is only generated during the active phase of the data frame (OCR.CLKCTRL = 0).

Table 21-2, **Table 21-3**, and **Table 21-4** show the definitions of the five data frame parameters that determine the layout of the data frame.

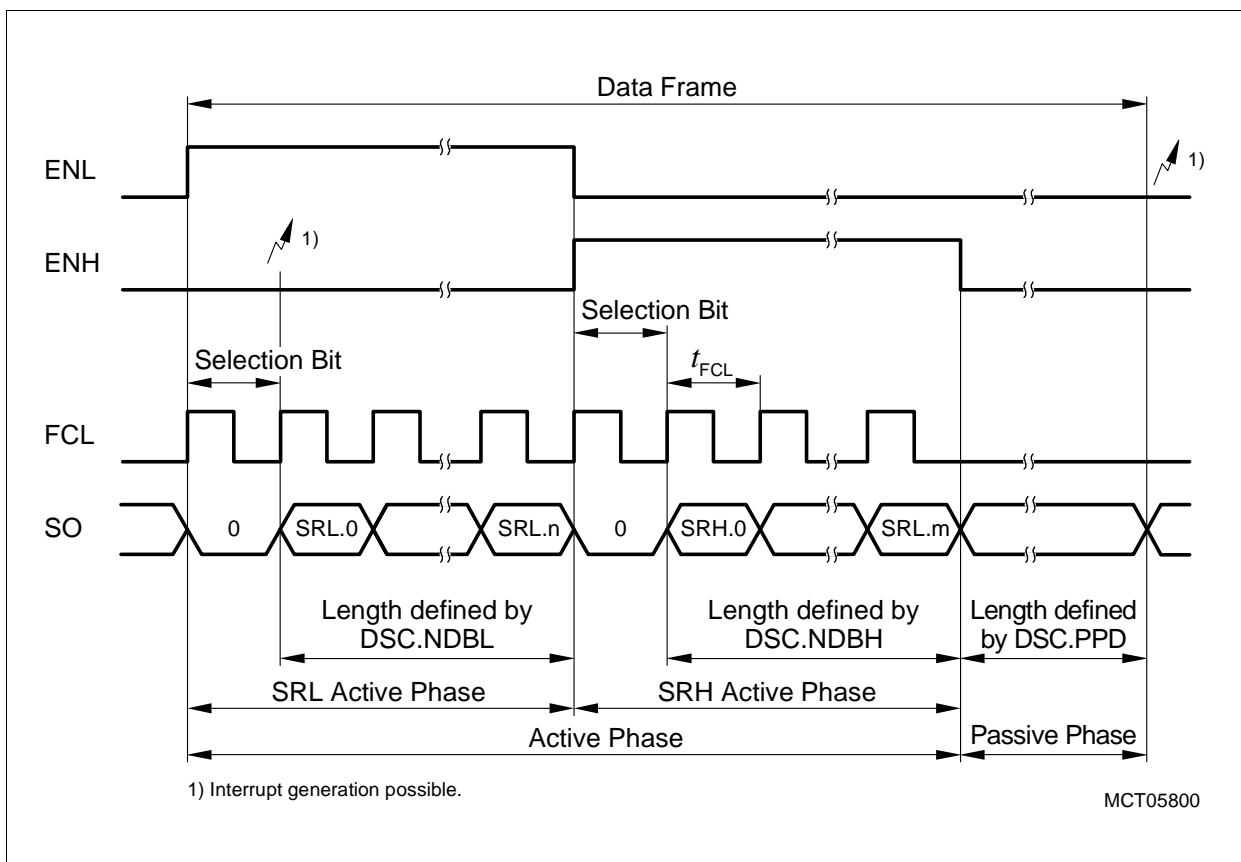


Figure 21-6 Data Frame Layout

Table 21-2 Data Frame Selection Bit Parameters

DSC.ENSELL	Selection Bit	DSC.ENSELH	Selection Bit
0	No selection bit inserted at the beginning of the SRL active phase	0	No selection bit inserted at the beginning of the SRH active phase
1	A low-level selection bit with is inserted at the beginning of the SRL active phase	1	A low-level selection bit is inserted at the beginning of the SRH active phase

Table 21-3 Data Frame SRL/SRH Length Parameters

DSC.NDBL	SRL Bits Transmitted in SRL Active Phase	DSC.NDBH	SRH Bits Transmitted in SRH Active Phase
00000 _B	No SRL bit transmitted	00000 _B	No SRH bit transmitted
00001 _B	SRL[0]	00001 _B	SRHL[0]
00010 _B	SRL[1:0]	00010 _B	SRH[1:0]
00011 _B	SRL[2:0]	00011 _B	SRH[2:0]
...
01111 _B	SRL[14:0]	01111 _B	SRH[14:0]
10000 _B	SRL[15:0]	10000 _B	SRH[15:0]
Other bit combinations	Reserved; do not use these bit combinations.	Other bit combinations	Reserved; do not use these bit combinations.

Table 21-4 Data Frame Passive Phase Length

DSC.PPD	Passive Phase Length
00000 _B	$2 \times t_{FCL}$
00001 _B	$2 \times t_{FCL}$
00010 _B	$2 \times t_{FCL}$
00011 _B	$3 \times t_{FCL}$
...	...
01110 _B	$30 \times t_{FCL}$
01111 _B	$31 \times t_{FCL}$

Micro Second Channel (MSC)

The following formula determines the number of t_{FCL} cycles of a data frame: All parameters (bits and bit fields) are located in register DSC.

$$\text{Number of cycles} = \text{ENSELL} + \text{NDBL} + \text{ENSELH} + \text{NDBH} + \text{PPD} \quad (21.1)$$

Note that in the formula above, PPD must be set to 2 when $\text{DSC.PPD} \leq 00010_B$.

Passive Time Frames

A passive time frame has the length defined by the five data frame parameters according [Equation \(21.1\)](#). They are generated only in Data Repetition Mode. Under special conditions (command frame insertion), passive time frames can be shortened (see [Figure 21-9](#)).

During passive time frames, the data output SO have to be considered as invalid at the receiving device and the clock output FCL may toggle or not (as selected by bit OCR.CLKCTRL). The ENL and ENH enable signals remain at low level during a passive time frame.

21.1.2.2 Shift Register Operation

This section describes the SRL and SRH shift register loading.

SRL Shift Register Loading

During the SRL/SRH shift register load operation at the beginning of each downstream frame transmission, several parameters determine which information is loaded into the bits of the shift register. **Figure 21-7** shows the logic that is implemented for the SRL shift register loading operation. The logic for the SRH shift register loading operation is equivalent to the one for the SRL register. Its differences in data sources and register controls are described later in this section.

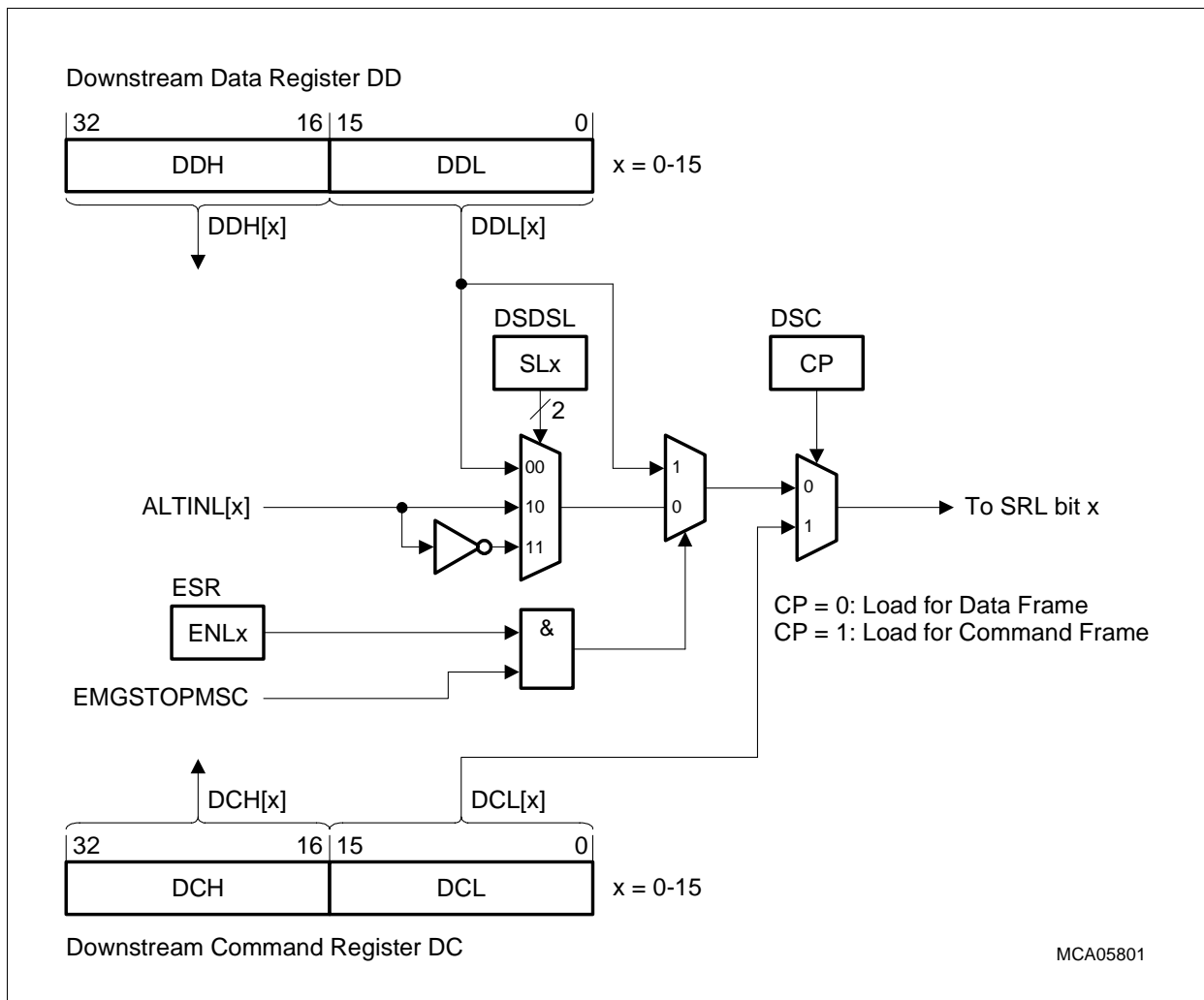


Figure 21-7 SRL Shift Register Data Loading Control

Micro Second Channel (MSC)

Four data sources can be selected for each SRL bit by using several control bits and one control signal:

- ALTINL input line (non-inverted)
- ALTINL input line (inverted)
- Bit of DD.DDL (downstream data register)
- Bit of DC.DCL (downstream control register)

When SRL is loaded for data frame transmission (DSC.CP = 0), bit fields DSDSL.SLx determine bit-wise which data is loaded into SRL bit x. The data source selection as controlled by DSDSL.SLx will only be effective when EMGSTOPMSC is inactive (at low level). When EMGSTOPMSC = 1 (active) during the load operation, all SRL[x] bits that are enabled for the emergency stop feature (bit ESR.ENLx = 1) are loaded directly with the corresponding bit DDL[x] of the downstream data register DD.

When SRL is loaded for command frame transmission (DSC.CP = 1), always the lower 16-bit part DCL of the downstream control register is loaded completely into SRL.

Table 21-5 summarizes all SRL data source selection capabilities (x = 0-15).

Table 21-5 SRL Data Source Selection Capabilities

DSC.CP	DSDSL.SLx	ESR.ENLx	EMGSTOPMSC	Selection
0	00 _B	0	–	Bit DD.DDL[x] is loaded into SRL[x].
	01 _B			Reserved.
	10 _B			State of ALTINL[x] input is loaded into SRL[x].
	11 _B			Inverted state of ALTINL[x] input is loaded into SRL[x].
	XX _B	1	1	Bit DD.DDL[x] is loaded into SRL[x].
1	XX _B	X	X	Bit fields DCL and DCH are completely loaded into SRL and SRH, respectively.

SRH Shift Register Loading

The SRH shift register load operation is equivalent to the SRL shift register load operation. The following differences must be taken into account for SRH shift register loading:

- Input lines ALTINH are connected instead of ALTINL input lines.
- DSDSH register bits control data source selection instead of DSDSL register.
- Emergency stop is enabled by ESR.ENHx bits instead of ESR.ENLx bits.
- Bits of the downstream data register high part DDH are selected instead of DDL.
- Downstream control register high part DCH is selected instead of DDL.

21.1.2.3 Transmission Modes

The downstream channel of the MSC makes it possible to select between two transmission modes:

- Triggered Mode, selected by DSC.TM = 0, or
- Data Repetition Mode, selected by DSC.TM = 1

Triggered Mode

In Triggered Mode, command frames or data frames are sent out as a result of a software event. When a frame transmission has been finished and no further frame transmission has been requested, the downstream channel returns to idle state and waits for the next frame transmission to be triggered by software.

When the Downstream Command Register DC is written, the command pending bit DSC.CP becomes set and a command frame will be immediately started and sent out if the downstream channel is idle. If a data or command frame is currently processed and output, the command frame transmission is delayed, and then started when the active downstream frame has been finished. The command pending bit DSC.CP becomes cleared by hardware when the first bit of the command frame is sent out.

If the downstream channel is idle and the data pending bit DSC.DP is set by writing bit ISC.SDP with 1, a data frame will be immediately started and sent out if the downstream channel is idle. If a data frame or a command frame is currently processed and output, the data frame transmission is delayed and started when the active downstream frame has been finished. The data pending bit DSC.DP becomes cleared by hardware when the first bit of the data frame is sent out.

A command frame always has priority over the data frame. This means that if both frame pending bits are set (DSC.DP = DSC.CP = 1), the command frame will always be sent first. Therefore, a pending data frame transmission will be delayed as long as no further command frame transmission is running or requested.

Figure 21-8 is a flow diagram of the Triggered Mode. This diagram especially shows the behavior of the data and command pending bits DSC.DP and DSC.CP. If both frame pending bits are set (DSC.DP = DSC.CP = 1), the command frame will always be sent first, followed by the data frame (assuming no further command frame has been requested).

The type of the active frame that is currently processed and output is indicated by two status flags: DSS.DFA is set during a data frame transmission and DSS.CFA is set during a command frame transmission. Further, the downstream counter DSS.DC indicates the number of shift clock periods that have been elapsed since the start of the current frame.

In Triggered Mode, the shift register loading event as described in [Section 21.1.2.2](#) occurs just before a command or data frame transmission is started.

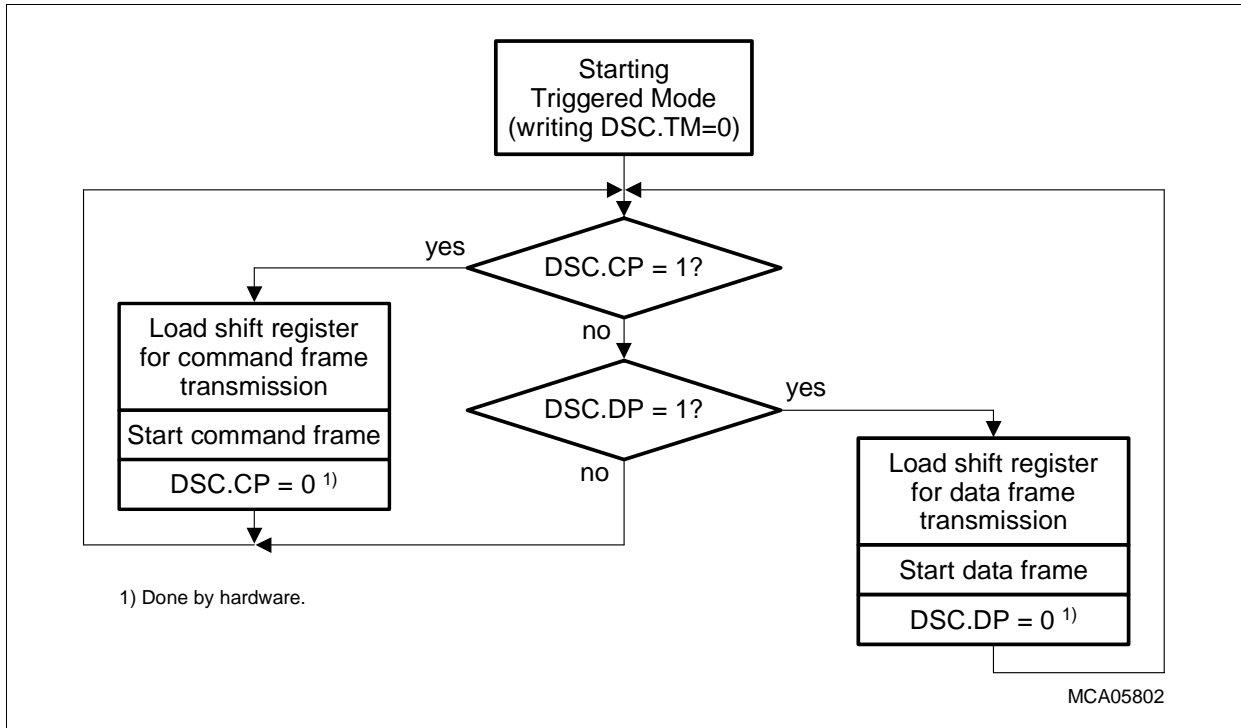


Figure 21-8 Triggered Mode Flow Diagram

Data Repetition Mode

In Data Repetition Mode, data frames are sent out continuously without any software interaction. In the time gap between two consecutive data frames, passive time frames can be inserted. The number of passive time frames to be inserted (0 to 15) is defined by bit field DSS.NPTF. The duration of data frame (t_{DF}) and passive time frames (t_{PTF}) is determined by the five data frame parameters (see [Equation \(21.1\)](#)). These parameters determine time reference points (TRPs) at which a data or passive time frames is started (see diagram A in [Figure 21-9](#)).

The automatic data frame generation is controlled by the data pending bit DSC.DP. This bit is set near the end of the last transmitted passive time frame. At the next TRP, a data frame is started (if no command frame has been requested) and DSC.DP is cleared again by hardware after the data frame has been started. Data Frames are always aligned to time reference points. This means they always start at a TRP. Passive time frames can be shortened. This is especially the case when command frames are inserted.

Continuous data frame transmission can be interrupted by insertion of command frames. Command frames are initiated by software. When the downstream control register DSC is written, the command pending bit DSC.CP is set by hardware. CP = 1 indicates that the MSC starts a command frame at the next TRP, independent of whether a data frame (indicated by DSC.DP = 1) or passive time frame should be started with the next TRP. This means that also command frames are always aligned to time reference points.

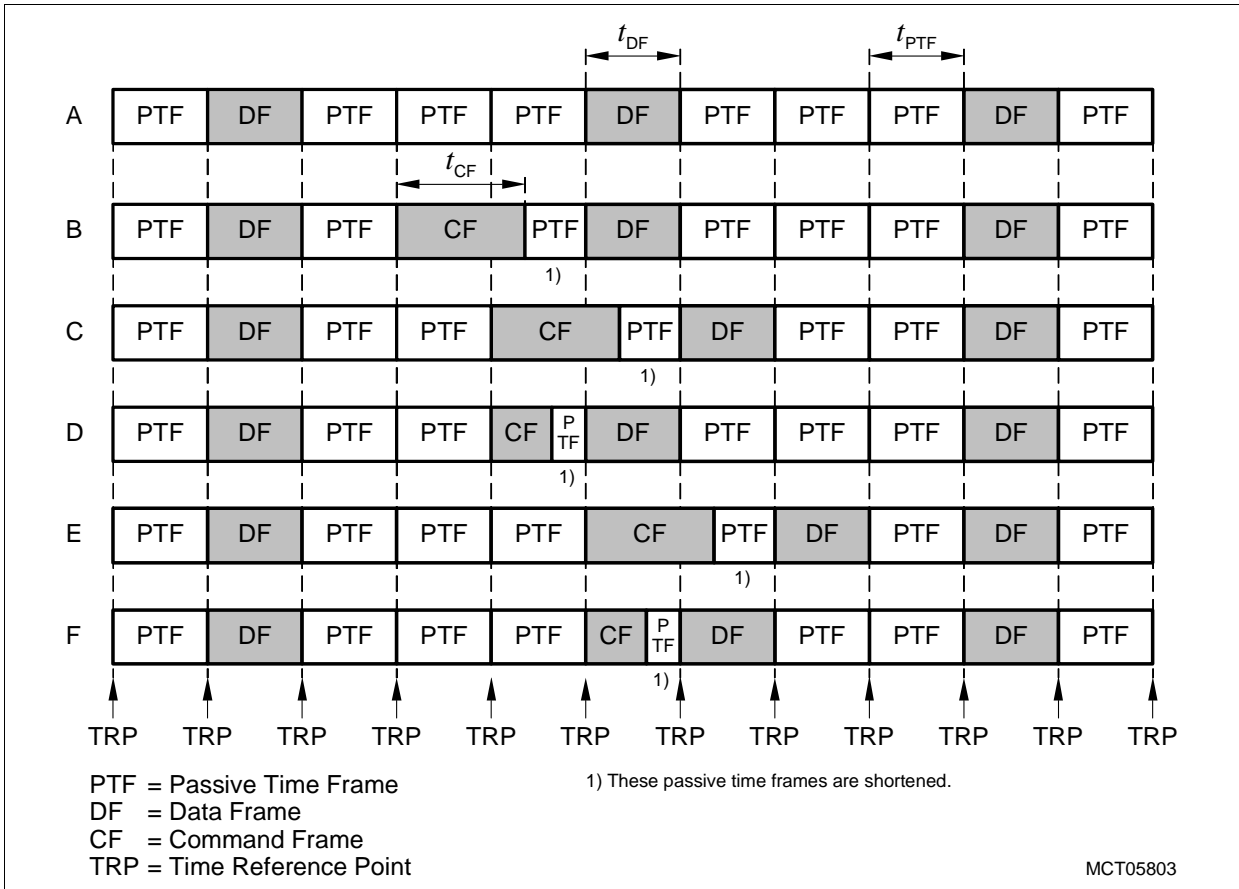


Figure 21-9 Data Repetition Mode Frame Examples with DSS.NPTF = 0011_B

Diagrams B to F in **Figure 21-9** show the command frame insertion in Data Repetition Mode.

In diagram B, a command frame has been requested during the first passive time frame after the data frame, and is inserted at the next TRP. In diagram C and D, a command frame has been requested during the second passive time frame, and is inserted at the time reference point of the last nominal passive time frame.

When the length of a command frame has not the same length of the data frame (as in diagrams B to F), a shortened passive time frame is inserted until the next TRP is reached. This assures that the next data or normal passive time frame is again aligned to a TRP.

Figure 21-10 is a flow diagram of the Data Repetition Mode. This diagram especially shows the behavior of the data and command pending bits DSC.DP and DSC.CP. If both frame pending bits are set (DSC.DP = DSC.CP = 1), the command frame will always be sent first, followed by the data frame when the next TRP is reached (assuming no further command frame has been requested).

When the last passive frame is transmitted, DSC.DP becomes set by hardware. This triggers the start of a data frame when the next TRP is reached.

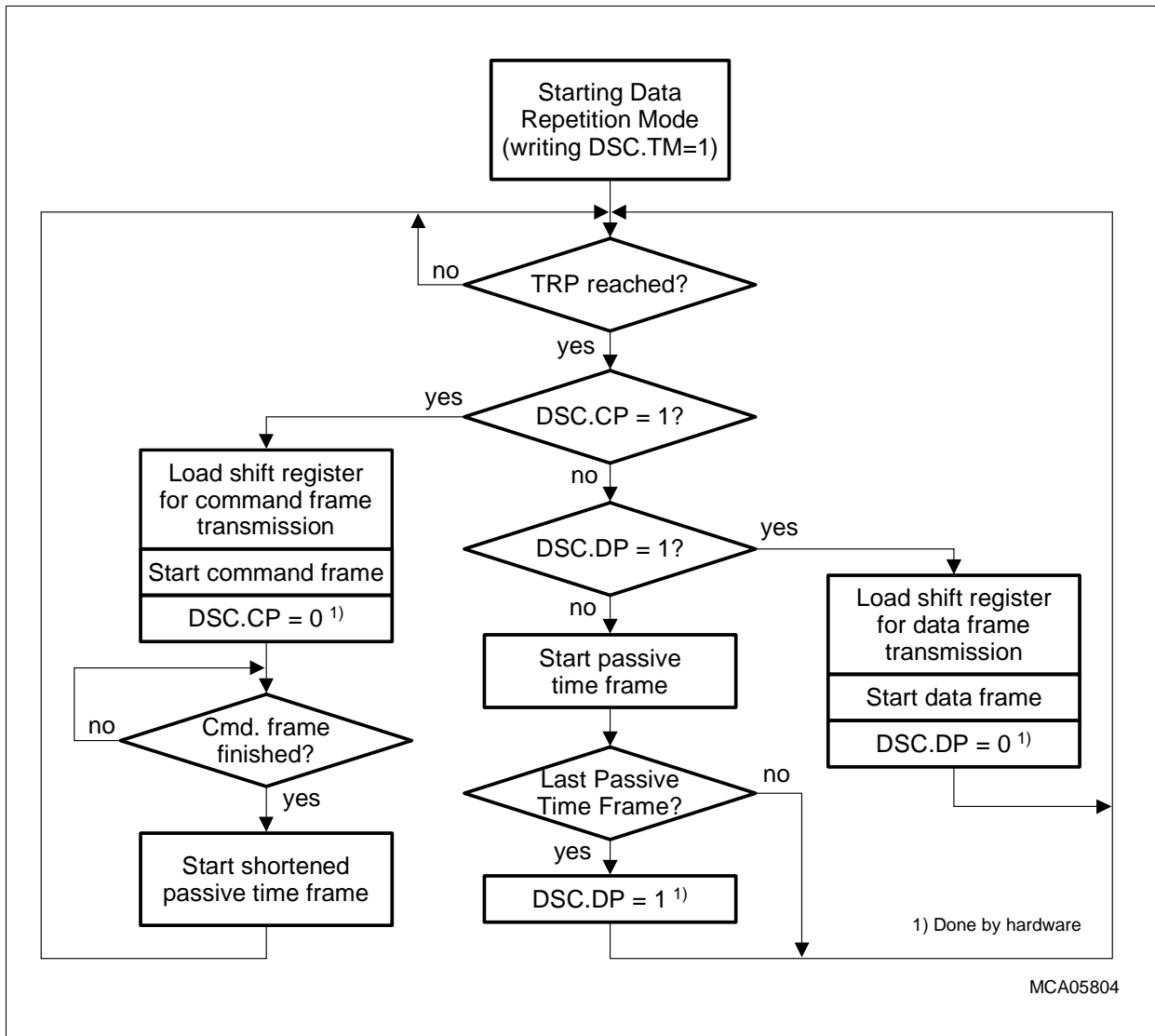


Figure 21-10 Data Repetition Mode Flow Diagram

The type of the active frame (data or command frame) that is currently processed and output is indicated by two status flags: DSS.DFA is set during a data frame transmission and DSS.CFA is set during a command frame transmission. Further, the downstream counter DSS.DC indicates the number of shift clock periods that have been elapsed since the start of the current data, command, or passive time frame.

As in Triggered Mode, the shift register loading event described in [Section 21.1.2.2](#) occurs in Data Repetition Mode just before a TRP, i.e. shortly before a command or data frame transmission is started.

Passive Frame Counter in Data Repetition Mode

In Data Repetition Mode, a passive time frame counter DSS.DC indicates how many time frames have been already transmitted after the last regular data frame occurrence. The passive time frame counter counts up from 0000_B to the value which has been written into bit field DSS.NPTF (number of passive time frames). $DSS.PFC = 0000_B$ indicates that a data frame is requested for transmission.

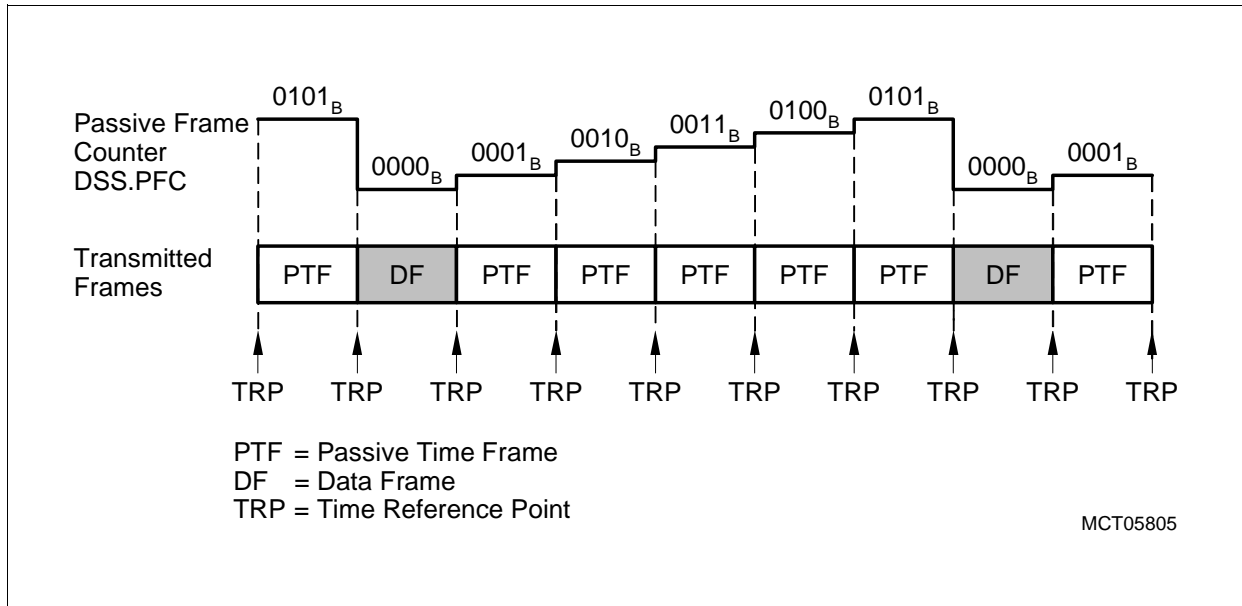


Figure 21-11 Passive Frame Counter Operation (with $DSS.NPTF = 0101_B$)

21.1.2.4 Downstream Counter and Enable Signals

During downstream channel operation, a 7-bit downstream counter DSS.DC is counting FCL shift clock periods. With the loading of the shift register, the downstream counter is reset to 00_H and started for counting up to the end of the downstream frame (end of passive phase).

In Triggered Mode, the downstream counter stops counting at the end of the passive phase and waits until a new downstream frame is started.

In repetition mode, the downstream counter does not stop at the end of the passive phase but is reset and starts counting up again with the next frame, independently whether a data frame, command frame, or passive time frame is started as next frame.

Figure 21-12 shows an example of downstream channel data frame transmission. In this example, the selection bit for the SRL active frame is enabled ($ENSELL = 1$), and the selection bit for the SRH active frame is disabled ($ENSELL = 0$). With loading of the shift register SRL/SRH, the downstream counter is reset and then starts counting up with each FCL clock until the end of the passive phase is reached. ENL is set to high level at the beginning of the SRL active frame selection bit.

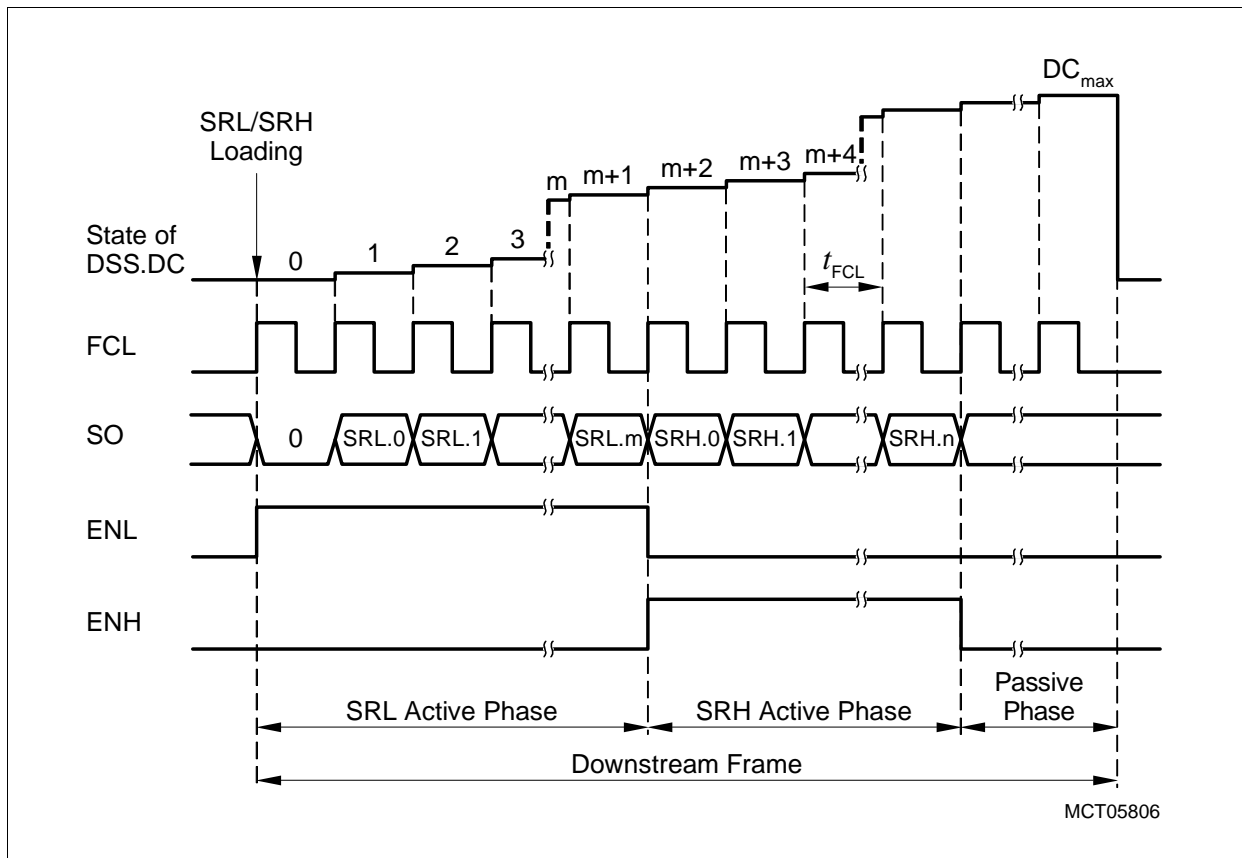


Figure 21-12 Shift Clock Counting: Data Frame with $ENSELL = 1$ and $ENSELH = 0$

Micro Second Channel (MSC)

When the selection bit for the SRL active frame is disabled ($ENSELL = 0$, see [Figure 21-13](#)), the loading of the shift register SRL/SRH (and reset of the downstream counter) occurs one FCL clock cycle before the first data bit SRL.0 is output. ENL is set to high level with the beginning of the first data bit SRL.0.

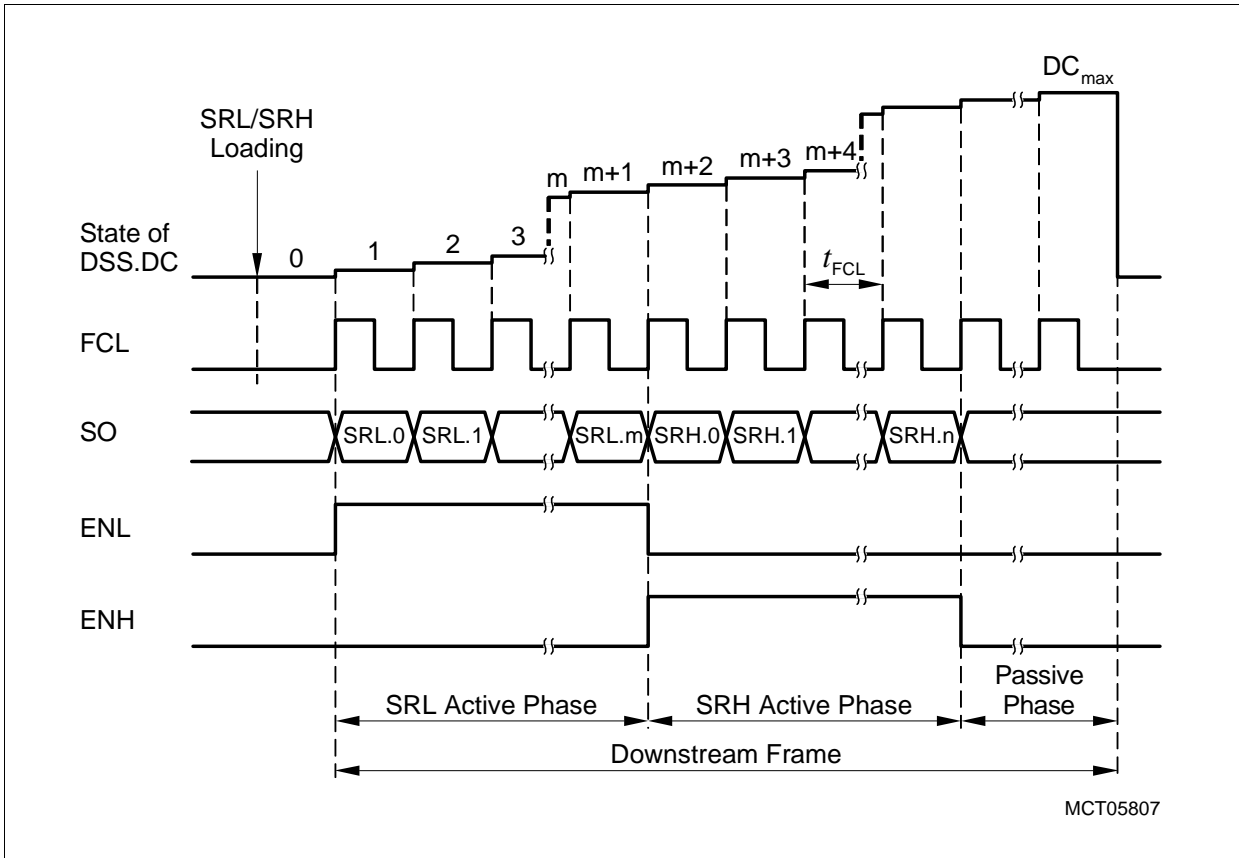


Figure 21-13 Shift Clock Counting: Data Frame with $ENSELL = 0$ and $ENSELH = 0$

21.1.2.5 Baud Rate

The baud rate of the downstream channel's serial transmission is defined by the frequency of the serial clock FCL, and is always $f_{MSC}/2$. The f_{MSC} generation is device specific and depends on the implementation of the MSC module. The TC1796 specific clock generation is described on [Page 21-65](#).

21.1.2.6 Abort of Frames

Only a reset condition of the device can abort a current transmission. The MSC module does not start a new frame transmission when the downstream channel becomes disabled, the suspend mode is requested, or the sleep mode is entered. If one of these three conditions becomes active during a running frame transmission, the frame transmission is completely finished before the requested abort state is entered. Note that in this case no time frame finished interrupt is generated any more.

21.1.3 Upstream Channel

The MSC upstream channel is an asynchronous serial receiver based on the standard asynchronous data transfer protocol. It is dedicated to receive a serial data stream from a peripheral device via its serial data input SDI, using two specific data frame formats.

Figure 21-14 is a block diagram of the MSC upstream channel.

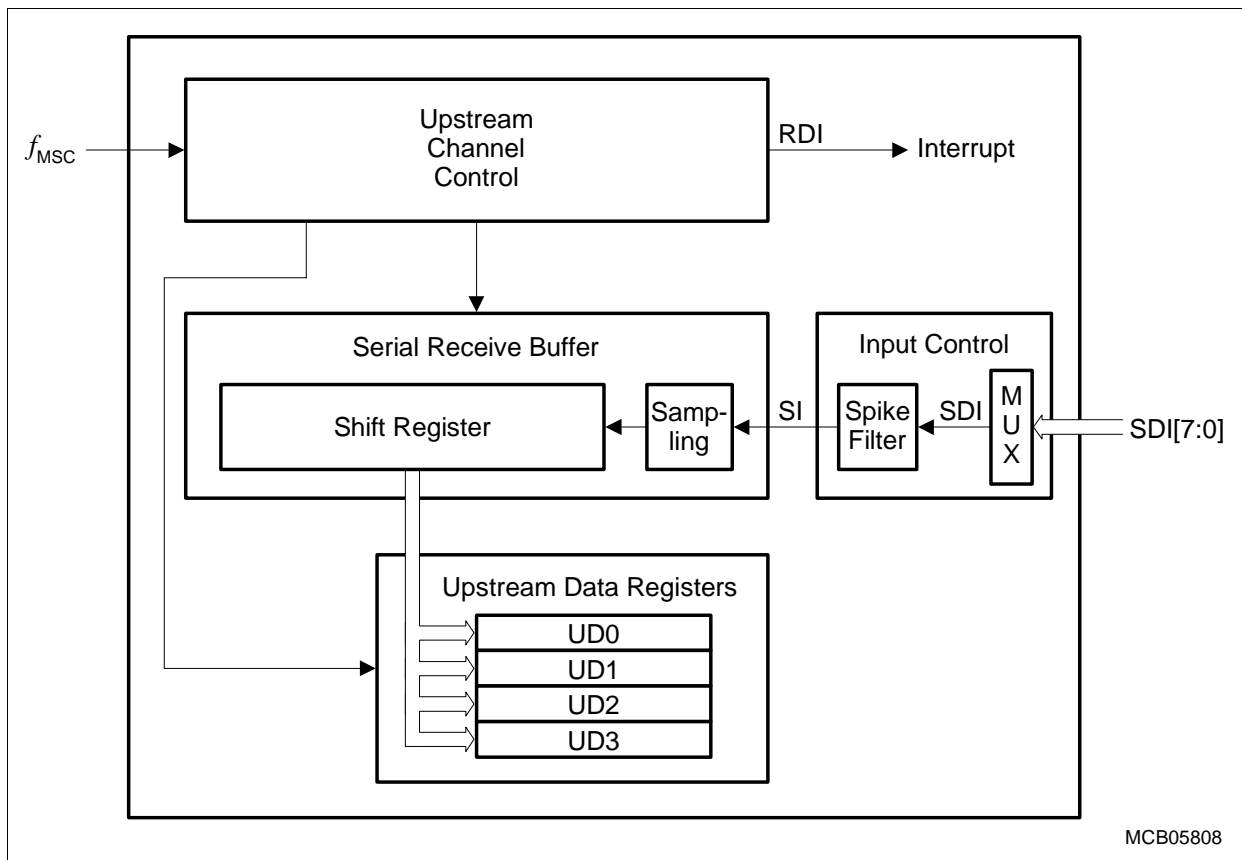


Figure 21-14 Upstream Channel Block Diagram

The incoming data at SI is sampled after it has been filtered for spikes. The detected logic states of the serial input are clocked into a shift register. After the complete reception of the serial data frame, the content of the shift register is transferred into one of the four data registers, and an interrupt can be generated optionally.

The receive baud rate is directly coupled to the module clock f_{MSC} , and can be within a range of $f_{MSC}/4$ up to $f_{MSC}/256$.

21.1.3.1 Data Frames

The asynchronous data frames used by the upstream channel include four basic parts:

1. One start bit, always at low level
2. An 8-bit data field D[7:0] with LSB first
3. An optional 4-bit address field A[3:0] with LSB first
4. One parity bit and two stop bits, that are always at high level

As shown in **Figure 21-15**, the 16-bit upstream data frame includes an additional 4-bit address field. The upstream frame type is selected by bit USR.UFT.

- USR.UFT = 0: 12-bit upstream data frame selected
- USR.UFT = 1: 16-bit upstream data frame with 4-bit address field selected

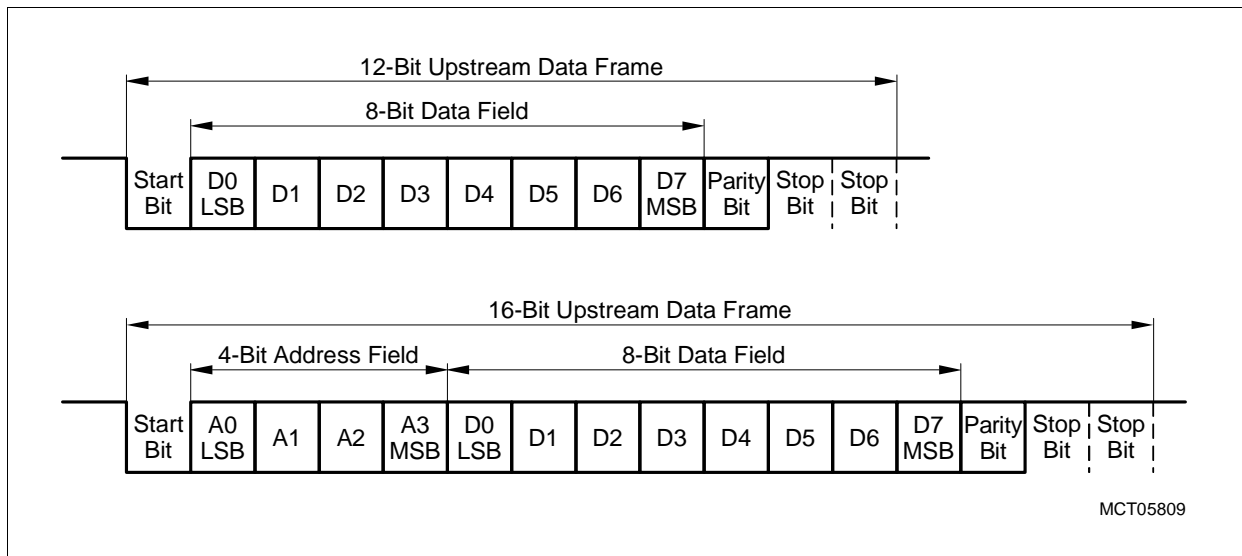


Figure 21-15 Upstream Channel Frame Types

21.1.3.2 Parity Checking

The incoming parity bit of the data frames can be checked by the upstream channel. When a parity error is detected, the parity error flag PERR in the related Upstream Data Register UDx is set. Note that a setting of the parity error flag PERR does not generate an interrupt. The PERR bits must be checked by software. Further, the UDx registers store the parity bit of the incoming data frame (UDx.P) and the parity bit that is generated internally (UDx.IPF).

Bit USR.PCTR determines the parity mode, even or odd, that is selected for parity checking. With USR.PCTR = 0, even parity mode is selected. Even parity means that the parity bit is set on an odd number of 1s in the data field (12-bit upstream data frame) or in the address plus data field (16-bit upstream data frame). With USR.PCTR = 1, odd parity mode is selected. In odd parity mode, the parity bit is set on an even number of 1s of the related data.

The parity checking logic in the upstream channel also controls whether start bit and the two stop bits of the upstream data frame are at correct logic level. If the start bit is not at low level and the two stop bits are not at high level at the end of the frame reception, the parity error flag UDx.PERR is set, too.

21.1.3.3 Data Reception

The reception of the upstream frame is started with a falling edge (1-to-0 transition) on the SI line. When the start bit is detected, serial reception is enabled and the receive circuit begins to sample the incoming serial data and to buffer it in the receive buffer. After the second stop bit has been detected, the content of the receive buffer is transferred to one of four upstream data registers UDx. The receive circuit then waits for the next start bit (1-to-0 transition) at the SI line. When the content of the receive buffer has been transferred to UDx, the valid bit UDx.V is set by hardware, and a receive interrupt can be generated.

Note: The SI input line is the filtered non-inverted (OCR.ILP = 0) or inverted (OCR.ILP = 1) SDI input signal. The SI input signal selection is described on [Page 21-30](#).

Frame Reception with Address Field

Frame reception for a 16-bit data frame (see [Figure 21-16](#)) is selected by USR.UFT = 1. When the content of the receive buffer has been received completely, it is transferred to one of the four UDx registers. The two most significant address bits A[3:2] of the received 4-bit address field select the number x of register UDx in which the received frame content is stored. Register UDx is loaded with the two least significant address bits A0 and A1 (UDx.LABF), the 8-bit data (UDx.DATA), the received parity bit (UDx.P), the calculated parity bit (UDx.IPF), and the parity checking result (UDx.PERR). Finally, the valid bit UDx.V is set to indicate that the UDx register contains valid data.

The current state of the frame reception is indicated by the content of an upstream counter that is readable via bit field USR.UC. The upstream counter is a 5-bit counter that counts the upstream frame bits during reception. As shown in [Figure 21-16](#), the upstream counter is loaded with 10000_B at the detection of a start bit. It counts down and is again at 00000_B when the second stop bit has been detected and the frame reception is finished.

The state of the serial input data line SI is sampled in the middle of a bit cell and shifted into the receive buffer at the end of the bit cell. The frequency of the shift clock f_{SHIFT} depends the selected baud rate (see [Page 21-25](#)).

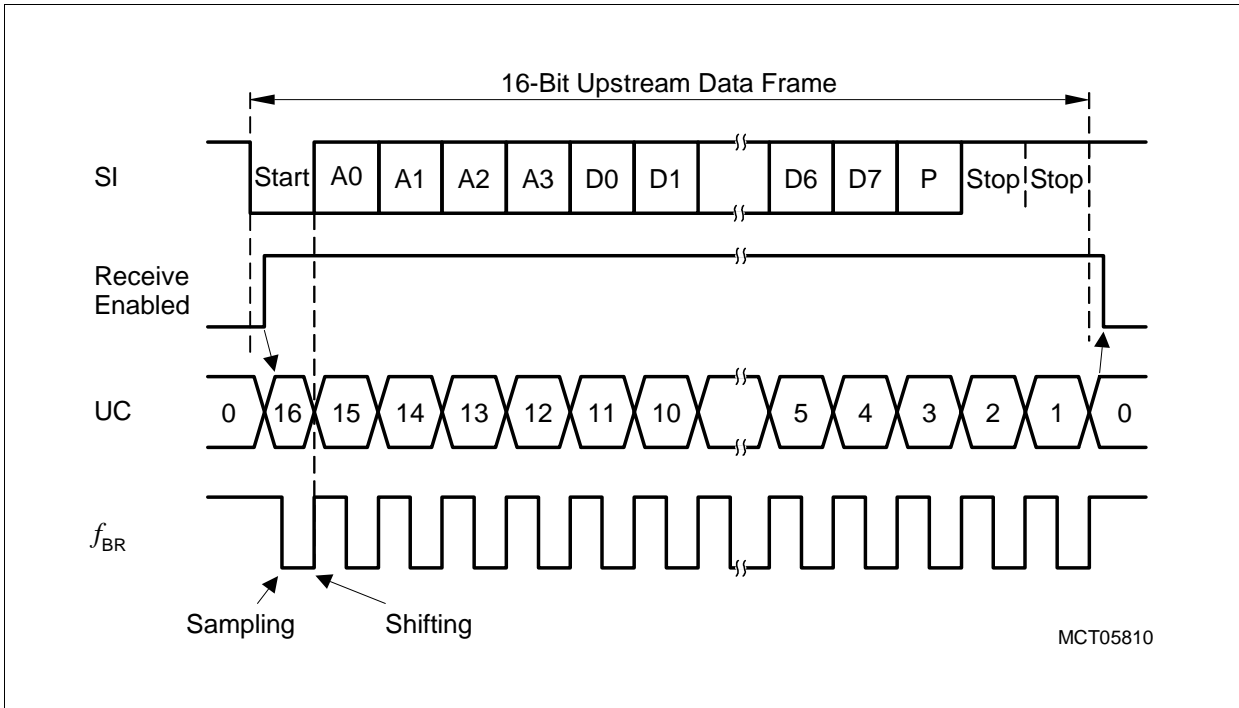


Figure 21-16 16-Bit Upstream Reception

Data Reception without Address Field

Frame reception for a 12-bit data frame is selected by `USR.UFT = 0`. The reception scheme is comparable with that of the 16-bit data frame reception but there are a few differences:

- The upstream counter `USR.UC` is initially loaded with 01100_B .
- The received frame content is always stored in register `UD0`.
- Bit field `UD0.LABF` is always loaded with 00_B when the frame is stored.

21.1.3.4 Baud Rate

The baud rate of the upstream channel is derived from the MSC module clock f_{MSC} . **Figure 21-17** shows the configuration of the upstream channel clock circuitry.

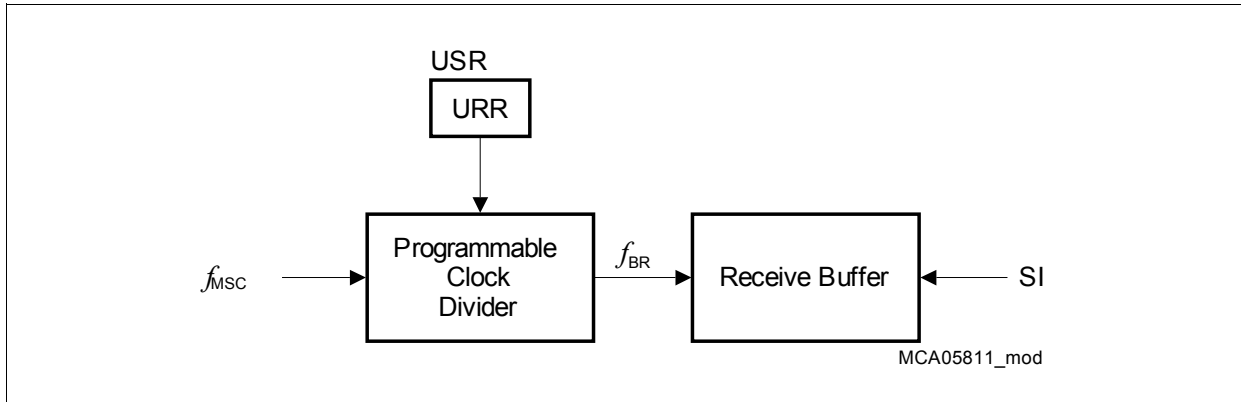


Figure 21-17 Upstream Channel Clock Circuitry

The serial data input SI is evaluated with the baud rate clock f_{BR} in the middle of each bit cell, and latched in case of a data bit. The baud rate clock f_{BR} is derived from f_{MSC} by a programmable clock divider. The frequency of f_{BR} determines the width of a received bit cell and therefore the baud rate for the receive data. The content of bit field USR.URR selects the baud rate according **Table 21-6**. The resulting baud rate formula is:

$$\text{Baud rate}_{\text{MSC Upstream Channel}} = \frac{f_{MSC}}{DF} \quad (21.2)$$

Table 21-6 Upstream Channel Divide Factor DF Selection & Baud Rate

USR.URR	Divide Factor DF	Baud Rate
000 _B	reception disabled	–
001 _B	4	$f_{MSC}/4$
010 _B	8	$f_{MSC}/8$
010 _B	16	$f_{MSC}/16$
100 _B	32	$f_{MSC}/32$
101 _B	64	$f_{MSC}/64$
110 _B	128	$f_{MSC}/128$
111 _B	256	$f_{MSC}/256$

Note: With the USR.URR = 000_B the upstream channel is disabled and data reception is not possible.

Micro Second Channel (MSC)

The content of bit field USR.URR determines the operation of an internal sampling reload counter that is clocked with f_{MSC} . **Figure 21-18** shows the operation of the sampling counter at the beginning of an upstream frame with a divide factor DF of 8 (OCSR.URR = 010_B is equal to DF = 8) which means eight sampling clocks per each frame bit cell.

When the upstream channel is in idle state it waits for a falling edge (1-to-0 transition) at SI. Therefore, the sample counter starts counting up and is reset when the selected divide factor DF shown in **Table 21-6** is reached. In the middle of the sampling counter's count range, the logic state at SI is evaluated and, in case of a data bit, latched in the receive buffer's shift register. With the reload of the sampling counter, the shift register is shifted by one bit position.

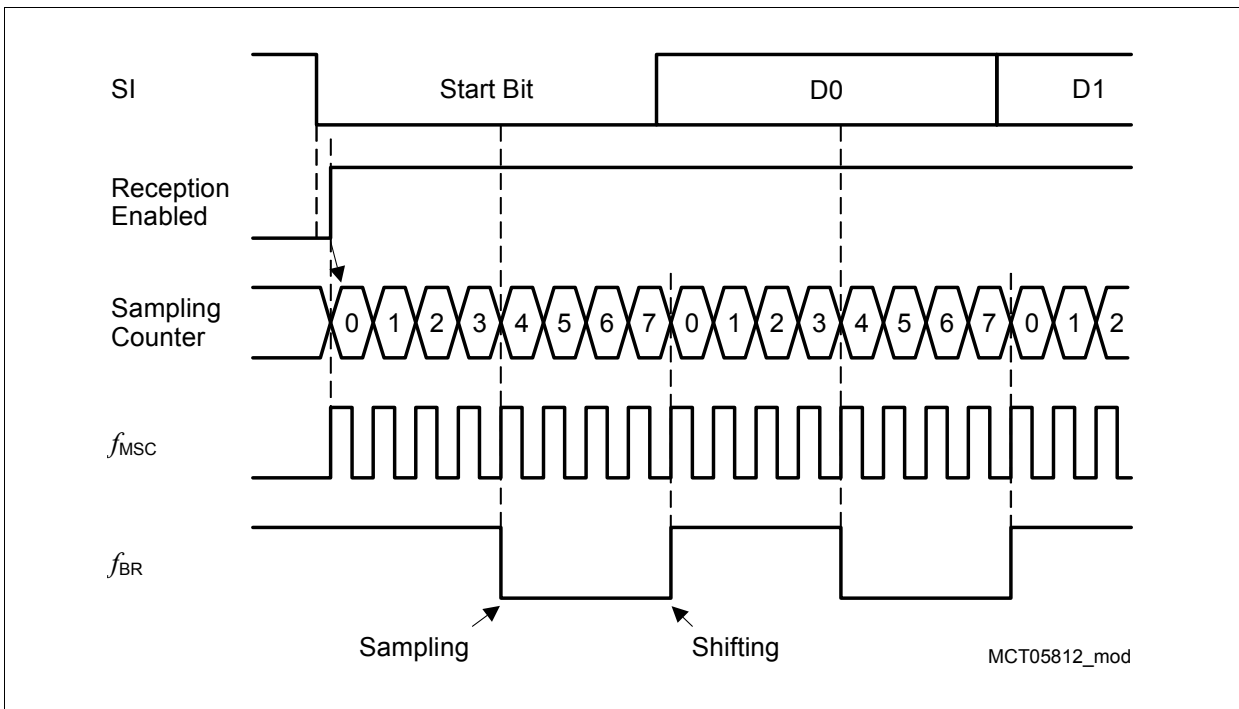


Figure 21-18 Upstream Channel Sampling with URR = 010_B

21.1.3.5 Spike Filter

The upstream channel input line SDI is sampled using a built-in spike filter with synchronization stage, both clocked with f_{MSC} . The spike filter is a chain of flip-flops with a majority decision logic (2 out of 3). A sampled value that is found at least twice in three samples is taken as data input value for SI.

21.1.4 I/O Control

The types of I/O control logic for the MSC module I/O lines are shown in **Figure 21-19**. The downstream channel generates five output signals that control eight MSC module outputs, split into four chip select outputs, two clock outputs, and two serial data outputs. The upstream channel has one input signal.

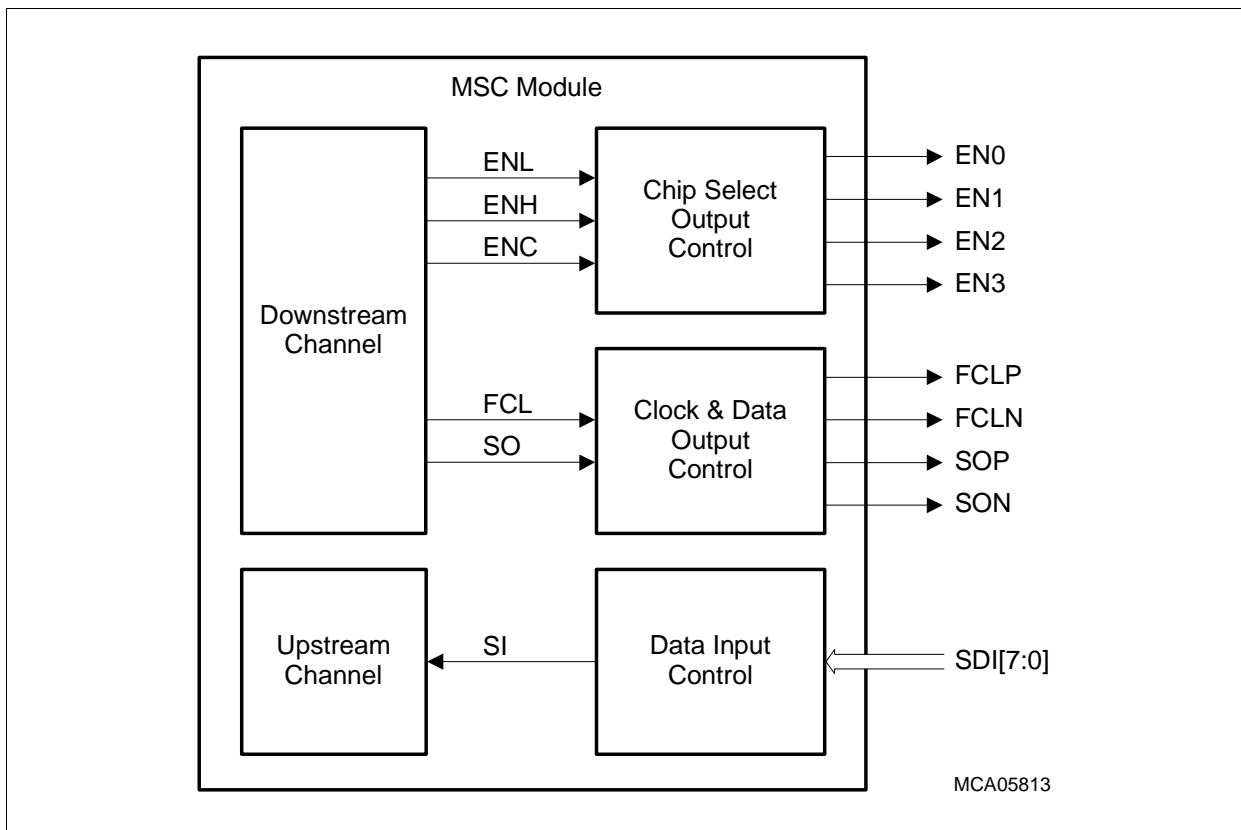


Figure 21-19 I/O Control

All MSC module I/O signals are controlled by bit fields that are located in the Output Control Register OCR.

21.1.4.1 Downstream Channel Output Control

As shown in **Figure 21-5** and **Figure 21-6**, the active phases during downstream channel operation are indicated by three enable signals:

- ENL indicates the SRL active phase of a data frame
- ENH indicates the SRH active phase of a data frame
- ENC indicates the active phase of a command frame

The chip select output control logic of the MSC uses a signal compressing scheme (similar to the interrupt request compressing scheme in **Figure 21-27**) that allows each of the three enable signals to be directed via a 2-bit selector to one of the four chip enable

Micro Second Channel (MSC)

outputs EN[3:0]. This also makes it possible to connect more than one internal enable signal (ENL, ENH, ENC) to one chip enable output ENx. Three bit fields in register OCR (CSL, CSH, and CSC) determine which chip enable output becomes active on a valid internal enable signal.

In the MSC, enable signals are high-level active signals. If required in a specific application, all chip enable outputs ENx can be assigned for low-level active polarity by setting bit OCR.CSLP.

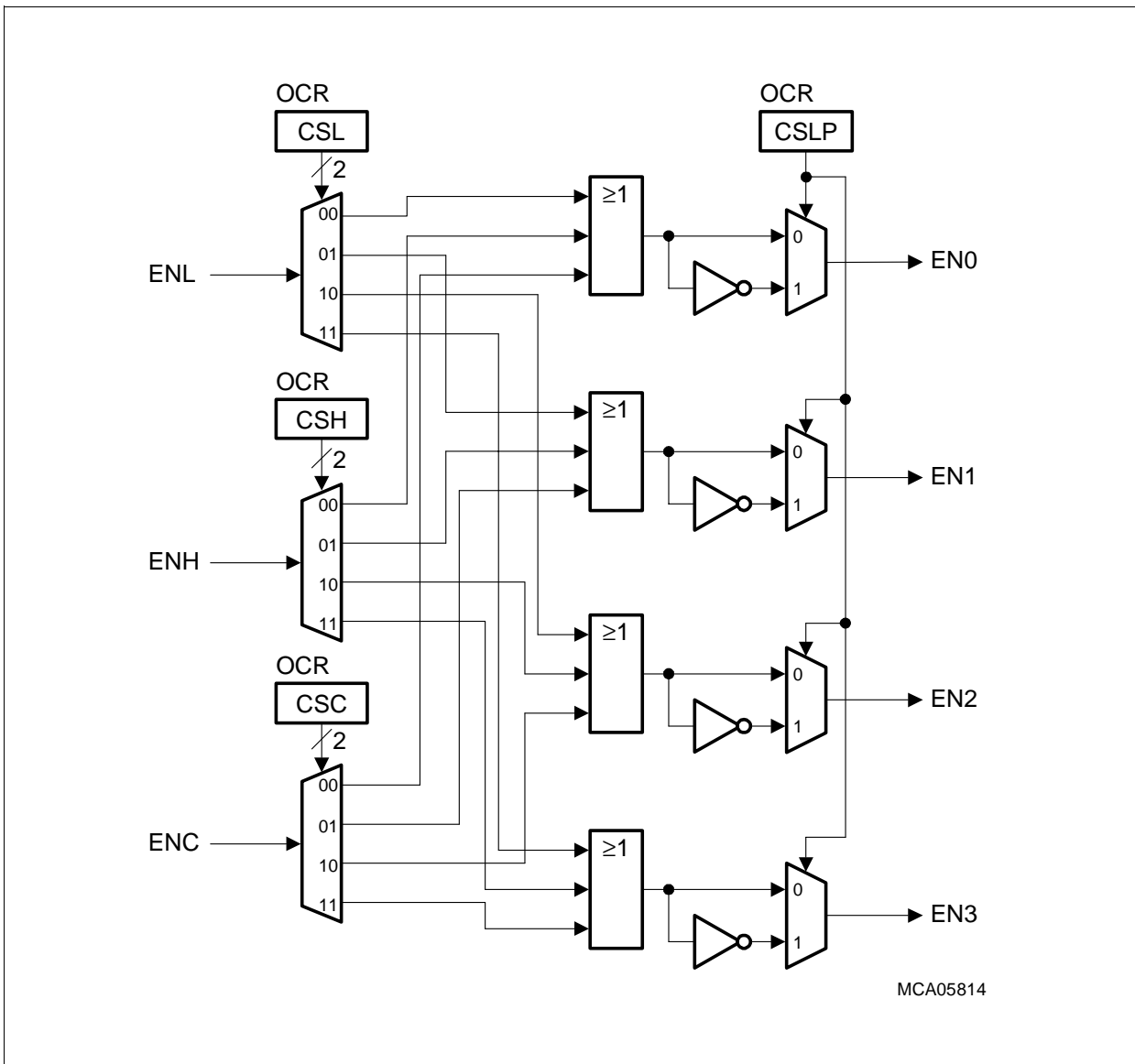


Figure 21-20 Downstream Channel: Chip Enable Output Control

Micro Second Channel (MSC)

At the MSC downstream channel, the internal serial clock output FCL and data output line SO are available outside the MSC module as two signal pairs with inverted signal polarity, FCLP/FCLN and SOP/SON. Both, clock and data outputs, are generated from the module internal signals FCL and SO according to **Figure 21-21**.

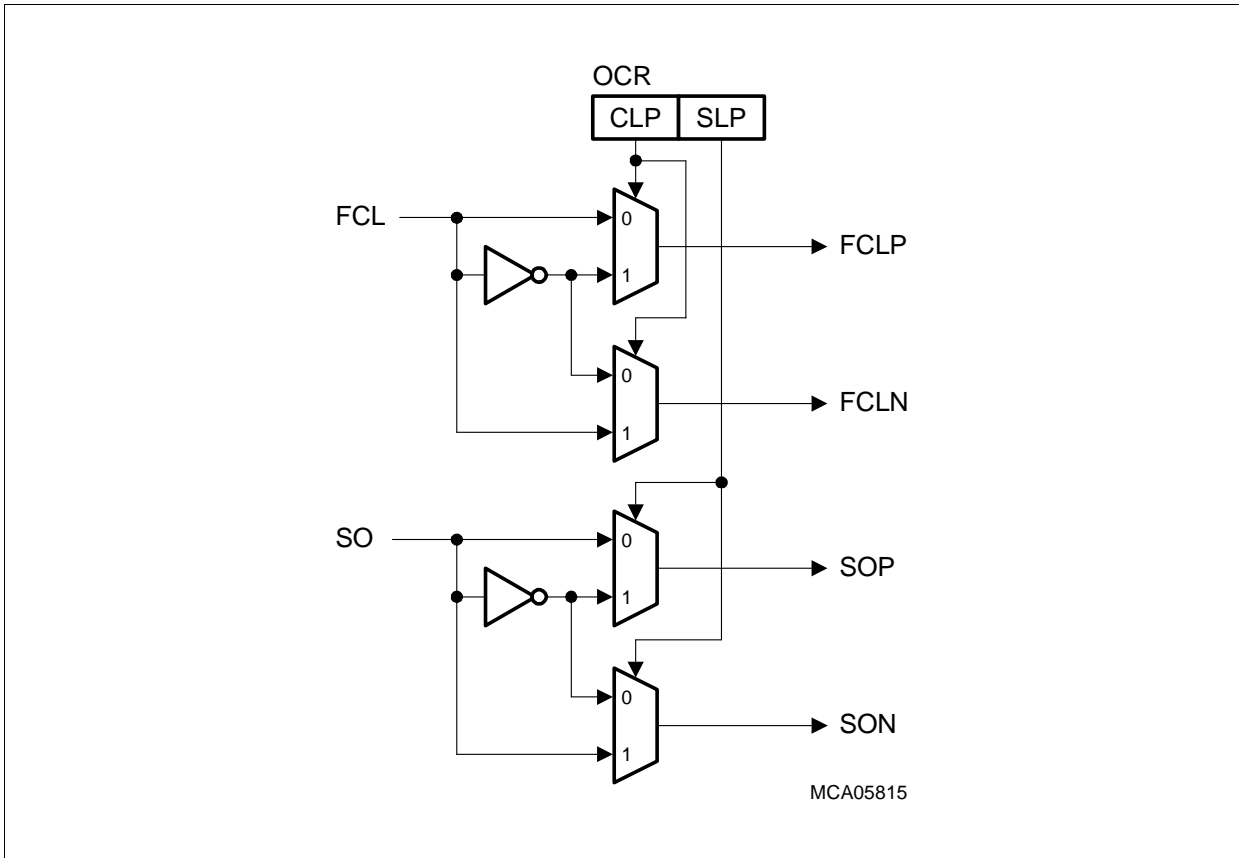


Figure 21-21 Downstream Channel: Clock and Data Output Control

With `OCR.CLP = 0`, FCLP has identical and FCLN has inverted polarity compared to FCL. Setting `OCR.CLP`, exchanges the signal polarities of FCLP and FCLN. An equivalent control capability is available for the SOP and SON data outputs (controlled by `OCR.SLP`).

One additional control capability not shown in **Figure 21-21** is available for the FCL signal. With `OCR.CLKCTRL = 1`, the FCL clock signal will always be generated, independently whether a downstream frame is currently transmitted or not. If `OCR.CLKCTRL = 0`, FCL becomes only active during the active phases of data or command frames (not during passive time frames).

21.1.4.2 Upstream Channel

As shown in [Figure 21-22](#), the MSC upstream channel can be connected to up to eight SDI[7:0] serial inputs. Bit field OCR.SDISEL selects one out of these input lines (input signal SDI). If OCR.ILP = 0, SDI is directly connected to the serial receive buffer input SI. If OCR.ILP = 1, SDI is connected to input SI via an inverter.

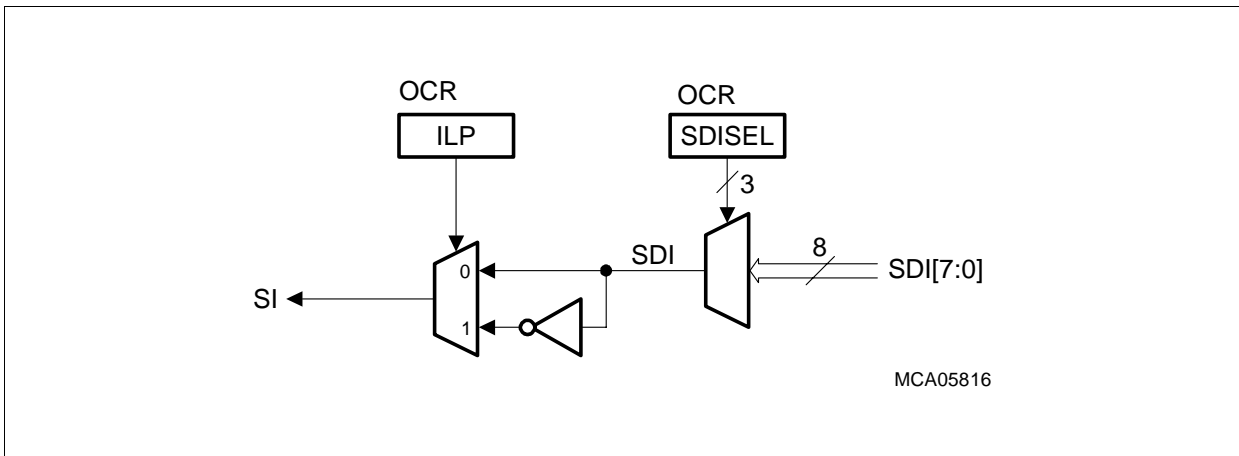


Figure 21-22 Upstream Channel Serial Data Input Control

21.1.5 MSC Interrupts

The MSC module has four interrupt sources and four service request outputs. A service request output is able to generate interrupts (controlled by a service request control register) or DMA requests. The service request output assignment, interrupt or DMA request, is specific for each microcontroller that is using the MSC (see also [Section 21.3.5.3](#) and [Section](#)). In this section, the term “interrupt request” has the meaning of “service request” that is able to handle interrupt or DMA requests.

Each interrupt source is provided with a status flag, enable bit(s) with software set/clear capability, and an interrupt node pointer. An interrupt event, internally generated as a request pulse, is always stored in an interrupt status flag that is located in the Interrupt Status Register ISR. All interrupt status flag can be set or cleared individually by software via the interrupt Set Clear Register ISC. Software-controlled interrupt generation can be initiated by setting the interrupt status flag of the corresponding interrupt. Each interrupt source can be enabled or disabled individually. When an interrupt event is enabled, a 2-bit interrupt node pointer determines which of the service request outputs will be activated.

[Table 21-7](#) shows the four MSC interrupt sources.

Table 21-7 MSC Interrupts

Interrupt Type	Generated by
Data frame interrupt	Downstream Channel
Command frame interrupt	
Time frame finished interrupt	
Receive data interrupt	Upstream Channel

21.1.5.1 Data Frame Interrupt

A data frame interrupt can be generated when either the first or the last data bit of the downstream channel is shifted out and becomes available at the SO output line (see also [Figure 21-6](#)). Bit ICR.EDIE selects which case is selected.

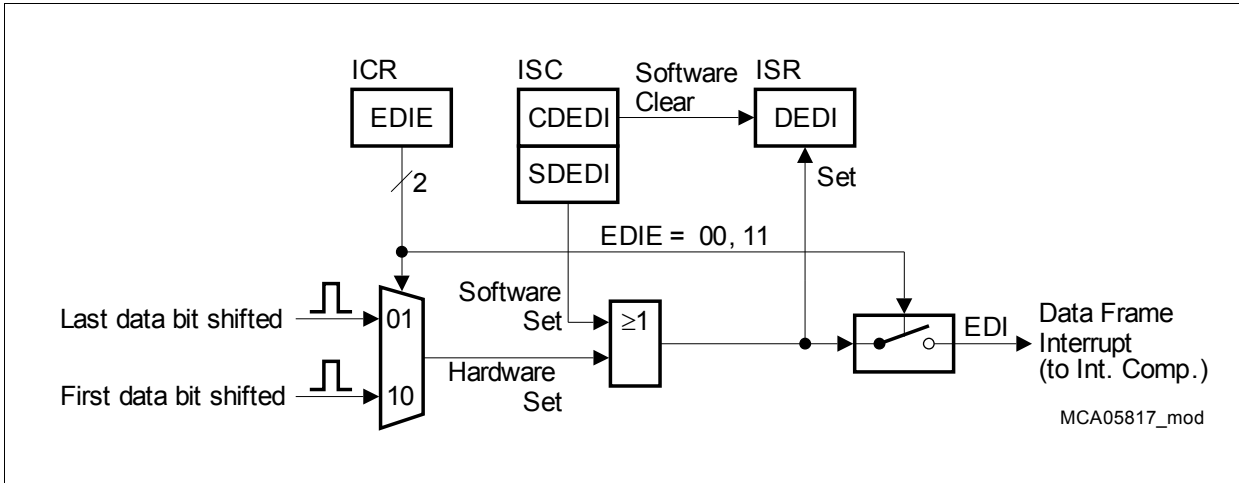


Figure 21-23 Data Frame Interrupt Control

21.1.5.2 Command Frame Interrupt

A command frame interrupt can be generated at the end of a downstream channel command frame (see also [Figure 21-5](#)).

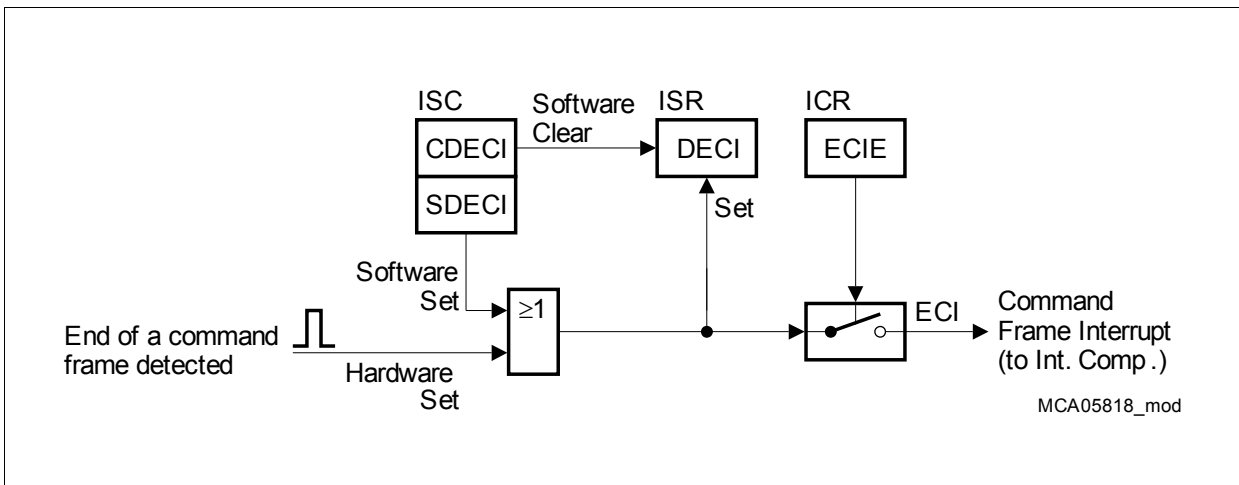


Figure 21-24 Command Frame Interrupt Control

21.1.5.3 Time Frame Finished Interrupt

A time frame finished interrupt can be generated at the end of a downstream channel passive time phase.

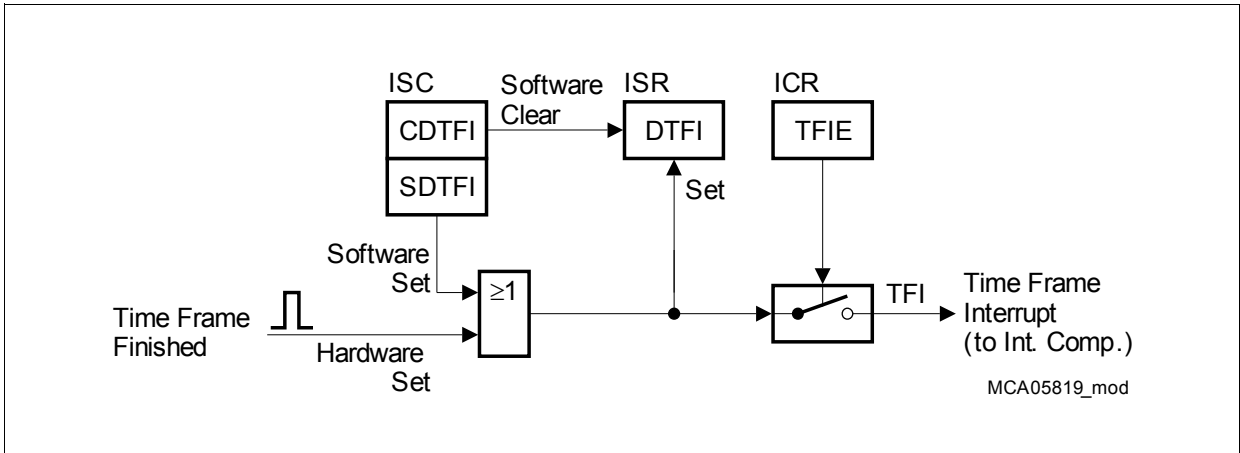


Figure 21-25 Time Frame Interrupt Control

21.1.5.4 Receive Data Interrupt

Whenever the upstream channel receives data in registers UDx (x = 0-3), the MSC is able to generate an interrupt. Three interrupt generation conditions can be selected for the receive data interrupt:

- Each update of UDx (x = 0-3) generates a receive data interrupt.
- Each update of UDx (x = 0-3) generates a receive data interrupt when the updated value is not equal 00_H.
- Only an update of register UD3 generates a receive data interrupt.

The selection of the interrupt generation condition is controlled by bit field ICR.RDIE. Setting ICR.RDIE = 0 disables the receive data interrupt in general. ISR.URDI is the interrupt status flag that can be set or cleared when writing bits ISC.SRDI or ISC.CRDI with a 1.

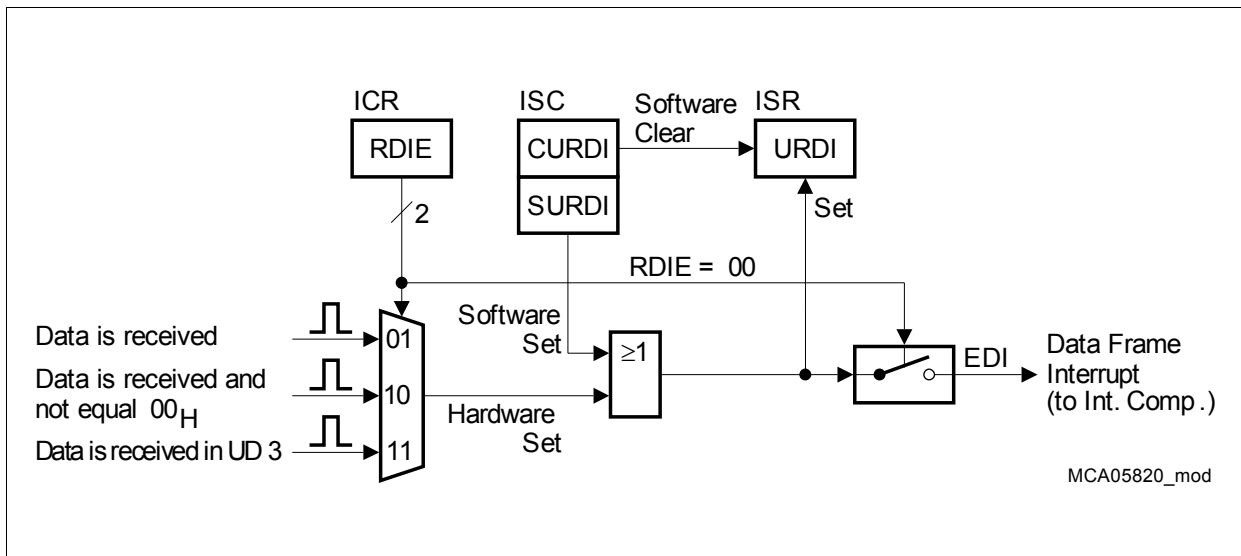


Figure 21-26 Receive Data Interrupt Control

21.1.5.5 Interrupt Request Compressor

The interrupt control logic of the MSC uses an interrupt compressing scheme that allows high flexibility in interrupt processing. Each of the four interrupt sources can be directed via a 2-bit interrupt node pointer to one of the four service request outputs SR[3:0]. This also makes it possible to connect more than one interrupt source to one interrupt output SRx.

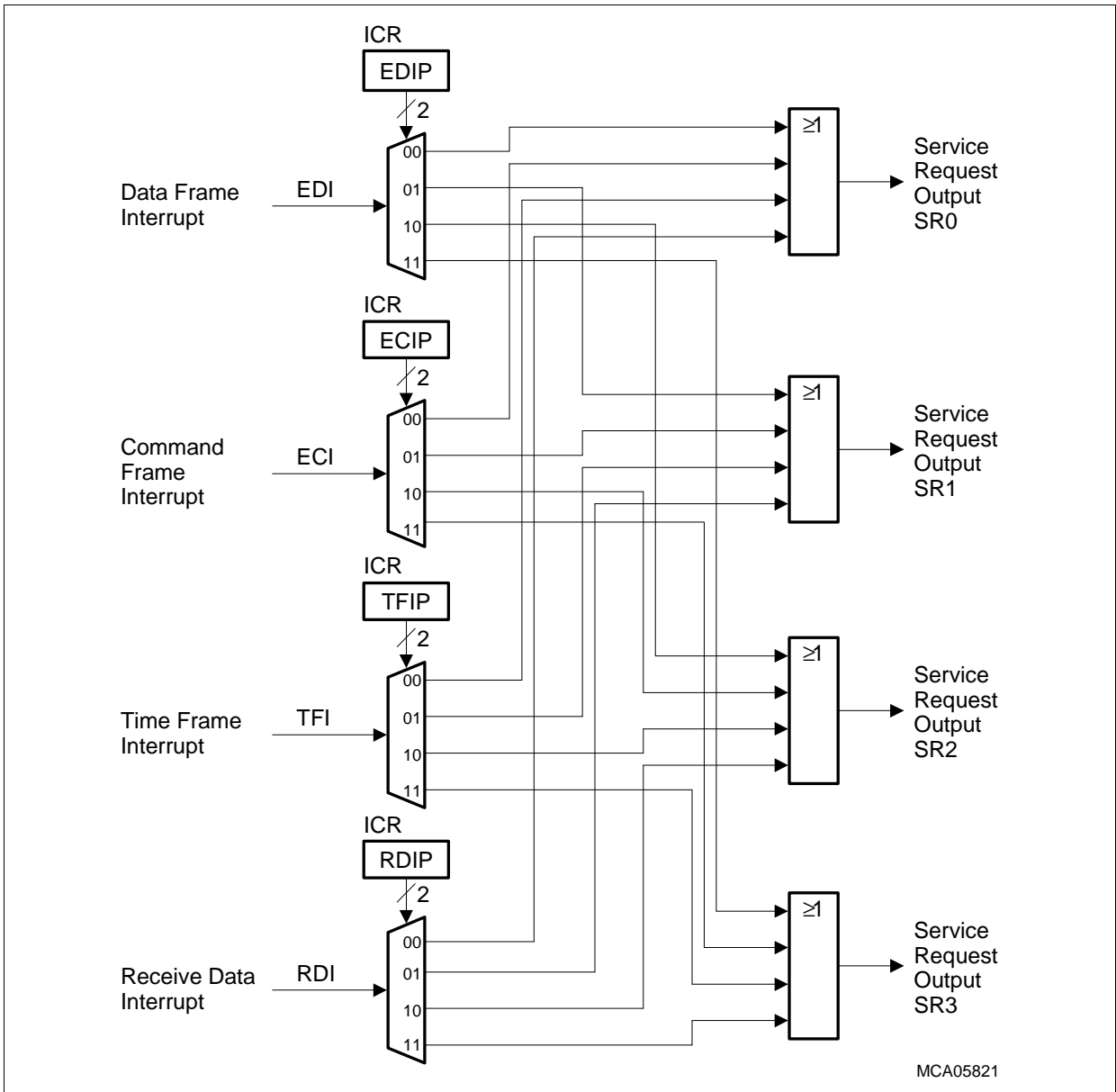


Figure 21-27 MSC Interrupt Request Compressor

Note: The number of available MSC interrupt outputs depends on the implementation of the MSC module(s) in the specific product (see [Section](#) for TC1796 details).

21.2 MSC Kernel Registers

This section describes the kernel registers of the MSC module. All MSC kernel register names described in this section will be referenced in other parts of the TC1796 User's Manual by the module name prefix "MSC0_" for the MSC0 interface.

MSC Kernel Register Overview

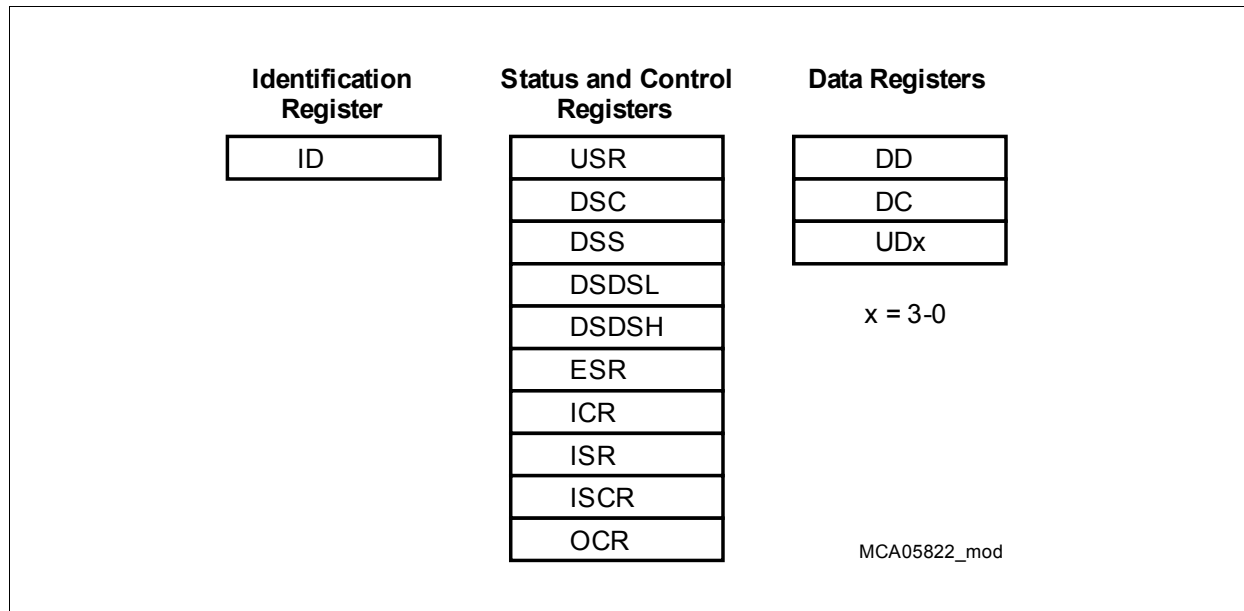


Figure 21-28 MSC Kernel Registers

The complete and detailed address map of the MSC0 module is described in [Table 18-7](#) on [Page 18-16](#) of the TC1796 User's Manual System Units part (Volume 1).

Table 21-8 Registers Address Space - MSC Kernel Registers

Module	Base Address	End Address	Note
MSC0	F000 0800 _H	F000 08FF _H	—
MSC1	F000 0900 _H	F000 09FF _H	—

Table 21-9 Registers Overview - MSC Kernel Registers

Register Short Name	Register Long Name	Offset Address ¹⁾	Description see
ID	Module Identification Register	08 _H	Page 21-38
USR	Upstream Status Register	10 _H	Page 21-39
DSC	Downstream Control Register	14 _H	Page 21-41

Micro Second Channel (MSC)

Table 21-9 Registers Overview - MSC Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Address ¹⁾	Description see
DSS	Downstream Status Register	18 _H	Page 21-44
DD	Downstream Data Register	1C _H	Page 21-59
DC	Downstream Command Register	20 _H	Page 21-59
DSDSL	Downstream Select Data Source Low Register	24 _H	Page 21-46
DSDSH	Downstream Select Data Source High Register	28 _H	Page 21-47
ESR	Emergency Stop Register	2C _H	Page 21-48
UD0	Upstream Data Register 0	30 _H	Page 21-60
UD1	Upstream Data Register 1	34 _H	
UD2	Upstream Data Register 2	38 _H	
UD3	Upstream Data Register 3	3C _H	
ICR	Interrupt Control Register	40 _H	Page 21-49
ISR	Interrupt Status Register	44 _H	Page 21-52
ISC	Interrupt Set Clear Register	48 _H	Page 21-54
OCR	Output Control Register	4C _H	Page 21-56

1) The absolute register address is calculated as follows:
Module Base Address ([Table 21-8](#)) + Offset Address (shown in this column)

Micro Second Channel (MSC)

21.2.2 Status and Control Registers

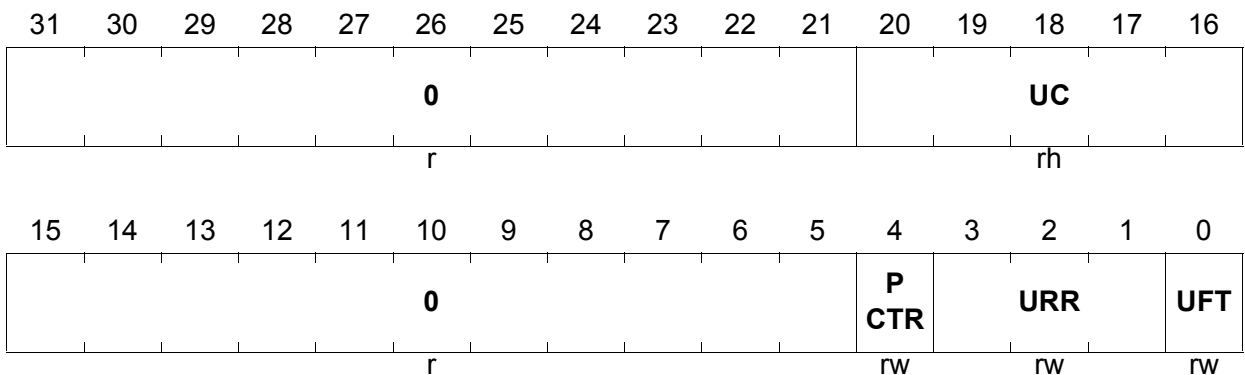
The Upstream Status Register DSR is used to configure the upstream channel data format, baud rate, and parity type. It also provides the status information of the upstream counter (UC).

USR

Upstream Status Register

(10_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
UFT	0	rw	Upstream Channel Frame Type This bit determines the frame type used by the upstream channel for data reception. 0 _B 12-bit upstream frame selected 1 _B 16-bit upstream frame selected (with 4-bit address field)
URR	[3:1]	rw	Upstream Channel Receiving Rate This bit field determines the baud rate for the upstream channel. 000 _B Upstream channel disabled; no reception is possible 001 _B Baud rate = $f_{MSC}/4$ 010 _B Baud rate = $f_{MSC}/8$ 011 _B Baud rate = $f_{MSC}/16$ 100 _B Baud rate = $f_{MSC}/32$ 101 _B Baud rate = $f_{MSC}/64$ 110 _B Baud rate = $f_{MSC}/128$ 111 _B Baud rate = $f_{MSC}/256$

Micro Second Channel (MSC)

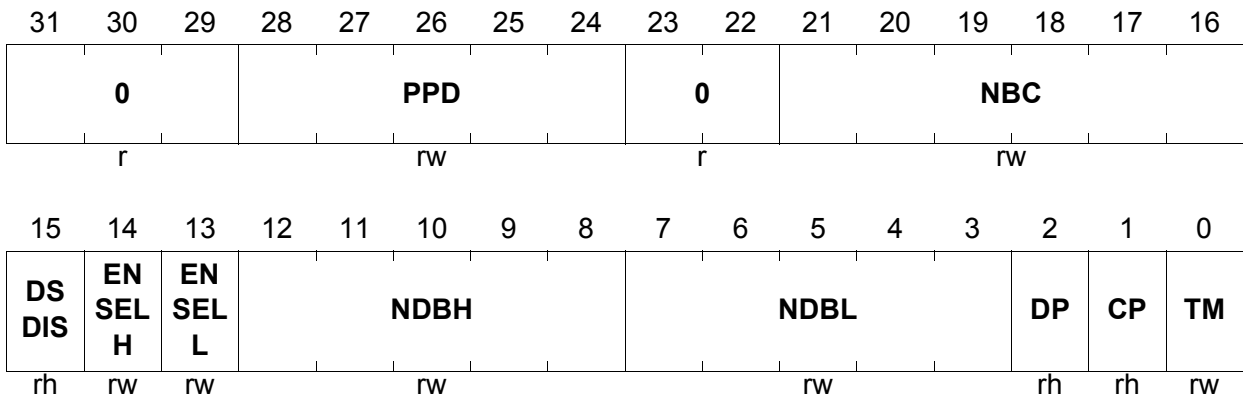
Field	Bits	Type	Description
PCTR	4	rw	<p>Parity Control This bit determines the parity mode used by the upstream channel for data reception.</p> <p>0_B Even parity mode is selected. A parity bit is set on an odd number of 1s in the serial address/data stream.</p> <p>1_B Odd parity mode is selected. A parity bit is set on an even number of 1s in the serial address/data stream.</p>
UC	[20:16]	rh	<p>Upstream Counter This bit field indicates the content of the upstream counter that counts the bits during upstream channel reception.</p>
0	[15:5], [31:21]	r	<p>Reserved Read as 0; should be written with 0.</p>

Micro Second Channel (MSC)

The Downstream Control Register DSC is used to control the operation mode and frame layout of the downstream channel transmission. It also contains the two pending status bits.

DSC

Downstream Control Register (14_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
TM	0	rw	<p>Transmission Mode</p> <p>This bit selects the transmission mode of the downstream channel.</p> <p>0_B Triggered Mode selected</p> <p>1_B Data Repetition Mode selected</p>
CP	1	rh	<p>Command Pending</p> <p>This bit is set when the downstream command register DC is written. CP is cleared when the first bit of the related command frame is sent out.</p>
DP	2	rh	<p>Data Pending</p> <p>In Triggered Mode, this bit is set when the set data pending bit ISC.SDP is set by software. In Data Repetition Mode, this bit is set by hardware at the last passive time frame. At the start of the data frame, DP is cleared by hardware.</p>

Micro Second Channel (MSC)

Field	Bits	Type	Description
NDBL	[7:3]	rw	<p>Number of SRL Bits Shifted at Data Frames NDBL determines the number of shift register low part (SRL) bits that are shifted out on SO during a data frame.</p> <p>00000_B No SRL bit shifted 00001_B SRL[0] shifted 00010_B SRL[1:0] shifted ..._B ... 01111_B SRL[14:0] shifted 10000_B SRL[15:0] shifted Other bit combinations are reserved; do not use these bit combinations.</p>
NDBH	[12:8]	rw	<p>Number of SRH Bits Shifted at Data Frames NDBH determines the number of shift register high part (SRH) bits that are shifted out on SO during a data frame.</p> <p>00000_B No SRH bit shifted; no selection bit is generated, the SRH active phase is completely skipped. 00001_B SRH[0] shifted 00010_B SRH[1:0] shifted ..._B ... 01111_B SRH[14:0] shifted 10000_B SRH[15:0] shifted Other bit combinations are reserved; do not use these bit combinations.</p>
ENSELL	13	rw	<p>Enable SRL Active Phase Selection Bit This bit determines whether a low-level selection bit is inserted at the beginning of a data frame's SRL active phase.</p> <p>0_B No selection bit inserted. 1_B Low-level selection bit inserted.</p>
ENSELH	14	rw	<p>Enable SRH Active Phase Selection Bit This bit determines whether a low-level selection bit is inserted at the beginning of a data frame's SRH active phase.</p> <p>0_B No selection bit inserted. 1_B Low-level selection bit inserted.</p>

Micro Second Channel (MSC)

Field	Bits	Type	Description
DSDIS	15	rh	<p>Downstream Disable This bit indicates the state of the downstream channel operation.</p> <p>0_B The downstream channel is enabled. A frame transmission can take place (Triggered Mode) or takes place (Data Repetition Mode).</p> <p>1_B Downstream Counter becomes disabled. No new frame transmission is started. A running frame transmission is always completed.</p>
NBC	[21:16]	rw	<p>Number of Bits Shifted at Command Frames This bit field determines how many bits of the SRL/SRH shift registers are shifted out during transmission of a command frame.</p> <p>000000_B No bit shifted 000001_B SRL[0] shifted 000010_B SRL[1:0] shifted 000011_B SRL[2:0] shifted ..._B ... 010000_B SRL[15:0] shifted 010001_B SRL[15:0] and SRH[0] shifted 010010_B SRL[15:0] and SRH[1:0] shifted ..._B ... 011111_B SRL[15:0] and SRH[14:0] shifted 100000_B SRL[15:0] and SRH[15:0] shifted Other bit combinations are reserved; do not use these bit combinations</p>
PPD	[28:24]	rw	<p>Passive Phase Length at Data Frames This bit field determines the length of the passive phase of a data frame.</p> <p>00000_B Passive phase length is $2 \times t_{FCL}$ 00001_B Passive phase length is $2 \times t_{FCL}$ 00010_B Passive phase length is $2 \times t_{FCL}$ 00011_B Passive phase length is $3 \times t_{FCL}$..._B ... 11111_B Passive phase length is $31 \times t_{FCL}$</p>
0	[23:22], [31:29]	r	<p>Reserved Read as 0; should be written with 0.</p>

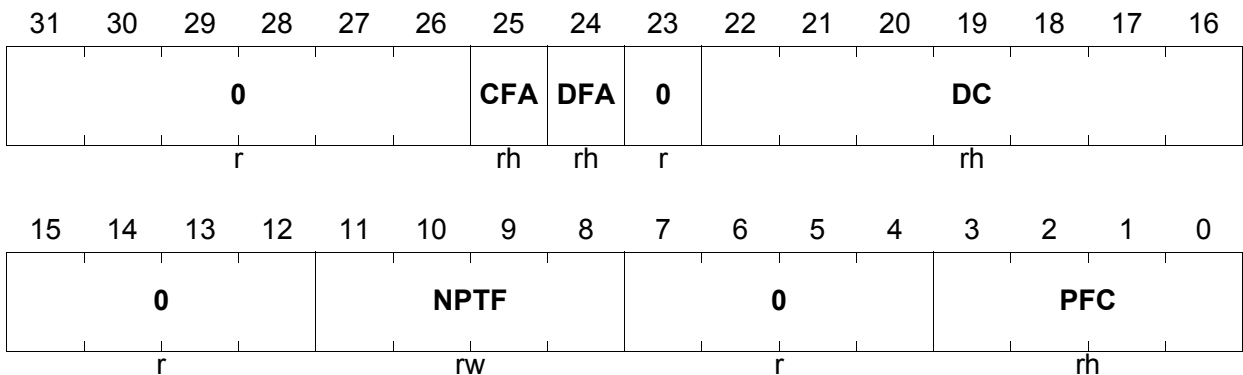
Note: The rw bits in the DSC register are buffered in a shadow buffer at the start of a corresponding frame transmission.

Micro Second Channel (MSC)

The Downstream Status Register DSS contains counter bit fields, status bits, and indicates the number of passive time frames.

DSS

Downstream Status Register (18_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
PFC	[3:0]	rh	<p>Passive Time Frame Counter</p> <p>In Data Repetition Mode, this bit field indicates the count of passive time frames that are currently transmitted. In Triggered Mode PFC remains at 0000_B.</p> <p>0000_B Data frame is transmitted. 0001_B First passive time frame is transmitted. 0010_B Second passive time frame is transmitted. ..._B ... 1111_B Fifteenth passive time frame is transmitted.</p>
NPTF	[11:8]	rw	<p>Number Of Passive Time Frames</p> <p>This bit field indicates the number of passive time frames that are inserted in Data Repetition Mode between two data frames.</p> <p>0000_B No passive time frame inserted. 0001_B One passive time frame inserted. 0010_B Two passive time frames inserted. ..._B ... 1111_B Fifteen passive time frames inserted.</p> <p><i>Note: NPTF is buffered in a shadow buffer at the start of each data frame.</i></p>

Micro Second Channel (MSC)

Field	Bits	Type	Description
DC	[22:16]	rh	<p>Downstream Counter</p> <p>This bit field indicates the number of downstream shift clock periods that have been elapsed since the start of the current frame.</p> <p>00_H No shift clock elapsed (after counter reset). 01_H 1 shift clock elapsed. ..._B ... 7F_H 127 shift clocks elapsed.</p> <p>The DC is reset at the end of a downstream frame.</p>
DFA	24	rh	<p>Data Frame Active</p> <p>This bit indicates if a data frame is currently sent out.</p> <p>0_B No data frame is currently sent out. 1_B A data frame is currently sent out.</p>
CFA	25	rh	<p>Command Frame Active</p> <p>This bit indicates if a command frame is currently sent out.</p> <p>0_B No command frame is currently sent out. 1_B A command frame is currently sent out.</p>
0	[7:4], [15:12], 23, [31:26]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

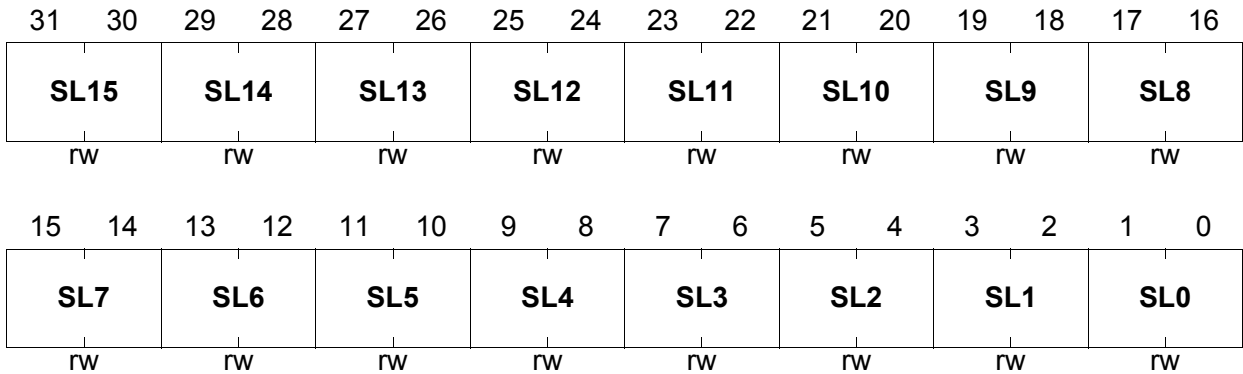
Micro Second Channel (MSC)

The bit fields of the Downstream Select Data Low Register DSDSL determine the data source for each bit in shift register SRL.

DSDSL

**Downstream Select Data Source Low Register
(24_H)**

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SLx (x = 0-15)	[2*x+1: 2*x]	rw	Select Source for SRL SLx determines which data source is used for the shift register bit SRL[x] during data frame transmission. 00 _B SRL[x] is taken from data register DD.DDL[x]. 01 _B Reserved. 10 _B SRL[x] is taken from the ALTINL input line x. 11 _B SRL[x] is taken from the ALTINL input line x in inverted state.

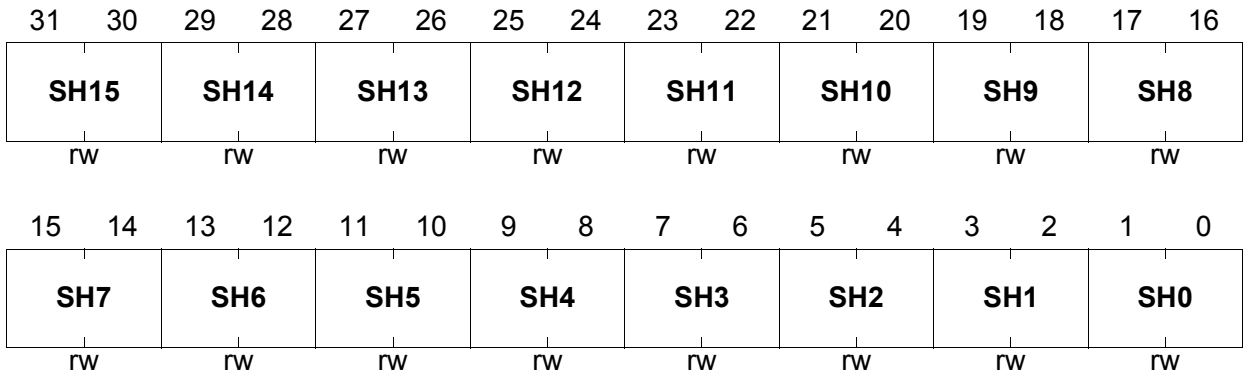
Micro Second Channel (MSC)

The bit fields of the Downstream Select Data Source High Register DSDSR determine the data source for each bit in shift register SRL.

DSDSH

**Downstream Select Data Source High Register
(28_H)**

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SHx (x = 0-15)	[2*x+1: 2*x]	rw	Select Source for SRH SHx determines which data source is used for the shift register bit SRL[x] during data frame transmission. 00 _B SRL[x] is taken from data register DD.DDL[x]. 01 _B Reserved. 10 _B SRL[x] is taken from the ALTINH input line x. 11 _B SRL[x] is taken from the ALTINH input line x in inverted state.

Micro Second Channel (MSC)

The Emergency Stop Register ESR determines which bits of SRL and SRH are enabled for emergency operation.

ESR

Emergency Stop Register

(2C_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENH	ENH	ENH	ENH	ENH	ENH	ENH	ENH	ENH	ENH	ENH	ENH	ENH	ENH	ENH	ENH
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENL	ENL	ENL	ENL	ENL	ENL	ENL	ENL	ENL	ENL	ENL	ENL	ENL	ENL	ENL	ENL
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

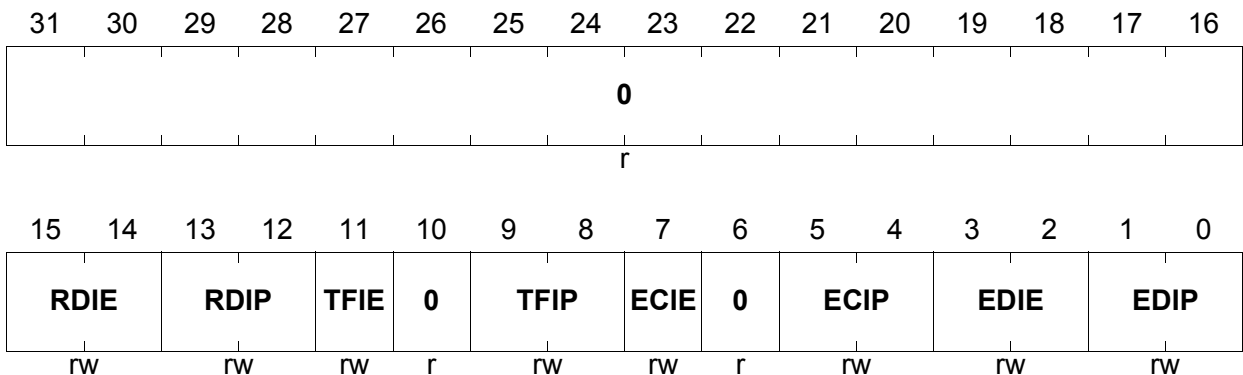
Field	Bits	Type	Description
ENLx (x = 0-15)	x	rw	<p>Emergency Stop Enable for Bit x in SRL</p> <p>This bit enables the emergency stop feature selectively for each SRL bit. If the emergency stop condition is met and enabled (ENLx = 1), the SRL[x] bit is of the data register DD.DDL[x] is used for the shift register load operation.</p> <p>0_B Emergency stop feature for bit SRL[x] is disabled.</p> <p>1_B The emergency stop feature for bit SRL[x] is enabled.</p>
ENHx (x = 0-15)	x+16	rw	<p>Emergency Stop Enable for Bit x in SRH</p> <p>This bit enables the emergency stop feature selectively for each SRH bit. If the emergency stop condition is met and enabled (ENHx = 1), the SRH[x] bit of the data register DD.DDH[x] is used for the shift register load operation.</p> <p>0_B Emergency stop feature for bit SRH[x] is disabled.</p> <p>1_B The emergency stop feature for bit SRH[x] is enabled.</p>

Micro Second Channel (MSC)

The Interrupt Control Register ICR holds the interrupt enable bits and interrupt pointers of all four MSC interrupts.

ICR

Interrupt Control Register (40_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
EDIP	[1:0]	rw	<p>Data Frame Interrupt Node Pointer EDIP selects the service request output line SR_n (n = 3-0) for the data frame interrupt.</p> <p>00_B Service request output SR0 selected 01_B Service request output SR1 selected 10_B Service request output SR2 selected 11_B Service request output SR3 selected</p>
EDIE	[3:2]	rw	<p>Data Frame Interrupt Enable This bit field determines the enable conditions for the data frame interrupt.</p> <p>00_B Interrupt generation disabled 01_B An interrupt is generated when the last data bit has been shifted out. 10_B An interrupt is generated when the first data bit has been shifted out, but only if DSC.NDBL is not equal 00000_B. This means, at least one SRL bit must be shifted out for the first data bit shifted interrupt to become active. 11_B Interrupt generation disabled</p>

Micro Second Channel (MSC)

Field	Bits	Type	Description
ECIP	[5:4]	rw	Command Frame Interrupt Node Pointer ECIP selects the service request output line SRn (n = 3-0) for the command frame interrupt. 00 _B Service request output SR0 selected 01 _B Service request output SR1 selected 10 _B Service request output SR2 selected 11 _B Service request output SR3 selected
ECIE	7	rw	Command Frame Interrupt Enable This bit enables the command frame interrupt. 0 _B Interrupt generation disabled. 1 _B Interrupt generation enabled.
TFIP	[9:8]	rw	Time Frame Interrupt Pointer TFIP selects the service request output line SRn (n = 3-0) for the time frame interrupt. 00 _B Service request output SR0 selected 01 _B Service request output SR1 selected 10 _B Service request output SR2 selected 11 _B Service request output SR3 selected
TFIE	11	rw	Time Frame Interrupt Enable This bit enables the time frame interrupt. 0 _B Interrupt generation disabled. 1 _B Interrupt generation enabled.
RDIP	[13:12]	rw	Receive Data Interrupt Pointer RDIP selects the service request output line SRn (n = 3-0) for the receive data interrupt. 00 _B Service request output SR0 selected 01 _B Service request output SR1 selected 10 _B Service request output SR2 selected 11 _B Service request output SR3 selected

Micro Second Channel (MSC)

Field	Bits	Type	Description
RDIE	[15:14]	rw	<p>Receive Data Interrupt Enable This bit field determines the enable conditions for the receive data interrupt.</p> <p>00_B Interrupt generation disabled.</p> <p>01_B An interrupt is generated when data is received and written into the upstream data registers UD_x (x = 0-3).</p> <p>10_B An interrupt is generated as with RDIE = 01_B but only if the received data is not equal to 00_H.</p> <p>11_B An interrupt is generated when data is received and written into register UD3.</p>
0	6, 10, [31:16]	r	<p>Reserved Read as 0; should be written with 0.</p>

Micro Second Channel (MSC)

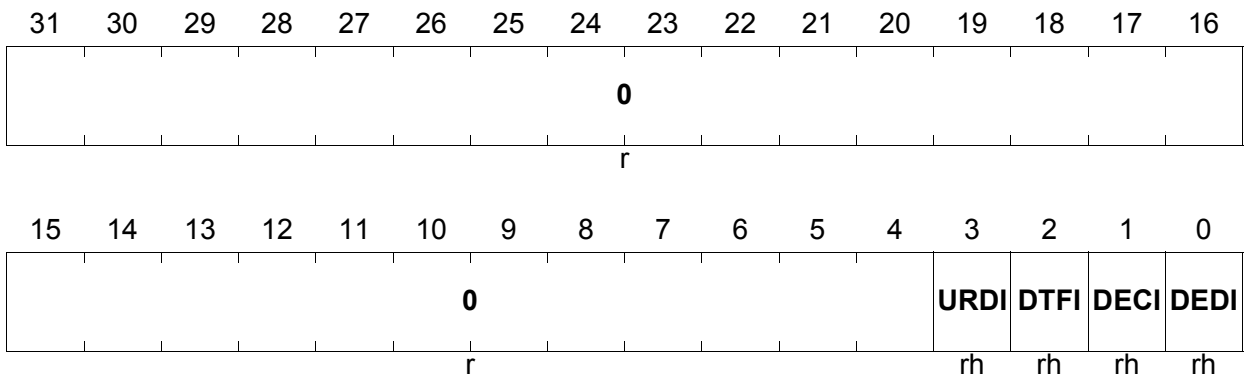
The Interrupt Status Register ISR holds the interrupt status flags that indicate an interrupt occurrence in downstream and upstream channels.

ISR

Interrupt Status Register

(44_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
DEDI	0	rh	<p>Data Frame Interrupt Flag</p> <p>This flag is always set by hardware when a downstream channel data frame interrupt is generated. DEDI can be set or cleared by software when writing to register ISC with the appropriate bits ISC.SDEDI or ISC.CDEDI set.</p>
DECI	1	rh	<p>Command Frame Interrupt Flag</p> <p>This flag is always set by hardware when a downstream channel command frame interrupt is generated, whether or not it is enabled. DECI can be set or cleared by software when writing to register ISC with the appropriate bits SDECI or CDECI set.</p>
DTFI	2	rh	<p>Time Frame Interrupt Flag</p> <p>This flag is always set by hardware when a downstream channel time frame interrupt is generated, whether or not it is enabled. DTFI can be set or cleared by software when writing to register ISC with the appropriate bits SDTFI or CDTFI set.</p>

Micro Second Channel (MSC)

Field	Bits	Type	Description
URDI	3	rh	Receive Data Interrupt Flag This flag is always set by hardware when an upstream channel receive data interrupt is generated, whether or not it is enabled. URDI can be set or cleared by software when writing to register ISC with the appropriate bits SURDI or CURDI set.
0	[31:4]	r	Reserved Read as 0; should be written with 0.

Micro Second Channel (MSC)

Field	Bits	Type	Description
SDEDI	16	w	Set DEDI Flag 0 _B No operation. 1 _B Bit ISR.DEDI is set.
SDECI	17	w	Set DECI Flag 0 _B No operation. 1 _B Bit ISR.DECI is set.
SDTFI	18	w	Set DTFI Flag 0 _B No operation. 1 _B Bit ISR.DTFI is set.
SURDI	19	w	Set URDI Flag 0 _B No operation. 1 _B Bit ISR.URDI is set.
SDP	20	w	Set DP Bit 0 _B No effect. 1 _B Bit DSC.DP is set.
SCP	21	w	Set CP Flag 0 _B No operation. 1 _B Bit DSC.CP is set.
SDDIS	22	w	Set DSDIS Flag 0 _B No operation. 1 _B Bit DSC.DSDIS is set.
0	[15:7], [31:23]	r	Reserved Read as 0; should be written with 0.

Note: When the ISC register is written with both bits set (clear and set bit) for a specific interrupt flag, the clear operation takes place and the set operation is ignored.

Micro Second Channel (MSC)

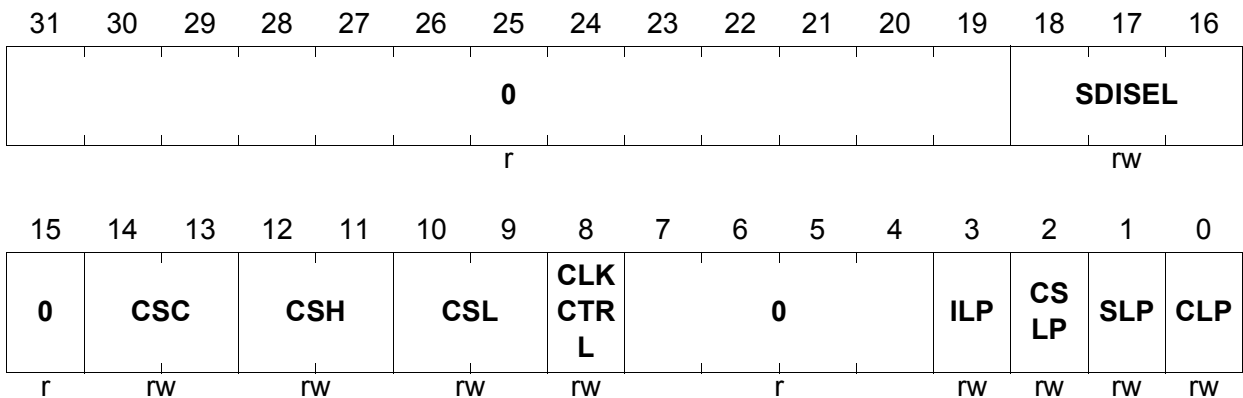
The Output Control Register OCR determines the MSC input/output signal polarities, the chip select output signal assignment, and the serial output clock generation.

OCR

Output Control Register

(4C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CLP	0	rw	FCLP Line Polarity 0 _B FCLP and FCL signal polarity is identical. FCLN signal has inverted FCL signal polarity. 1 _B FCLP signal has inverted FCL signal polarity. FCLN and FCL signal polarities are identical.
SLP	1	rw	SOP Line Polarity 0 _B SOP and SO signal polarity is identical. SON signal has inverted SO signal polarity. 1 _B SOP signal has inverted SO signal polarity. SON and SO signal polarities are identical.
CSLP	2	rw	Chip Selection Lines Polarity 0 _B EN[3:0] and ENL, ENH, ENC signal polarities are identical (high active). 1 _B EN[3:0] signal polarities are inverted (low active) to the ENL, ENH, ENC signal polarities. Bit CSLP is buffered during a frame transmission. This means that any change of CSLP becomes valid first with the start of the next frame transmission.
ILP	3	rw	SDI Line Polarity 0 _B SDI and SI signal polarities are identical. 1 _B SDI and SI signal polarities are inverted.

Micro Second Channel (MSC)

Field	Bits	Type	Description
CLKCTRL	8	rw	<p>Clock Control This bit determines the activation of clock output FCL.</p> <p>0_B FCL is activated only during the active phases of data or command frames (not during passive time frames).</p> <p>1_B FCL is always active whether or not a downstream frame is currently transmitted.</p>
CSL	[10:9]	rw	<p>Chip Enable Selection for ENL This bit field selects the chip enable output ENx that becomes active during the SRL active phase (ENL = 1) of a data frame. The active level of ENx is defined by bit CSLP.</p> <p>00_B EN0 line is selected for ENL. 01_B EN1 line is selected for ENL. 10_B EN2 line is selected for ENL. 11_B EN3 line is selected for ENL.</p>
CSH	[12:11]	rw	<p>Chip Enable Selection for ENH This bit field selects the chip enable output ENx that becomes active during the SRL active phase (ENH = 1) of a data frame. The active level of ENx is defined by bit CSLP.</p> <p>00_B EN0 line is selected for ENH. 01_B EN1 line is selected for ENH. 10_B EN2 line is selected for ENH. 11_B EN3 line is selected for ENH.</p>
CSC	[14:13]	rw	<p>Chip Enable Selection for ENC This bit field selects the chip enable output ENx that becomes active during the active phase (ENC = 1) of a command frame. The active level of ENx is defined by bit CSLP.</p> <p>00_B EN0 line is selected for ENC. 01_B EN1 line is selected for ENC. 10_B EN2 line is selected for ENC. 11_B EN3 line is selected for ENC.</p>

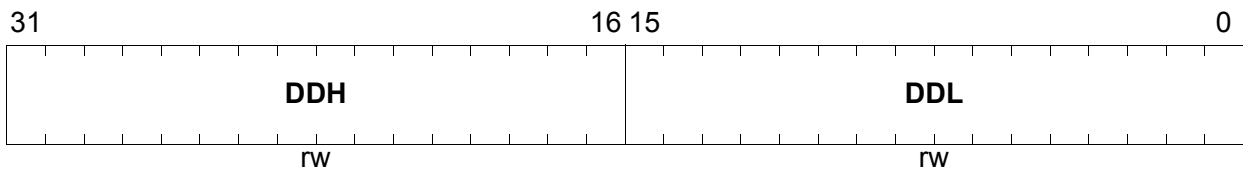
Micro Second Channel (MSC)

Field	Bits	Type	Description
SDISEL	[18:16]	rw	<p>Serial Data Input Selection This bit field selects the source for the serial data input SDI of the upstream channel.</p> <p>000_B SDI[0] input is selected as SDI. 001_B SDI[1] input is selected as SDI. 010_B SDI[2] input is selected as SDI. 011_B SDI[3] input is selected as SDI. 100_B SDI[4] input is selected as SDI. 101_B SDI[5] input is selected as SDI. 110_B SDI[6] input is selected as SDI. 111_B SDI[7] input is selected as SDI.</p>
0	[7:4], 15, [31:19]	r	<p>Reserved Read as 0; should be written with 0.</p>

21.2.3 Data Registers

The Downstream Data Register DD contains data to be transmitted during data frames.

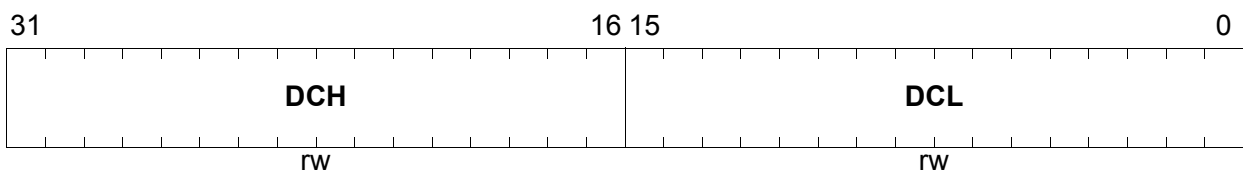
DD
Downstream Data Register (1C_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
DDL	[15:0]	rw	Downstream Data for SRL Shift Register Contains the data bits to be transmitted during the SRL active phase of a data frame.
DDH	[31:16]	rw	Downstream Data for SRH Shift Register Contains the data bits to be transmitted during the SRH active phase of a data frame.

The Downstream Command Register DC contains command information to be transmitted during command frames.

DC
Downstream Command Register (20_H) **Reset Value: 0000 0000_H**



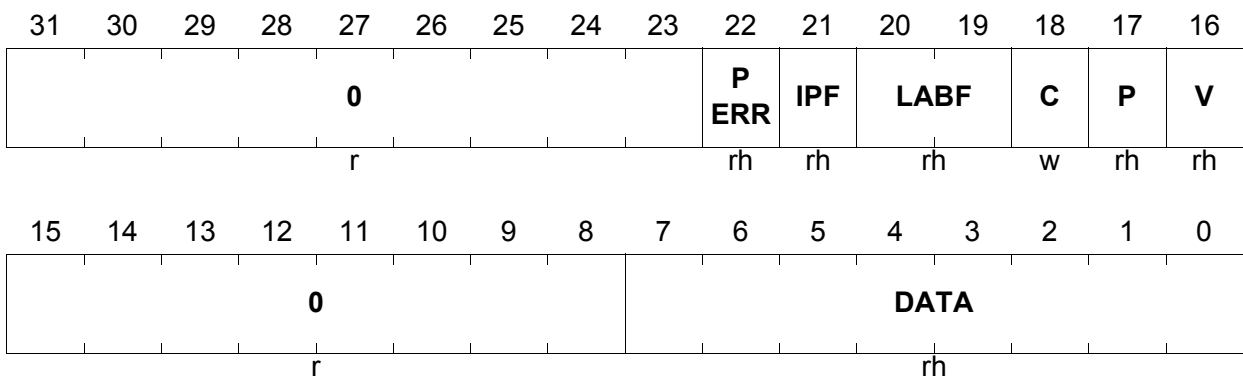
Field	Bits	Type	Description
DCL	[15:0]	rw	Downstream Command for SRL Shift Register Contains the data bits to be transmitted during the SRL active phase of a command frame.
DCH	[31:16]	rw	Downstream Command for SRH Shift Register Contains the data bits to be transmitted during the SRH active phase of a command frame.

Micro Second Channel (MSC)

The four Upstream Data Registers UDx store the content (data, addresses, received and calculated parity bit, parity error bit) of a received upstream channel data frame.

UDx (x = 0-3)

Upstream Data Register x **(30_H+x*4_H)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
DATA	[7:0]	rh	Received Data This bit field contains the 8-bit receive data.
V	16	rh	Valid Bit This bit is set by hardware when the received data is written to UDx. Writing bit C = 1 clears V. If hardware setting and software clearing of the valid bit occur simultaneously, bit V will be cleared.
P	17	rh	Parity Bit This flag contains the parity bit that has been received with the data frame.
C	18	w	Clear Bit 0 _B No operation. 1 _B Bit V is cleared. Bit C is always read as 0.
LABF	[20:19]	rh	Lower Address Bit Field This bit field contains the two address bits A[1:0] of the 4-bit address field (16-bit data frame). If 12-bit data frame is selected, LABF is always set to 00 _B .
IPF	21	rh	Internal Parity Flag This bit contains the parity bit that has been calculated in the MSC during data frame reception.

Micro Second Channel (MSC)

Field	Bits	Type	Description
PERR	22	rh	Parity Error This bit indicates if a start bit error, parity error, or stop bit error occurred during frame reception. 0 _B No error detected. 1 _B Error detected.
0	[15:8], [31:23]	r	Reserved Read as 0; should be written with 0.

21.3 MSC Module Implementation

This section describes the MSC module interfaces as they are implemented in the TC1796. It especially covers clock control, port and on-chip connections, interrupt control, and address decoding.

21.3.1 Interface Connections of the MSC Modules

Figure 21-29 shows the TC1796-specific implementation details and interconnections of the MSC0 and MSC1 modules.

Each MSC module is supplied by a separate clock control, address decoding, and interrupt control logic. Two of the four modules' service request outputs are connected to interrupt nodes, and two with the DMA controller. Outputs of the GPTA0, GPTA1, and LTCA2 modules are connected to the alternate input buses ALTINL/ALTINH. The emergency stop output from the SCU controls the corresponding inputs of both MSC modules.

The serial data and clock outputs of the downstream channels of each MSC module are connected to dedicated LVDS differential output drivers. After a reset operation, all LVDS outputs are disabled and in power-down mode. They must be enabled before usage by setting `SCU_CON.LDEN = 1`.

Additionally, the positive serial clock (FCLP) and the positive data output (SOP) are available at GPIO lines of Port 5 and Port 9. The device select outputs (ENxy) are wired to GPIO lines of Port 5 and Port 9. Two Port 5 input lines are connected to the upstream channel serial data inputs.

Each of the MSC module's downstream channel serial data inputs is connected to one port line of Port 5.

Micro Second Channel (MSC)

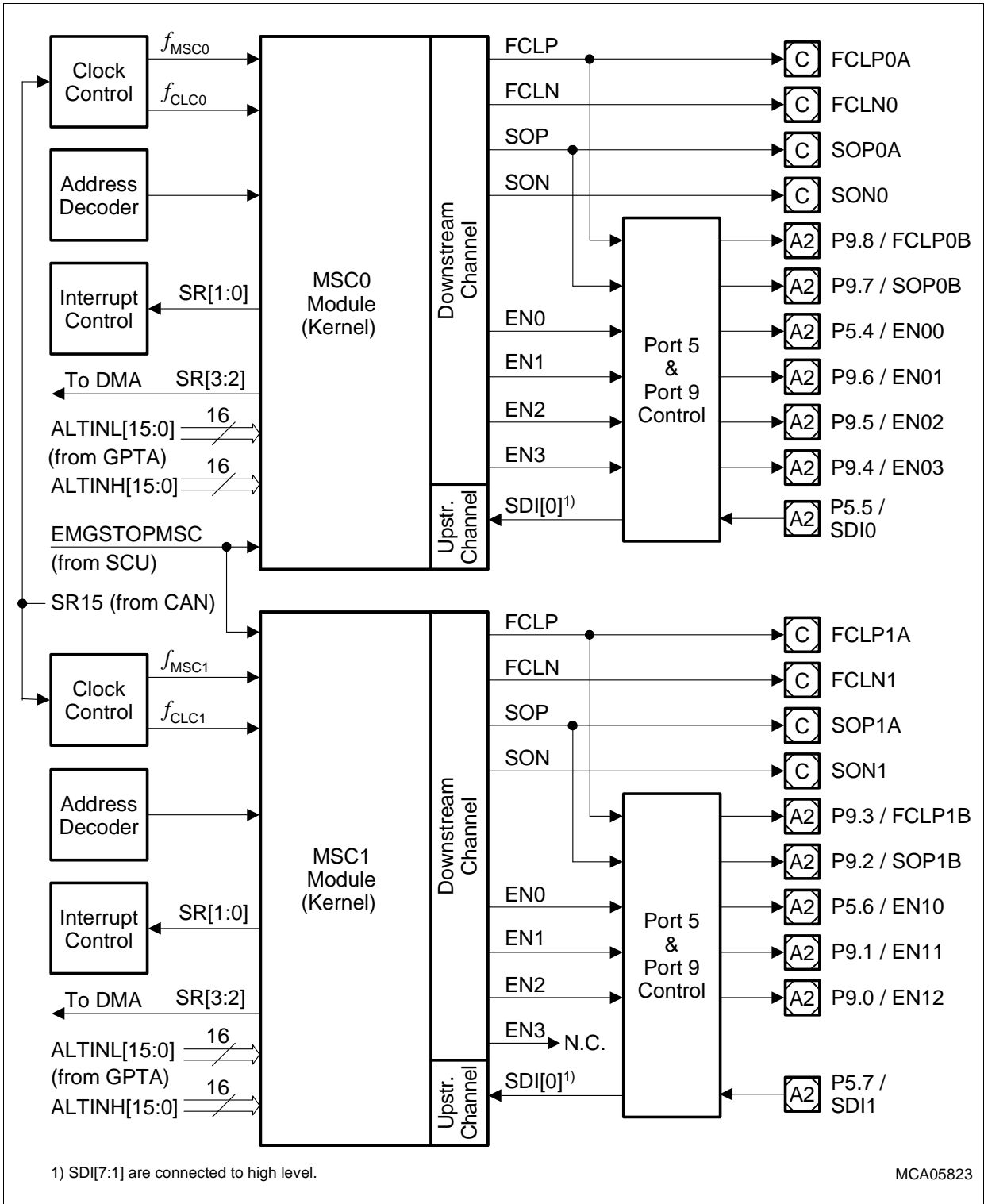


Figure 21-29 MSC0 and MSC1 Module Implementation and Interconnections

21.3.2 MSC0/MSC1 Module-Related External Registers

Figure 21-30 summarizes the module-related external registers which are required for programming of MSC0 and MSC1 (see also **Figure 21-28** for the module kernel specific registers). These registers are described in the following sections.

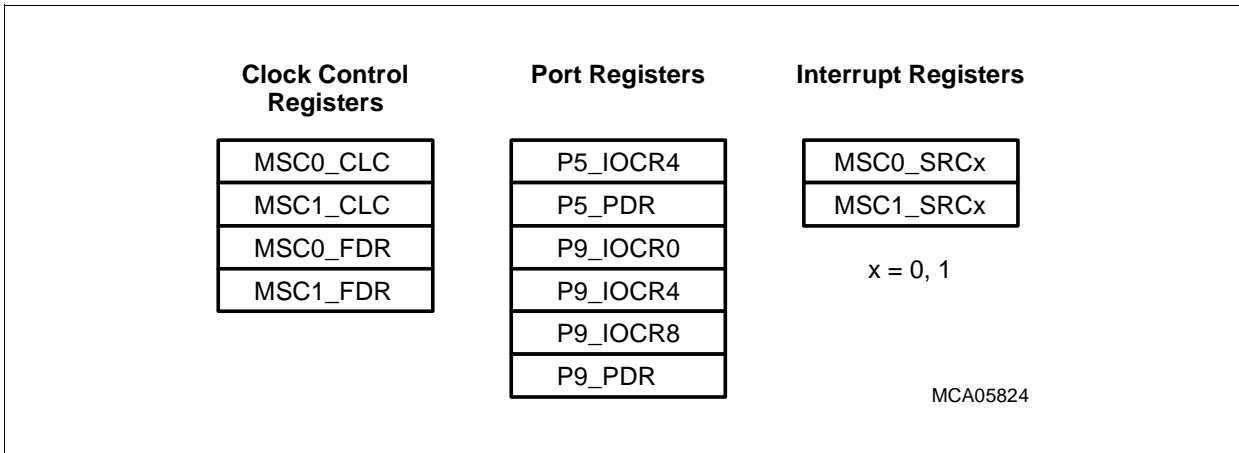


Figure 21-30 MSC Implementation-specific Special Function Registers

21.3.3 Clock Control

The MSC modules are provided both with two independent clock signals (**Figure 21-31**):

- f_{CLC0} and f_{CLC1}
These are the module clocks that are used inside the MSC kernel for control purposes such as clocking of control logic and register operations. The frequency of f_{CLC0} and f_{CLC1} is always identical to the system clock frequency f_{SYS} . The clock control registers MSC0_CLC and MSC1_CLC make it possible to enable/disable f_{CLC0} and f_{CLC1} under certain conditions.
- f_{MSC0} and f_{MSC1}
These clocks are the module clocks that are used inside the MSC for baud rate generation of the serial upstream and downstream channel. The fractional divider registers MSC0_FDR and MSC1_FDR control the frequency of f_{MSC0} and f_{MSC1} and make it possible to enable/disable it independently of f_{CLC0} and f_{CLC1} .

For module test purposes only, the service request output SR15 of the MultiCAN controller makes it possible to synchronize the fractional divider clock generation of both MSC modules to external events. This feature should not be used for normal MSC operation.

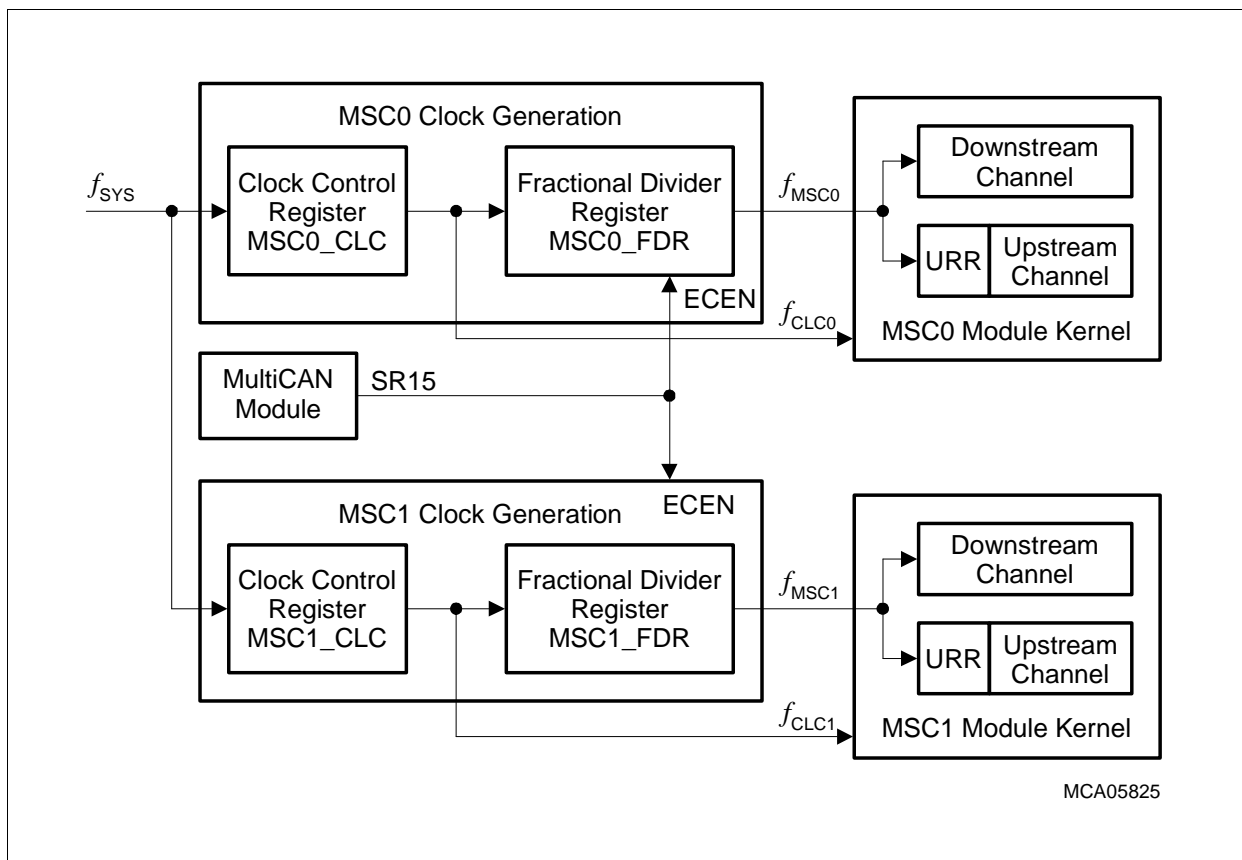


Figure 21-31 MSC Module Clock Generation

Micro Second Channel (MSC)

The following two formulas define the frequency of f_{MSC0} or f_{MSC1} ($x = 1, 0$):

$$f_{MSCx} = f_{SYS} \times \frac{1}{n} \text{ with } n = 1024 - MSCx.FDR.STEP \quad (21.3)$$

$$f_{MSCx} = f_{SYS} \times \frac{n}{1024} \text{ with } n = 0-1023 \quad (21.4)$$

Downstream Channel Baud Rate

As the clock signal FCL of the synchronous downstream channel is always half the frequency of f_{MSCx} , the resulting downstream channel baud rate is defined by:

$$\text{Baud rate}_{MSCx} = f_{SYS} \times \frac{1}{2 \times (1024 - MSCx.FDR.STEP)} \quad (21.5)$$

$$\text{Baud rate}_{MSCx} = f_{SYS} \times \frac{MSCx.FDR.STEP}{2 \times 1024} \quad (21.6)$$

Upstream Channel Baud Rate

The baud rate of the asynchronous upstream channel is derived from the module clock f_{MSCx} by a programmable clock divider selected by bit field MSCx_USR.URR (see also [Equation \(21.2\)](#) on [Page 21-25](#)). The divide factor DF can be at minimum 4 and at maximum 256.

$$\text{Baud rate}_{MSCx} = f_{SYS} \times \frac{1}{DF \times (1024 - MSCx.FDR.STEP)} \quad (21.7)$$

$$\text{Baud rate}_{MSCx} = f_{SYS} \times \frac{MSCx.FDR.STEP}{DF \times 1024} \quad (21.8)$$

[Equation \(21.3\)](#), [Equation \(21.5\)](#), and [Equation \(21.7\)](#) are valid for normal divider mode ($MSCx.FDR.DM = 01_B$). [Equation \(21.4\)](#), [Equation \(21.6\)](#), and [Equation \(21.8\)](#) are valid for fractional divider mode ($MSCx.FDR.DM = 10_B$).

Micro Second Channel (MSC)

21.3.3.1 Clock Control Register

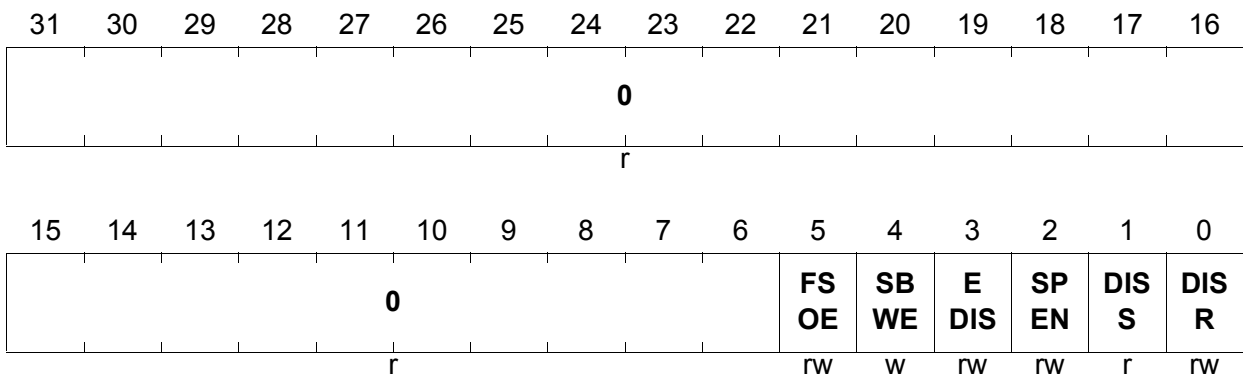
The Clock Control Registers allow the programmer to control (enable/disable) the clock signals to the MSC modules under certain conditions. The diagram below shows the clock control register functionality as is implemented for the MSC0 and MSC1 modules.

CLC

Clock Control Register

(00_H)

Reset Value: 0000 0003_H



Field	Bits	Type	Description
DISR	0	rw	Module Disable Request Bit Used for enable/disable control of the module.
DISS	1	r	Module Disable Status Bit Bit indicates the current status of the module.
SPEN	2	rw	Module Suspend Enable for OCDS Used to enable the suspend mode.
EDIS	3	rw	External Request Disable Used to control the external clock disable request.
SBWE	4	w	Module Suspend Bit Write Enable for OCDS Determines whether SPEN and FSOE are write-protected.
FSOE	5	rw	Fast Switch Off Enable Used for fast clock switch off in suspend mode.
0	[31:6]	r	Reserved Read as 0; should be written with 0.

Note: After a hardware reset operation, the f_{CLCx} and f_{MSCx} clocks are switched off and the MSC modules are disabled (DISS set).

Micro Second Channel (MSC)

2. Additional details of the clock control register functionality are described in section **“Clock Control Register CLC” on Page 3-24** of the *TC1796 User’s Manual System Units part (Volume 1)*.

21.3.3.2 Fractional Divider Register

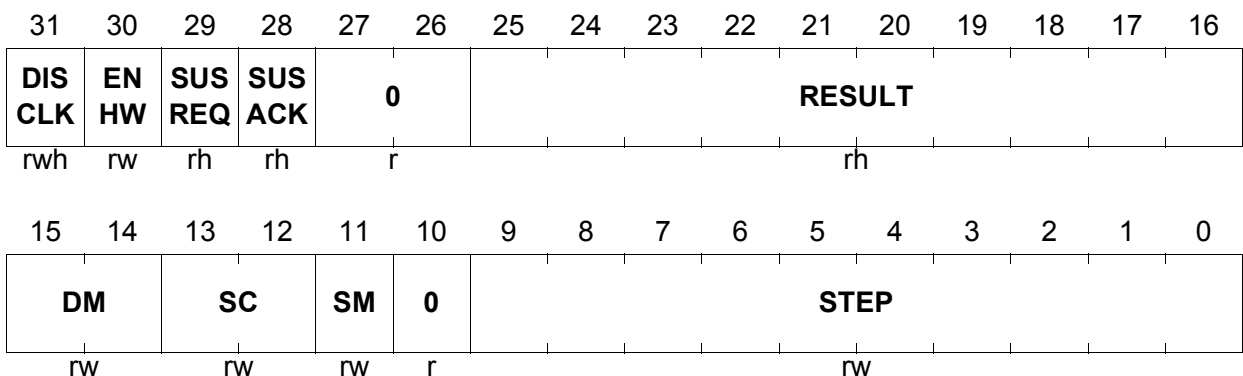
The Fractional Divider Registers control the clock rate of the shift clock f_{MSC0} and f_{MSC1} . Each MSC module has its own fractional divider register.

FDR

Fractional Divider Register

(0C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
STEP	[9:0]	rw	Step Value Reload or addition value for RESULT.
SM	11	rw	Suspend Mode SM selects between granted or immediate suspend mode.
SC	[13:12]	rw	Suspend Control This bit field determines the behavior of the fractional divider in suspend mode.
DM	[15:14]	rw	Divider Mode DM selects normal or fractional divider mode.
RESULT	[25:16]	rh	Result Value Bit field for the addition result.
SUSACK	28	rh	Suspend Mode Acknowledge Indicates state of SPNDACK signal.
SUSREQ	29	rh	Suspend Mode Request Indicates state of SPND signal.

Micro Second Channel (MSC)

Field	Bits	Type	Description
ENHW	30	rw	Enable Hardware Clock Control Controls operation of ECEN input and DISCLK bit.
DISCLK	31	rwh	Disable Clock Hardware controlled disable for f_{OUT} signal.
0	10, [27:26]	rw	Reserved Read as 0; should be written with 0.

*Note: Further details of the fractional divider register functionality are described in section **“Fractional Divider Operation”** on **Page 3-29** of the TC1796 User’s Manual System Units part (Volume 1).*

Micro Second Channel (MSC)

21.3.4 Port Control

MSC0 and MSC1 clock and data output lines are connected to dedicated differential output drivers. Some of the MSC module I/O lines are connected to I/O ports and therefore controlled in the port logic (see also [Figure 21-29](#)). The following port control operations selections must be executed for these I/O lines:

- Input/output function selection (IOCR registers)
- Pad driver characteristics selection for the outputs (PDR registers)

21.3.4.1 Input/Output Function Selection

The port input/output control registers contain the bit fields that select the digital output and input driver characteristics such as port direction (input/output) with alternate output selection, pull-up/down devices, and open-drain selections. The I/O lines for the MSC modules are controlled by the Port 5 and Port 9 input/output control registers.

[Table 21-10](#) shows in an overview how bits and bit fields must be programmed for the required I/O functionality of the MSC I/O lines.

Table 21-10 MSC0 and MSC1 I/O Line Selection and Setup

Module	Port Lines	Input/Output Control Register Bits ¹⁾	I/O
MSC0	P5.4 / EN00	P5_IOCR4.PC4 = 1X11 _B	Output
	P5.5 / SDI0 ²⁾	P5_IOCR4.PC5 = 0XXX _B	Input
	P9.4 / EN03	P9_IOCR4.PC4 = 1X11 _B	Output
	P9.5 / EN02	P9_IOCR4.PC5 = 1X11 _B	Output
	P9.6 / EN01	P9_IOCR4.PC6 = 1X11 _B	Output
	P9.7 / SOP0B	P9_IOCR4.PC7 = 1X11 _B	Output
	P9.8 / FCLP0B	P9_IOCR8.PC8 = 1X11 _B	Output
MSC1³⁾	P5.6 / EN10	P5_IOCR4.PC6 = 1X11 _B	Output
	P5.7 / SDI1 ²⁾	P5_IOCR4.PC7 = 0XXX _B	Input
	P9.0 / EN12	P9_IOCR8.PC0 = 1X11 _B	Output
	P9.1 / EN11	P9_IOCR8.PC1 = 1X11 _B	Output
	P9.2 / SOP1B	P9_IOCR8.PC2 = 1X11 _B	Output
	P9.3 / FCLP1B	P9_IOCR8.PC3 = 1X11 _B	Output

1) Possible Px bit field combinations see [Table 21-11](#).

2) For the upstream channel serial data inputs, additionally bit fields MSC0_OCR.SDISEL (for SDI0) and MSC1_OCR.SDISEL (for MSC1) must be set to 000_B.

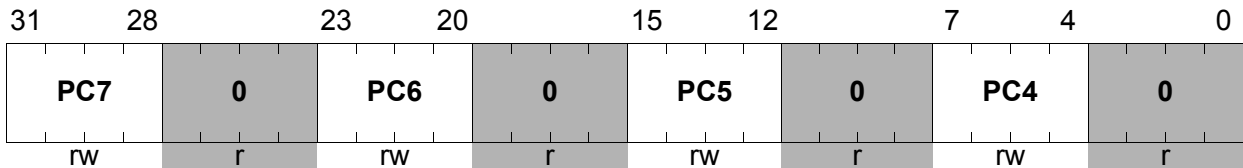
3) Chip enable output 3 (EN3) of the MSC1 module is not connected in the TC1796.

Input/Output Control Registers

P5_IOCRA4

Port 5 Input/Output Control Register 4 (14_H)

Reset Value: 2020 2020_H



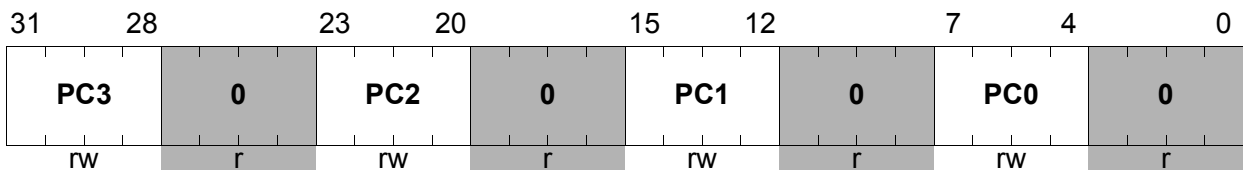
Field	Bits	Type	Description
PC4, PC5, PC6, PC7	[7:4], [15:12], [23:20], [31:28]	rw	Port Input/Output Control for Port 5.[7:4]¹⁾ Port input/output control for P5.4/EN00 Port input/output control for P5.5/SDI0 Port input/output control for P5.6/EN10 Port input/output control for P5.7/SDI1

1) Coding of bit field see [Table 21-11](#). Shaded bits and bit fields are “don’t care” for MSC I/O port control.

P9_IOCRO0

Port 9 Input/Output Control Register 0 (10_H)

Reset Value: 2020 2020_H



Field	Bits	Type	Description
PC0, PC1, PC2, PC3	[7:4], [15:12], [23:20], [31:28]	rw	Port Input/Output Control for Port 9 Pins 1-3¹⁾ Port input/output control for P9.0/EN12 Port input/output control for P9.1/EN11 Port input/output control for P9.2/SOP1B Port input/output control for P9.3/FCLP1B

1) Coding of bit field see [Table 21-11](#). Shaded bits and bit fields are “don’t care” for MSC I/O port control.

PCx Coding Table

Table 21-11 PCx Coding

PCx[3:0]	I/O	Output Characteristics	Selected Pull-up/Pull-down/ Selected Output Function
0X00 _B	Input	–	No pull device connected
0X01 _B			Pull-down device connected
0X10 _B ¹⁾			Pull-up device connected
0X11 _B			No pull device connected
1011 _B	Output	Push-pull	Output function ALT3
1111 _B		Open-drain	Output function ALT3

1) This bit field value is default after reset.

Micro Second Channel (MSC)

21.3.4.2 Pad Driver Mode Registers

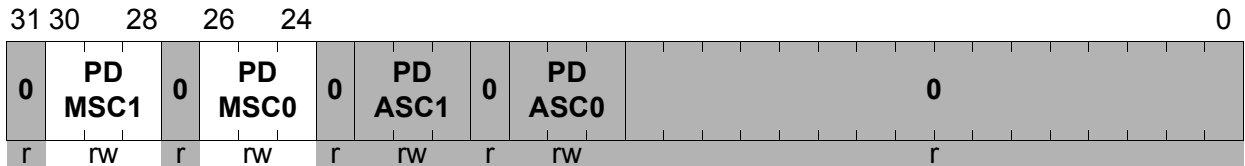
The Port 5 and Port 9 pad driver mode registers contain bit fields that determine the output driver strength and the slew rate of MSC output lines.

P5_PDR

Port 5 Pad Driver Mode Register

(40_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PDMSC0	[26:24]	rw	Pad Driver Mode for P5.4/EN00 ¹⁾
PDMSC1	[30:28]	rw	Pad Driver Mode for P5.6/EN10 ¹⁾

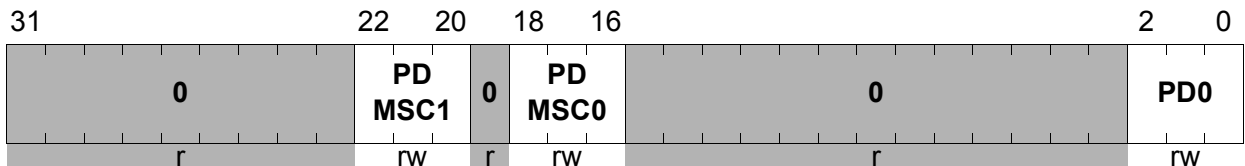
1) Coding of bit field see [Table 21-12](#). Shaded bits and bit fields are “don’t care” for MSC I/O port control.

P9_PDR

Port 9 Pad Driver Mode Register

(40_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PD0	[2:0]	rw	Pad Driver Mode for P9.8/FCLP0B ¹⁾
PDMSC0	[18:16]	rw	Pad Driver Mode for P9.7/SOP0B, P9.6/EN01, P9.5/EN02, P9.4/EN03 ¹⁾
PDMSC1	[22:20]	rw	Pad Driver Mode for P9.3/FCLP1B, P9.2/SOP1B, P9.1/EN11, P9.0/EN12 ¹⁾

1) Coding of bit field see [Table 21-12](#). Shaded bits and bit fields are “don’t care” for MSC I/O port control.

PDx Selection Table

Table 21-12 Pad Driver Mode Mode Selection (Class A2 Pads)

PDx Bit Field	Driver Strength	Signal Transitions
000 _B	Strong driver	Sharp edge ¹⁾
001 _B		Medium edge ¹⁾
010 _B		Soft edge ¹⁾
011 _B	Weak driver	–
100 _B	Medium driver	–
101 _B		–
110 _B		–
111 _B	Weak driver	–

1) In strong driver mode, the output driver characteristics of class A2 pads can be additionally controlled by the temperature compensation logic.

21.3.5 On-Chip Connections

This section describes the on-chip connections of the MSC0/MSC1 modules.

21.3.5.1 EMGSTOPMSC Signal (from SCU)

The emergency stop input signals EMGSTOPMSC of both MSC modules are connected to the output signal of the emergency stop input control logic. This logic is located in the SCU. Its functionality is controlled by the SCU emergency stop register.

Note: Further details of the emergency stop capabilities are described in [Section 5.10](#) on [Page 5-57](#) of the TC1796 System Units User's Manual (Volume 1).

21.3.5.2 ALTINH and ALTINL Connections

In the TC1796, output lines of GPTA0, GPTA1, and LTCA2 are connected to ALTINH and ALTINL inputs of the MSC0/MSC1 downstream channels as shown in [Figure 21-32](#). The setting of the GPTA-to-MSC multiplexer determines which output of the three GPTA sub-modules is connected to a specific ALTINL/ALTINH line. Details about the functionality of the GPTA-to-MSC multiplexer is given in the GPTA chapter, section [“GPTA-to-MSC Multiplexer”](#) on [Page 24-257](#).

Please note that the upper eight MSC1 input lines ALTINH[15:8] are connected to logic 0 level.

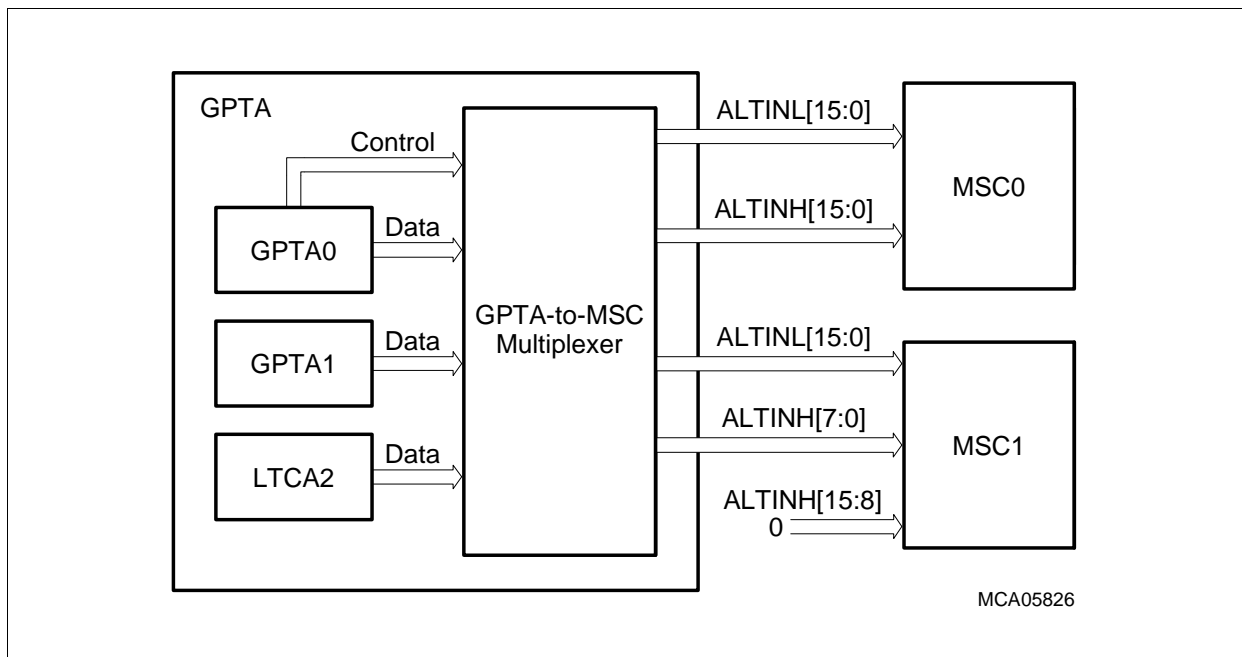


Figure 21-32 ALTINL/ALTINH Connections of the MSC0/MSC1

21.3.5.3 DMA Controller Service Requests

Two service request outputs (SR[3:2]) of each MSC module are connected as DMA request input to the DMA controller. The DMA request lines are connected to the DMA controller as shown in [Table 21-13](#).

Table 21-13 Service Request Lines of MSC0/MSC1

Module	Service Request Line	Connected To	Description
MSC0	SR0	MSC0_SRC0	MSC0 Service Request Node 0
	SR1	MSC0_SRC1	MSC0 Service Request Node 1
	SR2	CH06_REQI2	DMA Channel 06 Request Input 2
	SR3	CH07_REQI2 CH14_REQI5 CH16_REQI6	DMA Channel 07 Request Input 2 DMA Channel 14 Request Input 5 DMA Channel 16 Request Input 6
MSC1	SR0	MSC1_SRC0	MSC1 Service Request Node 0
	SR1	MSC1_SRC1	MSC1 Service Request Node 1
	SR2	CH06_REQI3	DMA Channel 06 Request Input 3
	SR3	CH07_REQI3 CH14_REQI6 CH17_REQI6	DMA Channel 07 Request Input 3 DMA Channel 14 Request Input 6 DMA Channel 17 Request Input 6

21.3.6 Interrupt Control Registers

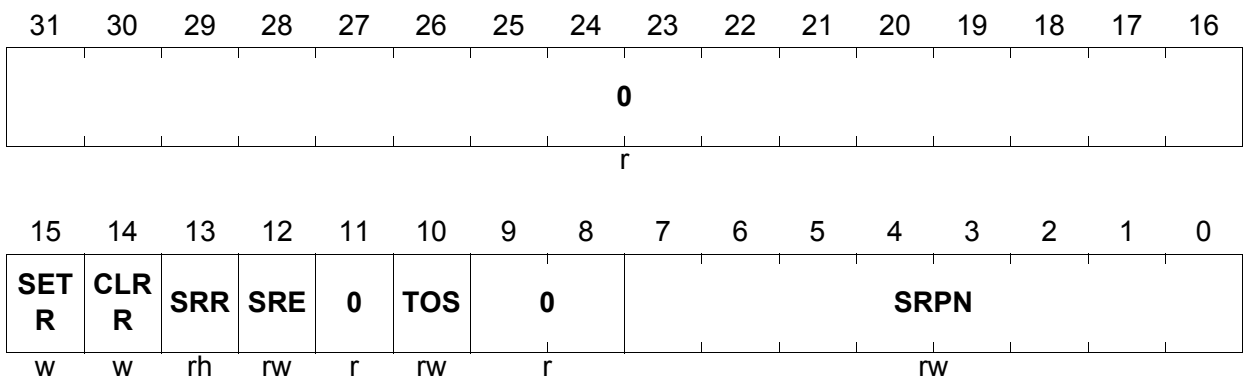
In the TC1796, the two service request outputs SR[1:0] of each MSC module are connected to one interrupt node. The upper two service request outputs SR[3:2] of each MSC module are not connected to interrupt nodes, but can be used as DMA requests (see [Table 21-13](#)).

SRC1

Service Request Control Register 1 (F8_H) **Reset Value: 0000 0000_H**

SRC0

Service Request Control Register 0 (FC_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
SRPN	[7:0]	rw	Service Request Priority Number
TOS	10	rw	Type of Service Control
SRE	12	rw	Service Request Enable
SRR	13	rh	Service Request Flag
CLRR	14	w	Request Clear Bit
SETR	15	w	Request Set Bit
0	[9:8], 11, [31:16]	r	Reserved Read as 0; should be written with 0.

Note: Additional details on service request nodes and the service request control registers are described on [Page 14-3](#) of the TC1796 User's Manual System Units part (Volume 1).

22 Controller Area Network (MultiCAN) Controller

This chapter describes the MultiCAN controller of the TC1796. It contains the following sections:

- CAN basics, including TTCAN (see [Page 22-2](#))
- Overview of the CAN Module in the TC1796 (see [Page 22-14](#))
- Functional description of the MultiCAN Kernel (see [Page 22-18](#))
- MultiCAN Kernel register description (see [Page 22-58](#))
- Functional description of the TTCAN extension (see [Page 22-114](#))
- TTCAN extension register description (see [Page 22-154](#))
- TC1796 implementation-specific details (port connections and control, interrupt control, address decoding, clock control, see [Page 22-199](#)).

Note: The MultiCAN/TTCAN register names described in this chapter will be referenced in the TC1796 User's Manual by the module name prefix "CAN_".

Controller Area Network (MultiCAN) Controller**22.1 CAN Basics**

CAN is an asynchronous serial bus system with one logical bus line. It has an open, linear bus structure with equal bus participants called nodes. A CAN bus consists of two or more nodes.

The bus logic corresponds to a “wired-AND” mechanism. **Recessive** bits (equivalent to the logic1 level) are overwritten by **dominant** bits (logic 0 level). As long as no bus node is sending a dominant bit, the bus is in the recessive state. In this state, a dominant bit from any bus node generates a dominant bus state. The maximum CAN bus speed is, by definition, 1 Mbit/s. This speed limits the CAN bus to a length of up to 40 m. For bus lengths longer than 40 m, the bus speed must be reduced.

The binary data of a CAN frame is coded in NRZ code (Non-Return-to-Zero). To ensure re-synchronization of all bus nodes, bit-stuffing is used. This means that during the transmission of a message, a maximum of five consecutive bits can have the same polarity. Whenever five consecutive bits of the same polarity have been transmitted, the transmitter will insert one additional bit (stuff bit) of the opposite polarity into the bit stream before transmitting further bits. The receiver also checks the number of bits with the same polarity and removes the stuff bits from the bit stream (= destuffing).

22.1.1 Addressing and Bus Arbitration

In the CAN protocol, address information is defined in the identifier field of a message. The identifier indicates the contents of the message and its priority. The lower the binary value of the identifier, the higher is the priority of the message.

For bus arbitration, CSMA/CD with NDA (Carrier Sense Multiple Access/Collision Detection with Non-Destructive Arbitration) is used. If bus node A attempts to transmit a message across the network, it first checks that the bus is in the idle state (“Carrier Sense”) i.e. no node is currently transmitting. If this is the case (and no other node wishes to start a transmission at the same moment), node A becomes the bus master and sends its message. All other nodes switch to receive mode during the first transmitted bit (Start-Of-Frame bit). After correct reception of the message (acknowledged by each node), each bus node checks the message identifier and stores the message, if required. Otherwise, the message is discarded.

If two or more bus nodes start their transmission at the same time (“Multiple Access”), bus collision of the messages is avoided by bit-wise arbitration (“Collision Detection / Non-Destructive Arbitration” together with the “Wired-AND” mechanism, dominant bits override recessive bits). Each node that sends also reads back the bus level. When a recessive bit is sent but a dominant one is read back, bus arbitration is lost and the transmitting node switches to receive mode. This condition occurs for example when the message identifier of a competing node has a lower binary value and therefore the competing node is sending a message with a higher priority. In this way, the bus node with the highest priority message wins arbitration without losing time by having to repeat the message. Other nodes that lost arbitration automatically try to repeat their

Controller Area Network (MultiCAN) Controller

transmission once the bus returns to idle state. Therefore, the same identifier can be sent in a Data Frame only by one node in the system. There must not be more than one node programmed to send Data Frames with the same identifier.

Standard message identifier have a length of 11 bits. CAN specification 2.0B extends the message identifier lengths to 29 bits, the extended identifier.

22.1.2 CAN Frame Formats

There are three types of CAN frames:

- Data Frames
- Remote Frames
- Error Frames

A Data Frame contains a Data Field of 0 to 8 bytes length. A Remote Frame contains no Data Field and is typically generated as a request for data (e.g. from a sensor). Data and Remote Frames can use an 11-bit “Standard” identifier or a 29-bit “Extended” identifier. An Error Frame can be generated by any node that detects a CAN bus error.

22.1.2.1 Data Frames

There are two types of Data Frames defined (see [Figure 22-1](#)):

- Standard Data Frame
- Extended Data Frame

Standard Data Frame

A Data Frame begins with the Start-Of-Frame bit (SOF = dominant level) for hard synchronization of all nodes. The SOF is followed by the Arbitration Field consisting of 12 bits, the 11-bit Identifier (reflecting the contents and priority of the message), and the RTR (Remote Transmission Request) bit. With RTR at dominant level, the frame is marked as Data Frame. With RTR at recessive level, the frame is defined as a Remote Frame.

The next field is the Control Field consisting of 6 bits. The first bit of this field is the IDE (Identifier Extension) bit and is at dominant level for the Standard Data Frame. The following bit is reserved and defined as a dominant bit. The remaining 4 bits of the Control Field are the Data Length Code (DLC) that specifies the number of bytes in the Data Field. The Data Field can be 0 to 8 bytes wide. The Cyclic Redundancy (CRC) Field that follows the data bytes is used to detect possible transmission errors. It consists of a 15-bit CRC sequence, completed by a recessive CRC delimiter bit.

The final field is the Acknowledge Field. During the ACK Slot, the transmitting node sends out a recessive bit. Any node that has received an error free frame acknowledges the correct reception of the frame by sending back a dominant bit, regardless of whether or not the node is configured to accept that specific message. This behavior assigns the

Controller Area Network (MultiCAN) Controller

CAN protocol to the “in-bit-response” group of protocols. The recessive ACK delimiter bit, which must not be overwritten by a dominant bit, completes the Acknowledge Field. Seven recessive End-of-Frame (EOF) bits finish the Data Frame. Between any two consecutive frames, the bus must remain in the recessive state for at least 3 bit times (called Inter Frame Space). If after the Inter Frame Space, no other nodes attempt to transmit the bus remains in idle state with a recessive level.

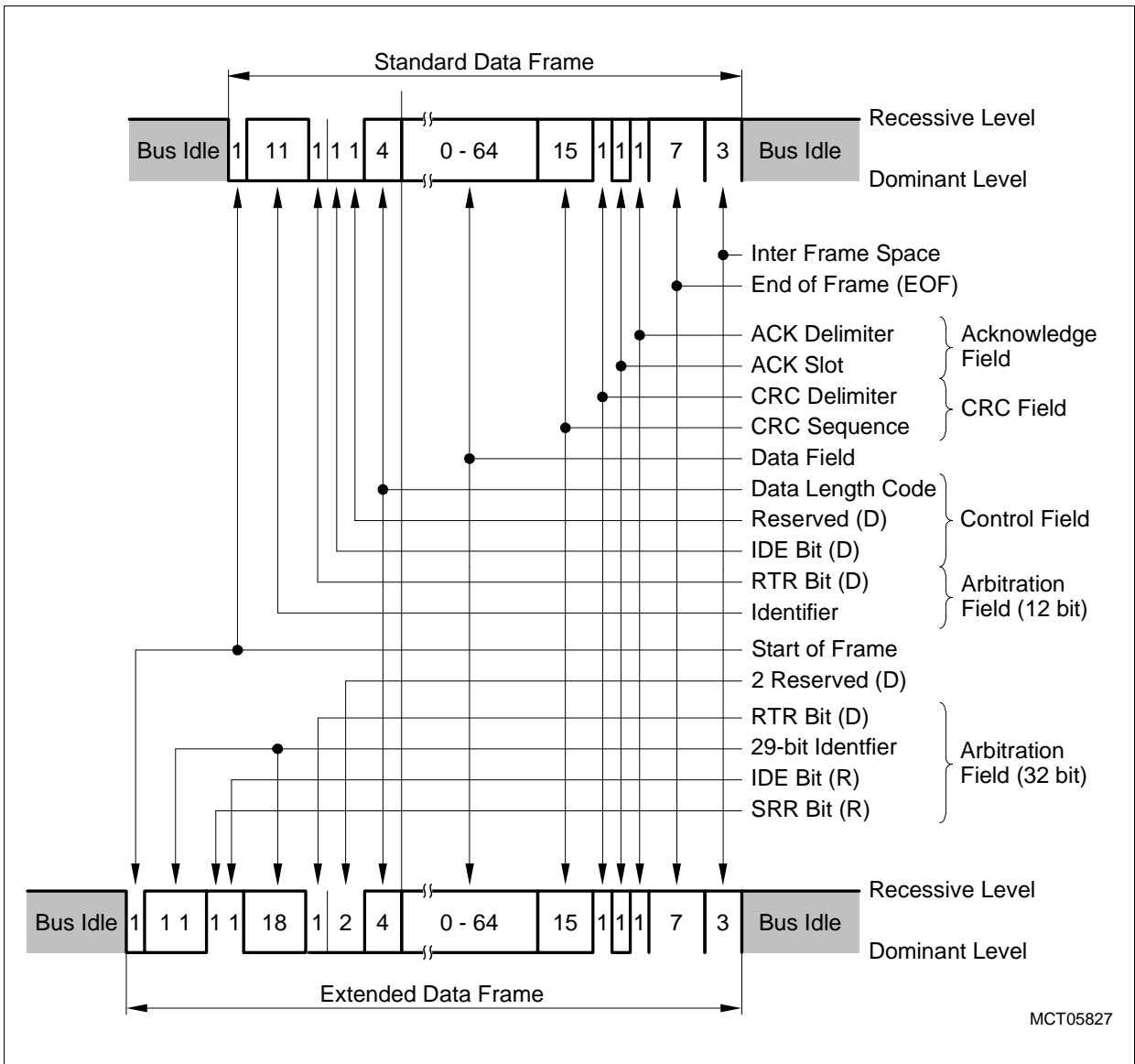


Figure 22-1 CAN Data Frame

Controller Area Network (MultiCAN) Controller**Extended Data Frame**

In the Extended CAN Data Frame, the message identifier of the standard frame has been extended to 29-bit. A split of the extended identifier into two parts, an 11-bit least significant section (as in standard CAN frame) and an 18-bit most significant section, ensures that the Identifier Extension bit (IDE) can remain at the same bit position in both standard and extended frames.

In the Extended CAN Data Frame, the SOF bit is followed by the 32-bit Arbitration Field. The first 11 bits are the least significant bits of the 29-bit Identifier ("Base-ID"). These 11 bits are followed by the recessive Substitute Remote Request (SRR) bit. The SRR is further followed by the recessive IDE bit, which indicates the frame to be an Extended CAN frame. If arbitration remains unresolved after transmission of the first 11 bits of the identifier, and if one of the nodes involved in arbitration is sending a Standard CAN frame, then the Standard CAN frame will win arbitration due to the assertion of its dominant IDE bit. Therefore, the SRR bit in an Extended CAN frame is recessive to allow the assertion of a dominant RTR bit by a node that is sending a Standard CAN Remote Frame. The SRR and IDE bits are followed by the remaining 18 bits of the extended identifier and the RTR bit.

Control field and frame termination is identical to the Standard Data Frame.

22.1.2.2 Remote Frames

Normally, data transmission is performed on an autonomous basis with the data source node (e.g. a sensor) sending out a Data Frame. It is also possible, however, for a destination node (or nodes) to request the data from the source. For this purpose, the destination node sends a Remote Frame with an identifier that matches the identifier of the required Data Frame. The appropriate data source node will then send a Data Frame as a response to this remote request.

There are 2 differences between a Remote Frame and a Data Frame.

- The RTR bit is in the recessive state in a Remote Frame.
- There is no Data Field in a Remote Frame.

If a Data Frame and a Remote Frame with the same identifier are transmitted at the same time, the Data Frame wins arbitration due to the dominant RTR bit following the identifier. In this way, the node that transmitted the Remote Frame receives the requested data immediately. The format of a Standard and Extended Remote Frames is shown in [Figure 22-2](#).

Controller Area Network (MultiCAN) Controller

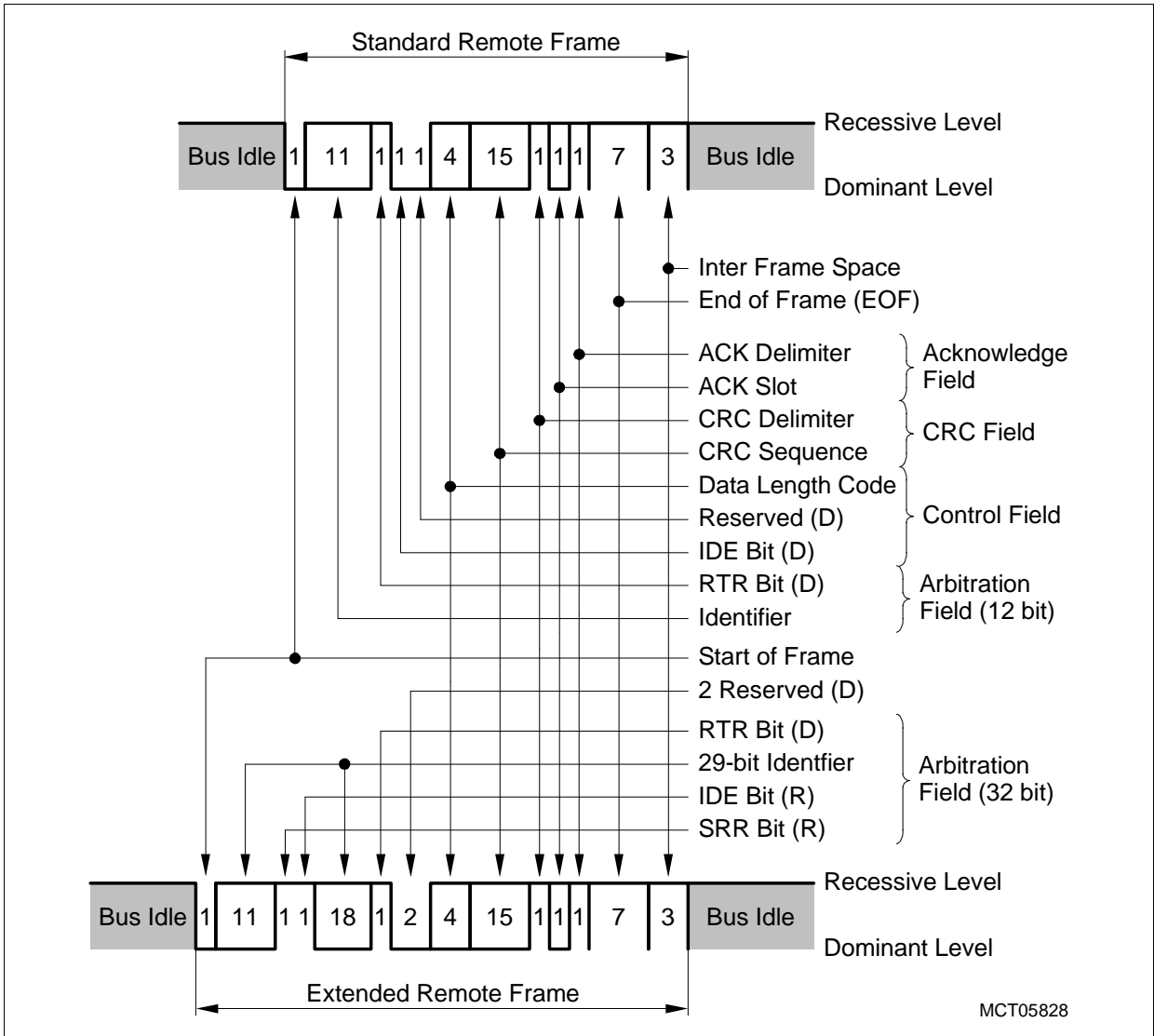


Figure 22-2 CAN Remote Frame

Controller Area Network (MultiCAN) Controller

22.1.2.3 Error Frames

An Error Frame is generated by any node that detects a bus error. An Error Frame consists of two fields, an Error Flag field followed by an Error Delimiter field. The Error Delimiter Field consists of 8 recessive bits and allows the bus nodes to restart bus communications after an error. There are, however, two forms of Error Flag fields. The form of the Error Flag field depends on the error status of the node that detects the error.

When an error-active node detects a bus error, the node generates an Error Frame with an active-error flag. The error-active flag is composed of six consecutive dominant bits that actively violate the bit-stuffing rule. All other stations recognize a bit-stuffing error and generate Error Frames themselves. The resulting Error Flag field on the CAN bus therefore consists of six to twelve consecutive dominant bits (generated by one or more nodes). The Error Delimiter field completes the Error Frame. After completion of the Error Frame, bus activity returns to normal and the interrupted node attempts to re-send the aborted message.

If an error-passive node detects a bus error, the node transmits an error-passive flag followed, again, by the Error Delimiter field. The error-passive flag consists of six consecutive recessive bits, and therefore the Error Frame (for an error-passive node) consists of 14 recessive bits (i.e. no dominant bits). Therefore, unless the bus error is detected by the node that is actually transmitting (i.e. is the bus master), the transmission of an Error Frame by an error-passive node will not affect any other node on the network. If the bus master node generates an error-passive flag, this may cause other nodes to generate Error Frames due to the resulting bit-stuffing violation. After transmission of an Error Frame an error-passive node must wait for 6 consecutive recessive bits on the bus before attempting to rejoin bus communications.

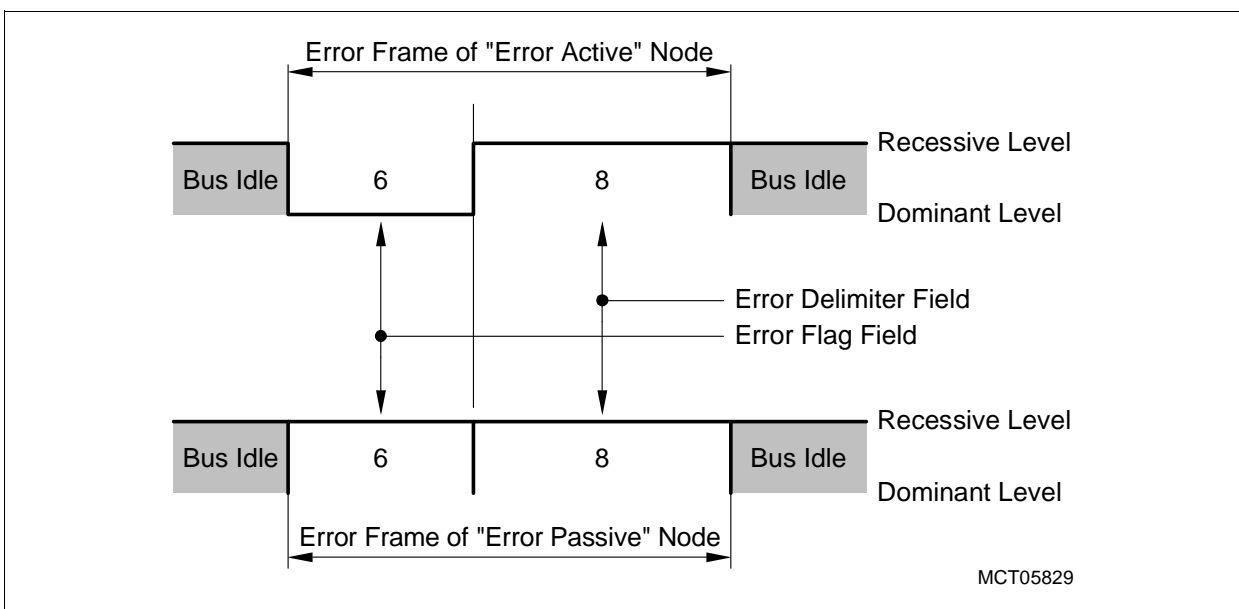


Figure 22-3 CAN Error Frames

Controller Area Network (MultiCAN) Controller

22.1.3 The Nominal Bit Time

One bit cell (this means one high or low pulse of the NRZ code) is composed by four segments. Each segment is an integer multiple of Time Quanta t_Q . The Time Quanta is the smallest discrete timing resolution used by a CAN node. The nominal bit time definition with its segments is shown in **Figure 22-4**.

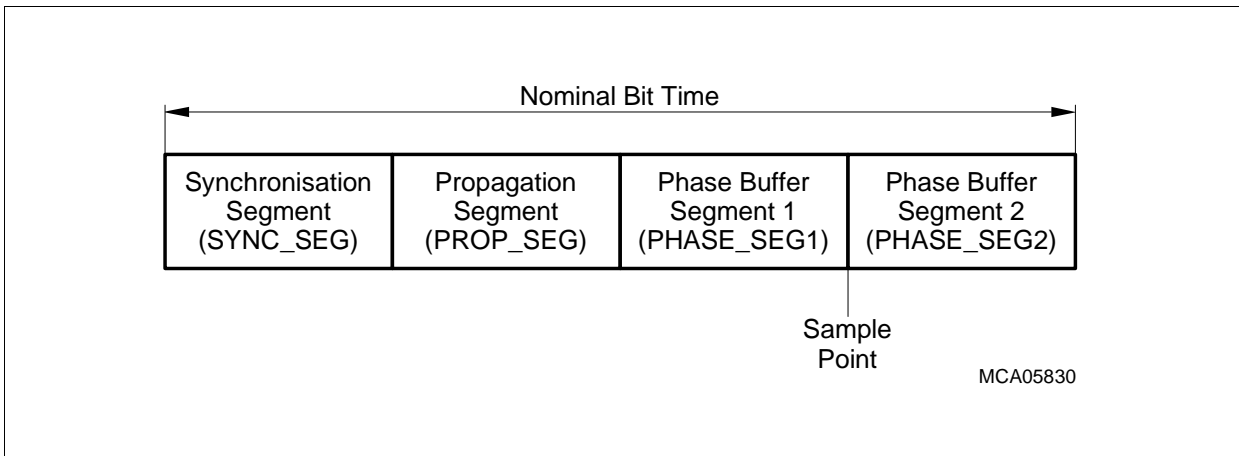


Figure 22-4 Partition of Nominal Bit Time

The Synchronization Segment (SYNC_SEG) is used to synchronize the various bus nodes. If there is a bit state change between the previous bit and the current bit, the bus state change is expected to occur within this segment. The length of this segment is always $1 t_Q$.

The Propagation Segment (PROP_SEG) is used to compensate for signal delays across the network. These delays are caused by signal propagation delay on the bus line and through the electronic interface circuits of the bus nodes.

The Phase Segments 1 and 2 (PHASE_SEG1, PHASE_SEG2) are used to compensate for edge phase errors. These segments can be lengthened or shortened by re-synchronization. PHASE_SEG2 is reserved for calculation of the subsequent bit level, and is $\geq 2 t_Q$. At the sample point, the bus level is read and interpreted as the value of the bit cell. It occurs at the end of PHASE_SEG1.

The total number of t_Q in a bit time is specified to be between 8 and 25.

As a result of re-synchronization, PHASE_SEG1 can be lengthened or PHASE_SEG2 can be shortened. The amount of lengthening or shortening the phase buffer segments has an upper limit given by the re-synchronization jump width. The re-synchronization jump width may be between 1 and $4 t_Q$, but it may not be longer than PHASE_SEG1.

Controller Area Network (MultiCAN) Controller**22.1.4 Error Detection and Error Handling**

The CAN protocol has sophisticated error detection mechanisms. The following errors can be detected:

- **Cyclic Redundancy Check (CRC) Error**
With the Cyclic Redundancy Check, the transmitter calculates special check bits for the bit sequence from the start of a frame until the end of the Data Field. This CRC sequence is transmitted in the CRC Field. The receiving node also calculates the CRC sequence using the same formula, and performs a comparison to the received sequence. If a mismatch is detected, a CRC error has occurred and an Error Frame is generated. The message is repeated.
- **Acknowledge Error**
In the Acknowledge Field of a message, the transmitter checks whether a dominant bit is read during the Acknowledge Slot (that is sent out as a recessive bit). If not, no other node has received the frame correctly, an Acknowledge Error has occurred, and the message must be repeated. No Error Frame is generated.
- **Form Error**
If a transmitter detects a dominant bit in one of the four segments End of Frame, Interframe Space, Acknowledge Delimiter, or CRC Delimiter, a Form Error has occurred, and an Error Frame is generated. The message is repeated.
- **Bit Error**
A Bit Error occurs if a) a transmitter sends a dominant bit and detects a recessive bit or b) if the transmitter sends a recessive bit and detects a dominant bit when monitoring the actual bus level and comparing it to the just transmitted bit. In case b), no error occurs during the Arbitration Field (ID, RTR, IDE) and the Acknowledge Slot.
- **Stuff Error**
If between Start of Frame and CRC Delimiter, six consecutive bits with the same polarity are detected, the bit-stuffing rule has been violated. A stuff error occurs and an Error Frame is generated. The message is repeated.

Detected errors are made public to all other nodes via Error Frames (except Acknowledge Errors). The transmission of the erroneous message is aborted and the frame is repeated as soon as possible. Furthermore, each CAN node is in one of the three error states (error-active, error-passive or bus-off) according to the value of the internal error counters. The error-active state is the usual state where the bus node can transmit messages and active-error frames (made of dominant bits) without any restrictions. In the error-passive state, messages and passive-error frames (made of recessive bits) may be transmitted. The bus-off state makes it temporarily impossible for the node to participate in the bus communication. During this state, messages can be neither received nor transmitted.

22.1.5 Different CAN Implementations

Basic-CAN and Full-CAN

There is one more CAN characteristic that is related to the interface of a CAN module (controller) and the host CPU: Basic-CAN and Full-CAN functionality.

In Basic-CAN devices, only basic functions of the protocol are implemented in hardware, such as the generation and the check of the bit stream. The decision, whether a received message has to be stored or not (acceptance filtering), and the complete message management must be done by software. Normally, the CAN device also provides only one transmit buffer and one or two receive buffers. Therefore, the host CPU load is quite high when using Basic-CAN modules. The main advantage of Basic-CAN is a reduced chip size leading to low costs of these devices.

Full-CAN devices (this is the case for the MultiCAN controller as implemented in TC1796) manage the whole bus protocol in hardware, including the acceptance filtering and message management. Full-CAN devices contain message objects that handle autonomously the identifier, the data, the direction (receive or transmit) and the information of Standard CAN/Extended CAN operation. During the initialization of the device, the host CPU determines which messages are to be sent and which are to be received. The host CPU is informed by interrupt if the identifier of a received message matches with one of the programmed (receive-) message objects. The CPU load of Full-CAN devices is greatly reduced. When using Full-CAN devices, high baud rates and high bus loads with many messages can be handled.

Controller Area Network (MultiCAN) Controller**22.1.6 TTCAN Basics**

In general, a CAN network is event-oriented due to the error handling and bus arbitration. The exact time for a message to be transferred is difficult to determine. In order to allow deterministic message transfers, the event-driven mechanism has been changed into a time-driven mechanism by a TTCAN extension, with “TT” standing for Time-Triggered.

TTCAN is a higher-layer protocol above the standard CAN protocol that synchronizes the communication schedules of all CAN nodes in a CAN network, and that provides a global system time. When CAN nodes are synchronized, messages can be transmitted at a specific time slot, without competing with other messages for the bus. Thus, the loss of arbitration is avoided, and the latency time becomes predictable. Apart from the synchronized communication schedule, TTCAN-compliant nodes operate according to the standard CAN protocol as defined by ISO 11898-4.

22.1.6.1 Time Reference

In TTCAN networks, a global time reference makes it possible to synchronize the activities of all CAN nodes. Each CAN node has its own local time represented by a counter that is incremented with each network time unit (NTU). The NTU is derived from the node’s local clock and local Time Unit Ratio (TUR).

In TTCAN, synchronization is achieved by a periodic transmission of a reference message. This reference message (transmitted by a time master) restarts the cycle time in each node. The reference message can be easily recognized by its identifier. Within TTCAN level 1, the reference message holds control information in one byte of the message Data Field. In TTCAN level 2, the reference message provides additional control information in four bytes of the message Data Field (e.g. the global time information of the current TTCAN time master) with downwards compatibility.

22.1.6.2 Basic Cycle

A reference message starts a new basic cycle and resets the cycle time. The cycle time counts the time since the start of the basic cycle. A basic cycle consists of several non-overlapping time windows of different sizes. The sequence of the time windows is described by time marks that determine when a time window begins. Time marks are compared to the cycle time, and particular actions are triggered when the cycle time matches the time mark.

Controller Area Network (MultiCAN) Controller

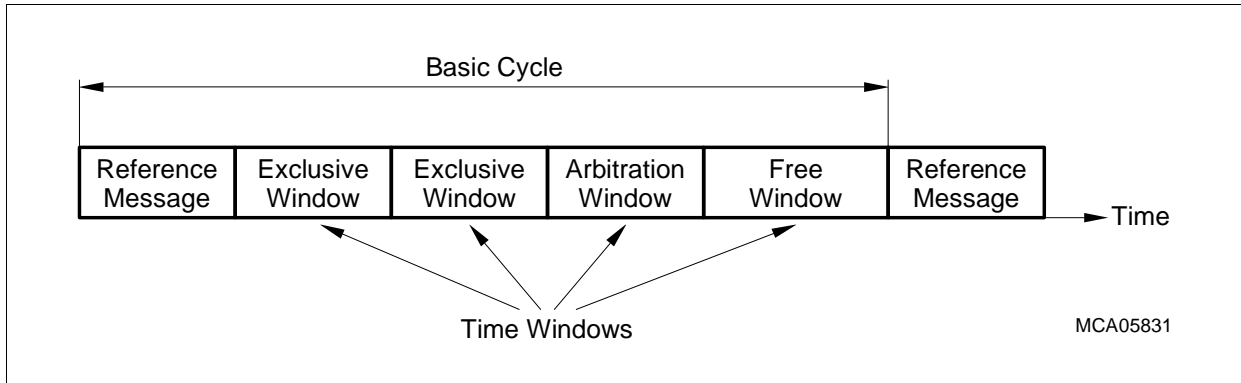


Figure 22-5 TTCAN Basic Cycle

The structure of the basic cycle is defined once for the whole TTCAN network. Several different basic cycles are combined to build a matrix cycle or the system matrix. The sequence of basic cycles in the matrix cycle is controlled by the reference messages.

All possible messages in the TTCAN system are assigned to time windows of the system matrix. A time window can be:

- An exclusive time window, in which only one particular message can be transmitted.
- An arbitrating time window, in which messages can arbitrate for bus access.
- A free time window, reserved for future extensions.

In exclusive time windows automatic retransmission of messages is disabled. This guarantees that messages are not delayed by other bus traffic.

22.1.6.3 Global System Time

Level 2 of TTCAN supports a global system time. The time master establishes its own local time as global time by transmitting reference message. The other CAN nodes in the network operate as time slaves that calculate their local time offset to the global time by comparing their local time with the received global time. To compensate for slightly different clock drifts in the CAN nodes and to provide a consistent view of the global time, the nodes can perform a drift compensation operation. They compare the length of the basic cycle in local time to the length of the basic cycle in global time. The difference (quotient) of the two values gives the adapted TUR.

22.1.6.4 The System Matrix

Typical CAN applications include many control loops and tasks with different periods. All of them need individual sending patterns for their information. One TTCAN basic cycle would not offer enough flexibility to satisfy this needs. Therefore, the TTCAN specification also makes it possible to use more than one basic cycle to build the communication matrix or system matrix. Several different basic cycles are connected together to build the matrix cycle. Most combinations of basic cycles are possible, such

Controller Area Network (MultiCAN) Controller

as sending every basic cycle, sending every second basic cycle, or sending only once within the matrix cycle.

22.1.6.5 Generation of the Network Time Unit (NTU)

The granularity of any timing information within TTCAN is the NTU. The cycle time is measured in NTU that is based on the nominal CAN bit time (TTCAN level 1), or on the Time Quantum t_Q (TTCAN level 2), or on the CAN input frequency. In level 2, the node's local relation between the physical oscillator of a TTCAN controller and the system wide NTU must be established. A clock signal in the node provides the system clock to a frequency divider. This frequency divider generates the system-wide NTU while a node local TUR takes care for the correct relationship between the system clock and NTU. NTU now can be used to calculate a local time and to build the global time.

22.1.6.6 Global Time Generation and Drift Correction

The node sending the reference message is the time master in the TTCAN network. The reference message includes the time master's (by definition correct) global time value for frame synchronization. All nodes take a snap-shot of their time values at the SOF bit sample point of the received reference message. After reference message reception, each node can calculate its local offset that indicates the difference between the master global time value and the own local time value. In the current basic cycle, the node can compute the global time for the next basic cycle by:

global time = local time + local offset

This equation ensures that all nodes have a consistent global time. Due to slightly different clock drifts of the different nodes in a CAN network, a mechanism is established that guarantees that local and global time run in parallel. This mechanism is the continuous update of the TUR. The initial value of TUR is locally known in the node by the node clock signal specification. During node operation, the node locally measures the length between two successive reference messages in number of oscillator periods and in local time. The quotient of these two values gives the actual TUR. The achievable precision determines a reasonable choice of the NTU value in physical seconds.

Controller Area Network (MultiCAN) Controller

22.2 Overview

This section describes the serial communication module called MultiCAN (CAN = Controller Area Network) with its Time-Triggered Extension module (called TTCAN). The MultiCAN module contains four independent CAN nodes, representing four serial communication interfaces.

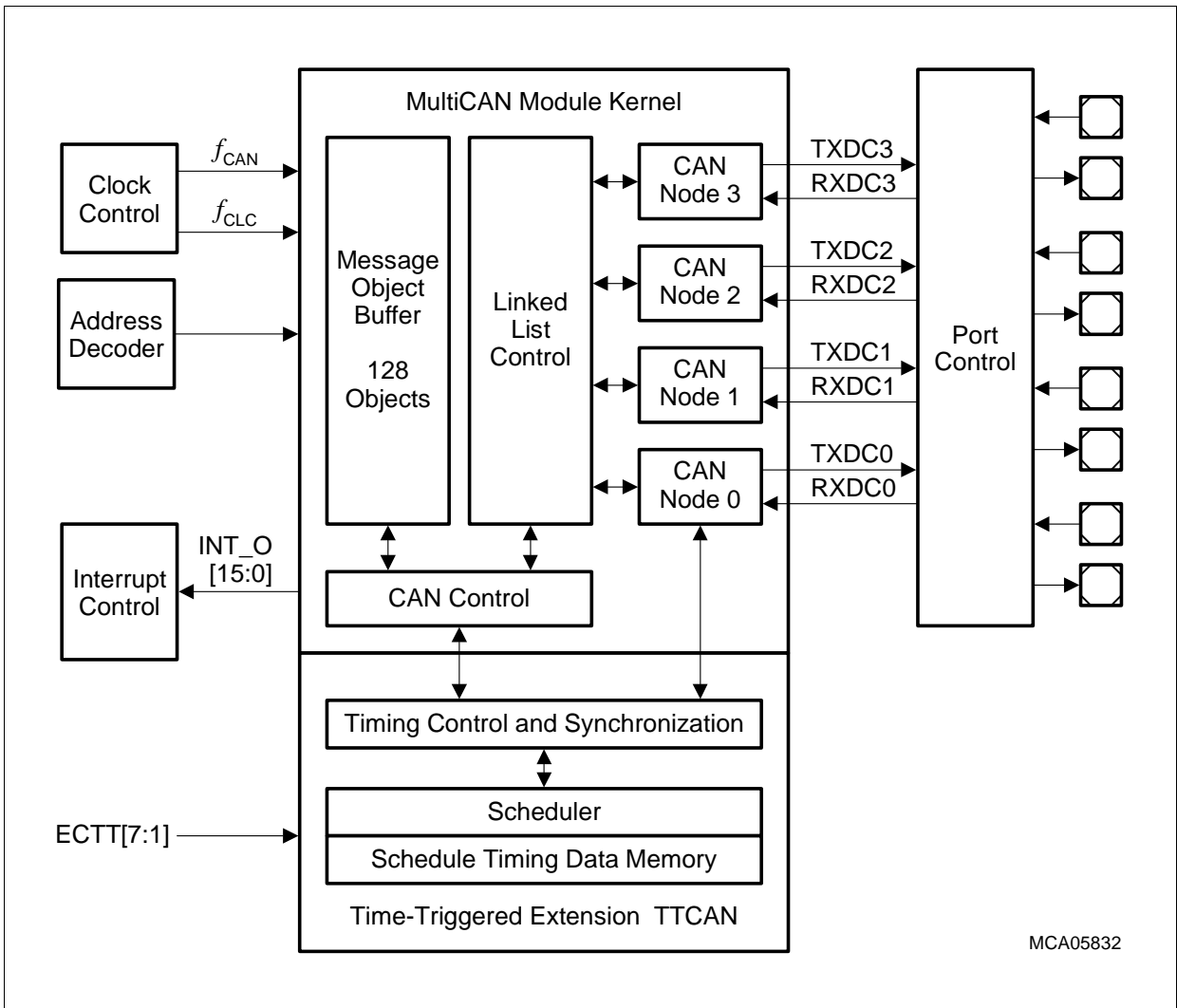


Figure 22-6 Overview of the MultiCAN Module with Time-Triggered Extension

Controller Area Network (MultiCAN) Controller**22.2.1 MultiCAN Module**

The MultiCAN module contains four independently operating CAN nodes with Full-CAN functionality that are able to exchange Data and Remote Frames via a gateway function. Transmission and reception of CAN frames is handled in accordance with CAN specification V2.0 B (active). Each CAN node can receive and transmit standard frames with 11-bit identifiers as well as extended frames with 29-bit identifiers.

All four CAN nodes share a common set of message objects. Each message object can be individually allocated to one of the CAN nodes. Besides serving as a storage container for incoming and outgoing frames, message objects can be combined to build gateways between the CAN nodes or to setup a FIFO buffer.

The message objects are organized in double-chained linked lists, where each CAN node has its own list of message objects. A CAN node stores frames only into message objects that are allocated to the message object list of the CAN node, and it transmits only messages belonging to this message object list. A powerful, command-driven list controller performs all message object list operations.

The bit timings for the CAN nodes are derived from the module timer clock (f_{CAN}), and are programmable up to a data rate of 1 Mbit/s. External bus transceivers are connected to a CAN node via a pair of receive and transmit pins.

Features

- Compliant with ISO 11898
- CAN functionality according to CAN specification V2.0 B active
- Dedicated control registers for each CAN node
- Data transfer rates up to 1 Mbit/s
- Flexible and powerful message transfer control and error handling capabilities
- Advanced CAN bus bit timing analysis and baud rate detection for each CAN node via a frame counter
- Full-CAN functionality: A set of 128 message objects can be individually
 - Allocated (assigned) to any CAN node
 - Configured as transmit or receive object
 - Set up to handle frames with 11-bit or 29-bit identifier
 - Identified by a timestamp via a frame counter
 - Configured to remote monitoring mode
- Advanced acceptance filtering
 - Each message object provides an individual acceptance mask to filter incoming frames.
 - A message object can be configured to accept standard or extended frames or to accept both standard and extended frames.
 - Message objects can be grouped into four priority classes for transmission and reception.

Controller Area Network (MultiCAN) Controller

- The selection of the message to be transmitted first can be based on frame identifier, IDE bit and RTR bit according to CAN arbitration rules, or according to its order in the list.
- Advanced message object functionality
 - Message objects can be combined to build FIFO message buffers of arbitrary size, limited only by the total number of message objects.
 - Message objects can be linked to form a gateway that automatically transfers frames between 2 different CAN buses. A single gateway can link any two CAN nodes. An arbitrary number of gateways can be defined.
- Advanced data management
 - The message objects are organized in double-chained lists.
 - List reorganizations can be performed at any time, even during full operation of the CAN nodes.
 - A powerful, command-driven list controller manages the organization of the list structure and ensures consistency of the list.
 - Message FIFOs are based on the list structure and can easily be scaled in size during CAN operation.
 - Static allocation commands offer compatibility with TwinCAN applications that are not list-based.
- Advanced interrupt handling
 - Up to 16 interrupt output lines are available. Interrupt requests can be routed individually to one of the 16 interrupt output lines.
 - Message post-processing notifications can be combined flexibly into a dedicated register field of 256 notification bits.
- Parity protected message object memory

Controller Area Network (MultiCAN) Controller**22.2.2 Time-Triggered Extension (TTCAN)**

In addition to the event-driven CAN functionality, a deterministic behavior can be achieved for CAN node 0 by an extension module that supports TTCAN functionality. The TTCAN protocol is compliant with the confirmed standardization proposal for ISO 11898-4 and fully conforms to the existing CAN protocol.

The time-triggered functionality is added as higher-layer extension (session layer) to the CAN protocol in order to be able to operate in safety-critical applications. The new features allow for deterministic behavior of a CAN network and the synchronization of networks. Global time information is available. The time-triggered extension is based on a scheduler mechanism with a timing control unit and a dedicated timing data part.

TTCAN Features

- Full support of basic cycle and system matrix functionality
- Support of reference messages level 1 and level 2
- Usable as time master
- Arbitration windows supported in time-triggered mode
- Global time information available
- CAN node 0 can be configured either for event-driven or time-triggered mode
- Built-in scheduler mechanism and a timing synchronization unit
- Write protection for scheduler timing data memory
- Timing-related interrupt functionality
- Parity protection for scheduler memory

Controller Area Network (MultiCAN) Controller

22.3 MultiCAN Kernel Functional Description

22.3.1 Module Structure

Figure 22-7 shows the general structure of the MultiCAN module.

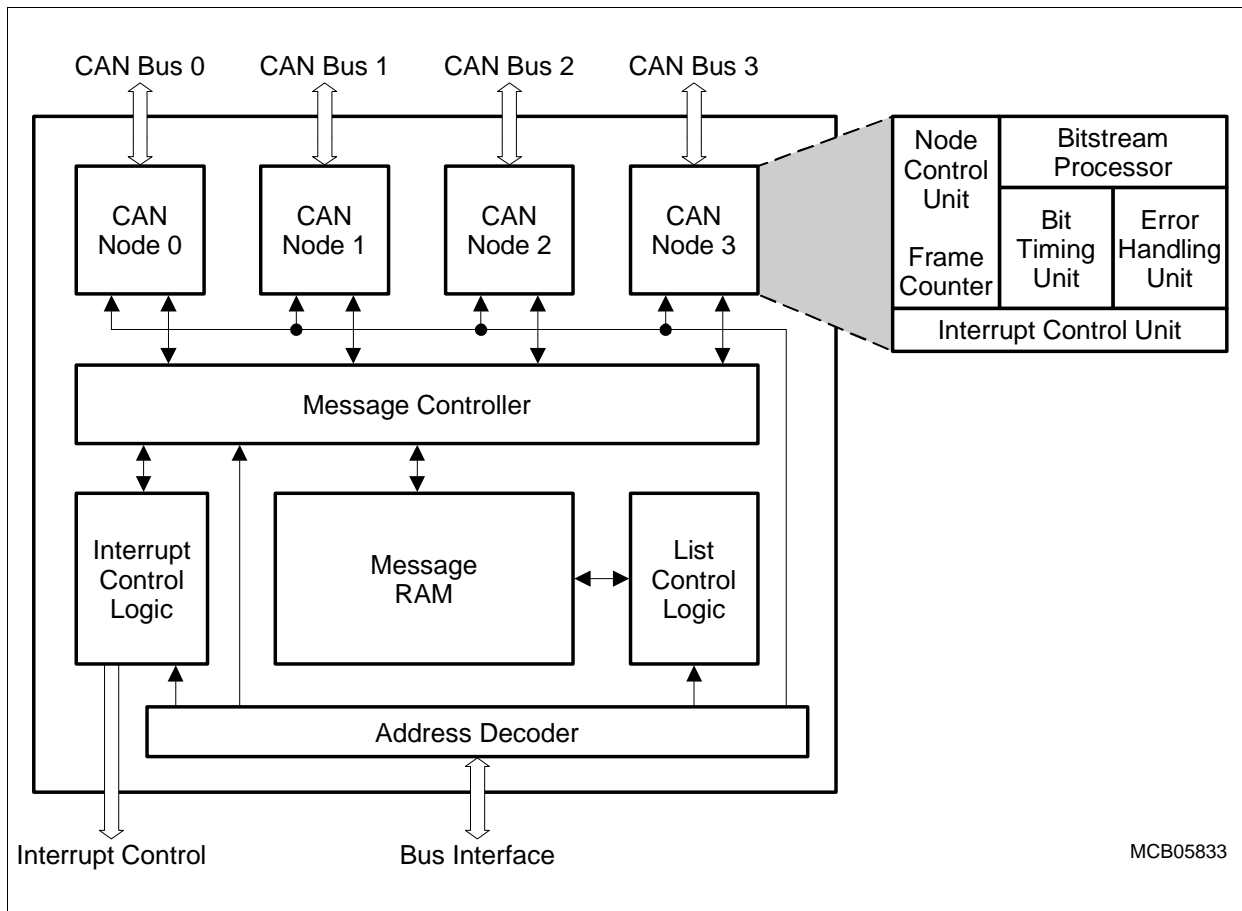


Figure 22-7 MultiCAN Block Diagram

CAN Nodes

Each CAN node consists of several sub-units.

- **Bitstream Processor**
The Bitstream Processor performs data, remote, error and overload frame processing according to the ISO 11898 standard. This includes conversion between the serial data stream and the input/output registers.
- **Bit Timing Unit**
The Bit Timing Unit determines the length of a bit time and the location of the sample point according to the user settings, taking into account propagation delays and phase shift errors. The Bit Timing Unit also performs re-synchronization.

Controller Area Network (MultiCAN) Controller

- **Error Handling Unit**
The Error Handling Unit manages the receive and transmit error counter. Depending on the contents of both counters, the CAN node is set into an error-active, error-passive or bus-off state.
- **Node Control Unit**
The Node Control Unit coordinates the operation of the CAN node:
 - Enable/disable CAN transfer of the node
 - Enable/disable and generate node-specific events that lead to an interrupt request (CAN bus errors, successful frame transfers etc.)
 - Administration of the Frame Counter
- **Interrupt Control Unit**
The Interrupt Control Unit in the CAN node controls the interrupt generation for the different conditions that can occur in the CAN node.

Message Controller

The Message Controller handles the exchange of CAN frames between the CAN nodes and the message objects that are stored in the Message RAM. The Message Controller performs several functions:

- Receive acceptance filtering to determine the correct message object for storing of a received CAN frame
- Transmit acceptance filtering to determine the message object to be transmitted first, individually for each CAN node
- Transfer contents between message objects and the CAN nodes, taking into account the status/control bits of the message objects
- Handling of the FIFO buffering and gateway functionality
- Aggregation of message-pending notification bits

List Controller

The List Controller performs all operations that lead to a modification of the double-chained message object lists. Only the list controller is allowed to modify the list structure. The allocation/deallocation or reallocation of a message object can be requested via a user command interface (command panel). The list controller state machine then performs the requested command autonomously.

Interrupt Control

The general interrupt structure is shown in [Figure 22-8](#). The interrupt event can trigger the interrupt generation. The interrupt pulse is generated independently of the interrupt flag in the interrupt status register. The interrupt flag can be cleared by software by writing a 0 to it.

If enabled by the related interrupt enable bit in the interrupt enable register, an interrupt pulse can be generated at one of the 16 interrupt output lines INT_Om of the MultiCAN

Controller Area Network (MultiCAN) Controller

module. If more than one interrupt source is connected to the same interrupt node pointer (in the interrupt node pointer register), the requests are combined to one common line.

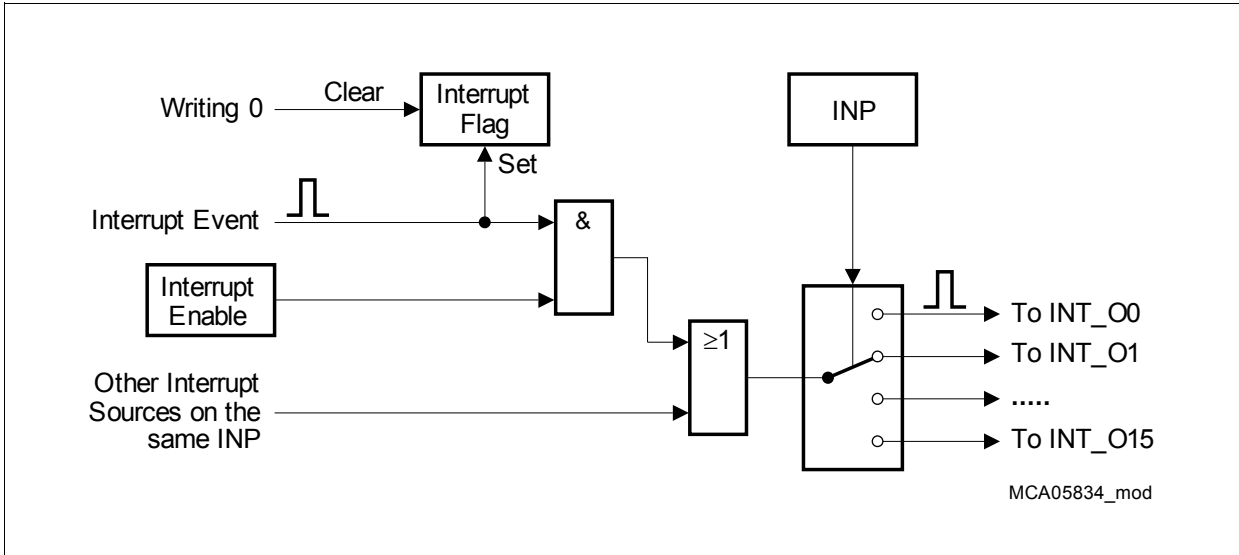


Figure 22-8 General Interrupt Structure

22.3.2 Clock Control

The CAN module timer clock f_{CAN} of the functional blocks of the MultiCAN module is derived from the module control clock f_{CLC} . The Fractional Divider is used to generate f_{CAN} used for the bit timing calculation, the generation of the NTU, and the local time of the TTCAN part. The frequency of f_{CAN} is identical for all CAN nodes. The TTCAN scheduler and the register file operate with the module control clock f_{CLC} . See also [“Module Clock Generation” on Page 22-201](#).

The output clock f_{CAN} of the Fractional Divider is based on the system clock f_{CLC} , but only every n-th clock pulse is taken. A suspend signal (coming as acknowledge from the MultiCAN module in response to a OCDS suspend request) freezes or resets the Fractional Divider.

Controller Area Network (MultiCAN) Controller

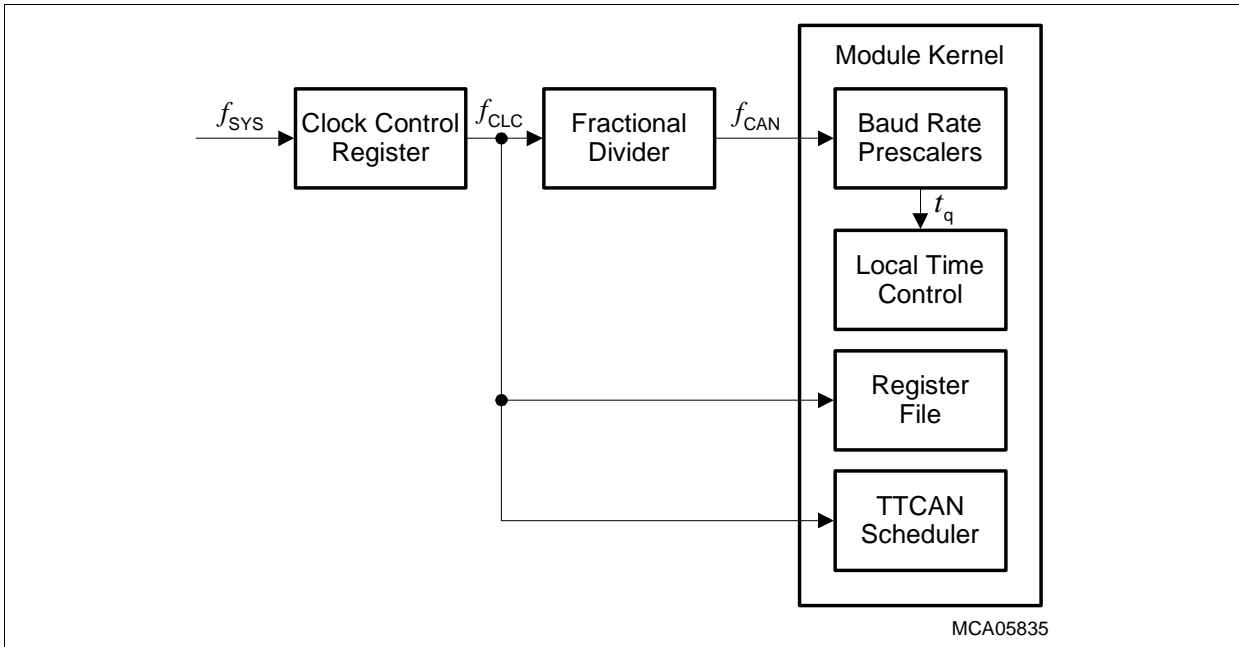


Figure 22-9 MultiCAN Clock Generation

Table 22-1 indicates the minimum operating frequencies in MHz for f_{CLC} that are required for a baud rate of 1 Mbit/s for the active CAN nodes. If a lower baud rate is desired, the values can be scaled linearly (e.g. for a maximum of 500 kbit/s, 50% of the indicated value are required).

The values imply that the CPU (or PCP, or DMA) executes a maximum of accesses to the MultiCAN module. The values may contain rounding effects.

Table 22-1 Minimum Operating Frequencies [MHz]

Number of Allocated Message Objects MO ¹⁾ , with or without TTCAN Functionality	with 1 CAN Node Active	with 2 CAN Nodes Active	with 3 CAN Nodes Active	with 4 CAN Nodes Active
16 MO, without TTCAN	12	19	26	33
16 MO, with 1 TTCAN	14	21	28	35
32 MO, without TTCAN	15	23	30	37
32 MO, with 1 TTCAN	17	25	32	39
64 MO, without TTCAN	21	28	37	46
64 MO, with 1 TTCAN	23	30	39	48
128 MO, without TTCAN	40	45	50	55
128 MO, with 1 TTCAN	42	47	52	57

1) Only those message objects that are allocated to a CAN node must be taken into account. The unallocated message objects have no influence on the minimum operating frequency.

Controller Area Network (MultiCAN) Controller**22.3.3 Port Input Control**

It is possible to select the input lines for the RXDCANx inputs for the CAN nodes. The selected input is connected to the CAN node and is also available to wake-up the system. More details are defined in [Section 22.9.4.3](#) at [Page 22-207](#).

22.3.4 Suspend Mode

The Suspend Mode can be triggered by the OCDS in order to freeze the state of the module and to permit access to the registers (at least for read actions). The MultiCAN module provides two types of Suspend Modes:

- All actions are immediately stopped (Hard Suspend Mode):
The module clocks f_{CLC} and f_{CAN} are switched off as soon as the suspend request becomes active. Read and write operations to the module are no longer possible. This means that the CAN registers cannot be accessed anymore. In this mode, there is a very high probability that the communication with other CAN devices is made impossible, and that the CAN bus is blocked (e.g. if the suspended CAN module just sends a dominant level). A reset operation must be executed to leave Hard Suspend Mode.
- The current action is finished (Soft Suspend Mode):
The module clock f_{CLC} keeps running. Module functions are stopped automatically after internal actions have been finished (for example, after a CAN frame has been sent out). The end of the internal actions is indicated to the fractional divider by a suspend mode acknowledged signal. Due to this behavior, the communication network is not blocked. Furthermore, all registers are accessible for read and write actions. As a result, the debugger can stop the module actions and modify registers. These modifications are taken into account after the Suspend Mode is left.

The Hard Suspend Mode can be enabled/disabled only for the complete MultiCAN module. The Soft Suspend Mode can be individually enabled for each CAN node.

The fractional divider disables module clock f_{CAN} only if all CAN nodes signal that they can be suspended. A CAN node that is not active can always be suspended.

The user has to be aware that the Soft Suspend Mode can corrupt the TTCAN timing values, because the counter clock can be disabled. In order to guarantee correct TTCAN behavior, a TTCAN node should not enter any Suspend Mode if the consistency of the timing data must be ensured.

Controller Area Network (MultiCAN) Controller**22.3.5 CAN Node Control**

Each CAN node may be configured and run independently from the other CAN nodes. Each CAN node is equipped with its own node control logic to configure the global behavior and to provide status information.

Note: In the following descriptions, index “x” stands for the node number and index “n” represents the message object number.

Configuration Mode is activated when bit NCRx.CCE is set to 1. This mode allows CAN bit timing parameters and the error counter registers to be modified.

CAN Analyze Mode is activated when bit NCRx.CALM is set to 1. In this operation mode, Data and Remote Frames are monitored without active participation in any CAN transfer (CAN transmit pin is held on recessive level). Incoming Remote Frames are stored in a corresponding transmit message object, while arriving Data Frames are saved in a matching receive message object.

In CAN Analyze Mode, the entire configuration information of the received frame is stored in the corresponding message object, and can be evaluated by the CPU to determine their identifier, XTD bit information and data length code (ID and DLC optionally if the Remote Monitoring Mode is active, bit MOFCRn.RMM = 1). Incoming frames are not acknowledged, and no Error Frames are generated. If CAN Analyze Mode is enabled, Remote Frames are not responded by the corresponding Data Frame, and Data Frames cannot be transmitted by setting the transmit request bit MOSTATn.TXRQ. Receive interrupts are generated. In CAN Analyze Mode (if enabled) for all error free received frames.

The node-specific interrupt configuration is also defined by the Node Control Logic via the NCRx register bits TRIE, ALIE and LECIE:

- If control bit TRIE is set to 1, a transfer interrupt is generated when the NSRx register has been updated (after each successfully completed message transfer).
- If control bit ALIE is set to 1, an error interrupt is generated when a “bus-off” condition has been recognized or the Error Warning Level has been exceeded or under-run. Additionally, list or object errors lead to this type of interrupt.
- If control bit LECIE is set to 1, a last error code interrupt is generated when an error code > 0 is written into bit field NSRx.LEC by hardware.

The Node x Status Register NSRx provides an overview about the current state of the respective CAN node x, comprising information about CAN transfers, CAN node status, and error conditions.

The CAN frame counter can be used to check the transfer sequence of message objects or to obtain information about the instant a frame has been transmitted or received from the associated CAN bus. CAN frame counting is performed by a 16-bit counter, controlled by register NFCRx. Bit fields NFCRx.CFMODE and NFCRx.CFSEL determine the operation mode and the trigger event incrementing the frame counter.

Controller Area Network (MultiCAN) Controller

22.3.5.1 Bit Timing Unit

According to the ISO 11898 standard, a CAN bit time is subdivided into different segments (see [Figure 22-10](#)). Each segment consists of multiples of a time quantum t_q . The magnitude of t_q is adjusted by Node x Bit Timing Register bit fields NBTRx.BRP and NBTRx.DIV8, both controlling the baud rate prescaler (register NBTRx is described on [Page 22-84](#)). The baud rate prescaler is driven by the module timer clock f_{CAN} (generation and control of f_{CAN} is described at [Page 22-201](#)).

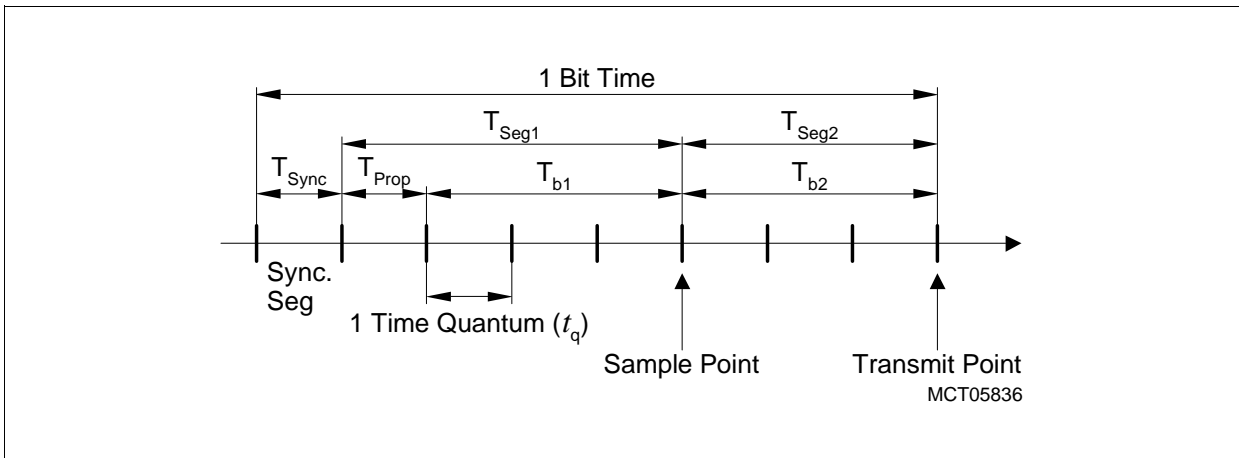


Figure 22-10 CAN Bus Bit Timing Standard

The Synchronization Segment (T_{Sync}) allows a phase synchronization between transmitter and receiver time base. The Synchronization Segment length is always one t_q . The Propagation Time Segment (T_{Prop}) takes into account the physical propagation delay in the transmitter output driver on the CAN bus line and in the transceiver circuit. For a working collision detection mechanism, T_{Prop} must be two times the sum of all propagation delay quantities rounded up to a multiple of t_q . The phase buffer segments 1 and 2 (T_{b1} , T_{b2}) before and after the signal sample point are used to compensate for a mismatch between transmitter and receiver clock phase detected in the synchronization segment.

The maximum number of time quanta allowed for re-synchronization is defined by bit field NBTRx.SJW. The Propagation Time Segment and the Phase Buffer Segment 1 are combined to parameter T_{Seg1} , which is defined by the value NBTRx.TSEG1. A minimum of 3 time quanta is demanded by the ISO standard. Parameter T_{Seg2} , which is defined by the value of NBTRx.TSEG2, covers the Phase Buffer Segment 2. A minimum of 2 time quanta is demanded by the ISO standard. According ISO standard, a CAN bit time, calculated as the sum of T_{Sync} , T_{Seg1} and T_{Seg2} , must not fall below 8 time quanta.

Controller Area Network (MultiCAN) Controller

Calculation of the bit time:

$$\begin{aligned}
 t_q &= (\text{BRP} + 1) / f_{\text{CAN}} && \text{if DIV8} = 0 \\
 &= (\text{BRP} + 1) / 8 \times f_{\text{CAN}} && \text{if DIV8} = 1 \\
 T_{\text{Sync}} &= 1 \times t_q \\
 T_{\text{Seg1}} &= (\text{TSEG1} + 1) \times t_q && (\text{min. } 3 t_q) \\
 T_{\text{Seg2}} &= (\text{TSEG2} + 1) \times t_q && (\text{min. } 2 t_q) \\
 \text{bit time} &= T_{\text{Sync}} + T_{\text{Seg1}} + T_{\text{Seg2}} && (\text{min. } 8 t_q)
 \end{aligned}$$

To compensate phase shifts between clocks of different CAN controllers, the CAN controller must synchronize on any edge from the recessive to the dominant bus level. If the hard synchronization is enabled (at the start of frame), the bit time is restarted at the synchronization segment. Otherwise, the re-synchronization jump width T_{SJW} defines the maximum number of time quanta, a bit time may be shortened or lengthened by one re-synchronization. The value of SJW is defined by bit field NBTRx.SJW.

$$\begin{aligned}
 T_{\text{SJW}} &= (\text{SJW} + 1) \times t_q \\
 T_{\text{Seg1}} &\geq T_{\text{SJW}} + T_{\text{prop}} \\
 T_{\text{Seg2}} &\geq T_{\text{SJW}}
 \end{aligned}$$

The maximum relative tolerance for f_{CAN} depends on the Phase Buffer Segments and the re-synchronization jump width.

$$\begin{aligned}
 df_{\text{CAN}} &\leq \min(T_{b1}, T_{b2}) / 2 \times (13 \times \text{bit time} - T_{b2}) \quad \text{AND} \\
 df_{\text{CAN}} &\leq T_{\text{SJW}} / 20 \times \text{bit time}
 \end{aligned}$$

A valid CAN bit timing must be written to the CAN Node Bit Timing Register NBTR before clearing the INIT bit in the Node Control Register, i.e. before enabling the operation of the CAN node.

The Node Bit Timing Register may be written only if bit CCE (Configuration Change Enable) is set in the corresponding Node Control Register.

22.3.5.2 Bitstream Processor

Based on the message objects in the message buffer, the Bitstream Processor generates the remote and Data Frames to be transmitted via the CAN bus. It controls the CRC generator and adds the checksum information to the new remote or Data Frame. After including the SOF bit and the EOF field, the Bitstream Processor starts the CAN

Controller Area Network (MultiCAN) Controller

bus arbitration procedure and continues with the frame transmission when the bus was found in idle state. While the data transmission is running, the Bitstream Processor continuously monitors the I/O line. If (outside the CAN bus arbitration phase or the acknowledge slot) a mismatch is detected between the voltage level on the I/O line and the logic state of the bit currently sent out by the transmit shift register, a CAN error interrupt request is generated, and the error code is indicated by the Node x Status Register bit field NSRx.LEC.

The data consistency of an incoming frame is verified by checking the associated CRC field. When an error has been detected, a CAN error interrupt request is generated and the associated error code is presented in the Node x Status Register NSRx. Furthermore, an Error Frame is generated and transmitted on the CAN bus. After decomposing a faultless frame into identifier and data portion, the received information is transferred to the message buffer executing remote and Data Frame handling, interrupt generation and status processing.

22.3.5.3 Error Handling Unit

The Error Handling Unit of a CAN node x is responsible for the fault confinement of the CAN device. Its two counters, the Receive Error Counter REC and the Transmit Error Counter TEC (bit fields of the Node x Error Counter Register NECNTx, see [Page 22-86](#)) are incremented and decremented by commands from the Bitstream Processor. If the Bitstream Processor itself detects an error while a transmit operation is running, the Transmit Error Counter is incremented by 8. An increment of 1 is used when the error condition was reported by an external CAN node via an Error Frame generation. For error analysis, the transfer direction of the disturbed message and the node that recognizes the transfer error, are indicated for the respective CAN node x in register NECNTx. Depending on the values of the error counters, the CAN node is set into error-active, error-passive, or bus-off state.

The CAN node is in error-active state if both error counters are below the error-passive limit of 128. The CAN node is in error-passive state, if at least one of the error counters is equal to or greater than 128.

The bus-off state is activated if the Transmit Error Counter is equal to or greater than the bus-off limit of 256. This state is reported for CAN node x by the Node x Status Register flag NSRx.BOFF. The device remains in this state, until the “bus-off” recovery sequence is finished. Additionally, Node x Status Register flag NSRx.EWRN is set when at least one of the error counters is equal to or greater than the error warning limit defined by the Node x Error Count Register bit field NECNTx.EWRNLVL. Bit NSRx.EWRN is cleared if both error counters fall below the error warning limit again.

Controller Area Network (MultiCAN) Controller**22.3.5.4 CAN Frame Counter**

Each CAN node is equipped with a frame counter that counts transmitted/received CAN frames or obtains information about the time when a frame has been started to transmit or be received by the CAN node. CAN frame counting/bit time counting is performed by a 16-bit counter that is controlled by Node x Frame Counter Register NFCRx (see [Page 22-87](#)). Bit field NFCRx.CFSEL determines the operation mode of the frame counter:

- **Frame Count Mode:**
The frame counter is incremented after the successful transmission and/or reception of a CAN frame. The incremented value is copied into the CFCVAL bit field of the MOIPRn register of the message object involved in the transfer.
- **Time Stamp Mode:**
The frame counter is incremented with the beginning of a new bit time. When the transmission/reception of a frame starts, the value of the frame counter is captured and stored to the CFC bit field of the NFCRx register. After the successful transfer of the frame the captured value is copied to the CFCVAL bit field of the MOIPRn register of the message object involved in the transfer.
- **Bit Timing Mode:** Used for baud rate detection and analysis of the bit timing (see [Page 22-35](#)).

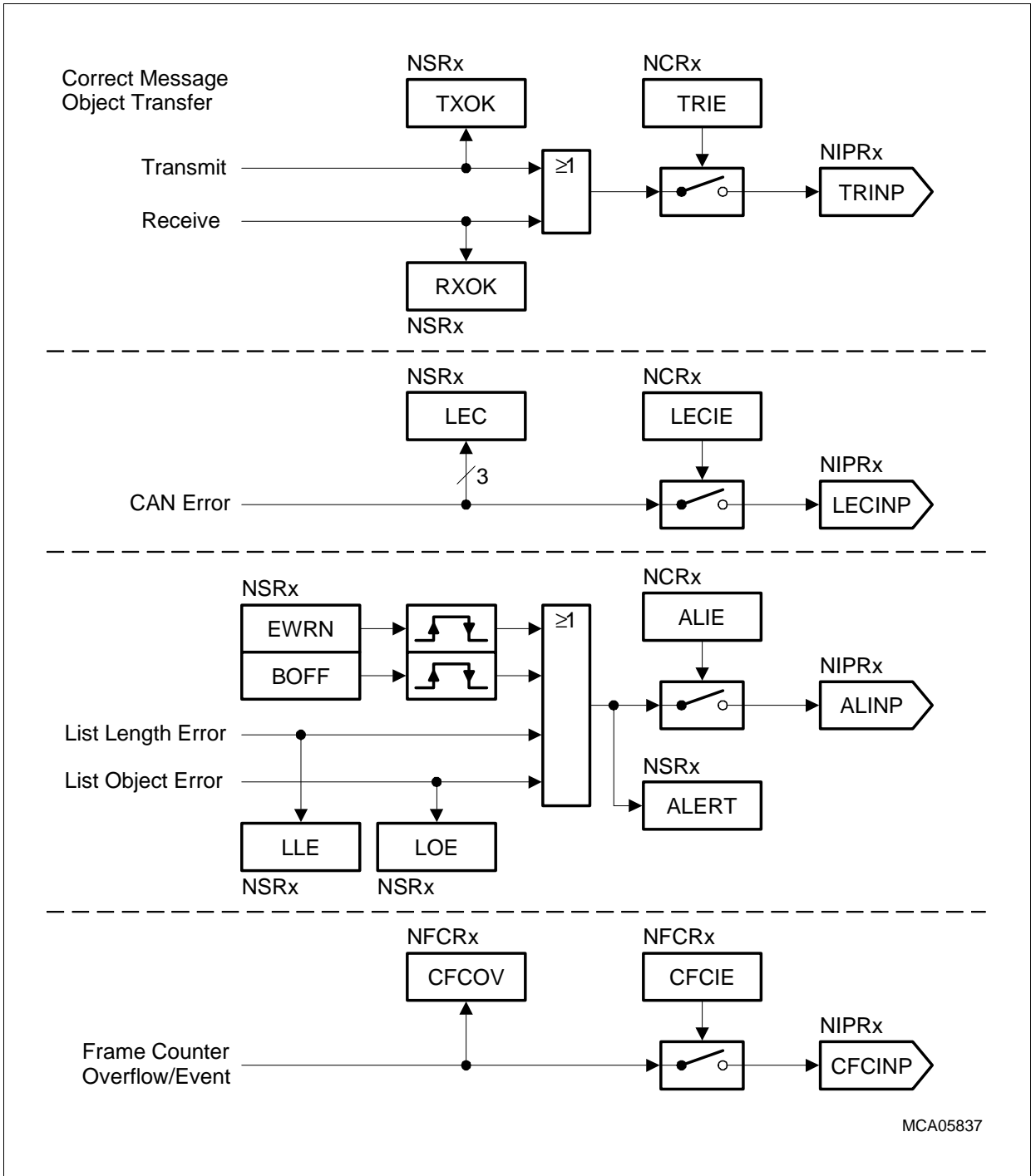
22.3.5.5 CAN Node Interrupts

Each CAN node has four hardware triggered interrupt request types that are able to generate an interrupt request upon:

- The successful transmission or reception of a frame
- A CAN protocol error with a last error code
- An alert condition: Transmit/receive error counters reach the warning limit, bus-off state changes, a List Length Error occurs, or a List Object Error
- An overflow of the frame counter

Besides the hardware generated interrupts, software initiated interrupts can be generated using the Module Interrupt Trigger Register MITR. Writing a 1 to bit n of bit field MITR.IT generates an interrupt request signal on the corresponding interrupt output line INT_On. When writing MITR.IT more than one bit can be set resulting in activation of multiple INT_On interrupt output lines at the same time. See also [“Interrupt Control” on Page 22-210](#) for further processing of the CAN node interrupts.

Controller Area Network (MultiCAN) Controller



MCA05837

Figure 22-11 CAN Node Interrupts

Controller Area Network (MultiCAN) Controller

22.3.6 Message Object List Structure

22.3.6.1 Basics

The message objects of the MultiCAN module are organized in double-chained lists, where each message object has a pointer to the previous message object in the list as well as a pointer to the next message object in the list. The MultiCAN module provides eight lists. Each message object is allocated to one of these lists. In the example in [Figure 22-12](#), the three message objects (3, 5, and 16) are allocated to the list with index 2 (List Register LIST2).

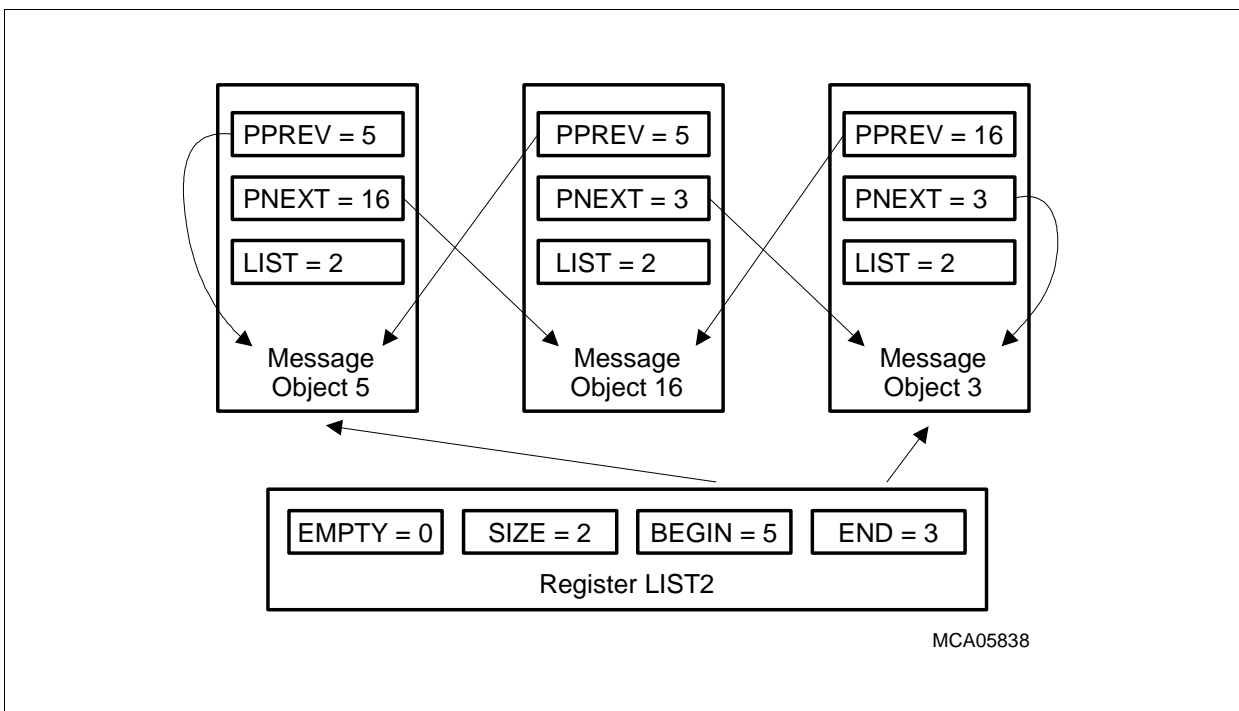


Figure 22-12 Example Allocation of Message Objects to a List

Bit field BEGIN in the List Register (for definition, see [Page 22-68](#)) points to the first element in the list (object 5 in the example), and bit field END points to the last element in the list (object 3 in the example). The number of elements in the list is indicated by bit field SIZE of the List Register (SIZE = number of list elements - 1, thus SIZE = 2 for the 3 elements in the example). The EMPTY bit of the List Register indicates whether or not a list is empty (EMPTY = 0 in the example, because list 2 is not empty).

Each message object n has a pointer PNEXT in its Message Object n Control Register MOCTRn (see [Page 22-91](#)) that points to the next message object in the list, and a pointer PPREV that points to the previous message object in the list. PPREV of the first message object points to the message object itself because the first message object has no predecessor (in the example message object 5 is the first message object in the list,

Controller Area Network (MultiCAN) Controller

indicated by PPREV = 5). PNEXT of the last message object also points to the message object itself because the last message object has no successor (in the example, object 3 is the last message object in the list, indicated by PNEXT = 3).

Bit field MOCTRn.LIST indicates the list index number to which the message object is currently allocated. The message object of the example are allocated to list 2. Therefore, all LIST bit fields for the message objects assigned to list 2 are set to LIST = 2.

22.3.6.2 List of Unallocated Elements

The list with list index 0 has a special meaning: it is the list of all unallocated elements. An element is called unallocated if it belongs to list 0 (MOCTRn.LIST = 0). It is called allocated if it belongs to a list with an index not equal to 0 (MOCTRn.LIST > 0).

After reset, all message objects are unallocated. This means that they are assigned to the list of unallocated elements with MOCTRn.LIST = 0. After this initial allocation of the message objects caused by reset, the list of all unallocated message objects is ordered by message number (predecessor of message object n is object n-1, successor of object n is object n+1).

22.3.6.3 Connection to the CAN Nodes

Each CAN node is linked to one unique list of message objects. A CAN node performs message transfers only with the message objects that are allocated to the list of the CAN node. This is illustrated in [Figure 22-13](#). Frames that are received on a CAN node may only be stored in one of the message objects that belongs to the CAN node; frames to be transmitted on a CAN node are selected only from the message objects that are allocated to that node, as indicated by the vertical arrows.

There are more lists (eight) than CAN nodes (four). This means that some lists are not linked to one of the CAN nodes. A message object that is allocated to one of these unlinked lists cannot receive messages directly from a CAN node and it may not transmit messages.

FIFO and gateway mechanisms refer to message numbers and not directly to a specific list. The user must take care that the message objects targeted by FIFO/gateway belong to the desired list. The mechanisms make it possible to work with lists that do belong to the CAN node.

Controller Area Network (MultiCAN) Controller

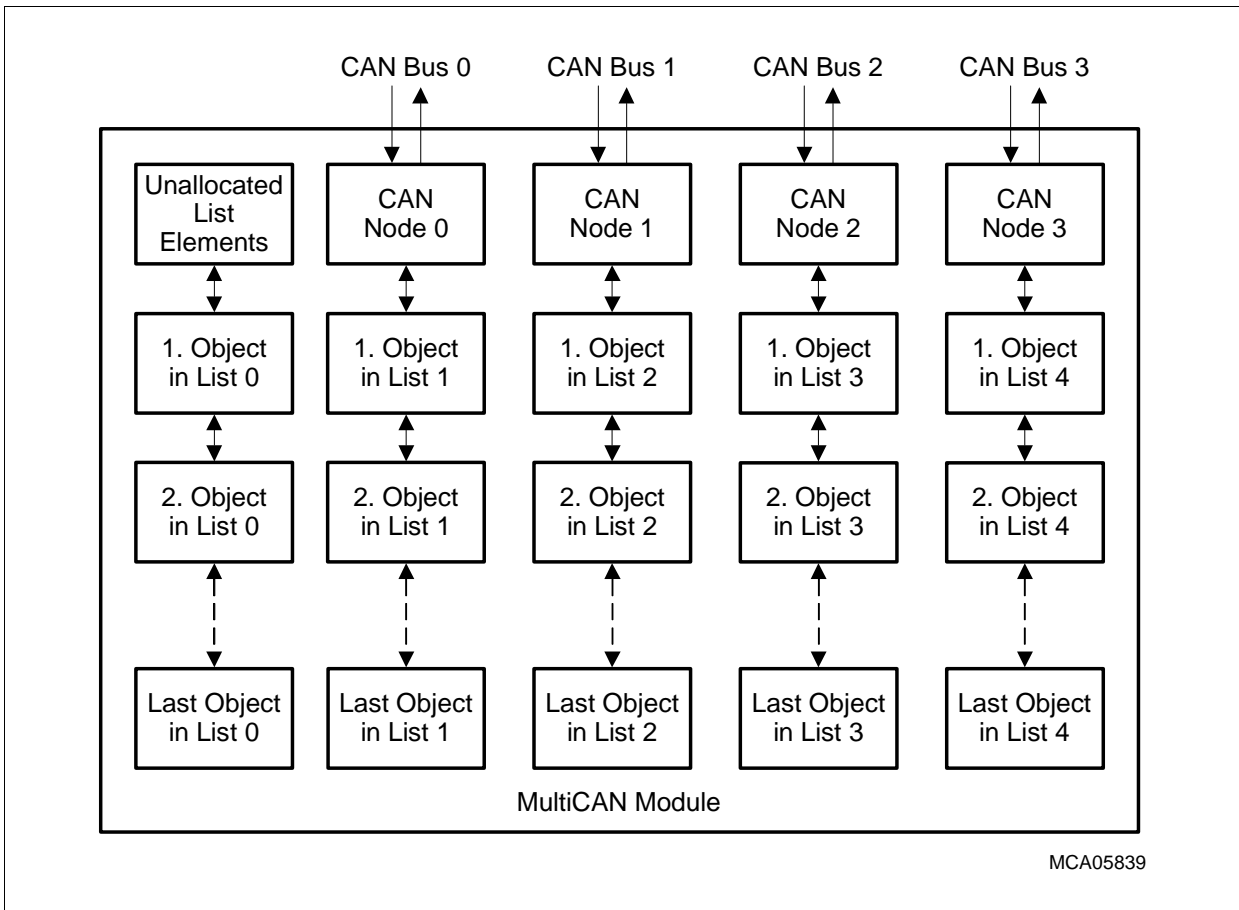


Figure 22-13 Message Objects Linked to CAN Nodes

22.3.6.4 List Command Panel

The list structure cannot be modified directly by write accesses to the LIST registers and the PPREV, PNEXT and LIST bit fields in the Message Object Control Registers, as they are read only. The list structure is managed by and limited to the list controller inside the MultiCAN module. The list controller is controlled via a command panel allowing the user to issue list allocation commands to the list controller. The list controller has two main purposes:

1. Ensure that all operations that modify the list structure result in a consistent list structure.
2. Present maximum ease of use and flexibility to the user.

The list controller and the associated command panel allows the programmer to concentrate on the final properties of the list, which are characterized by the allocation of message objects to a CAN node, and the ordering relation between objects that are allocated to the same list. The process of list (re-)building is done in the list controller.

Controller Area Network (MultiCAN) Controller

Table 22-2 gives an overview on the available panel commands while **Table 22-6** on **Page 22-63** describes the panel commands in more detail.

Table 22-2 Panel Commands Overview

Command Name	Description
No Operation	No new command is started.
Initialize Lists	Run the initialization sequence to reset the CTRL and LIST field of all message objects.
Static Allocate	Allocate message object to a list.
Dynamic Allocate	Allocate the first message object of the list of unallocated objects to the selected list.
Static Insert Before	Remove a message object (source object) from the list that it currently belongs to, and insert it before a given destination object into the list structure of the destination object.
Dynamic Insert Before	Insert a new message object before a given destination object.
Static Insert Behind	Remove a message object (source object) from the list that it currently belongs to, and insert it behind a given destination object into the list structure of the destination object.
Dynamic Insert Behind	Insert a new message object behind a given destination object.

A panel command is started by writing the respective command code into the Panel Control Register bit field PANCTR.PANCMD (see **Page 22-62**). The corresponding command arguments must be written into bit fields PANCTR.PANAR1 and PANCTR.PANAR2 before writing the command code, or latest along with the command code in a single 32-bit write access to the Panel Control Register.

With the write operation of a valid command code, the PANCTR.BUSY flag is set and further write accesses to the Panel Control Register are ignored. The BUSY flag remains active and the control panel remains locked until the execution of the requested command has been completed. After a reset, the list controller builds up list 0. During this operation, BUSY is set and other accesses to the CAN RAM are forbidden. The CAN RAM can be accessed again when BUSY becomes inactive.

Note: The CAN RAM is automatically initialized after reset by the list controller in order to ensure correct list pointers in each message object. The end of this CAN RAM initialization is indicated by bit PANCTR.BUSY becoming inactive.

In case of a dynamic allocation command that takes an element from the list of unallocated objects, the PANCTR.RBUSY bit is also set along with the BUSY bit

Controller Area Network (MultiCAN) Controller

(RBUSY = BUSY = 1). This indicates that bit fields PANAR1 and PANAR2 are going to be updated by the list controller in the following way:

1. The message number of the message object taken from the list of unallocated elements is written to PANAR1.
2. If ERR (bit 7 of PANAR2) is set to 1, the list of unallocated elements was empty and the command is aborted. If ERR is 0, the list was not empty and the command will be performed successfully.

The results of a dynamic allocation command are written before the list controller starts the actual allocation process. As soon as the results are available, RBUSY becomes inactive (RBUSY = 0) again, while BUSY still remains active until completion of the command. This allows the user to set up the new message object while it is still in the process of list allocation. The access to message objects is not limited during ongoing list operations. However, any access to a register resource located inside the RAM delays the ongoing allocation process by one access cycle.

As soon as the command is finished, the BUSY flag becomes inactive (BUSY = 0) and write accesses to the Panel Control Register are enabled again. Also, the "No Operation" command code is automatically written to the PANCTR.PANCMD field. A new command may be started any time when BUSY = 0.

All fields of the Panel Control Register PANCTR except BUSY and RBUSY may be written by the user. This makes it possible to save and restore the Panel Control Register if the Command Panel shall be used within independent (mutually interruptible) interrupt service routines. If this is the case, any task that uses the Command Panel and that may interrupt another task that also uses the Command Panel should poll the BUSY flag until it becomes inactive and save the whole PANCTR register to a memory location before issuing a command. At the end of the interrupt service routine, the task should restore PANCTR from the memory location.

Before a message object that is allocated to the list of an active CAN node shall be moved to another list or to another position within the same list, bit MOCTRn.MSGVAL ("Message Valid") of message object n must be cleared.

22.3.7 CAN Node Analysis Features

22.3.7.1 Analyze Mode

The CAN Analyze Mode makes it possible to monitor the CAN traffic for each CAN node individually without affecting the logical state of the CAN bus. The CAN Analyze Mode for CAN node x is selected by setting Node x Control Register bit NCRx.CALM.

In CAN Analyze Mode, the transmit pin of a CAN node is held at a recessive level permanently. The CAN node may receive frames (data, remote, and Error Frames) but is not allowed to transmit. Received Data/Remote Frames are not acknowledged (i.e. acknowledge slot is sent recessive) but will be received and stored in matching message objects as long as there is any other node that acknowledges the frame. The complete message object functionality is available, but no transmit request will be executed.

22.3.7.2 Loop-Back Mode

The MultiCAN module provides a Loop-Back Mode to enable an in-system test of the MultiCAN module as well as the development of CAN driver software without access to an external CAN bus.

The loop-back feature consists of an internal CAN bus (inside the MultiCAN module) and a bus select switch for each CAN node (see [Figure 22-14](#)). With the switch, each CAN node can be connected either to the internal CAN bus (Loop-Back Mode activated) or the external CAN bus, respectively to transmit and receive pins (normal operation). The CAN bus that is not currently selected is driven recessive; this means the transmit pin is held at 1, and the receive pin is ignored by the CAN nodes that are in Loop-Back Mode.

The Loop-Back Mode is selected for CAN node x by setting the Node x Port Control Register bit NPCRx.LBM. All CAN nodes that are in Loop-Back Mode may communicate together via the internal CAN bus without affecting the normal operation of the other CAN nodes that are not in Loop-Back Mode.

Controller Area Network (MultiCAN) Controller

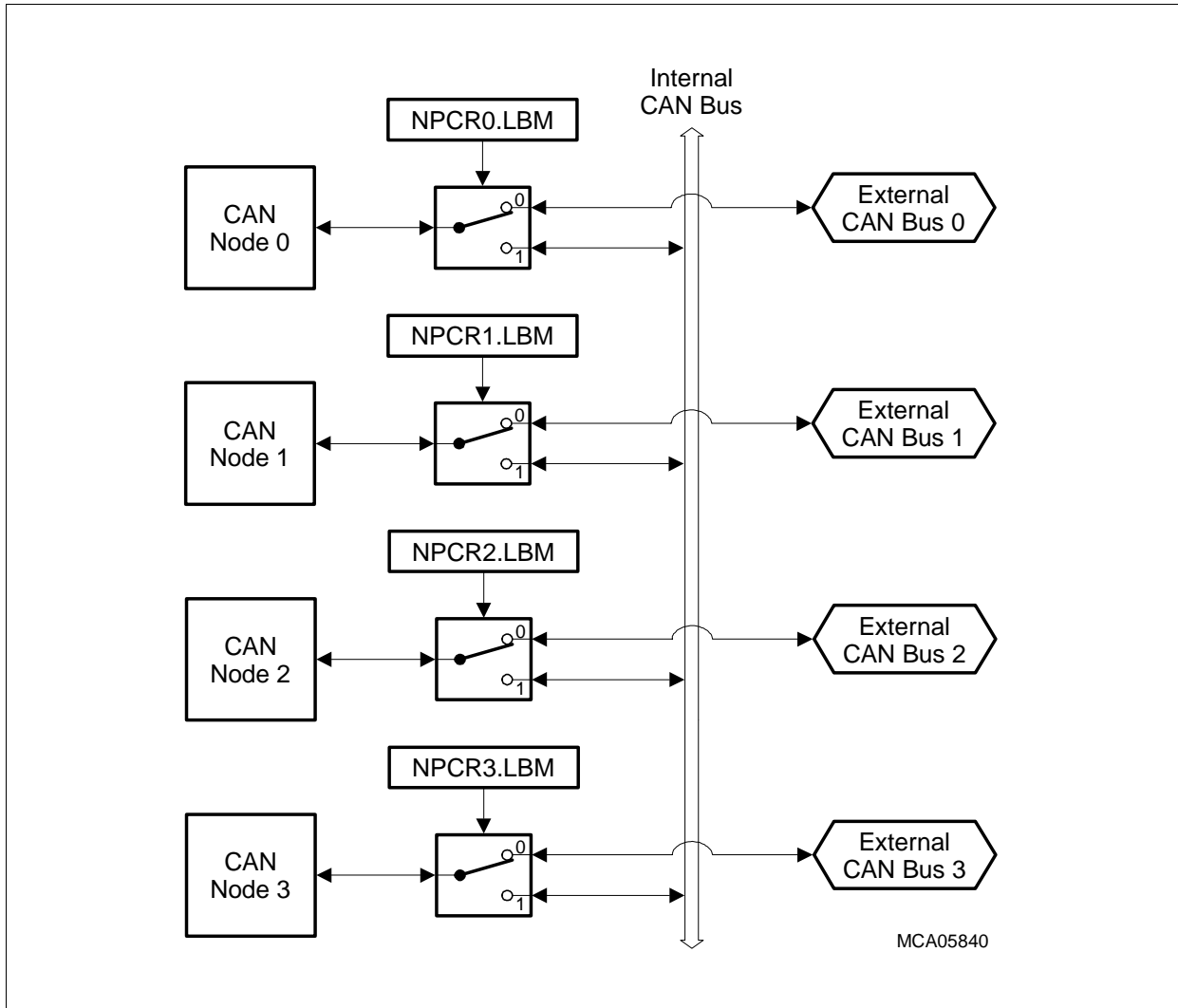


Figure 22-14 Loop-Back Mode

22.3.7.3 Bit Timing Analysis

Detailed analysis of the bit timing can be performed for each CAN node using the analysis modes of the CAN frame counter. The bit timing analysis functionality of the frame counter may be used for automatic detection of the CAN baud rate, as well as to analyze the timing of the CAN network.

Bit timing analysis for CAN node x is selected when bit field $NFCRx.CFMODE = 10_B$. Bit timing analysis does not affect the operation of the CAN node.

The bit timing measurement results are written into the $NFCRx.CFC$ bit field. Whenever $NFCRx.CFC$ is updated in bit timing analysis mode, bit $NFCRx.CFCOV$ is also set to indicate the CFC update event. If $NFCRx.CFCIE$ is set, an interrupt request can be generated (see [Figure 22-11](#)).

Controller Area Network (MultiCAN) Controller**Automatic Baud Rate Detection**

For automatic baud rate detection, the time between the observation of subsequent dominant edges on the CAN bus must be measured. This measurement is automatically performed if bit field NFCRx.CFSEL = 000_B. With each dominant edge monitored on the CAN receive input line, the time (measured in f_{CAN} clock cycles) between this edge and the most recent dominant edge is stored in the NFCRx.CFC bit field.

Synchronization Analysis

The bit time synchronization is monitored if NFCRx.CFSEL = 010_B. The time between the first dominant edge and the sample point is measured and stored in the NFCRx.CFC bit field. The bit timing synchronization offset may be derived from this time as the first edge after the sample point triggers synchronization and there is only one synchronization between consecutive sample points.

Synchronization analysis can be used, for example, for fine tuning of the baud rate during reception of the first CAN frame with the measured baud rate.

Driver Delay Measurement

The delay between a transmitted edge and the corresponding received edge is measured when NFCRx.CFSEL = 011_B (dominant to dominant) and NFCRx.CFSEL = 100_B (recessive to recessive). These delays indicate the time needed to represent a new bit value on the physical implementation of the CAN bus.

22.3.8 Message Acceptance Filtering

22.3.8.1 Receive Acceptance Filtering

When a CAN frame is received by a CAN node, a unique message object is determined in which the received frame is stored after successful frame reception. A message object is qualified for reception of a frame if the following six conditions are met:

- The message object is allocated to the message object list of the CAN node by which the frame is received.
- Bit MOSTATn.MSGVAL in the Message Status Register (see [Page 22-94](#)) is set.
- Bit MOSTATn.RXEN is set.
- Bit MOSTATn.DIR is equal to bit RTR of the received frame.
If bit MOSTATn.DIR = 1 (transmit object,) the message object accepts only Remote Frames. If bit MOSTATn.DIR = 0 (receive object), the message object accepts only Data Frames.
- If bit MOAMRn.MIDE = 1, the IDE bit of the received frame becomes evaluated in the following way: If MOARn.IDE = 1, the IDE bit of the received frame must be set (indicates extended identifier). If MOARn.IDE = 0, the IDE bit of the received frame must be cleared (indicates standard identifier).
If bit MOAMRn.MIDE = 0, the IDE bit of the received frame is “don’t care”. In this case, message objects with standard and extended frames are accepted.
- The identifier of the received frame matches the identifier stored in the Arbitration Register of the message object as qualified by the acceptance mask in the MOAMRn register. This means that each bit of the received message object identifier is equal to the bit field MOARn.ID, except those bits for which the corresponding acceptance mask bits in bit field MOAMRn.AM are cleared. These identifier bits are “don’t care” for reception. [Figure 22-15](#) illustrates this receive message identifier check.

Among all messages that fulfill all six qualifying criteria, the message object with the highest receive priority wins receive acceptance filtering round, and is selected to store the received frame. All other message objects lose receive acceptance filtering round.

The following priority scheme is defined for the message objects:

A message object a (MOa) has higher receive priority than a message object b (MOb) if the following two conditions are fulfilled (see [Page 22-110](#)):

1. MOa has a higher priority class than MOb. This means, the 2-bit priority bit field MOARa.PRI must be equal or less than bit field MOARb.PRI.
2. If both message objects have the same priority class (MOARa.PRI = MOARb.PRI), MOb is a list successor of MOa. This means that MOb can be reached by means of successively stepping forward in the list, starting from a.

Controller Area Network (MultiCAN) Controller

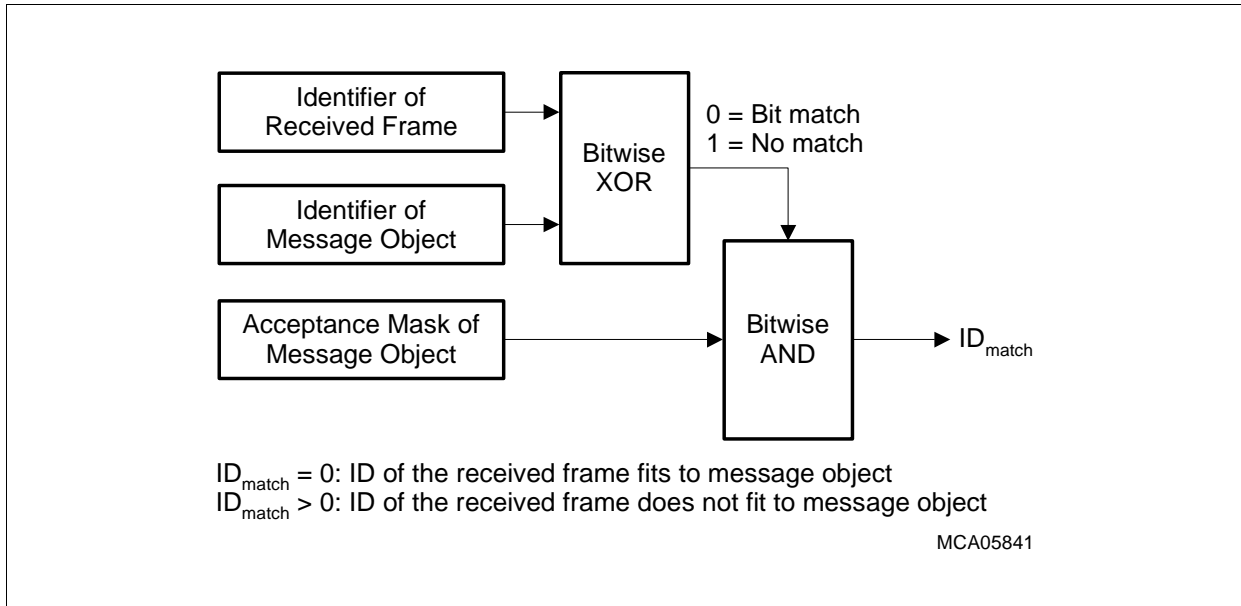


Figure 22-15 Received Message Identifier Acceptance Check

22.3.8.2 Transmit Acceptance Filtering

A message is requested for transmission by setting a transmit request in the message object that holds the message. If more than one message object have a valid transmit request for the same CAN node, one of these message objects is chosen for transmission, because only one single message object can be transmitted at the same time on one CAN bus.

A message object is qualified for transmission on a CAN node if the following four conditions are met (see also [Figure 22-16](#)).

- The message object is allocated to the message object list of the CAN node.
- Bit MOSTATn.MSGVAL is set.
- Bit MOSTATn.TXRQ is set.
- Bit MOSTATn.TXEN0 and MOSTATn.TXEN1 are set.

A priority scheme determines which one of all qualifying message objects is transmitted first. It is assumed that message object a (MOa) and message object b (MOb) are two message objects qualified for transmission. MOa is a list successor of MOb. For both message objects, CAN messages CANa and CANb are defined (identifier, IDE, and RTR are taken from the message-specific bit fields and bits MOARn.ID, MOARn.IDE and MOCTRn.DIR).

If both message objects belong to the same priority class (identical PRI bit field in register MOARn), MOa has a higher transmit priority than MOb if one of the following conditions is fulfilled.

- $PRI = 10_B$ and CAN message MOa has higher or equal priority than CAN message MOb with respect to CAN arbitration rules (see [Table 22-12](#) on [Page 22-111](#)).

Controller Area Network (MultiCAN) Controller

- $PRI = 01_B$ or $PRI = 11_B$ (priority by list order).
- $PRI = 00_B$ and the actual matrix position in a TTCAN network matches the matrix position(s) given in MOAMRn of MOa, i.e.
 $CYCLE \ \& \ MCYCLE = CYCCTM.BCC \ \& \ MCYCLE$ and
 $COLUMN \ \& \ MCOLUMN = CYCCTM.CSM \ \& \ MCOLUMN$
 A message object with $PRI = 00_B$ can be transmitted only by an active TTCAN node.

The message object that is qualified for transmission and has highest transmit priority wins the transmit acceptance filtering, and will be transmitted first. All other message objects lose the current transmit acceptance filtering round. They get a new chance in subsequent acceptance filtering rounds.

The three priority rules listed before are valid for normal CAN operation (without TTCAN functionality) as well as for arbitration windows within a TTCAN system matrix.

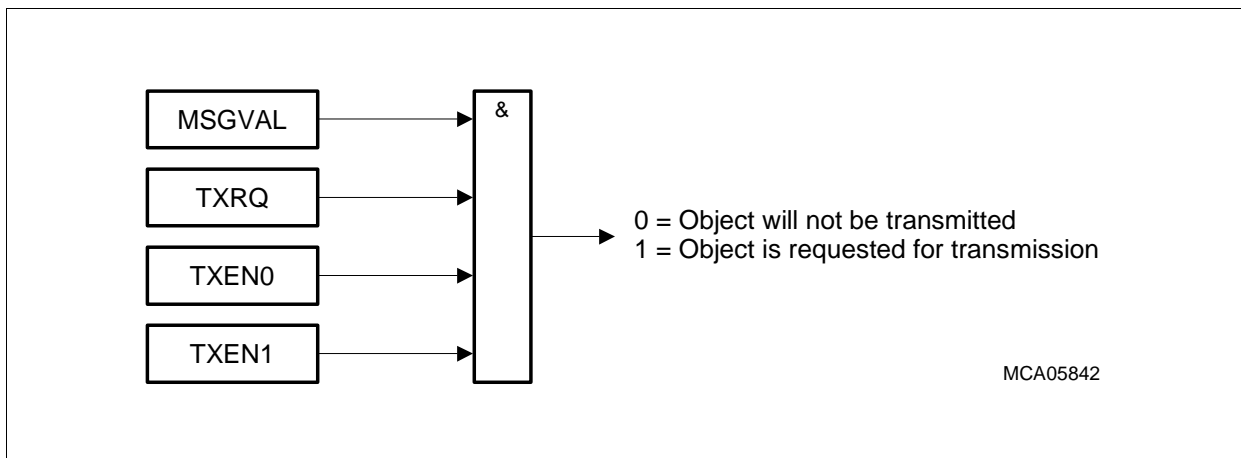


Figure 22-16 Effective Transmit Request of Message Object

Transmission Acceptance Filtering in TTCAN Exclusive Windows

In exclusive windows of a TTCAN, transmit acceptance filtering is performed as described in the previous section but with the exception that only message objects with $MOARn.PRI = 00_B$ can be transmitted. Message objects with other PRI values are not taken into account. Hence the transmit acceptance filtering process selects the first message object in the list that meets the following conditions:

1. $MOARn.PRI = 00_B$
 - a) $MSGVAL \ \& \ TXRQ \ \& \ TXEN0 \ \& \ TXEN1 = 1$ (i.e. the message object qualifies for transmission)
 - b) The actual matrix position in a TTCAN network matches the matrix position(s) given in MOAMR of MOa, i.e.
 $CYCLE \ \& \ MCYCLE = CYCCTM.BCC \ \& \ MCYCLE$ and
 $COLUMN \ \& \ MCOLUMN = CYCCTM.CSM \ \& \ MCOLUMN$

22.3.9 Message Postprocessing

After a message object has successfully received or transmitted a frame, the CPU can be notified to perform a postprocessing on the message object. The postprocessing of the MultiCAN module consists of two elements:

1. Message interrupts to trigger postprocessing.
2. Message pending registers to collect pending message interrupts into a common structure for postprocessing.

22.3.9.1 Message Object Interrupts

When the storage of a received frame into a message object or the successful transmission of a frame is completed, a message interrupt can be issued. For each message object, a transmit and a receive interrupt can be generated and routed to one of the sixteen CAN interrupt output lines (see [Figure 22-17](#)). A receive interrupt occurs also after a frame storage event that has been induced by a FIFO or a gateway action. The status bits TXPND and RXPND in the Message Object n Status Register are always set after a successful transmission/reception, whether or not the respective message interrupt is enabled.

A third FIFO full interrupt condition of a message object is provided. If bit field MOFCRn.OVIE (Overflow Interrupt Enable) is set, the FIFO full interrupt be activated depending on the actual message object type.

In case of a Receive FIFO Base Object (MOFCRn.MMC = 0001_B), the FIFO full interrupt is routed to the interrupt output line INT_Om as defined by the transmit interrupt node pointer MOIPRn.TXINP.

In case of a Transmit FIFO Base Object (MOFCRn.MMC = 0010_B), the FIFO full interrupt becomes routed to the interrupt output line INT_Om as defined by the receive interrupt node pointer MOIPRn.RXINP.

See also [“Interrupt Control” on Page 22-210](#) for further processing of the message object interrupts.

Controller Area Network (MultiCAN) Controller

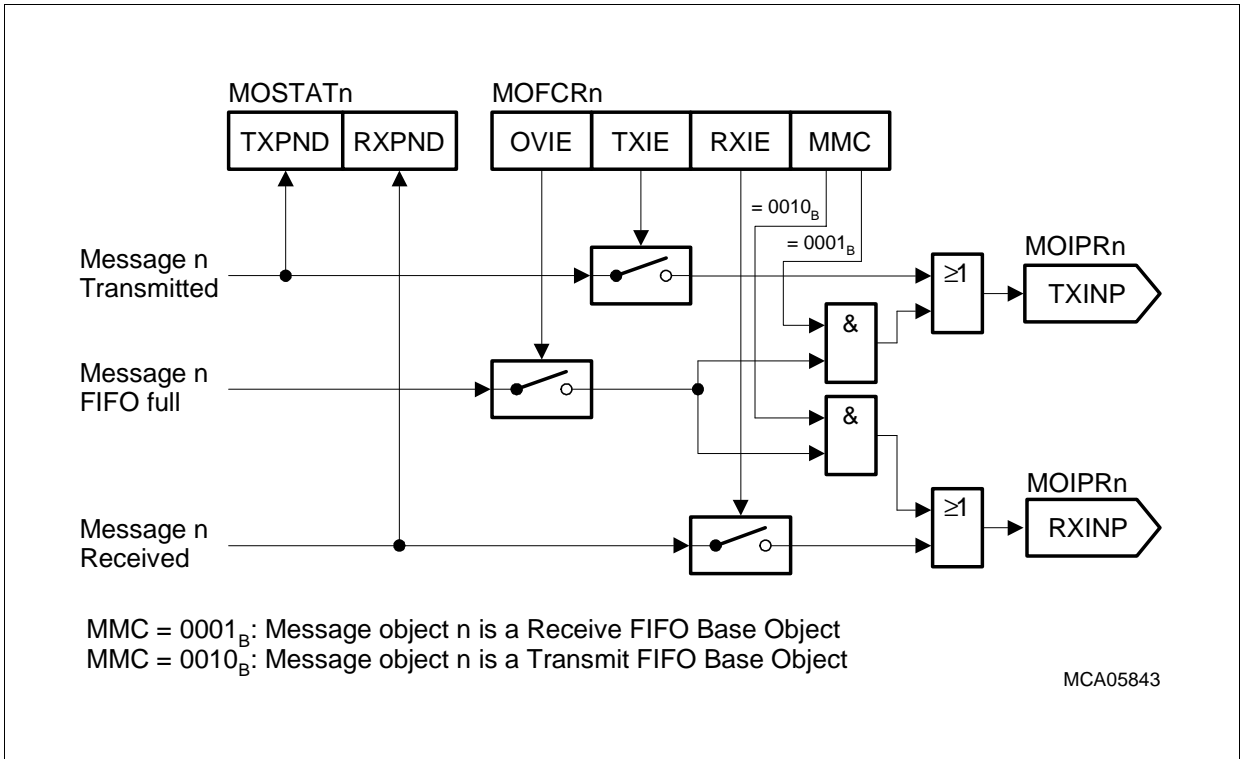


Figure 22-17 Message Interrupt Request Routing

Controller Area Network (MultiCAN) Controller

22.3.9.2 Pending Messages

When a message interrupt request is generated, a message pending bit is set in one of the Message Pending Registers. There are eight Message Pending Registers, MSPNDk (k = 0-7), with 32 pending bits available each, resulting in 256 pending bits. Figure 22-18 shows the allocation of the message pending bits.

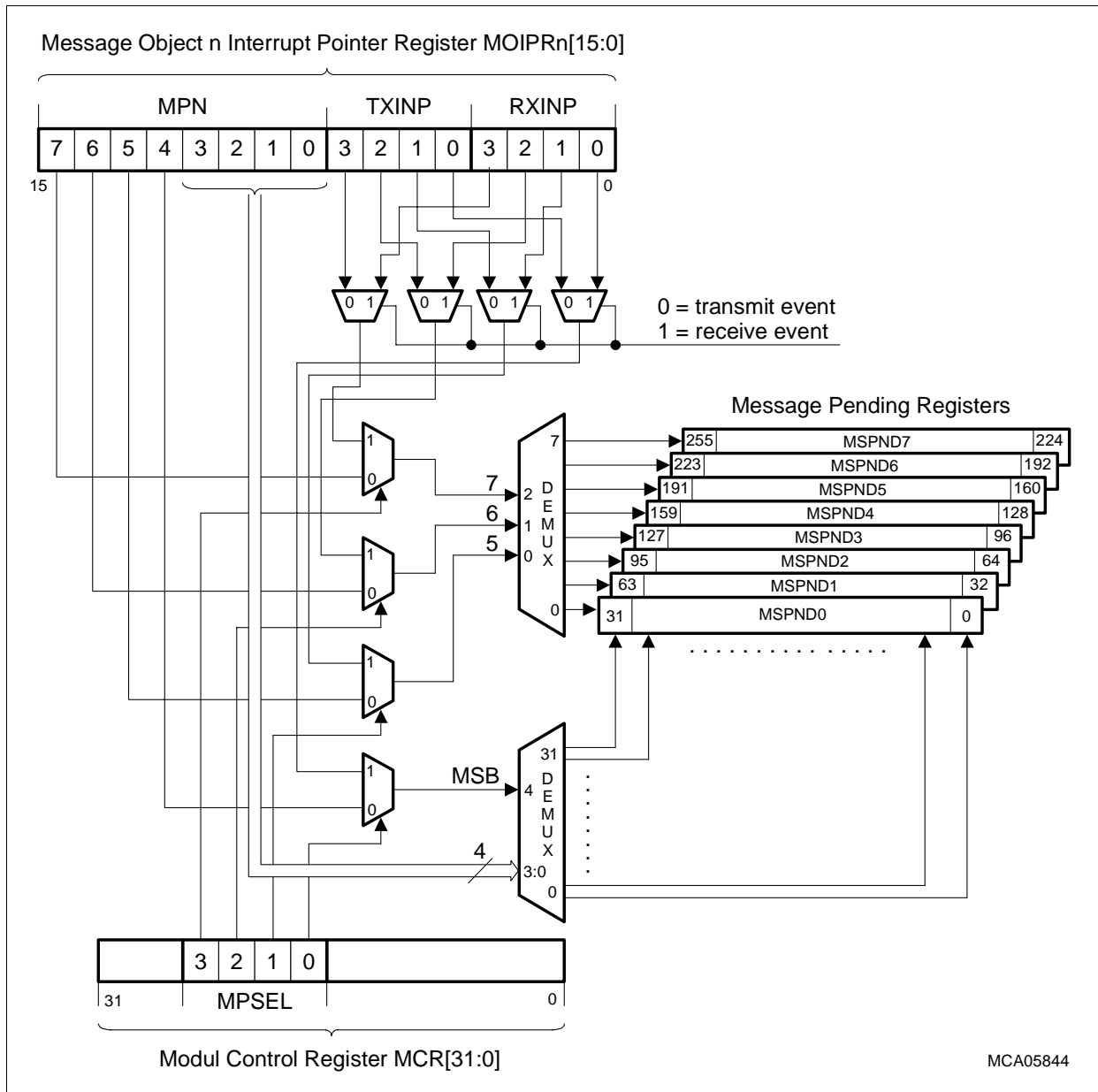


Figure 22-18 Message Pending Bit Allocation

The location of a pending bit is defined by two demultiplexers selecting the number k of the MSPNDk registers (3-bit demux), and the bit location within the corresponding MSPNDk register (5-bit demux).

Controller Area Network (MultiCAN) Controller**Allocation Case 1**

In this allocation case, bit field MCR.MPSEL = 0000_B (see [Page 22-66](#)). The location selection consists of 2 parts:

- The upper three bits of MOINPRn.MPN (MPN[7:5]) select the number k of a Message Pending Register MSPNDk in which the pending bit will be set.
- The lower five bits of MOINPRn.MPN (MPN[4:0]) select the bit position (0-31) in MSPNDk for the pending bit to be set.

Allocation Case 2

In this allocation case, bit field MCR.MPSEL is taken into account for pending bit allocation. Bit field MCR.MPSEL makes it possible to include the interrupt request node pointer for reception (MOIPRn.RXINP) or transmission (MOIPRn.TXINP) for pending bit allocation in such a way that different target locations for the pending bits are used in receive and transmit case. If MPSEL = 1111_B, the location selection operates in the following way:

- At a transmit event, the upper 3 bits of TXINP determine the number k of a Message Pending Register MSPNDk in which the pending bit will be set. At a receive event, the upper 3 bits of RXINP determine the number k.
- The bit position (0-31) in MSPNDk for the pending bit to be set is selected by the lowest bit of TXINP or RXINP (selects between low and high half-word of MSPNDk) and the four least significant bits of MPN.

General Hints

The Message Pending Registers MSPNDk can be written by software. Bits that are written with 1 are left unchanged, and bits which are written with 0 are cleared. This makes it possible to clear individual MSPNDk bits with a single register write access. Therefore, access conflicts are avoided when the MultiCAN module (hardware) sets another pending bit at the same time when software writes to the register.

Each Message Pending Register MSPNDk is associated with a Message Index Register MSIDk (see [Page 22-71](#)) which indicates the lowest bit position of all set (1) bits in Message Pending Register k. The MSIDk register is a read-only register that is updated immediately when a value in the corresponding Message Pending Register k is changed by software or hardware.

22.3.10 Message Object Data Handling

22.3.10.1 Frame Reception

After the reception of a message, it is stored in a message object according to the scheme shown in [Figure 22-19](#). The MultiCAN module not only copies the received data into the message object, and it provides advanced features to enable consistent data exchange between MultiCAN and CPU.

MSGVAL

Bit MSGVAL (Message Valid) in the Message Object n Status Register MOSTATn is the main switch of the message object. During the frame reception, information is stored in the message object only when MSGVAL = 1. If bit MSGVAL is cleared by the CPU, the MultiCAN module stops all ongoing write accesses to the message object. Now the message object can be re-configured by the CPU with subsequent write accesses to it without being disturbed by the MultiCAN.

RTSEL

When the CPU re-configures a message object during CAN operation (for example clears MSGVAL, modifies the message object and sets MSGVAL again), the following scenario can occur:

1. The message object wins receive acceptance filtering.
2. The CPU clears MSGVAL to re-configure the message object.
3. The CPU sets MSGVAL again after re-configuration.
4. The end of the received frame is reached. As MSGVAL is set, the received data is stored in the message object, a message interrupt request is generated, gateway and FIFO actions are processed etc.

After the re-configuration of the message object (after step 3 above) the storage of further received data may be undesirable. This can be achieved through bit MOCTRn.RTSEL (Receive/Transmit Selected) that makes it possible to disconnect a message object from an ongoing frame reception.

When a message object wins the receive acceptance filtering, its RTSEL bit is set by the MultiCAN module to indicate an upcoming frame delivery. The MultiCAN module checks RTSEL whether it is set on successful frame reception to verify that the object is still ready for receiving the frame. The received frame is then stored in the message object (along with all subsequent actions such as message interrupts, FIFO & gateway actions, flag updates) only if RTSEL = 1.

When a message object is invalidated during CAN operation (clearing bit MSGVAL), RTSEL should be cleared before setting MSGVAL again (latest with the same write access that sets MSGVAL) to prevent the storage of a frame that belongs to the old

Controller Area Network (MultiCAN) Controller

context of the message object. Therefore, a message object re-configuration should consist of the following steps:

1. Clear MSGVAL bit
2. Re-configure the message object while MSGVAL = 0
3. Clear RTSEL bit and set MSGVAL again

RXEN

Bit MOSTATn.RXEN enables a message object for frame reception. A message object can receive CAN messages from the CAN bus only if RXEN = 1. The MultiCAN module evaluates RXEN only during receive acceptance filtering. After receive acceptance filtering, RXEN is ignored and has no further influence on the actual storage of a received message in a message object.

Bit RXEN enables the “soft phase out” of a message object: after clearing RXEN, a currently received CAN message for which the message object has won acceptance filtering is still stored in the message object but for subsequent messages the message object no longer wins receive acceptance filtering.

RXUPD, NEWDAT and MSGLST

An ongoing frame storage process is indicated by the RXUPD (Receive Updating) flag in the MOSTATn register. RXUPD is set with the start and cleared with the end of a message object update, which consists of frame storage as well as flag updates.

After storing the received frame (identifier, IDE bit, DLC and for Data Frames also the Data Field), the NEWDAT (New Data) bit of the message object is set. If NEWDAT was already set before it becomes set again, bit MSGLST (Message Lost) is set to indicate a data loss condition.

The RXUPD and NEWDAT flags can help to read consistent frame data from the message object during an ongoing CAN operation. The following steps are recommended to be executed:

1. Clear NEWDAT bit.
2. Read message content (identifier, data etc.) from the message object.
3. Check that both, NEWDAT and RXUPD, are cleared. If this is not the case go back to step 1.
4. When step 3 was successful, the message object contents are consistent and have not been updated by the MultiCAN module while reading.

Bits RXUPD, NEWDAT and MSGLST have the same behavior for the reception of data as well as Remote Frames.

Controller Area Network (MultiCAN) Controller

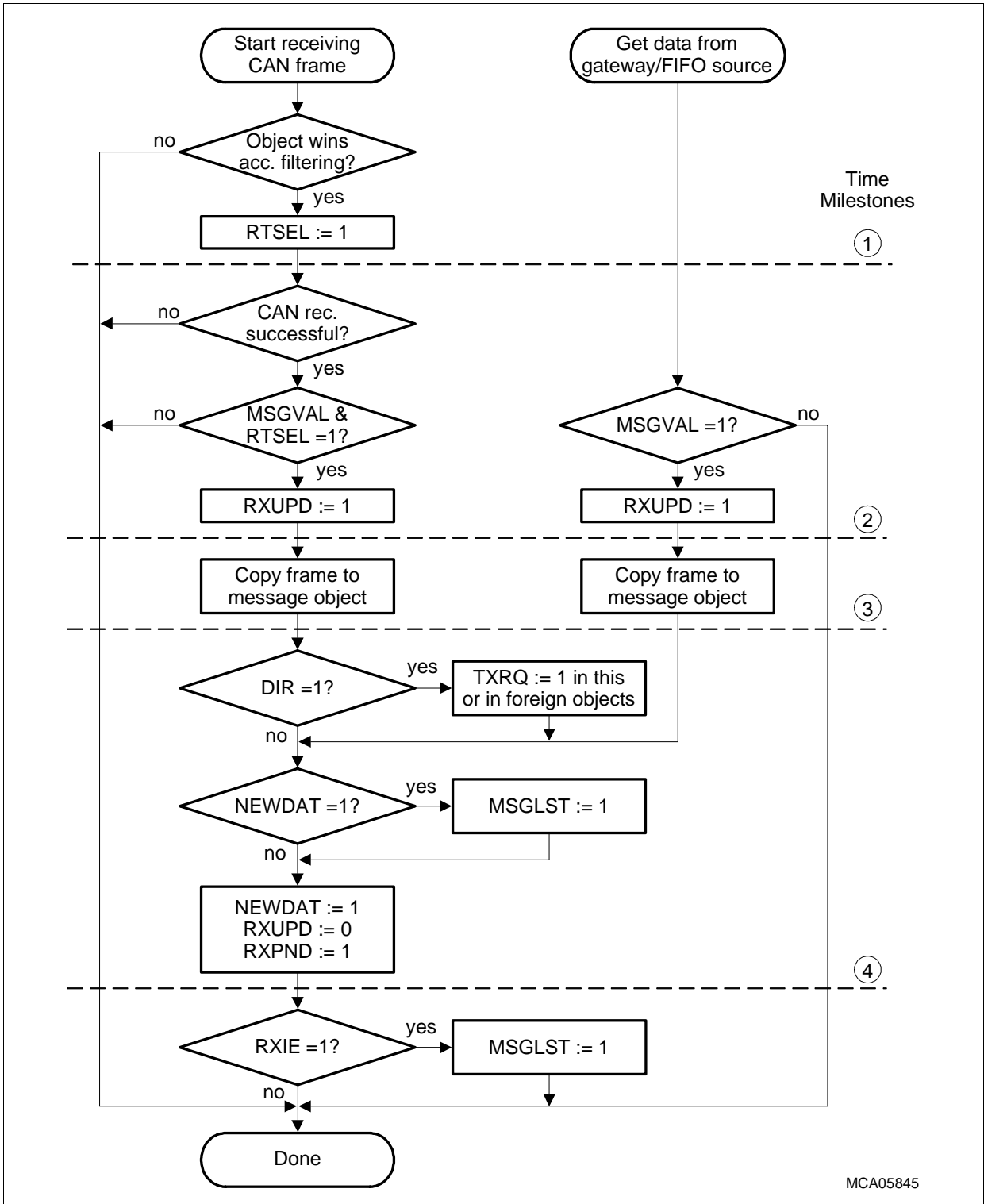


Figure 22-19 Reception of a Message Object

Controller Area Network (MultiCAN) Controller

22.3.10.2 Frame Transmission

The process of a message object transmission is shown in [Figure 22-20](#). Along with the copy of the message object content to be transmitted (identifier, IDE bit, RTR = DIR bit, DLC, and for Data Frames also the Data Field) into the internal transmit buffer of the assigned CAN node, several status flags are also served and monitored to control consistent data handling.

The transmission process of a message object starting after the transmit acceptance filtering is identical for Remote and Data Frames.

MSGVAL, TXRQ, TXEN0, TXEN1

A message can only be transmitted if all four bits in register MOSTATn, MSGVAL (Message Valid), TXRQ (Transmit Request), TXEN0 (Transmit Enable 0), TXEN1 (Transmit Enable 1) are set as shown [Figure 22-16](#). Although these bits are equivalent with respect to the transmission process, they have different semantics:

Table 22-3 Message Transmission Bit Definitions

Bit	Description
MSGVAL	<p>Message Valid This is the main switch bit of the message object.</p>
TXRQ	<p>Transmit Request This is the standard transmit request bit. This bit must be set whenever a message object should be transmitted. TXRQ is cleared by hardware at the end of a successful transmission, except when there is new data (indicated by NEWDAT = 1) to be transmitted. When bit MOFCRn.STT (Single Transmit Trial) is set, TXRQ becomes already cleared when the contents of the message object are copied into the transmit frame buffer of the CAN node. A received remote request (after a Remote Frame reception) sets bit TXRQ to request the transmission of the requested Data Frame.</p>
TXEN0	<p>Transmit Enable 0 This bit can be temporarily cleared by software to suppress the transmission of this message object when it writes new content to the Data Field. This avoids transmission of inconsistent frames that consist of a mixture of old and new data. Remote requests are still accepted when TXEN0 = 0, but transmission of the Data Frame is suspended until transmission is re-enabled by software (setting TXEN0).</p>

Controller Area Network (MultiCAN) Controller

Table 22-3 Message Transmission Bit Definitions (cont'd)

Bit	Description
TXEN1	<p>Transmit Enable 1</p> <p>This bit is used in Transmit FIFOs to select the message object that is transmit active within the FIFO structure.</p> <p>For message objects that are not Transmit FIFO elements, TXEN1 can either be set permanently to 1 or can be used as a second independent transmission enable bit.</p>

RTSEL

When a message object has been identified to be transmitted next after transmission acceptance filtering, bit MOCTRn.RTSEL (Receive/Transmit Selected) is set.

When the message object is copied into the internal transmit buffer, bit RTSEL is checked, and the message is transmitted only if RTSEL = 1. After the successful transmission of the message, bit RTSEL is checked again and the message postprocessing is only executed if RTSEL = 1.

For a complete re-configuration of a valid message object, the following steps should be executed:

1. Clear MSGVAL bit
2. Re-configure the message object while MSGVAL = 0
3. Clear RTSEL and set MSGVAL

Clearing RTSEL ensures that the message object is disconnected from an ongoing/scheduled transmission and no message object processing (copying message to transmit buffer including clearing NEWDAT, clearing TXRQ, time stamp update, message interrupt, etc.) within the old context of the object can occur after the message object becomes valid again, but within a new context.

NEWDAT

When the contents of a message object have been transferred to the internal transmit buffer of the CAN node, bit MOSTATn.NEWDAT (New Data) is cleared by hardware to indicate that the transmit message object data is no longer new.

When the transmission of the frame is successful and NEWDAT is still cleared (if no new data has been copied into the message object meanwhile), TXRQ (Transmit Request) is cleared automatically by hardware.

If, however, the NEWDAT bit has been set again by the software (because a new frame should be transmitted), TXRQ is not cleared to enable the transmission of the new data.

Controller Area Network (MultiCAN) Controller

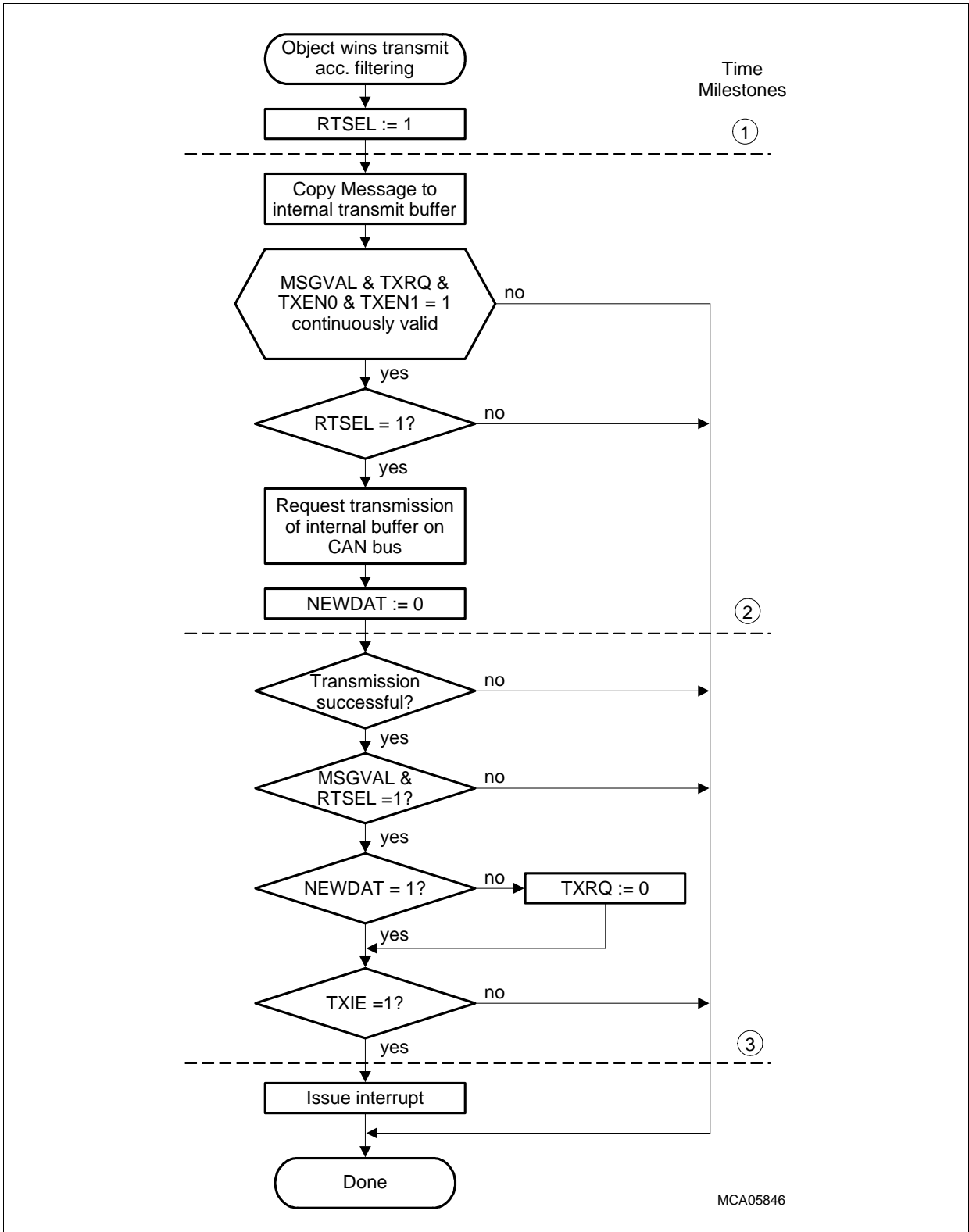


Figure 22-20 Transmission of a Message Object

22.3.11 Message Object Functionality

22.3.11.1 Standard Message Object

A message object is selected as standard message object when bit field MOFCRn.MMC = 0000_B (see [Page 22-101](#)). The standard message object can transmit and receive CAN frames according to the basic rules described in the previous sections. Additional services such as Single Data Transfer Mode or Single Transmit Trial (see following sections) are available and can be individually selected.

22.3.11.2 Single Data Transfer Mode

Single Data Transfer Mode is a useful feature in order to broadcast data over the CAN bus without unintended duplication of information. Single Data Transfer Mode is selected via bit MOFCRn.SDT.

Message Reception

When a received message stored in a message object is overwritten by a new received message, the contents of the first message are lost and replaced with the contents of the new received message (indicated by MSGLST = 1).

If SDT is set (Single Data Transfer Mode activated), bit MSGVAL of the message object is automatically cleared by hardware after the storage of a received Data Frame. This prevents the reception of further messages.

After the reception of a Remote Frame, bit MSGVAL is not automatically cleared.

Message Transmission

When a message object receives a series of multiple remote requests, it transmits several Data Frames in response to the remote requests. If the data within the message object has not been updated in the time between the transmissions, the same data can be sent more than once on the CAN bus.

In Single Data Transfer Mode (SDT = 1), this is avoided because MSGVAL is automatically cleared after the successful transmission of a Data Frame.

After the transmission of a Remote Frame, bit MSGVAL is not automatically cleared.

22.3.11.3 Single Transmit Trial

If the bit STT in the message object function register is set (STT = 1), the transmission request is cleared (TXRQ := 0) when the frame contents of the message object have been copied to the internal transmit buffer of the CAN node. Thus, the transmission of the message object is not tried again when it fails due to CAN bus errors.

Controller Area Network (MultiCAN) Controller

22.3.11.4 Message Object FIFO Structure

In case of high CPU load it may be difficult to process a series of CAN frames in time. This may happen if multiple messages are received or must be transmitted in short time.

Therefore, a FIFO buffer structure is available to avoid loss of incoming messages and to minimize the setup time for outgoing messages. The FIFO structure can also be used to automate the reception or transmission of a series of CAN messages and to generate a single message interrupt when the whole CAN frame series is done.

There can be several FIFOs in parallel. The number of FIFOs and their size are limited only by the number of available message objects. A FIFO can be installed, resized and de-installed at any time, even during CAN operation.

The basic structure of a FIFO is shown in [Figure 22-21](#). A FIFO consists of one base object and n slave objects. The slave objects are chained together in a list structure (similar as in message object lists). The base object may be allocated to any list. Although [Figure 22-21](#) shows the base object as a separate part beside the slave objects, it is also possible to integrate the base object at any place into the chain of slave objects. This means that the base object is slave object, too (not possible for gateways). The absolute object numbers of the message objects have no impact on the operation of the FIFO.

The base object does not need be allocated to the same list as the slave objects. Only the slave object must be allocated to a common list (as they are chained together). Several pointers (BOT, CUR and TOP) that are located in the Message Object n FIFO/Gateway Pointer Register MOFGPRn link the base object to the slave objects, regardless whether the base object is allocated to the same or to another **list** than the slave objects.

The smallest FIFO would be a single message object which is both, FIFO base and FIFO slave (not very useful). The biggest possible FIFO structure would include all message objects of the MultiCAN module. Any FIFO sizes between these limits are possible.

In the FIFO base object, the FIFO boundaries are defined. Bit field MOFGPRn.BOT of the base object points to (includes the number of) the bottom slave object in the FIFO structure. The MOFGPRn.TOP bit field points to (includes the number of) the top slave object in the FIFO structure. The MOFGPRn.CUR bit field points to (includes the number of) the slave object that is actually selected by the MultiCAN module for message transfer. When a message transfer takes place with this object, CUR is set to the next message object in the list structure of the slave objects (CUR := PNEXT of current object). If CUR was equal to TOP (top of the FIFO reached), the next update of CUR will result in CUR = BOT (wrap-around from the top to the bottom of the FIFO). This scheme represents a circular FIFO structure where the bit fields BOT and TOP establish the link from the last to the first element.

Bit field MOFGPRn.SEL of the base object can be used for monitoring purposes. It makes it possible to define a slave object within the list at which a message interrupt is

Controller Area Network (MultiCAN) Controller

generated whenever the CUR pointer reaches the value of the SEL pointer. Thus SEL makes it possible to detect the end of a predefined message transfer series or to issue a warning interrupt when the FIFO becomes full.

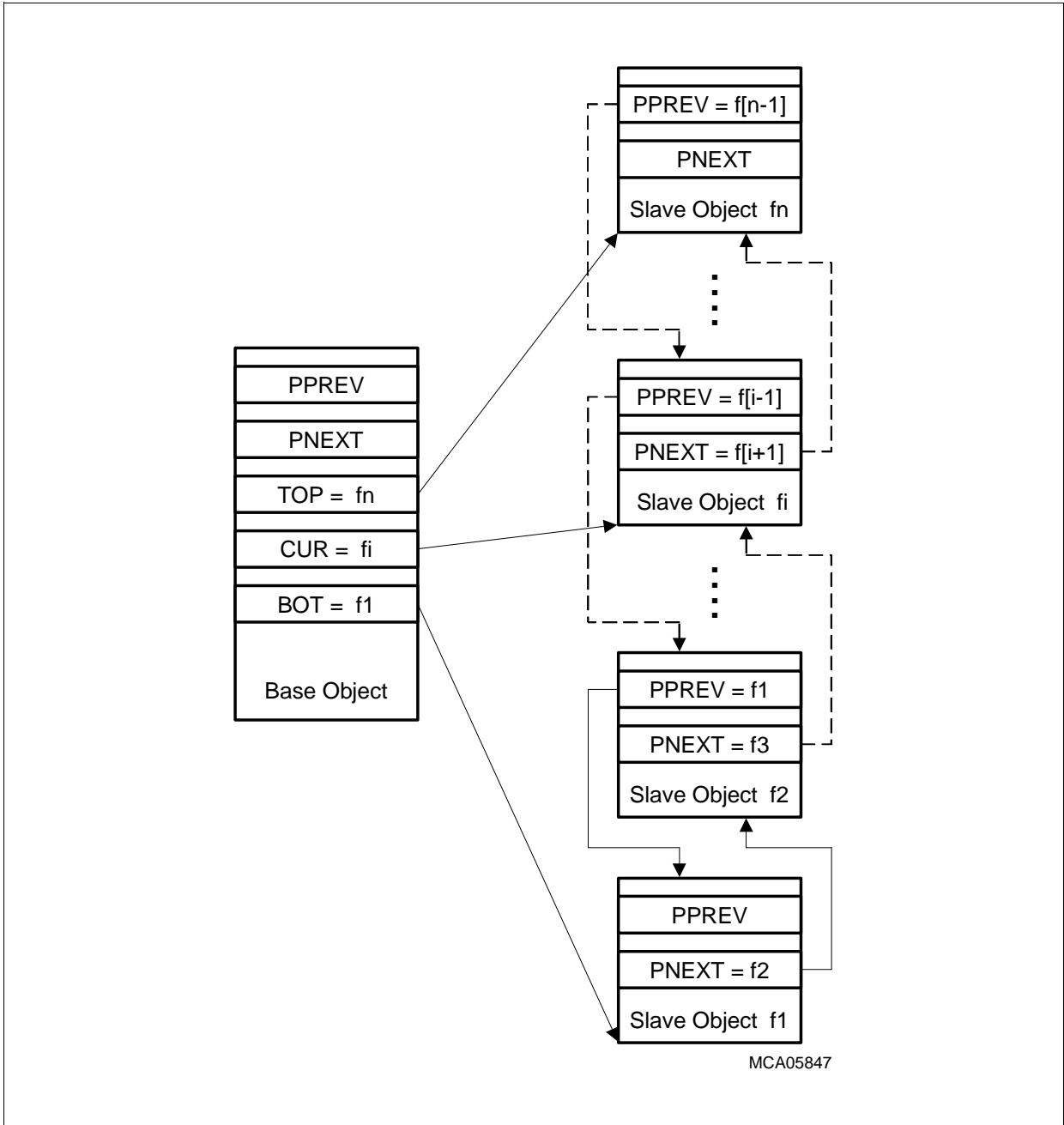


Figure 22-21 FIFO Structure with FIFO Base Object and n FIFO Slave Objects

Controller Area Network (MultiCAN) Controller**22.3.11.5 Receive FIFO**

The Receive FIFO structure is used to buffer incoming (received) Remote or Data Frames.

A Receive FIFO is selected by setting $\text{MOFCRn.MMC} = 0001_{\text{B}}$ in the FIFO base object. This MMC code automatically designates a message object as FIFO base object. The message modes of the FIFO slave objects are not relevant for the operation of the Receive FIFO.

When the FIFO base object receives a frame from the CAN node it belongs to, the frame is not stored in the base object itself but in the message object that is selected by the base object's MOFGPRn.CUR pointer. This message object receives the CAN message as if it is the direct receiver of the message. However, $\text{MOFCRn.MMC} = 0000_{\text{B}}$ is implicitly assumed for the FIFO slave object, and a standard message delivery is performed. The actual message mode (MMC setting) of the FIFO slave object is ignored. For the slave object, no acceptance filtering takes place that checks the received frame for a match with the identifier, IDE bit, and DIR bit.

With the reception of a CAN frame, the current pointer CUR of the base object is set to the number of the next message object in the FIFO structure. This message object will then be used to store the next incoming message.

If bit field MOFCRn.OVIE ("Overflow Interrupt Enable") of the FIFO base object is set and the current pointer MOFGPRn.CUR becomes equal to MOFGPRn.SEL , a FIFO overflow interrupt request is generated. This interrupt request is generated on interrupt node TXINP of the base object immediately after the storage of the received frame in the slave object. Transmit interrupts are still generated if TXIE is set.

A CAN message is stored in FIFO base and slave object only if $\text{MSGVAL} = 1$.

In order to avoid direct reception of a message by a slave message object, as if it was an independent message object and not a part of a FIFO, the bit RXEN of each slave object must be cleared. The setting of the bit RXEN is "don't care" only if the slave object is located in a list not assigned to a CAN node.

Controller Area Network (MultiCAN) Controller**22.3.11.6 Transmit FIFO**

The Transmit FIFO structure is used to buffer a series of Data or Remote Frames that must be transmitted. A transmit FIFO consists of one base message object and one or more slave message objects."

A Transmit FIFO is selected by setting MOFCRn.MMC = 0010_B in the FIFO base object. Unlike the Receive FIFO, slave objects assigned to the Transmit FIFO must explicitly set their bit fields MOFCRn.MMC = 0011_B. The CUR pointer in all slave objects must point back to the Transmit FIFO Base Object (to be initialized by software).

The MOSTATn.TXEN1 bits (Transmit Enable 1) of all message objects except the one which is selected by the CUR pointer of the base object must be cleared by software. TXEN1 of the message (slave) object selected by CUR must be set. CUR (of the base object) may be initialized to any FIFO slave object.

When tagging the message objects of the FIFO as valid to start the operation of the FIFO, then the base object must be tagged valid (MSGVAL := 1) first.

Before a Transmit FIFO becomes de-installed during operation, its slave objects must be tagged invalid (MSGVAL = 0).

The Transmit FIFO uses the TXEN1 bit in the Message Object Control Register of all FIFO elements to select the actual message for transmission. Transmit acceptance filtering evaluates TXEN1 for each message object and a message object can win transmit acceptance filtering only if its TXEN bit is set. When a FIFO object has transmitted a message, the hardware clears its TXEN1 bit in addition to standard transmit postprocessing (clear TXRQ, transmit interrupt etc.), and moves the CUR pointer in the next FIFO base object to be transmitted. TXEN1 is set automatically (by hardware) in the next message object. Thus, TXEN1 moves along the Transmit FIFO structure as a token that selects the active element.

If bit field MOFCRn.OVIE ("Overflow Interrupt Enable") of the FIFO base object is set and the current pointer CUR becomes equal to MOFGPRn.SEL, a FIFO overflow interrupt request is generated. The interrupt request is generated on interrupt node RXINP of the base object after postprocessing of the received frame. Receive interrupts are still generated for the Transmit FIFO base object if bit RXIE is set.

Controller Area Network (MultiCAN) Controller

22.3.11.7 Gateway Mode

The Gateway Mode makes it possible to establish an automatic information transfer between two independent CAN buses without CPU interaction.

The Gateway Mode operates on message object level. In Gateway mode, information is transferred between two message objects, resulting in an information transfer between the two CAN nodes to which the message objects are allocated. A gateway may be established with any pair of CAN nodes, and there can be as many gateways as there are message objects available to build the gateway structure.

Gateway Mode is selected by setting MOFCRs.MMC = 0100_B for the gateway source object *s*. The gateway destination object *d* is selected by the MOFGPRd.CUR pointer of the source object. The gateway destination object only needs to be valid (its MSGVAL = 1). All other settings are not relevant for the information transfer from the source object to the destination object.

Gateway source objects behave as a standard message objects with the difference that some additional actions are performed by the MultiCAN module when a CAN frame has been received and stored in the source object (see [Figure 22-22](#)):

1. If bit MOFCR.DLCC is set, the data length code MOFCRs.DLC is copied from the gateway source object to the gateway destination object.
2. If bit MOFCRs.IDC is set, the identifier MOARs.ID and the identifier extension MOARs.IDE are copied from the gateway source object to the gateway destination object.
3. If bit MOFCRs.DATC is set, the data bytes stored in the two data registers MODATALs and MODATAHs are copied from the gateway source object to the gateway destination object. All 8 data bytes are copied, even if MOFCRs.DLC indicates less than 8 data bytes.
4. If bit MOFCRs.GDFS is set, the transmit request flag MOSTATd.TXRQ is set in the gateway destination object.
5. The receive pending bit MOSTATd.RXPND and the new data bit MOSTATd.NEWDAT are set in the gateway destination object.
6. A message interrupt request is generated for the gateway destination object if its MOSTATd.RXIE is set.
7. The current object pointer MOFGPRs.CUR of the gateway source object is moved to the next destination object according to the FIFO rules as described on [Page 22-51](#). A gateway with a single (static) destination object is obtained by setting MOFGPRs.TOP = MOFGPRs.BOT = MOFGPRs.CUR = destination object.

The link from the gateway source object to the gateway destination object works in the same way as the link from a FIFO base to a FIFO slave. This means that a gateway with an integrated destination FIFO may be created; in [Figure 22-21](#), the object on the left is the gateway source object and the message objects on the right side are the gateway destination objects.

Controller Area Network (MultiCAN) Controller

The gateway operates equivalent for the reception of Data Frames (source object is receive object, i.e. DIR = 0), as well as for the reception of Remote Frames (source object is transmit object).

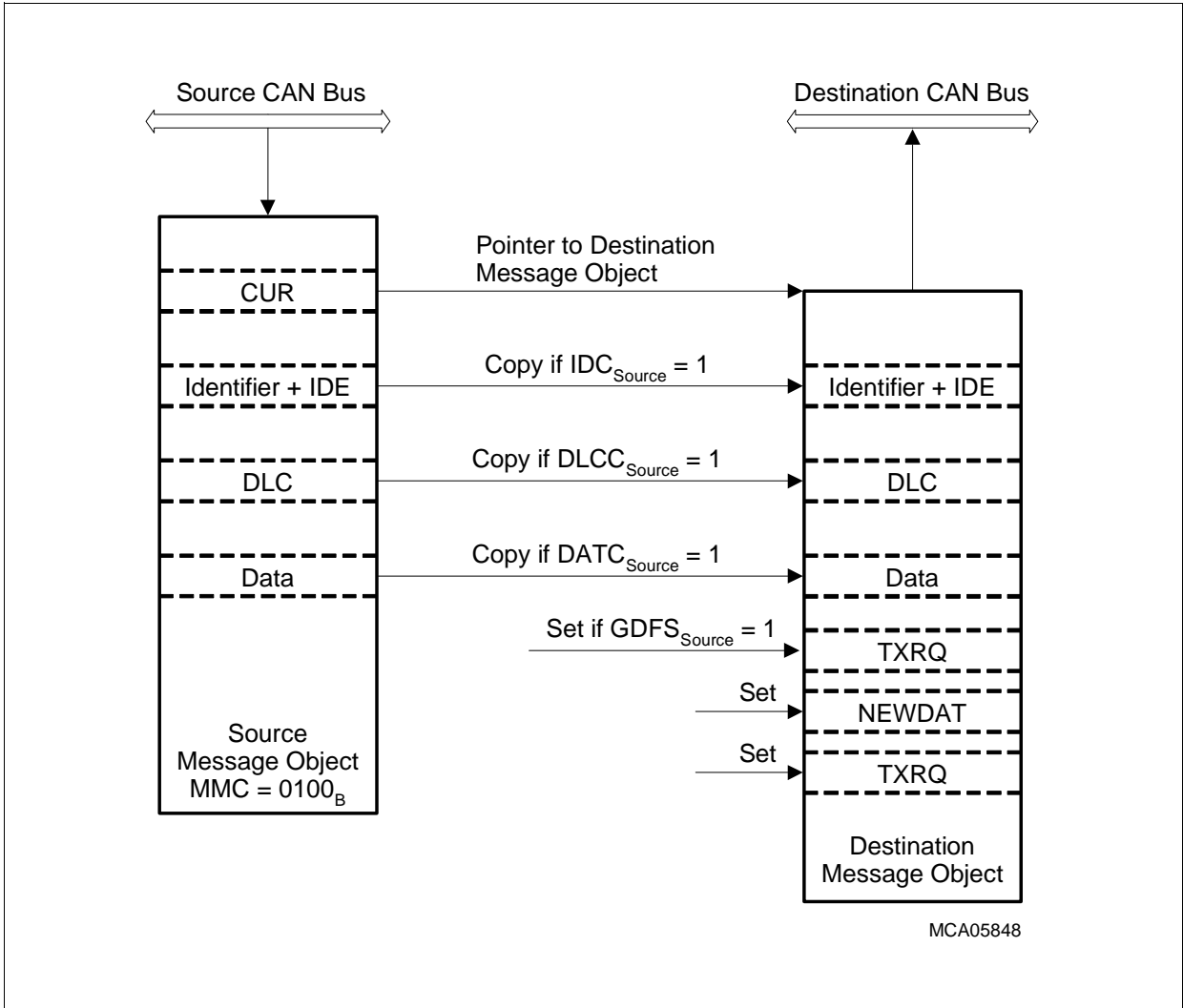


Figure 22-22 Gateway Transfer from Source to Destination

Controller Area Network (MultiCAN) Controller**22.3.11.8 Foreign Remote Requests**

When a Remote Frame has been received on a CAN node and is stored in a message object, a transmit request is set to trigger the answer (transmission of a Data Frame) to the request or to automatically issue a secondary request. If the Foreign Remote Request Enable bit MOFCRn.FRREN is cleared in the message object in which the remote request is stored, MOSTATn.TXRQ is set in the same message object.

If bit FRREN is set (FRREN = 1: foreign remote request enabled), TXRQ is set in the message object that is referenced by pointer MOFGPRn.CUR. The value of CUR is, however, not changed by this feature.

Although the foreign remote request feature works independently of the selected message mode, it is especially useful for gateways to issue a remote request on the source bus of a gateway after the reception of a remote request on the gateway destination bus. According to the setting of FRREN in the gateway destination object, there are two capabilities to handle remote requests that appear on the destination side (assuming that the source object is a receive object and the destination is a transmit object, i.e. $DIR_{source} = 0$ and $DIR_{destination} = 1$):

FRREN = 0 in the Gateway Destination Object

1. A Remote Frame is received by gateway destination object.
2. TXRQ is set automatically in the gateway destination object.
3. A Data Frame with the current data stored in the destination object is transmitted on the destination bus.

FRREN = 1 in the Gateway Destination Object

1. A Remote Frame is received by gateway destination object.
2. TXRQ is set automatically in the gateway source object (must be referenced by CUR pointer of the destination object).
3. A remote request is transmitted by the source object (which is a receive object) on the source CAN bus.
4. The receiver of the remote request responds with a Data Frame on the source bus.
5. The Data Frame is stored in the source object.
6. The Data Frame is copied to the destination object (gateway action).
7. TXRQ is set in the destination object (assuming $GDFS_{source} = 1$).
8. The new data stored in the destination object is transmitted on the destination bus, in response to the initial remote request on the destination bus.

Controller Area Network (MultiCAN) Controller

22.4 MultiCAN Kernel Registers

This section describes the kernel registers of the MultiCAN module. All MultiCAN kernel register names described in this section are also referenced in other parts of the TC1796 User's Manual by the module name prefix "CAN_".

MultiCAN Kernel Register Overview

The MultiCAN Kernel include three blocks of registers:

- Global Module Registers
- Node Registers, for each CAN node x
- Message Object Registers, for each message object n

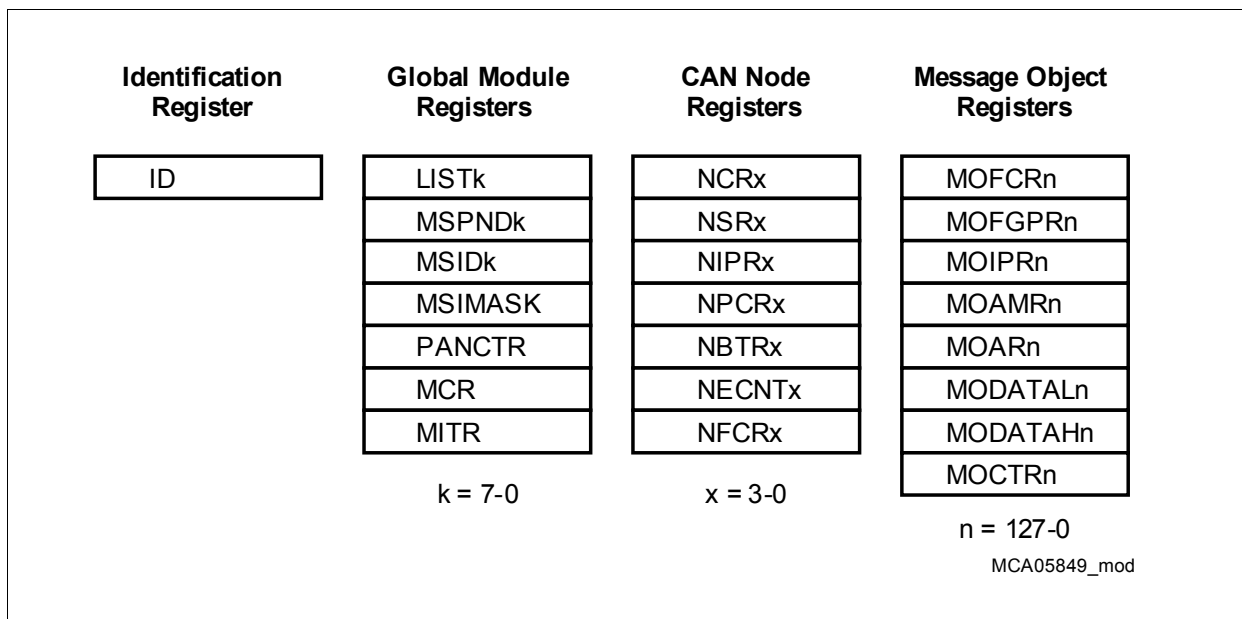


Figure 22-23 MultiCAN Kernel Registers

The complete and detailed address map of the MultiCAN module is described in [Table 18-24](#) on [Page 18-65](#) of the TC1796 User's Manual System Units part (Volume 1).

Table 22-4 Registers Address Space - MultiCAN Kernel Registers

Module	Base Address	End Address	Note
CAN	F000 4000 _H	F000 5FFF _H	—

Controller Area Network (MultiCAN) Controller

Table 22-5 Registers Overview - MultiCAN Kernel Registers

Register Short Name	Register Long Name	Offset Address¹⁾	Description see
ID	Module Identification Register i	008 _H	Page 22-61
LISTk	List Register k	100 _H + k × 4 _H	Page 22-68
MSPNDk	Message Pending Register k	120 _H + k × 4 _H	Page 22-70
MSIDk	Message Index Register k	140 _H + k × 4 _H	Page 22-71
MSIMASK	Message Index Mask Register	1C0 _H	Page 22-72
PANCTR	Panel Control Register	1C4 _H	Page 22-62
MCR	Module Control Register	1C8 _H	Page 22-66
MITR	Module Interrupt Trigger Reg.	1CC _H	Page 22-67
NCRx	Node x Control Register	200 _H + x × 100 _H	Page 22-73
NSRx	Node x Status Register	204 _H + x × 100 _H	Page 22-77
NIPRx	Node x Interrupt Pointer Reg.	208 _H + x × 100 _H	Page 22-81
NPCRx	Node x Port Control Register	20C _H + x × 100 _H	Page 22-83
NBTRx	Node x Bit Timing Register	210 _H + x × 100 _H	Page 22-84
NECNTx	Node x Error Counter Register	214 _H + x × 100 _H	Page 22-86
NFCRx	Node x Frame Counter Register	218 _H + x × 100 _H	Page 22-87
MOFCRn	Message Object n Function Control Register	400 _H + n × 20 _H	Page 22-101
MOFGPRn	Message Object n FIFO/Gateway Pointer Register	404 _H + n × 20 _H	Page 22-106
MOIPRn	Message Object n Interrupt Pointer Register	408 _H + n × 20 _H	Page 22-99
MOAMRn	Message Object n Acceptance Mask Register	40C _H + n × 20 _H	Page 22-107
MODATALn	Message Object n Data Register Low	410 _H + n × 20 _H	Page 22-112
MODATAHn	Message Object n Data Register High	414 _H + n × 20 _H	Page 22-113
MOARN	Message Object n Arbitration Register	418 _H + n × 20 _H	Page 22-109
MOCTRn MOSTATn	Message Object n Control Reg. Message Object n Status Reg.	41C _H + n × 20 _H	Page 22-91 Page 22-94

Controller Area Network (MultiCAN) Controller

- 1) The absolute register address is calculated as follows:
 Module Base Address (**Table 22-4**) + Offset Address (shown in this column)
 Further, the following ranges for parameters i, k, x, and n are valid: i = 0-7, k = 0-7, x = 0-1, n = 0-63.

Figure 22-24 shows the MultiCAN register address map.

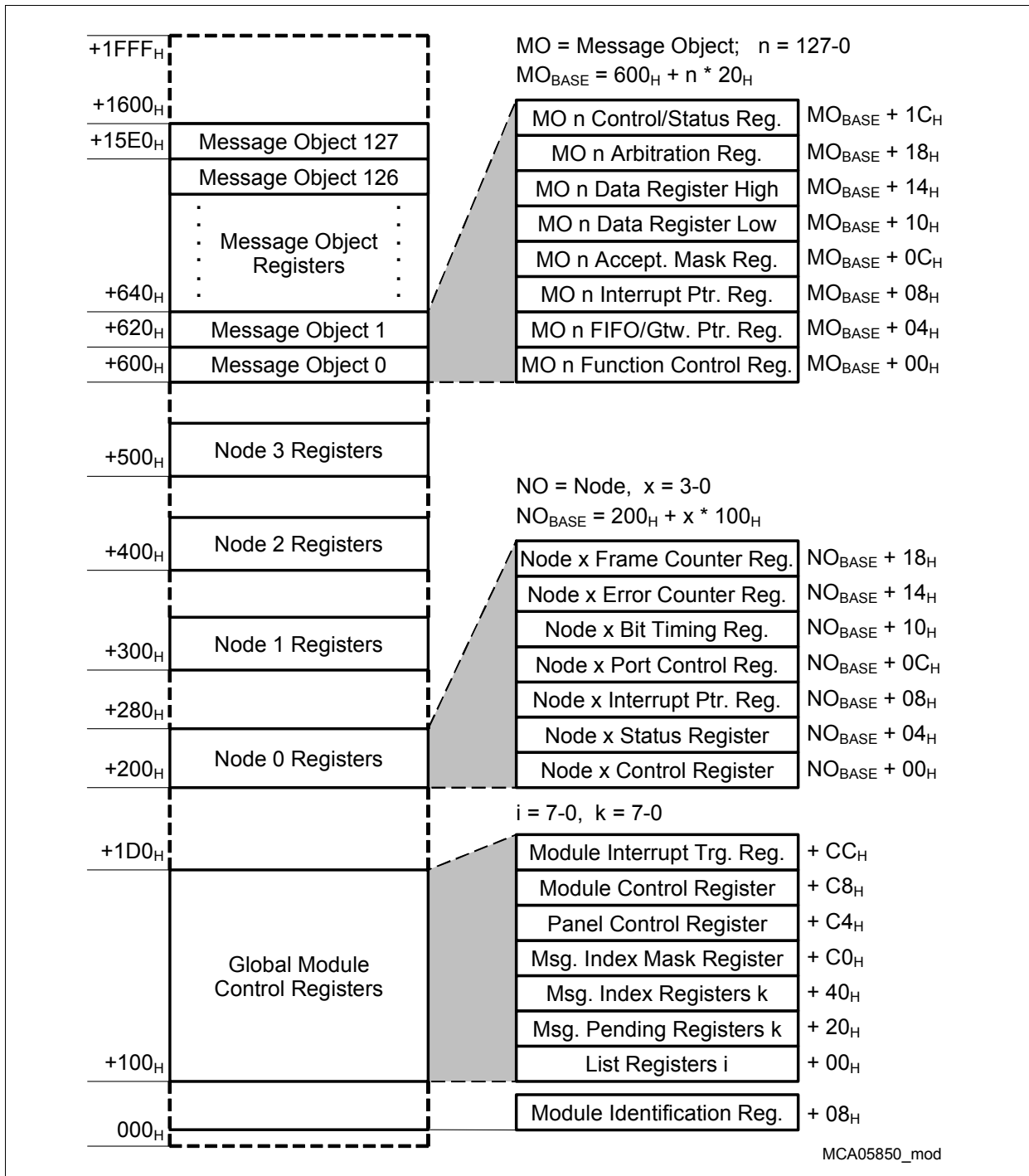


Figure 22-24 MultiCAN Kernel Register Address Map

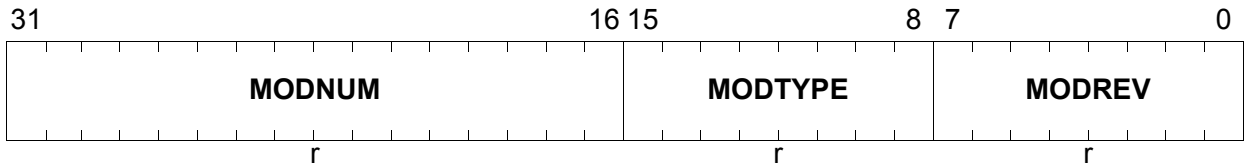
Controller Area Network (MultiCAN) Controller

22.4.1 Module Identification Register

The Module Identification Register ID contains read-only information about the MultiCAN module version.

ID

Module Identification Register (008_H) **Reset Value: 002B C0XX_H**



Field	Bits	Type	Description
MODREV	[7:0]	r	Module Revision Number MODREV defines the module revision number. The value of a module revision starts with 01 _H (first revision).
MODTYPE	[15:8]	r	Module Number Value This bit field defines the module as a 32-bit module: C0 _H
MODNUM	[31:16]	r	Module Number Value This bit field defines the module identification number for the MultiCAN: 002B _H

Controller Area Network (MultiCAN) Controller

22.4.2 Global Module Register

All list operations such as allocation, de-allocation and relocation of message objects within the list structure are performed via the Command Panel. It is not possible to modify the list structure directly by software by writing to the message objects and the LIST registers.

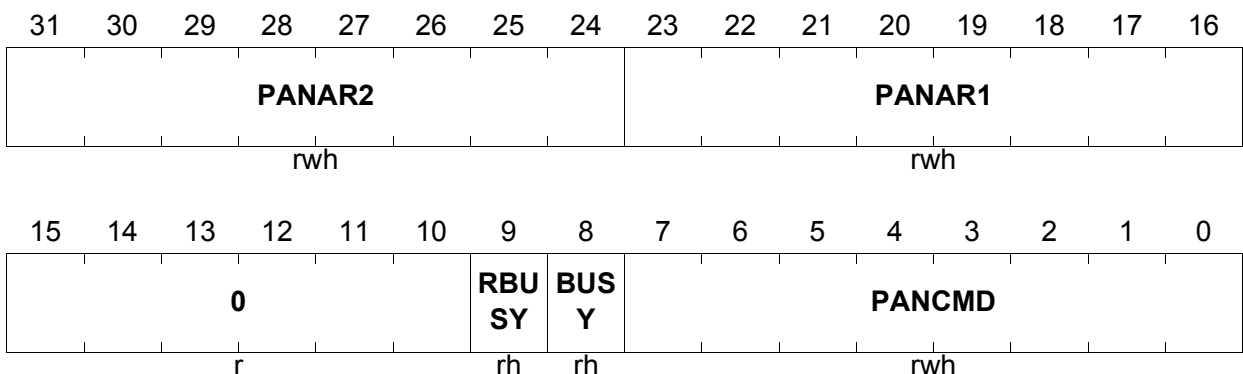
The Panel Control Register PANCTR is used to start a new command by writing the command arguments and the command code into its bit fields.

PANCTR

Panel Control Register

(1C4_H)

Reset Value: 0000 0301_H



Field	Bits	Type	Description
PANCMD	[7:0]	rwh	Panel Command This bit field is used to start a new command by writing a panel command code into it. At the end of a panel command, the NOP (no operation) command code is automatically written into PANCMD. The coding of PANCMD is defined in Table 22-6 .
BUSY	8	rh	Panel Busy Flag 0 _B Panel has finished command and is ready to accept a new command. 1 _B Panel operation is in progress.
RBUSY	9	rh	Result Busy Flag 0 _B No update of PANAR1 and PANAR2 is scheduled by the list controller. 1 _B A list command is running (BUSY = 1) that will write results to PANAR1 and PANAR2, but the results are not yet available.

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
PANAR1	[23:16]	rwh	Panel Argument 1 See Table 22-6 .
PANAR2	[31:24]	rwh	Panel Argument 2 See Table 22-6 .
0	[15:10]	r	Reserved Read as 0; should be written with 0.

Panel Commands

A panel operation consists of a command code (PANCMD) and up to two panel arguments (PANAR1, PANAR2). Commands that have a return value deliver it to the PANAR1 bit field. Commands that return an error flag deliver it to bit 31 of the Panel Control Register, this means to bit 7 of PANAR2.

Table 22-6 Panel Commands

PANCMD	PANAR2	PANAR1	Command Description
00 _H	–	–	No Operation Writing 00 _H to PANCMD has no effect. No new command is started.
01 _H	Result: Bit 7: ERR Bit 6-0: undefined	–	Initialize Lists Run the initialization sequence to reset the CTRL and LIST fields of all message objects. List registers LIST[7:0] are set to their reset values. This results in the de-allocation of all message objects. The initialization command requires that bits NCRx.INIT and NCRx.CCE are set for all CAN nodes (x = 0-3). Bit 7 of PANAR2 (ERR) reports the success of the operation: 0 _B Initialization was successful 1 _B Not all NCRx.INIT and NCRx.CCE bits are set. Therefore, no initialization was performed. The initialize lists command is automatically performed with each reset of the MultiCAN module, but with the exception that all message object registers are reset, too.

Controller Area Network (MultiCAN) Controller

Table 22-6 Panel Commands (cont'd)

PANCMD	PANAR2	PANAR1	Command Description
02 _H	Argument: List Index	Argument: Message Object Number	Static Allocate Allocate message object to a list. The message object is removed from the list that it currently belongs to, and appended to the end of the list. given by PANAR2. This command is also used to deallocate a message object. In this case, the target list is the list of unallocated elements (PANAR2 = 0).
03 _H	Argument: List Index Result: Bit 7: ERR Bit 6-0: undefined	Result: Message Object Number	Dynamic Allocate Allocate the first message object of the list of unallocated objects to the selected list. The message object is appended to the end of the list. The message number of the message object is returned in PANAR1. An ERR bit (bit 7 of PANAR2) reports the success of the operation: 0 _B Success. 1 _B The operation has not been performed because the list of unallocated elements was empty.
04 _H	Argument: Destination Object Number	Argument: Source Object Number	Static Insert Before Remove a message object (source object) from the list that it currently belongs to, and insert it before a given destination object into the list structure of the destination object. The source object thus becomes the predecessor of the destination object.

Controller Area Network (MultiCAN) Controller

Table 22-6 Panel Commands (cont'd)

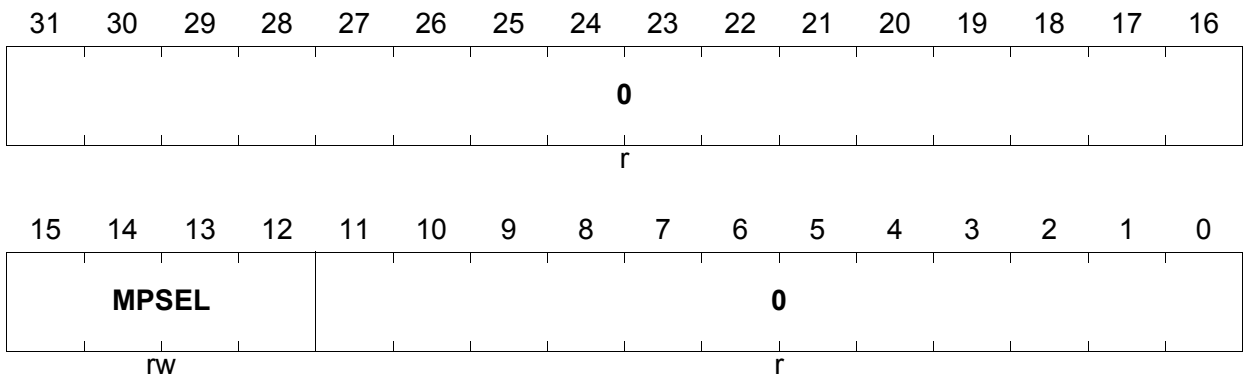
PANCMD	PANAR2	PANAR1	Command Description
05 _H	Argument: Destination Object Number Result: Bit 7: ERR Bit 6-0: undefined	Result: Object Number of inserted object	Dynamic Insert Before Insert a new message object before a given destination object. The new object is taken from the list of unallocated elements (the first element is chosen). The number of the new object is delivered as a result to PANAR1. An ERR bit (bit 7 of PANAR2) reports the success of the operation: 0 _B Success. 1 _B The operation has not been performed because the list of unallocated elements was empty.
06 _H	Argument: Destination Object Number	Argument: Source Object Number	Static Insert Behind Remove a message object (source object) from the list that it currently belongs to, and insert it behind a given destination object into the list structure of the destination object. The source object thus becomes the successor of the destination object.
07 _H	Argument: Destination Object Number Result: Bit 7: ERR Bit 6-0: undefined	Result: Object Number of inserted object	Dynamic Insert Behind Insert a new message object behind a given destination object. The new object is taken from the list of unallocated elements (the first element is chosen). The number of the new object is delivered as result to PANAR1. An ERR bit (bit 7 of PANAR2) reports the success of the operation: 0 _B Success. 1 _B The operation has not been performed because the list of unallocated elements was empty.
08 _H - FF _H	–	–	Reserved

Controller Area Network (MultiCAN) Controller

The Module Control Register MCR contains basic settings to determine the operation of the MultiCAN module.

MCR

Module Control Register (1C8_H) **Reset Value: 0000 0000_H**



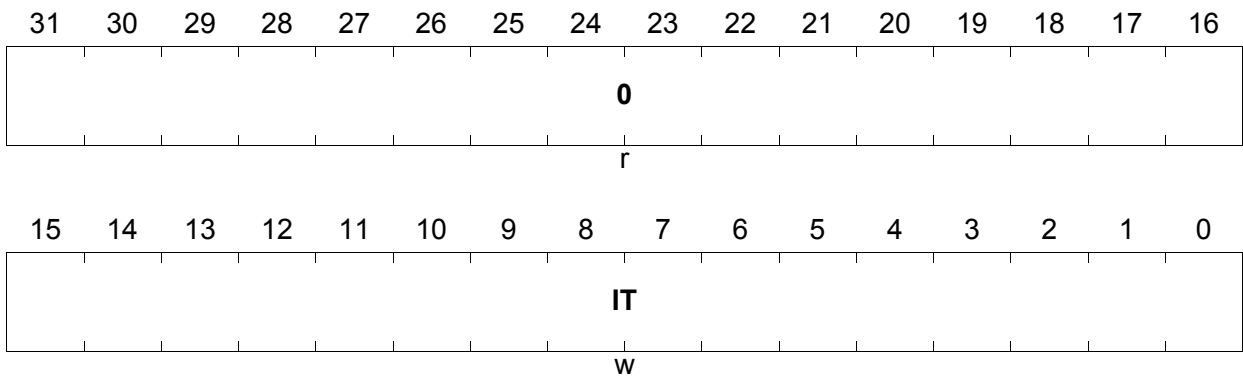
Field	Bits	Type	Description
MPSEL	[15:12]	rw	Message Pending Selector Bit field MPSEL makes it possible to select the bit position of the message pending bit after a message reception/transmission by a mixture of the MOIPRn register bit fields RXINP, TXINP, and MPN. Selection details are given in Figure 22-18 on Page 22-42 .
0	[31:16], [11:0]	r	Reserved Read as 0; should be written with 0.

Controller Area Network (MultiCAN) Controller

The Interrupt Trigger Register ITR is used to trigger interrupt requests on each interrupt output line by software.

MITR

Module Interrupt Trigger Register (1CC_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
IT	[15:0]	w	Interrupt Trigger Writing a 1 to IT[n] (n = 0-15) generates an interrupt request on interrupt output line INT_O[n]. Writing a 0 to IT[n] has no effect. Bit field IT is always read as 0. Multiple interrupt requests can be generated with a single write operation to MITR by writing a 1 to several bit positions of IT.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

Controller Area Network (MultiCAN) Controller

List Pointer and List Register

Each of the four CAN nodes has a list that determines the allocated message objects. Additionally, a list of all unallocated objects is available. Furthermore, general purpose lists are available which are not associated to a CAN node. The List Registers are assigned in the following way:

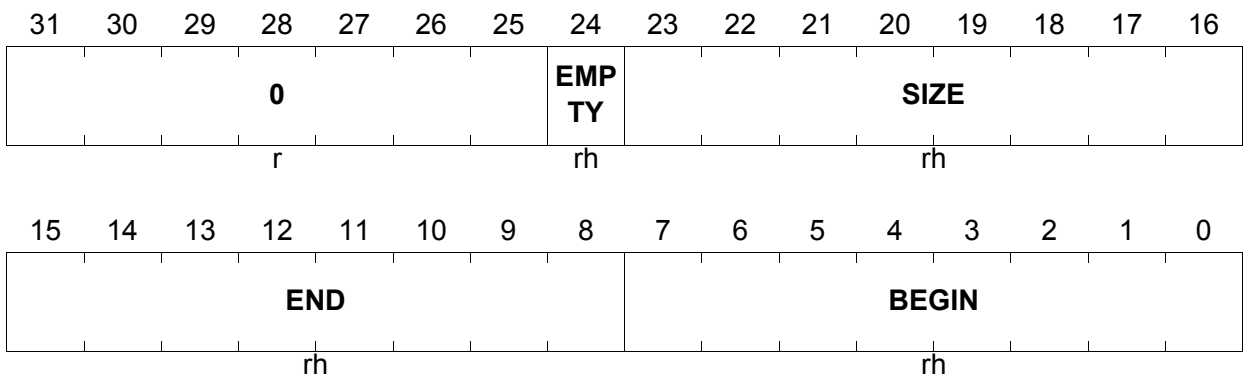
- LIST0 provides the list of all unallocated objects
- LIST1 provides the list for CAN node 0
- LIST2 provides the list for CAN node 1
- LIST3 provides the list for CAN node 2
- LIST4 provides the list for CAN node 3
- LIST[7:5] are not associated to a CAN node (free lists)

Each list is defined by a List Register that contains the first and the last element (message object number) and the size (number of message objects assigned to the list) of the list.

LIST0

List Register 0 (100_H) Reset Value: 007F 7F00_H

LISTk (k = 1-7)
List Register k (100_H+k*4_H) Reset Value: 0100 0000_H



Field	Bits	Type	Description
BEGIN	[7:0]	rh	List Begin BEGIN indicates the number of the first message object in list k.
END	[15:8]	rh	List End END indicates the number of the last message object in list k.

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
SIZE	[23:16]	rh	List Size SIZE indicates the number of elements in the list k. SIZE = number of list elements - 1
EMPTY	24	rh	List Empty Indication 0 _B At least one message object is allocated to list k. 1 _B No message object is allocated to the list k. List k is empty.
0	[31:25]	r	Reserved Read as 0.

Controller Area Network (MultiCAN) Controller

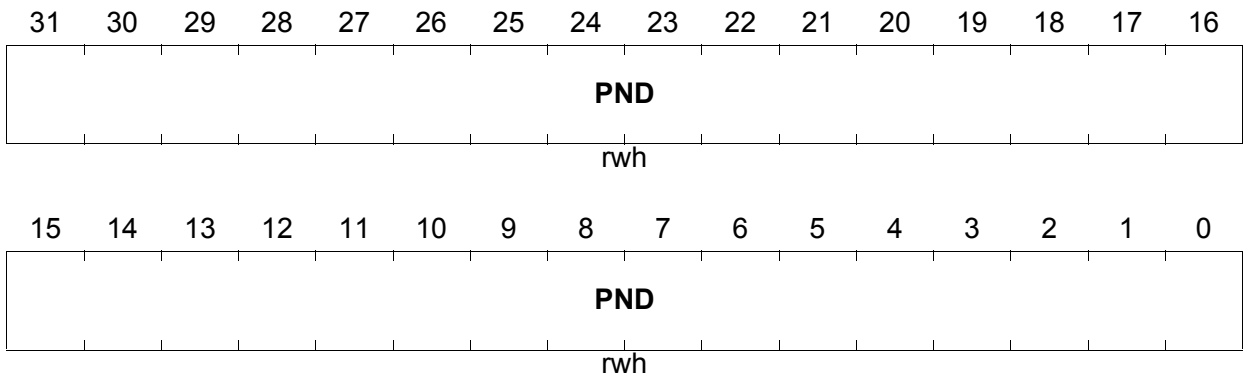
Message Notifications

When a message object n generates an interrupt request upon the transmission or reception of a message, then the request is routed to the interrupt output line selected by the bit fields MOIPRn.TXIPND or MOIPRn.RXIPND of the message object n . As there are more message objects than interrupt output lines, an interrupt routine typically processes requests from more than one message object. Therefore, a priority selection mechanism is implemented in the MultiCAN module to select the highest priority object within a collection of message objects.

The Message Pending Register MSPNDk contains the pending interrupt notification of list k .

MSPNDk (k = 0-7)

Message Pending Register k **(120_H+k*4_H)** **Reset Value: 0000 0000_H**



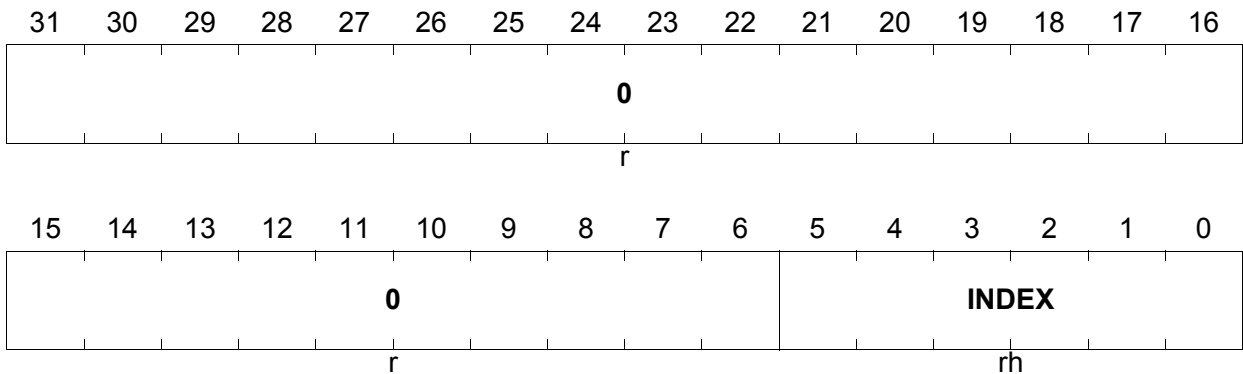
Field	Bits	Type	Description
PND	[31:0]	rwh	<p>Message Pending</p> <p>When a message interrupt occurs, the message object sets a bit in one of the MSGPND register, where the bit position is given by the MPN[4:0] field of the IPR register of the message object. The register selection n is given by the higher bits of MPN.</p> <p>The register bits can be cleared by software (write 0). Writing a 1 has no effect.</p>

Controller Area Network (MultiCAN) Controller

Each Message Pending Register has a Message Index Register MSIDk associated with it. The Message Index Register MSIDk shows the active (set) pending bit with lowest bit position within groups of pending bits.

MSIDk (k = 0-7)

Message Index Register k **(140_H+k*4_H)** **Reset Value: 0000 0020_H**



Field	Bits	Type	Description
INDEX	[5:0]	rh	<p>Message Pending Index</p> <p>The value of INDEX is given by the bit position i of the pending bit of MSPNDk with the following properties:</p> <ol style="list-style-type: none"> 1. MSPNDk[i] & IM[i] = 1 2. i = 0 or MSPNDk[i-1:0] & IM[i-1:0] = 0 <p>If no bit of MSPNDk satisfies these conditions then INDEX reads 100000_B.</p> <p>Thus INDEX shows the position of the first pending bit of MSPNDk, in which only those bits of MSPNDk that are selected in the Message Index Mask Register are taken into account.</p>
0	[31:6]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Controller Area Network (MultiCAN) Controller

22.4.3 CAN Node Registers

The CAN node registers are built in for each CAN node of the MultiCAN module. They contain information that is directly related to the operation of the CAN nodes and are shared among the nodes.

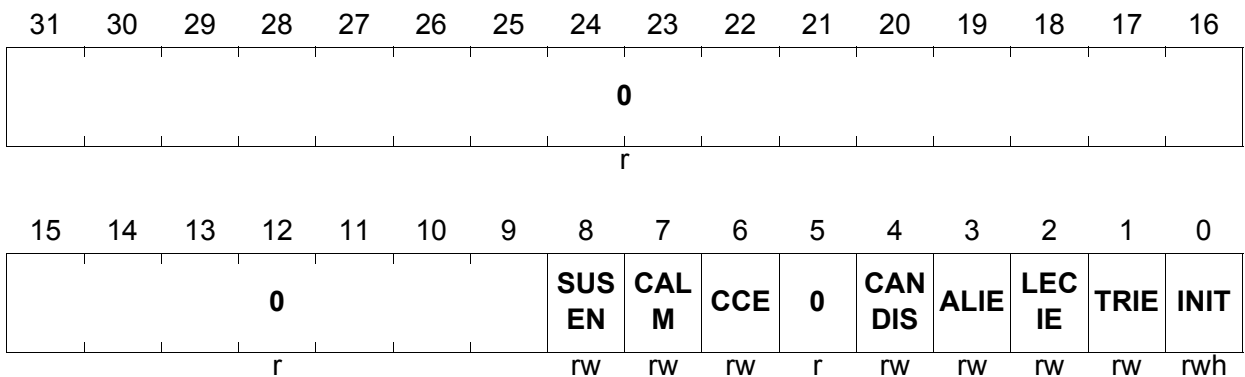
The Node Control Register contains basic settings that determine the operation of the CAN node.

NCRx (x = 0-3)

Node x Control Register

(200_H+x*100_H)

Reset Value: 0000 0001_H



Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
INIT	0	rwh	<p>Node Initialization</p> <p>0_B Clearing bit INIT enables the participation of the node in the CAN traffic. If the CAN node is in the bus-off state, the ongoing bus-off recovery (which does not depend on the INIT bit) is continued. With the end of the bus-off recovery sequence, the CAN node is allowed to take part in the CAN traffic if INIT has been cleared by software. If the CAN node is not in the bus-off state, a sequence of 11 consecutive recessive bits must be detected before the node is allowed to take part in the CAN traffic.</p> <p>1_B Setting this bit terminates the participation of this node in the CAN traffic. Any ongoing frame transfer is cancelled and the transmit line goes recessive. If the CAN node is in the bus-off state, then the running bus-off recovery sequence is continued. If the INIT bit is still set after the successful completion of the bus-off recovery sequence, i.e. after detecting 128 sequences of 11 consecutive recessive bits (11 × 1), then the CAN node leaves the bus-off state but remains inactive as long as INIT remains set.</p> <p>Bit INIT is automatically set when the CAN node enters the bus-off state.</p>
TRIE	1	rw	<p>Transfer Interrupt Enable</p> <p>TRIE enables the transfer interrupt of CAN node x. This interrupt is generated after the successful reception or transmission of a CAN frame in node x.</p> <p>0_B Transfer interrupt is disabled.</p> <p>1_B Transfer interrupt is enabled.</p> <p>Bit field NIPRx.TRINP selects the interrupt output line which becomes activated at this type of interrupt.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
LECIE	2	rw	<p>Last Error Code Interrupt Enable</p> <p>LECIE enables the last error code interrupt of CAN node x. This interrupt is generated with each update of bit field NSRx.LEC with LEC > 0 (CAN protocol error).</p> <p>0_B Last error code interrupt is disabled. 1_B Last error code interrupt is enabled.</p> <p>Bit field NIPRx.LECINP selects the interrupt output line which becomes activated at this type of interrupt.</p>
ALIE	3	rw	<p>Alert Interrupt Enable</p> <p>ALIE enables the alert interrupt of CAN node x. This interrupt is generated by any one of the following events:</p> <ul style="list-style-type: none"> • A change of bit NSRx.BoFF • A change of bit NSRx.EWRN • A List Length Error, which also sets bit NSRx.LLE • A List Object Error, which also sets bit NSRx.LOE • A Bit INIT is set by hardware <p>0_B Alert interrupt is disabled. 1_B Alert interrupt is enabled.</p> <p>Bit field NIPRx.ALINP selects the interrupt output line which becomes activated at this type of interrupt.</p>
CANDIS	4	rw	<p>CAN Disable</p> <p>Setting this bit disables the CAN node. The CAN node first waits until it is bus-idle or bus-off. Then bit INIT is automatically set, and an alert interrupt is generated if bit ALIE is set.</p>
CCE	6	rw	<p>Configuration Change Enable</p> <p>0_B The Bit Timing Register, the Port Control Register, and the Error Counter Register can only be read. All attempts to modify them are ignored.</p> <p>1_B The Bit Timing Register, the Port Control Register, and the Error Counter Register may be read and written.</p>
CALM	7	rw	<p>CAN Analyze Mode</p> <p>If this bit is set, then the CAN node operates in Analyze Mode. This means that messages may be received, but not transmitted. No acknowledge is sent on the CAN bus upon frame reception. Active-error flags are sent recessive instead of dominant. The transmit line is continuously held at recessive (1) level.</p> <p>Bit CALM can be written only while bit INIT is set.</p>

Controller Area Network (MultiCAN) Controller

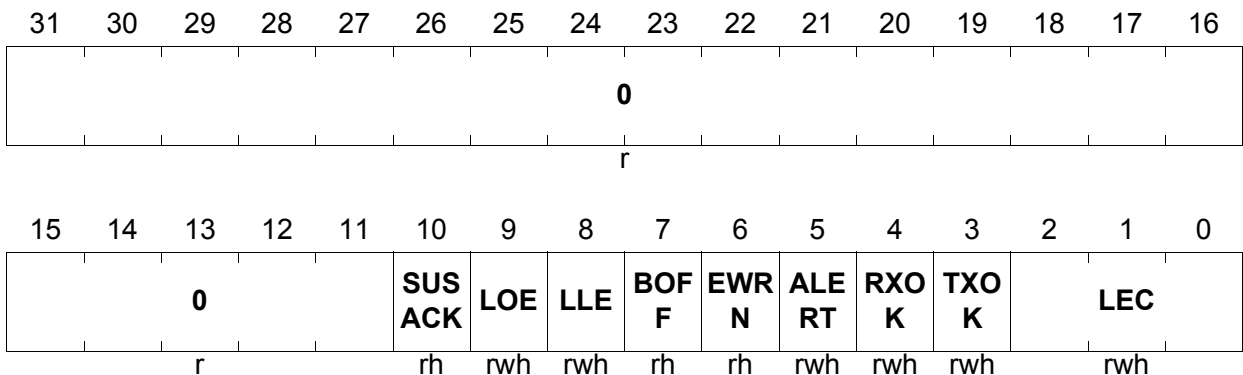
Field	Bits	Type	Description
SUSEN	8	rw	<p>Suspend Enable</p> <p>This bit makes it possible to set the CAN node into Suspend Mode via OCDS (on chip debug support):</p> <p>0_B An OCDS suspend trigger is ignored by the CAN node.</p> <p>1_B An OCDS suspend trigger disables the CAN node: As soon as the CAN node becomes bus-idle or bus-off, bit INIT is internally forced to 1 to disable the CAN node. The actual value of bit INIT remains unchanged.</p> <p>Bit SUSEN is cleared via OCDS Reset.</p>
0	[31:9], 5	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Controller Area Network (MultiCAN) Controller

The Node Status Register NSRx reports errors as well as successfully transferred CAN frames.

NSRx (x = 0-3)

Node x Status Register (204_H+x*100_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
LEC	[2:0]	rwh	<p>Last Error Code</p> <p>This bit field indicates the type of the last (most recent) CAN error. The encoding of this bit field is described in Table 22-7.</p>
TXOK	3	rwh	<p>Message Transmitted Successfully</p> <p>0_B No successful transmission since the last (most recent) clearance of the flag.</p> <p>1_B A message has been transmitted successfully (error-free and acknowledged by at least another node).</p> <p>TXOK must be cleared by software (write 0). Writing 1 has no effect.</p>
RXOK	4	rwh	<p>Message Received Successfully</p> <p>0_B No successful reception since the last (most recent) clearance of the flag.</p> <p>1_B A message has been received successfully.</p> <p>RXOK must be cleared by software (write 0). Writing 1 has no effect.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
ALERT	5	rwh	<p>Alert Warning</p> <p>The ALERT bit is set upon the occurrence of one of the following events (the same events which also trigger an alert interrupt if ALIE is set):</p> <ul style="list-style-type: none"> • A change of bit NSRx.BOFF • A change of bit NSRx.EWRN • A List Length Error, which also sets bit NSRx.LLE • A List Object Error, which also sets bit NSRx.LOE • Bit INIT has been set by hardware <p>ALERT must be cleared by software (write 0). Writing 1 has no effect.</p>
EWRN	6	rh	<p>Error Warning Status</p> <p>0_B No warning limit exceeded. 1_B One of the error counters REC or TEC reached the warning limit EWRNLVL.</p>
BOFF	7	rh	<p>Bus-off Status</p> <p>0_B CAN controller is not in the bus-off state. 1_B CAN controller is in the bus-off state.</p>
LLE	8	rwh	<p>List Length Error</p> <p>0_B No List Length Error since the last (most recent) clearance of the flag. 1_B A List Length Error has been detected during message acceptance filtering. The number of elements in the list that belongs to this CAN node differs from the list SIZE given in the list termination pointer.</p> <p>LLE must be cleared by software (write 0). Writing 1 has no effect.</p>
LOE	9	rwh	<p>List Object Error</p> <p>0_B No List Object Error since the last (most recent) clearance of the flag. 1_B A List Object Error has been detected during message acceptance filtering. A message object with wrong LIST index entry in the Message Object Control Register has been detected.</p> <p>LOE must be cleared by software (write 0). Writing 1 has no effect.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
SUSACK	10	rh	Suspend Acknowledge 0_B The CAN node is not in Suspend Mode or a suspend request is pending, but the CAN node has not yet reached bus-idle or bus-off. 1_B The CAN node is in Suspend Mode: The CAN node is inactive (bit NCR.INIT internally forced to 1) due to an OCDS suspend request.
0	[31:11]	r	Reserved Read as 0; should be written with 0.

Encoding of the LEC Bit Field

Table 22-7 Encoding of the LEC Bit Field

LEC Value	Signification
000_B	No Error: No error was detected for the last (most recent) message on the CAN bus.
001_B	Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.
010_B	Form Error: A fixed format part of a received frame has the wrong format.
011_B	Ack Error: The transmitted message was not acknowledged by another node.
100_B	Bit1 Error: During a message transmission, the CAN node tried to send a recessive level (1) outside the arbitration field and the acknowledge slot, but the monitored bus value was dominant.
101_B	Bit0 Error: Two different conditions are signaled by this code: <ol style="list-style-type: none"> 1. During transmission of a message (or acknowledge bit, active-error flag, overload flag), the CAN node tried to send a dominant level (0), but the monitored bus value was recessive. 2. During bus-off recovery, this code is set each time a sequence of 11 recessive bits has been monitored. The CPU may use this code as indication that the bus is not continuously disturbed.

Controller Area Network (MultiCAN) Controller

Table 22-7 Encoding of the LEC Bit Field (cont'd)

LEC Value	Signification
110 _B	CRC Error: The CRC checksum of the received message was incorrect.
111 _B	CPU write to LEC: Whenever the CPU writes the value 111 _B to LEC, it takes the value 111 _B . Whenever the CPU writes another value to LEC, the written LEC value is ignored.

Controller Area Network (MultiCAN) Controller

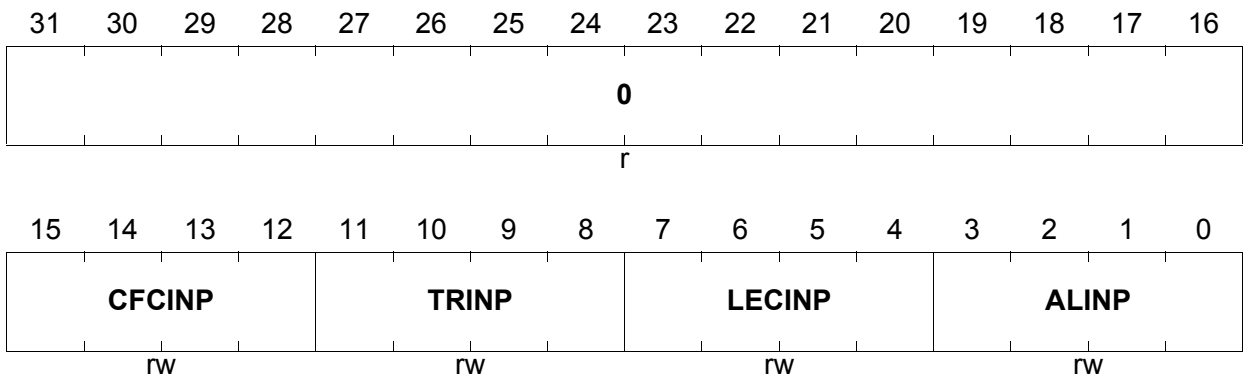
The four interrupt pointers in the Node Interrupt Pointer Register NIPRx select one of the sixteen interrupt outputs individually for each type of CAN node interrupt. See also [Page 22-27](#) for more CAN node interrupt details.

NIPRx (x = 0-3)

Node x Interrupt Pointer Register

(208_H+x*100_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
ALINP	[3:0]	rw	<p>Alert Interrupt Node Pointer</p> <p>ALINP selects the interrupt output line INT_Om (m = 0-15) for an alert interrupt of CAN Node x.</p> <p>0000_B Interrupt output line INT_O0 is selected.</p> <p>0001_B Interrupt output line INT_O1 is selected.</p> <p>..._B ...</p> <p>1110_B Interrupt output line INT_O14 is selected.</p> <p>1111_B Interrupt output line INT_O15 is selected.</p>
LECINP	[7:4]	rw	<p>Last Error Code Interrupt Node Pointer</p> <p>LECINP selects the interrupt output line INT_Om (m = 0-15) for a LEC interrupt of CAN Node x.</p> <p>0000_B Interrupt output line INT_O0 is selected.</p> <p>0001_B Interrupt output line INT_O1 is selected.</p> <p>..._B ...</p> <p>1110_B Interrupt output line INT_O14 is selected.</p> <p>1111_B Interrupt output line INT_O15 is selected.</p>

Controller Area Network (MultiCAN) Controller

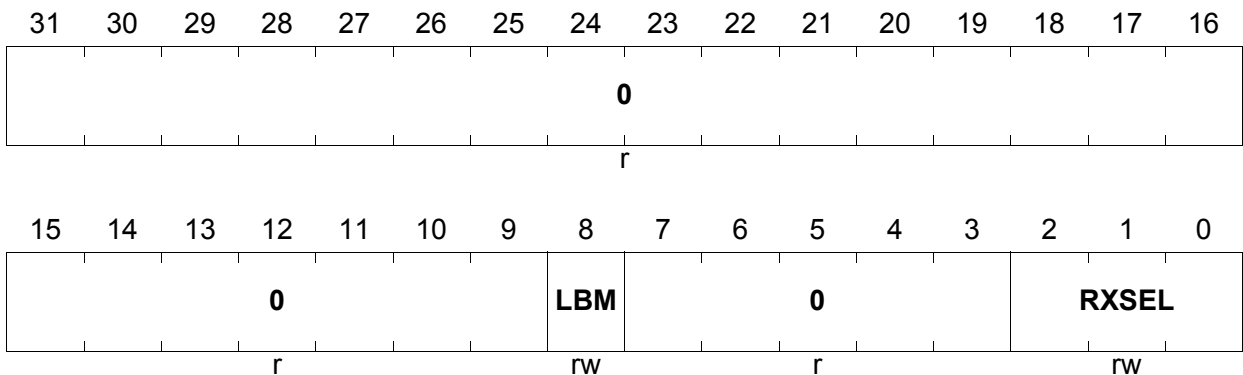
Field	Bits	Type	Description
TRINP	[11:8]	rw	<p>Transfer OK Interrupt Node Pointer TRINP selects the interrupt output line INT_Om (m = 0-15) for a transfer OK interrupt of CAN Node x.</p> <p>0000_B Interrupt output line INT_O0 is selected. 0001_B Interrupt output line INT_O1 is selected. ..._B ... 1110_B Interrupt output line INT_O14 is selected. 1111_B Interrupt output line INT_O15 is selected.</p>
CFCINP	[15:12]	rw	<p>Frame Counter Interrupt Node Pointer CFCINP selects the interrupt output line INT_Om (m = 0-15) for a frame counter overflow interrupt of CAN Node x.</p> <p>0000_B Interrupt output line INT_O0 is selected. 0001_B Interrupt output line INT_O1 is selected. ..._B ... 1110_B Interrupt output line INT_O14 is selected. 1111_B Interrupt output line INT_O15 is selected.</p>
0	[31:16]	r	<p>Reserved Read as 0; should be written with 0.</p>

Controller Area Network (MultiCAN) Controller

The Node Port Control Register NPCR_x configures the CAN bus transmit/receive ports. NPCR_x can be written only if bit NCR_x.CCE is set.

NPCR_x (x = 0-3)

Node x Port Control Register ($20C_H + x \cdot 100_H$) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
RXSEL	[2:0]	rw	<p>Receive Select</p> <p>RXSEL selects one out of 8 possible receive inputs of CAN node x. The CAN receive signal is performed only through the selected input.</p> <p><i>Note: In TC1796, only specific combinations of RXSEL are available (see also “Receive Input Selection” on Page 22-207).</i></p>
LBM	8	rw	<p>Loop-Back Mode</p> <p>0_B Loop-Back Mode is disabled.</p> <p>1_B Loop-Back Mode is enabled. This node is connected to an internal (virtual) loop-back CAN bus. All CAN nodes which are in Loop-Back Mode are connected to this virtual CAN bus so that they can communicate with each other internally. The external transmit line is forced recessive in Loop-Back Mode.</p>
0	[7:3], [31:9]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Controller Area Network (MultiCAN) Controller

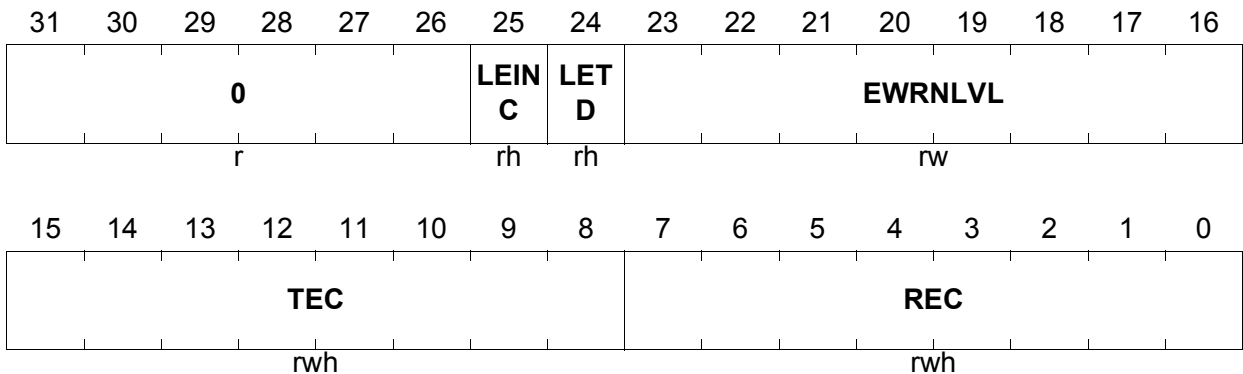
Field	Bits	Type	Description
DIV8	15	rw	<p>Divide Prescaler Clock by 8</p> <p>0_B A time quantum lasts (BRP+1) clock cycles. 1_B A time quantum lasts 8 × (BRP+1) clock cycles.</p>
FTX	16	rw	<p>Fast Transmit (TTC only)</p> <p>When a message is requested for transmission on the CAN bus, then the SOF is sent with the beginning of a new bit time.</p> <p>If the CAN bus is in the idle state and bit FTX is set (FTX = 1), a new bit time is started immediately with the transmit trigger of a new message. This eliminates the variable delay between the transmit trigger of a message and the actual SOF signal on the transmit output. Such a variable delay occurs when transmit triggers occur at different positions within a CAN bit time.</p>
0	[31:17]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Controller Area Network (MultiCAN) Controller

The Node Error Counter Register NECNTRx contains the CAN receive and transmit error counter as well as some additional bits to ease error analysis. NECNTRx can be written only if bit NCRx.CCE is set.

NECNTx (x = 0-3)

Node x Error Counter Register $(214_H + x * 100_H)$ **Reset Value: 0060 0000_H**



Field	Bits	Type	Description
REC	[7:0]	rwh	Receive Error Counter Bit field REC contains the value of the receive error counter of CAN node x.
TEC	[15:8]	rwh	Transmit Error Counter Bit field TEC contains the value of the transmit error counter of CAN node x.
EWRNLVL	[23:16]	rw	Error Warning Level Bit field EWRNLVL determines the threshold value (warning level, default 96) to be reached in order to set the corresponding error warning bit EWRN.
LETD	24	rh	Last Error Transfer Direction 0 _B The last error occurred while the CAN node x was receiver (REC has been incremented). 1 _B The last error occurred while the CAN node x was transmitter (TEC has been incremented).
LEINC	25	rh	Last Error Increment 0 _B The last error led to an error counter increment of 1. 1 _B The last error led to an error counter increment of 8.

Controller Area Network (MultiCAN) Controller

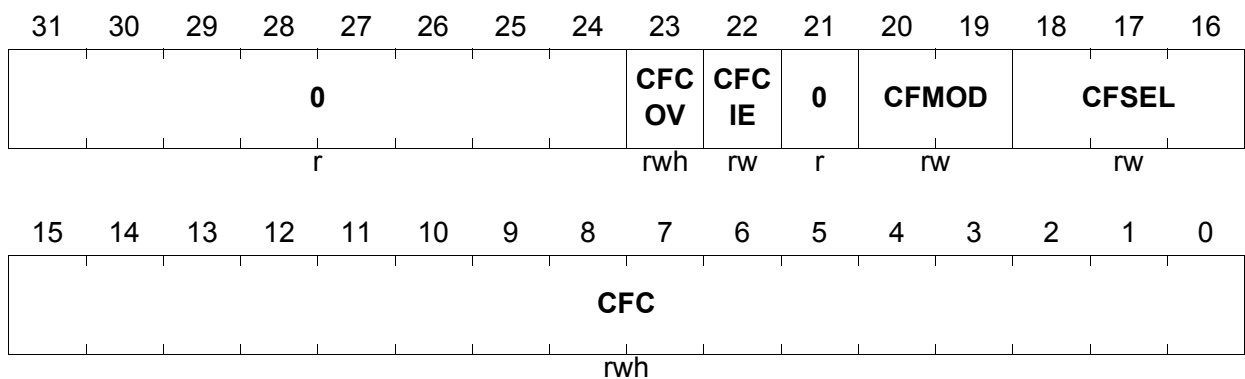
Field	Bits	Type	Description
0	[31:26]	r	Reserved Read as 0; should be written with 0.

The Node Frame Counter Register NFCRx contains the actual value of the frame counter as well as control and status bits of the frame counter.

NFCRx (x = 0-3)

Node x Frame Counter Register (218_H+x*100_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CFC	[15:0]	rwh	<p>CAN Frame Counter</p> <p>In Frame Count Mode (CFMOD = 00_B), this bit field contains the frame count value.</p> <p>In Time Stamp Mode (CFMOD = 01_B), this bit field contains the captured bit time count value, captured with the start of a new frame.</p> <p>In all Bit Timing Analysis Modes (CFMOD = 10_B), CFC always displays the number of f_{CLC} clock cycles (measurement result) minus 1. Example: a CFC value of 34 in measurement mode CFSEL = 000_B means that 35 f_{CLC} clock cycles have been elapsed between the most recent two dominant edges on the receive input.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
CFSEL	[18:16]	rw	<p>CAN Frame Count Selection This bit field selects the function of the frame counter for the chosen frame count mode.</p> <p>Frame Count Mode</p> <p>Bit 0 If Bit 0 of CFSEL is set, then CFC is incremented each time a foreign frame (i.e. a frame not matching to a message object) has been received on the CAN bus.</p> <p>Bit 1 If Bit 1 of CFSEL is set, then CFC is incremented each time a frame matching to a message object has been received on the CAN bus.</p> <p>Bit 2 If Bit 2 of CFSEL is set, then CFC is incremented each time a frame has been transmitted successfully by the node.</p> <p>Time Stamp Mode 000_B The frame counter is incremented (internally) at the beginning of a new bit time. The value is sampled during the SOF bit of a new frame. The sampled value is visible in the CFC field.</p> <p>Bit Timing Mode The available bit timing measurement modes are shown in Table 22-8. This bit field selects the function of the frame counter for the chosen frame count mode.</p>
CFMOD	[20:19]	rw	<p>CAN Frame Counter Mode This bit field determines the operation mode of the frame counter.</p> <p>00_B Frame Count Mode: The frame counter is incremented upon the reception and transmission of frames.</p> <p>01_B Time Stamp Mode: The frame counter is used to count bit times.</p> <p>10_B Bit Timing Mode: The frame counter is used for analysis of the bit timing.</p>
CFCIE	22	rw	<p>CAN Frame Count Interrupt Enable CFCIE enables the CAN frame counter overflow interrupt of CAN node x.</p> <p>0_B CAN frame counter overflow interrupt is disabled.</p> <p>1_B CAN frame counter overflow interrupt is enabled.</p> <p>Bit field NIPRx.CFCINP selects the interrupt output line that is activated at this type of interrupt.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
CFCOV	23	rwh	<p>CAN Frame Counter Overflow Flag Flag CFCOV is set upon a frame counter overflow (transition from FFFF_H to 0000_H). In bit timing analysis mode, CFCOV is set upon an update of CFC. An interrupt request is generated if CFCIE = 1.</p> <p>0_B No overflow has occurred since the last (most recent) clearance of the flag.</p> <p>1_B An overflow has occurred since the last (most recent) clearance of the flag.</p> <p>CFCOV must be cleared by software.</p>
0	21, [31:24]	r	<p>Reserved Read as 0; should be written with 0.</p>

Bit Timing Analysis Modes

Table 22-8 Bit Timing Analysis Modes (CFMOD = 10)

CFSEL	Measurement
000 _B	Whenever a dominant edge (transition from 1 to 0) is monitored on the receive input, the time (measured in clock cycles) between this edge and the most recent dominant edge is stored in CFC.
001 _B	Whenever a recessive edge (transition from 0 to 1) is monitored on the receive input, the time (measured in clock cycles) between this edge and the most recent dominant edge is stored in CFC.
010 _B	Whenever a dominant edge is received as a result of a transmitted dominant edge, the time (clock cycles) between both edges is stored in CFC.
011 _B	Whenever a recessive edge is received as a result of a transmitted recessive edge, the time (clock cycles) between both edges is stored in CFC.
100 _B	Whenever a dominant edge that qualifies for synchronization is monitored on the receive input, the time (measured in clock cycles) between this edge and the most recent sample point is stored in CFC.
101 _B	<p>With each sample point, the time (measured in clock cycles) between the start of the new bit time and the start of the previous bit time is stored in CFC[11:0]. Additional information is written to CFC[15:12] at each sample point:</p> <p>CFC[15]: Transmit value of actual bit time CFC[14]: Receive sample value of actual bit time CFC[13:12]: CAN bus information (see Table 22-9)</p>
111 _B	Reserved; do not use this combination.

Controller Area Network (MultiCAN) Controller

Table 22-9 CAN Bus State Information

CFC[13:12]	CAN Bus State
00 _B	<p>NoBit The CAN bus is idle, performs bit (de-) stuffing or is in one of the following frame segments: SOF, SRR, CRC, delimiters, first 6 EOF bits, IFS</p>
01 _B	<p>NewBit This code represents the first bit of a new frame segment. The current bit is the first bit in one of the following frame segments: Bit 10 (MSB) of standard ID (transmit only), RTR, reserved bits, IDE, DLC(MSB), bit 7 (MSB) in each data byte and the first bit of the ID extension</p>
10 _B	<p>Bit This code represents a bit inside a frame segment with a length of more than one bit (not the first bit of those frame segments that is indicated by NewBit). The current bit is processed within one of the following frame segments: ID bits (except first bit of standard ID for transmission and first bit of ID extension), DLC (3 LSB) and bits 6-0 in each data byte.</p>
11 _B	<p>Done The current bit is in one of the following frame segments: Acknowledge slot, last bit of EOF, active/passive-error frame, overload frame. Two or more directly consecutive Done codes signal an Error Frame.</p>

Controller Area Network (MultiCAN) Controller

22.4.4 Message Object Registers

The Message Object Control Register MOCTR_n and the Message Object Status Register MOSTAT_n are located at the same address offset within a message object address block (offset address 1C_H). The MOCTR_n is a write-only register that makes it possible to set/clear CAN transfer related control bits through software.

When reading from the MOCTR_n address after reset, the reset value of register MOSTAT_n will always be delivered (see also [Table 22-11](#)).

MOCTR0

Message Object 0 Control Register (061C_H) **Reset Value: 0100 0000H**

MOCTR127

Message Object 127 Control Register (15FC_H) **Reset Value: 7F7E 0000H**

MOCTR_n (n = 1-126)

Message Object n Control Register

(0061C_H+n*20_H)

Reset Value: ((n+1)*01000000_H)+((n-1)*00010000_H)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				SET DIR	SET TX EN1	SET TX EN0	SET TX RQ	SET RX EN	SET RT SEL	SET MSG VAL	SET MSG LST	SET NEW DAT	SET RX UPD	SET TX PND	SET RX PND
W				W	W	W	W	W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				RES DIR	RES TX EN1	RES TX EN0	RES TX RQ	RES RX EN	RES RT SEL	RES MSG VAL	RES MSG LST	RES NEW DAT	RES RX UPD	RES TX PND	RES RX PND
W				W	W	W	W	W	W	W	W	W	W	W	W

Field	Bits	Type	Description
RESRXPND, SETRXPND	0, 16	w	Reset/Set Receive Pending These bits control the clear/set condition for RXPND (see Table 22-20).
RESTXPND, SETTXPND	1, 17	w	Reset/Set Transmit Pending These bits control the clear/set condition for TXPND (see Table 22-20).
RESRXUPD, SETRXUPD	2, 18	w	Reset/Set Receive Updating These bits control the clear/set condition for RXUPD (see Table 22-20).

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
RESNEWDAT, SETNEWDAT	3, 19	w	Reset/Set New Data These bits control the clear/set condition for NEWDAT (see Table 22-20).
RESMSGLST, SETMSGLST	4, 20	w	Reset/Set Message Lost These bits control the clear/set condition for MSGLST (see Table 22-20).
RESMSGVAL, SETMSGVAL	5, 21	w	Reset/Set Message Valid These bits control the clear/set condition for MSGVAL (see Table 22-20).
RESRTSEL, SETRTSEL	6, 22	w	Reset/Set Receive/Transmit Selected These bits control the clear/set condition for RTSEL (see Table 22-20).
RESRXEN, SETRXEN	7, 23	w	Reset/Set Receive Enable These bits control the clear/set condition for RXEN (see Table 22-20).
RESTXRQ, SETTXRQ	8, 24	w	Reset/Set Transmit Request These bits control the clear/set condition for TXRQ (see Table 22-20).
RESTXEN0, SETTXEN0	9, 25	w	Reset/Set Transmit Enable 0 These bits control the clear/set condition for TXEN0 (see Table 22-20).
RESTXEN1, SETTXEN1	10, 26	w	Reset/Set Transmit Enable 1 These bits control the clear/set condition for TXEN1 (see Table 22-20).
RESDIR, SETDIR	11, 27	w	Reset/Set Message Direction These bits control the clear/set condition for DIR (see Table 22-20).
0	[15:12], [31:28]	w	Reserved Should be written with 0.

Controller Area Network (MultiCAN) Controller

Table 22-10 Clear/Set Conditions for Bits in Register MOCTRn

RESy Bit¹⁾	SETy Bit	Action on Write
Write 0	Write 0	Leave element unchanged
	No write	
No write	Write 0	
Write 1	Write 1	
Write 1	Write 0	Clear element
	No write	
Write 0	Write 1	Set element
No write		

1) The parameter “y” stands for the second part of the bit name (“RXPND“, “TXPND“, ... up to “DIR“).

Controller Area Network (MultiCAN) Controller

The MOSTATn is a read-only register that indicates message object list status information such as the number of the current message object predecessor and successor message object, as well as the list number to which the message object is assigned.

MOSTAT0

Message Object 0 Status Register (061C_H)

Reset Value: 0100 0000_H

MOSTAT127

Message Object 127 Status Register (15FC_H)

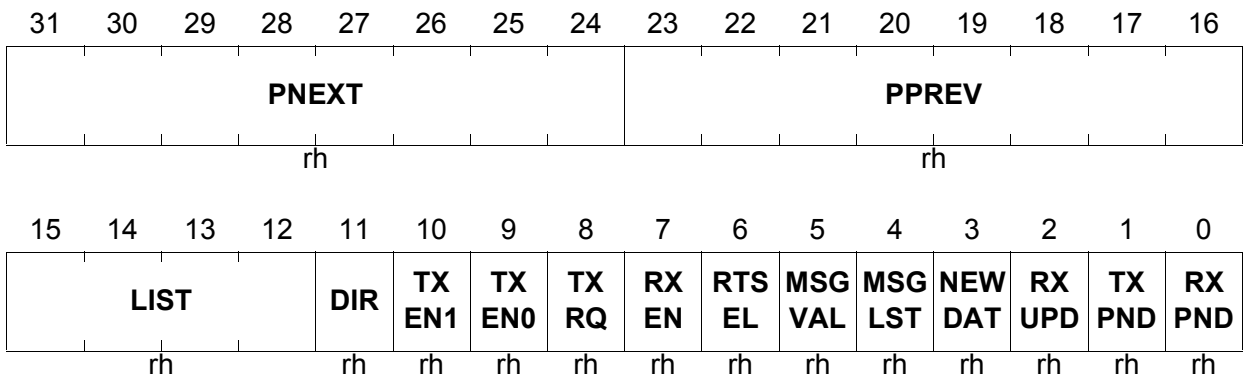
Reset Value: 7F7E 0000_H

MOSTATn (n = 1-126)

Message Object n Status Register

(061C_H+n*20_H)

Reset Value: ((n+1)*01000000_H)+((n-1)*00010000_H)



Field	Bits	Type	Description
RXPND	0	rh	<p>Receive Pending</p> <p>0_B No CAN message has been received. 1_B A CAN message has been received by the message object n, either directly or via gateway copy action.</p> <p>RXPND is not cleared by hardware but must be cleared by software.</p>
TXPND	1	rh	<p>Transmit Pending</p> <p>0_B No CAN message has been transmitted. 1_B A CAN message from message object n has been transmitted successfully over the CAN bus.</p> <p>TXPND is cleared by hardware but must be cleared by software.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
RXUPD	2	rh	<p>Receive Updating</p> <p>0_B No receive update ongoing. 1_B Message identifier, DLC, and data of the message object are currently updated.</p>
NEWDAT	3	rh	<p>New Data</p> <p>0_B No update of the message object n since the last clearance of the flag. 1_B Message object n has been updated. NEWDAT is set by hardware after a received CAN frame has been stored in message object n. NEWDAT is cleared by hardware when a CAN transmission of message object n has been started. NEWDAT should be set by software after the new transmit data has been stored in message object n to prevent the automatic clearing of TXRQ at the end of an ongoing transmission.</p>
MSGLST	4	rh	<p>Message Lost</p> <p>0_B No CAN message is lost. 1_B A CAN message is lost because NEWDAT has become set again when it has already been set.</p>
MSGVAL	5	rh	<p>Message Valid</p> <p>0_B Message object n is not valid. 1_B Message object n is valid. Only a valid message object takes part in CAN transfers.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
RTSEL	6	rh	<p>Receive/Transmit Selected</p> <p>0_B Message object n is not selected for receive or transmit operation.</p> <p>1_B Message object n is selected for receive or transmit operation.</p> <p>Frame Reception: RTSEL is set by hardware when message object n has been identified for storage of a CAN frame that is currently received. Before a received frame becomes finally stored in message object n, a check is performed to determine if RTSEL is set. Thus the CPU can suppress a scheduled frame delivery to this message object n by clearing RTSEL by software.</p> <p>Frame Transmission: RTSEL is set by hardware when message object n has been identified to be transmitted next. A check is performed to determine if RTSEL is still set before message object n is actually set up for transmission and bit NEWDAT is cleared. It is also checked that RTSEL is still set before it message object n is verified due to the successful transmission of a frame. RTSEL needs to be checked only when the context of message object n changes, and a conflict with an ongoing frame transfer shall be avoided. In all other cases, RTSEL can be ignored. RTSEL has no impact on message acceptance filtering. RTSEL is not cleared by hardware.</p>
RXEN	7	rh	<p>Receive Enable</p> <p>0_B Message object n is not enabled for frame reception.</p> <p>1_B Message object n is enabled for frame reception.</p> <p>RXEN is evaluated for receive acceptance filtering only.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
TXRQ	8	rh	<p>Transmit Request</p> <p>0_B No transmission of message object n is requested.</p> <p>1_B Transmission of message object n on the CAN bus is requested.</p> <p>The transmit request becomes valid only if TXRQ, TXEN0, TXEN1 and MSGVAL are set. TXRQ is set by hardware if a matching Remote Frame has been received correctly. TXRQ is cleared by hardware if message object n has been transmitted successfully and NEWDAT is not set again by software.</p>
TXEN0	9	rh	<p>Transmit Enable 0</p> <p>0_B Message object n is not enabled for frame transmission.</p> <p>1_B Message object n is enabled for frame transmission.</p> <p>Message object n can be transmitted only if both bits, TXEN0 and TXEN1, are set.</p> <p>The user may clear TXEN0 in order to inhibit the transmission of a message that is currently updated, or to disable automatic response of Remote Frames.</p>
TXEN1	10	rh	<p>Transmit Enable 1</p> <p>0_B Message object n is not enabled for frame transmission.</p> <p>1_B Message object n is enabled for frame transmission.</p> <p>Message object n can be transmitted only if both bits, TXEN0 and TXEN1, are set.</p> <p>TXEN1 is used by the MultiCAN module for selecting the active message object in the Transmit FIFOs.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
DIR	11	rh	<p>Message Direction</p> <p>0_B Receive Object selected: With TXRQ = 1, a Remote Frame with the identifier of message object n is scheduled for transmission. On reception of a Data Frame with matching identifier, the message is stored in message object n.</p> <p>1_B Transmit Object selected: If TXRQ = 1, message object n is scheduled for transmission of a Data Frame. On reception of a Remote Frame with matching identifier, bit TXRQ is set.</p>
LIST	[15:12]	rh	<p>List Allocation</p> <p>LIST indicates the number of the message list to which message object n is allocated. LIST is updated by hardware when the list allocation of the object is modified by a panel command.</p>
PPREV	[23:16]	rh	<p>Pointer to Previous Message Object</p> <p>PPREV holds the message object number of the previous message object in a message list structure.</p>
PNEXT	[31:24]	rh	<p>Pointer to Next Message Object</p> <p>PNEXT holds the message object number of the next message object in a message list structure.</p>

Table 22-11 MOSTATn Reset Values

Message Object	PNEXT	PPREV	Reset Value
0	1	0	0100 0000 _H
1	2	0	0200 0000 _H
2	3	1	0301 0000 _H
3	4	2	0402 0000 _H
...
124	125	123	7D7B 0000 _H
125	126	124	7E7C 0000 _H
126	127	125	7F7D 0000 _H
127	127	126	7F7E 0000 _H

Controller Area Network (MultiCAN) Controller

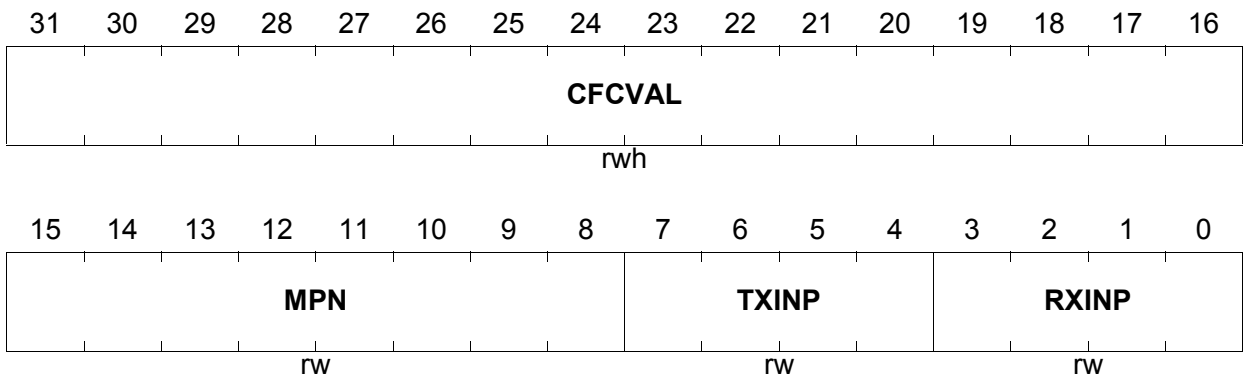
The Message Object Interrupt Pointer Register MOIPR_n holds the message interrupt pointers, the message pending number, and the frame counter value of message object n.

MOIPR_n (n = 0-127)

Message Object n Interrupt Pointer Register

(0608_H+n*20_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
RXINP	[3:0]	rw	<p>Receive Interrupt Node Pointer</p> <p>RXINP selects the interrupt output line INT_0_m (m = 0-15) for a receive interrupt event of message object n. RXINP can also be taken for message pending bit selection (see Page 22-42).</p> <p>0000_B Interrupt output line INT_00 is selected. 0001_B Interrupt output line INT_01 is selected. ..._B ... 1110_B Interrupt output line INT_014 is selected. 1111_B Interrupt output line INT_015 is selected.</p>
TXINP	[7:4]	rw	<p>Transmit Interrupt Node Pointer</p> <p>TXINP selects the interrupt output line INT_0_m (m = 0-15) for a transmit interrupt event of message object n. TXINP can also be taken for message pending bit selection (see Page 22-42).</p> <p>0000_B Interrupt output line INT_00 is selected. 0001_B Interrupt output line INT_01 is selected. ..._B ... 1110_B Interrupt output line INT_014 is selected. 1111_B Interrupt output line INT_015 is selected.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
MPN	[15:8]	rw	Message Pending Number This bit field selects the bit position of the bit in the Message Pending Register that is set upon a message object n receive/transmit interrupt.
CFCVAL	[31:16]	rwh	CAN Frame Counter Value When a message is stored in message object n or message object n has been successfully transmitted, the CAN frame counter value NFCRx.CFC is then copied to CFCVAL.

Controller Area Network (MultiCAN) Controller

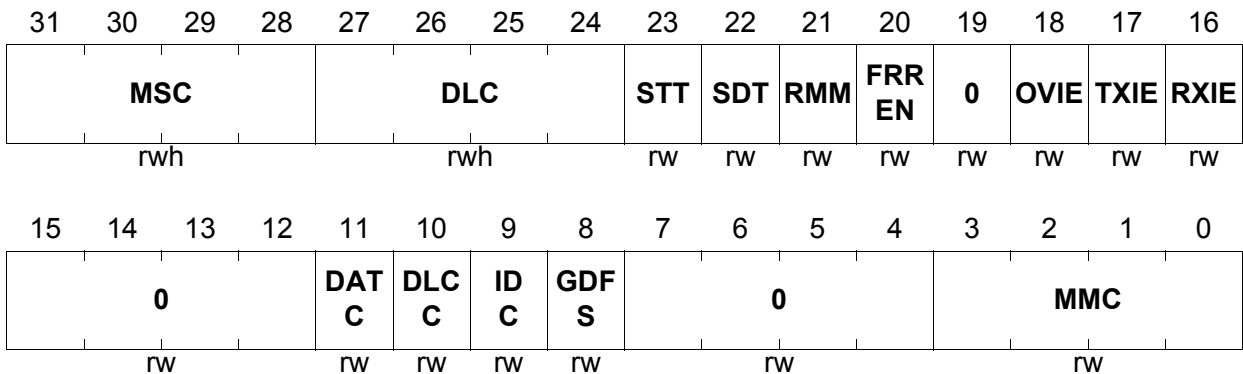
The Message Object Function Control Register MOFCR_n contains bits to select and to configure the function of message object n. It also holds the CAN data length code.

MOFCR_n (n = 0-127)

Message Object n Function Control Register

(0600_H+n*20_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
MMC	[3:0]	rw	Message Mode Control MMC controls the message mode of message object n. 0000 _B Standard Message Object 0001 _B Receive FIFO Base Object 0010 _B Transmit FIFO Base Object 0011 _B Transmit FIFO Slave Object 0100 _B Gateway Source Object All other combinations are reserved.
GDFS	8	rw	Gateway Data Frame Send 0 _B TXRQ is unchanged in the destination object. 1 _B TXRQ is set in the gateway destination object after the internal transfer from the gateway source to the gateway destination object. Applicable only to a gateway source object; ignored in other nodes.

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
IDC	9	rw	<p>Identifier Copy</p> <p>0_B The identifier of the gateway source object is not copied.</p> <p>1_B The identifier of the gateway source object (after storing the received frame in the source) is copied to the gateway destination object.</p> <p>Applicable only to a gateway source object; ignored in other nodes.</p>
DLCC	10	rw	<p>Data Length Code Copy</p> <p>0_B Data length code is not copied.</p> <p>1_B Data length code of the gateway source object (after storing the received frame in the source) is copied to the gateway destination object.</p> <p>Applicable only to a gateway source object; ignored in other nodes.</p>
DATC	11	rw	<p>Data Copy</p> <p>0_B Data fields are not copied.</p> <p>1_B Data fields in registers MODATALn and MODATAHn of the gateway source object (after storing the received frame in the source) are copied to the gateway destination.</p> <p>Applicable only to a gateway source object. Ignored in other nodes.</p>
RXIE	16	rw	<p>Receive Interrupt Enable</p> <p>RXIE enables the message receive interrupt of message object n. This interrupt is generated after reception of a CAN message (independent whether the CAN message is received directly or indirectly via a gateway action).</p> <p>0_B Message receive interrupt is disabled.</p> <p>1_B Message receive interrupt is enabled.</p> <p>Bit field MOIPRn.RXINP selects the interrupt output line which becomes activated at this type of interrupt.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
TXIE	17	rw	<p>Transmit Interrupt Enable TXIE enables the message transmit interrupt of message object n. This interrupt is generated after the transmission of a CAN message.</p> <p>0_B Message transmit interrupt is disabled. 1_B Message transmit interrupt is enabled. Bit field MOIPRn.TXINP selects the interrupt output line which becomes activated at this type of interrupt.</p>
OVIE	18	rw	<p>Overflow Interrupt Enable OVIE enables the FIFO full interrupt of message object n. This interrupt is generated when the pointer to the current message object (CUR) reaches the value of SEL in the FIFO/Gateway Pointer Register.</p> <p>0_B FIFO full interrupt is disabled. 1_B FIFO full interrupt is enabled. If message object n is a Receive FIFO base object, bit field MOIPRn.TXINP selects the interrupt output line which becomes activated at this type of interrupt. If message object n is a Transmit FIFO base object, bit field MOIPRn.RXINP selects the interrupt output line which becomes activated at this type of interrupt. For all other message object modes, bit OVIE has no effect.</p>
FRREN	20	rw	<p>Foreign Remote Request Enable Specifies whether the TXRQ bit is set in message object n or in a foreign message object referenced by the pointer CUR.</p> <p>0_B TXRQ of message object n is set on reception of a matching Remote Frame. 1_B TXRQ of the message object referenced by the pointer CUR is set on reception of a matching Remote Frame.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
RMM	21	rw	<p>Transmit Object Remote Monitoring</p> <p>0_B Remote monitoring is disabled: Identifier, IDE bit, and DLC of message object n remain unchanged upon the reception of a matching Remote Frame.</p> <p>1_B Remote monitoring is enabled: Identifier, IDE bit, and DLC of a matching Remote Frame are copied to transmit object n in order to monitor incoming remote frames.</p> <p>Bit RMM applies only to transmit objects and has no effect on receive objects.</p>
SDT	22	rw	<p>Single Data Transfer</p> <p>If SDT = 1 and message object n is not a FIFO base object, then MSGVAL is cleared when this object has taken part in a successful data transfer (receive or transmit).</p> <p>If SDT = 1 and message object n is a FIFO base object, then MSGVAL is cleared when the pointer to the current object CUR reaches the value of SEL in the FIFO/Gateway Pointer Register.</p> <p>With SDT = 0, bit MSGVAL is not affected.</p>
STT	23	rw	<p>Single Transmit Trial</p> <p>If this bit is set, TXRQ is cleared on transmission start of message object n. Thus, no transmission retry is performed in case of transmission failure.</p>
DLC	[27:24]	rwh	<p>Data Length Code</p> <p>Bit field determines the number of data bytes for message object n. Valid values for DLC are 0 to 8. A value of DLC > 8 results in a data length of 8 data bytes.</p> <p>If a frame with DLC > 8 is received, the received value is stored in the message object.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
MSC	[31:28]	rwh	<p>Message Status Count (TTCAN only) MSC can have values between 0000_B and 0111_B. An increment of MSC = 0111_B results in MSC = 0111_B (no overflow). A decrement of MSC = 0000_B results in MSC = 0000_B (no underflow).</p> <p>Transmission: When message object n is scheduled for transmission in an exclusive time window and the transmission is successful within this time window and the TTCAN is in error state S0 or S1, MSC is decremented by one. If the transmission fails, bit field MSC is incremented by one. If the TTCAN is in error state S2 and detects CAN bus-idle during the transmit enable window for message object n, MSC is decremented by one (although this message object is not transmitted).</p> <p>Reception: When message object n is scheduled for reception in an exclusive time window and the reception is successful within this time window, MSC is decremented by one. If the reception does not take place within this time window, MSC is incremented by one. If TTCAN function is not available/enabled, bit field MSC is unchanged.</p>
0	[7:4], [15:12], 19	rw	<p>Reserved Read as 0 after reset; value last written is read back.</p>

Controller Area Network (MultiCAN) Controller

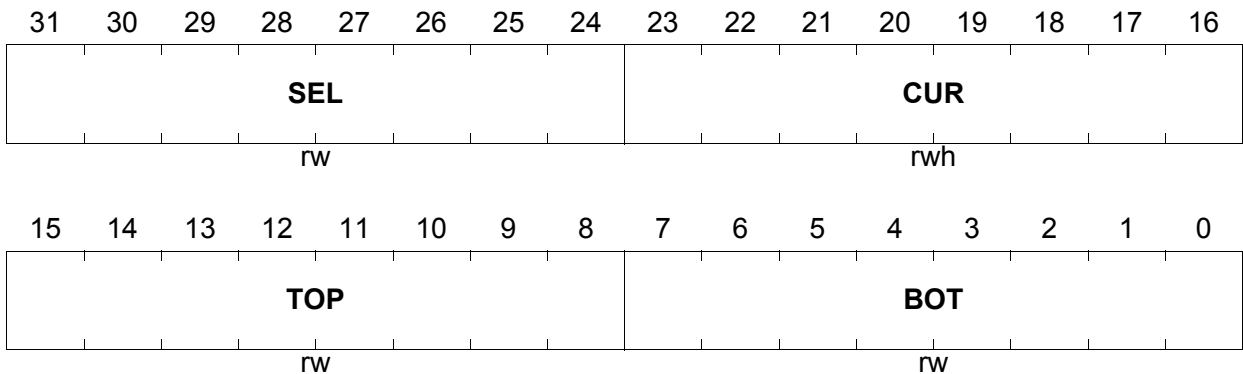
The Message Object FIFO/Gateway Pointer Register MOFGPR_n contains a set of message object link pointers used for FIFO and gateway operations.

MOFGPR_n (n = 0-127)

Message Object n FIFO/Gateway Pointer Register

(0604_H+n*20_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
BOT	[7:0]	rw	Bottom Pointer Bit field BOT points to the first element in a FIFO structure.
TOP	[15:8]	rw	Top Pointer Bit field TOP points to the last element in a FIFO structure.
CUR	[23:16]	rwh	Current Object Pointer Bit field CUR points to the actual target object within a FIFO/Gateway structure. After a FIFO/gateway operation CUR is updated with the message number of the next message object in the list structure (given by PNEXT of the message control register) until it reaches the FIFO top element (given by TOP) when it is reset to the bottom element (given by BOT).
SEL	[31:24]	rw	Object Select Pointer Bit field SEL is the second (software) pointer to complement the hardware pointer CUR in the FIFO structure. SEL is used for monitoring purposes (FIFO interrupt generation).

Controller Area Network (MultiCAN) Controller

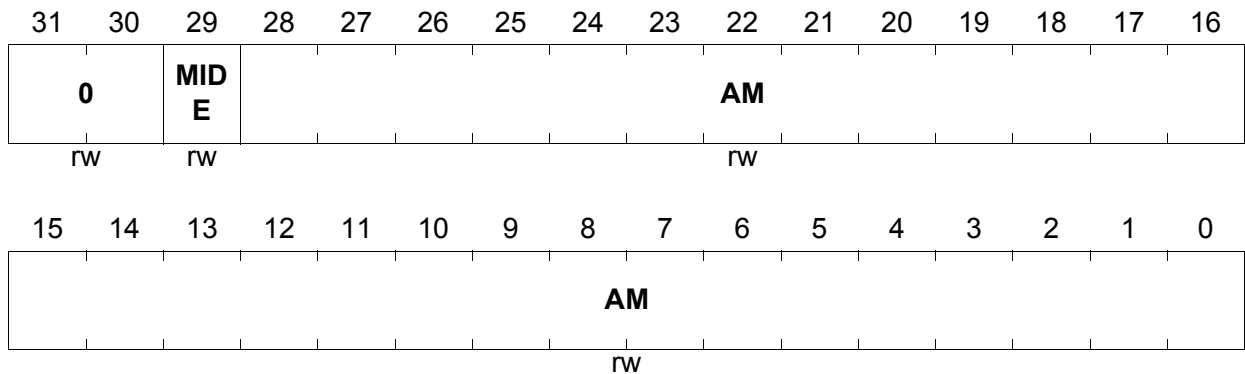
Message Object n Acceptance Mask Register MOAMRn contains the mask bits for the acceptance filtering of message object n.

MOAMRn (n = 0-127)

Message Object n Acceptance Mask Register

(060C_H+n*20_H)

Reset Value: 3FFF FFFF_H



Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
AM	[28:0]	rw	<p>Acceptance Mask for Message Identifier</p> <p>Bit field AM is the 29-bit mask for filtering incoming messages with standard identifiers (AM[28:18]) or extended identifiers (AM[28:0]). For standard identifiers bits AM[17:0] are “don’t care”.</p> <p>For TTC based transmit acceptance filtering (MOARn.PRI = 00_B) register MOAMRn is used to specify the transmit column/row in which this message object may be transmitted in a time-triggered communication. It is then no longer used as an identifier mask. This means that all identifier bits are used for receive acceptance filtering. In this case AMR contains the following information:</p> <ul style="list-style-type: none"> • MOAMR[5:0] = CYCLE • MOAMR[13:8] = MCYCLE • MOAMR[21:16] = COLUMN • MOAMR[29:24] = MCOLUMN <p>where CYCLE is the cycle count (transmit row) and COLUMN is the transmission column of the system matrix. MCYCLE and MCOLUMN are acceptance masks for matching CYCLE and COLUMN against the actual position within the system matrix of the time-triggered communication network. Message object n may be transmitted only if CYCLE and COLUMN match the actual matrix position (given by CYCTMR.BCC and CYCTMR.CSM). If more than one object with valid transmit request matches a given matrix position, the actually transmitted message object is chosen by list order.</p>
MIDE	29	rw	<p>Acceptance Mask Bit for Message IDE Bit</p> <p>0_B Message object n accepts the reception of both, standard and extended frames.</p> <p>1_B Message object n receives frames only with matching IDE bit.</p>
0	[31:30]	rw	<p>Reserved</p> <p>Read as 0 after reset; value last written is read back.</p>

Controller Area Network (MultiCAN) Controller

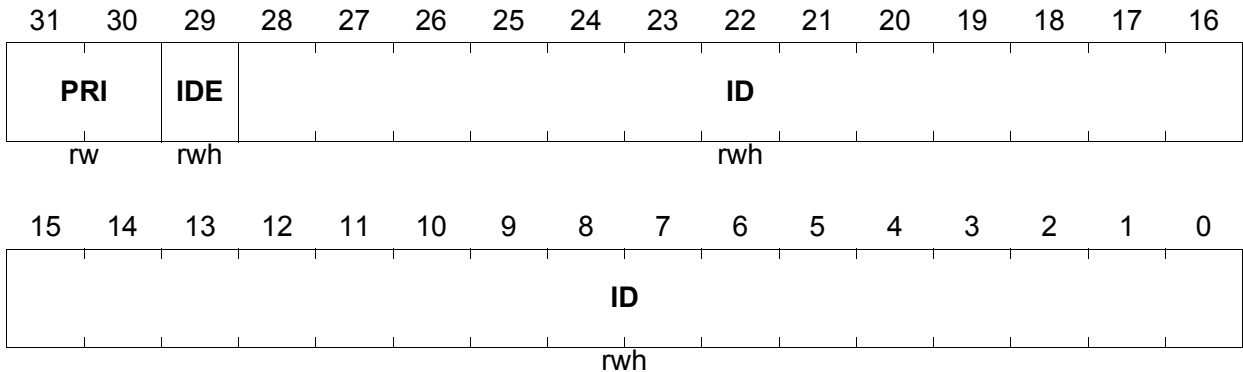
Message Object n Arbitration Register MOARn contains the CAN identifier of message object n.

MOARn (n = 0-127)

Message Object n Arbitration Register

(0618_H+n*20_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
ID	[28:0]	rwh	CAN Identifier of Message Object n Identifier of a standard message (ID[28:18]) or an extended message (ID[28:0]). For standard identifiers, bits ID[17:0] are “don’t care”.
IDE	29	rwh	Identifier Extension Bit of Message Object n 0 _B Message object n handles standard frames with 11-bit identifier. 1 _B Message object n handles extended frames with 29-bit identifier.

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
PRI	[31:30]	rw	<p>Priority Class</p> <p>PRI assigns one of the four priority classes 0, 1, 2, 3 to message object n. A lower PRI number defines a higher priority. Message objects with lower PRI value always win acceptance filtering for frame reception and transmission over message objects with higher PRI value. Acceptance filtering based on identifier/mask and list position is performed only between message objects of the same priority class. PRI also determines the acceptance filtering method for transmission:</p> <p>00_B Time-Triggered CAN (TTCAN): The Message Object Mask Register specifies the position(s) within the transmission matrix at which the message object n may be transmitted. Transmit acceptance filtering is based on the list order. This means that message object n is considered for transmission only if there is no other message object with valid transmit request (MSGVAL & TXEN0 & TXEN1 = 1) that matches the actual position within the transmission matrix somewhere before this object in the list.</p> <p>01_B Transmit acceptance filtering is based on the list order. This means that message object n is considered for transmission only if there is no other message object with valid transmit request (MSGVAL & TXEN0 & TXEN1 = 1) somewhere before this object in the list.</p> <p>10_B Transmit acceptance filtering is based on the CAN identifier. This means, message object n is considered for transmission only if there is no other message object with higher priority identifier + IDE + DIR (with respect to CAN arbitration rules) somewhere in the list (see Table 22-12).</p> <p>11_B Transmit acceptance filtering is based on the list order (as PRI = 01_B).</p>

Controller Area Network (MultiCAN) Controller

Transmit Priority of Msg. Objects based on CAN Arbitration Rules

Table 22-12 Transmit Priority of Msg. Objects Based on CAN Arbitration Rules

Settings of Arbitrarily Chosen Message Objects A and B, (A has higher transmit priority than B)	Comment
A.MOAR[28:18] < B.MOAR[28:18] (11 bit standard identifier of A less than 11 bit standard identifier of B)	Messages with lower standard identifier have higher priority than messages with higher standard identifier. MOAR[28] is the most significant bit (MSB) of the standard identifier. MOAR[18] is the least significant bit of the standard identifier.
A.MOAR[28:18] = B.MOAR[28:18] A.MOAR.IDE = 0 (send Standard Frame) B.MOAR.IDE = 1 (send Extended Frame)	Standard Frames have higher transmit priority than extended frames with equal standard identifier.
A.MOAR[28:18] = B.MOAR[28:18] A.MOAR.IDE = B.MOAR.IDE = 0 A.MOCTR.DIR = 1 (send Data Frame) B.MOCTR.DIR = 0 (send Remote Fame)	Standard Data Frames have higher transmit priority than Standard Remote Frames with equal identifier.
A.MOAR[28:0] = B.MOAR[28:0] A.MOAR.IDE = B.MOAR.IDE = 1 A.MOCTR.DIR = 1 (send Data Frame) B.MOCTR.DIR = 0 (send Remote Frame)	Extended Data Frames have higher transmit priority than Extended Remote Frames with equal identifier.
A.MOAR[28:0] < B.MOAR[28:0] A.MOAR.IDE = B.MOAR.IDE = 1 (29-bit identifier)	Extended Frames with lower identifier have higher transmit priority than Extended Frames with higher identifier. MOAR[28] is the most significant bit (MSB) of the overall identifier (standard identifier MOAR[28:18] and identifier extension MOAR[17:0]). MOAR[0] is the least significant bit (LSB) of the overall identifier.

Controller Area Network (MultiCAN) Controller

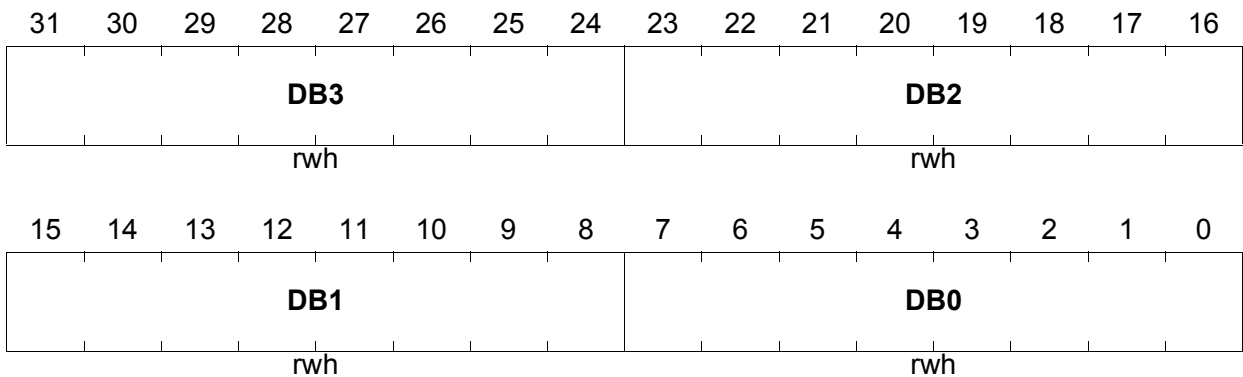
Message Object n Data Register Low MODATAL contains the lowest four data bytes of message object n. Unused data bytes are set to zero upon reception and ignored for transmission.

MODATALn (n = 0-127)

Message Object n Data Register Low

(0610_H+n*20_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
DB0	[7:0]	rwh	Data Byte 0 of Message Object n
DB1	[15:8]	rwh	Data Byte 1 of Message Object n
DB2	[23:16]	rwh	Data Byte 2 of Message Object n
DB3	[31:24]	rwh	Data Byte 3 of Message Object n

Controller Area Network (MultiCAN) Controller

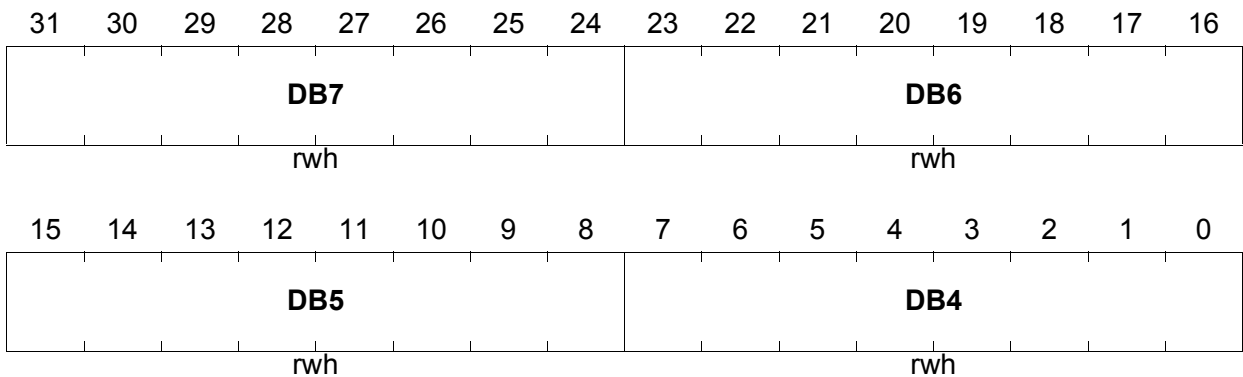
Message Object n Data Register High MODATAH contains the highest four data bytes of message object n. Unused data bytes are set to zero upon reception and ignored for transmission.

MODATAH_n (n = 0-127)

Message Object n Data Register High

(0614_H+n*20_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
DB4	[7:0]	rwh	Data Byte 4 of Message Object n
DB5	[15:8]	rwh	Data Byte 5 of Message Object n
DB6	[23:16]	rwh	Data Byte 6 of Message Object n
DB7	[31:24]	rwh	Data Byte 7 of Message Object n

Controller Area Network (MultiCAN) Controller

22.5 Time-Triggered Extension (TTCAN Controller)

22.5.1 Generation of the Local Time

The Nominal Time Unit (NTU) of the TTCAN controller is generated by the NTU timer with a clock derived from the fractional divider (an NTU timer action takes place with t_q or with a multiple of $1/f_{CAN}$). The sum of the NTU is the local time LT, which is a 16-bit value (to be considered as integer number) and a fractional part LTFR.

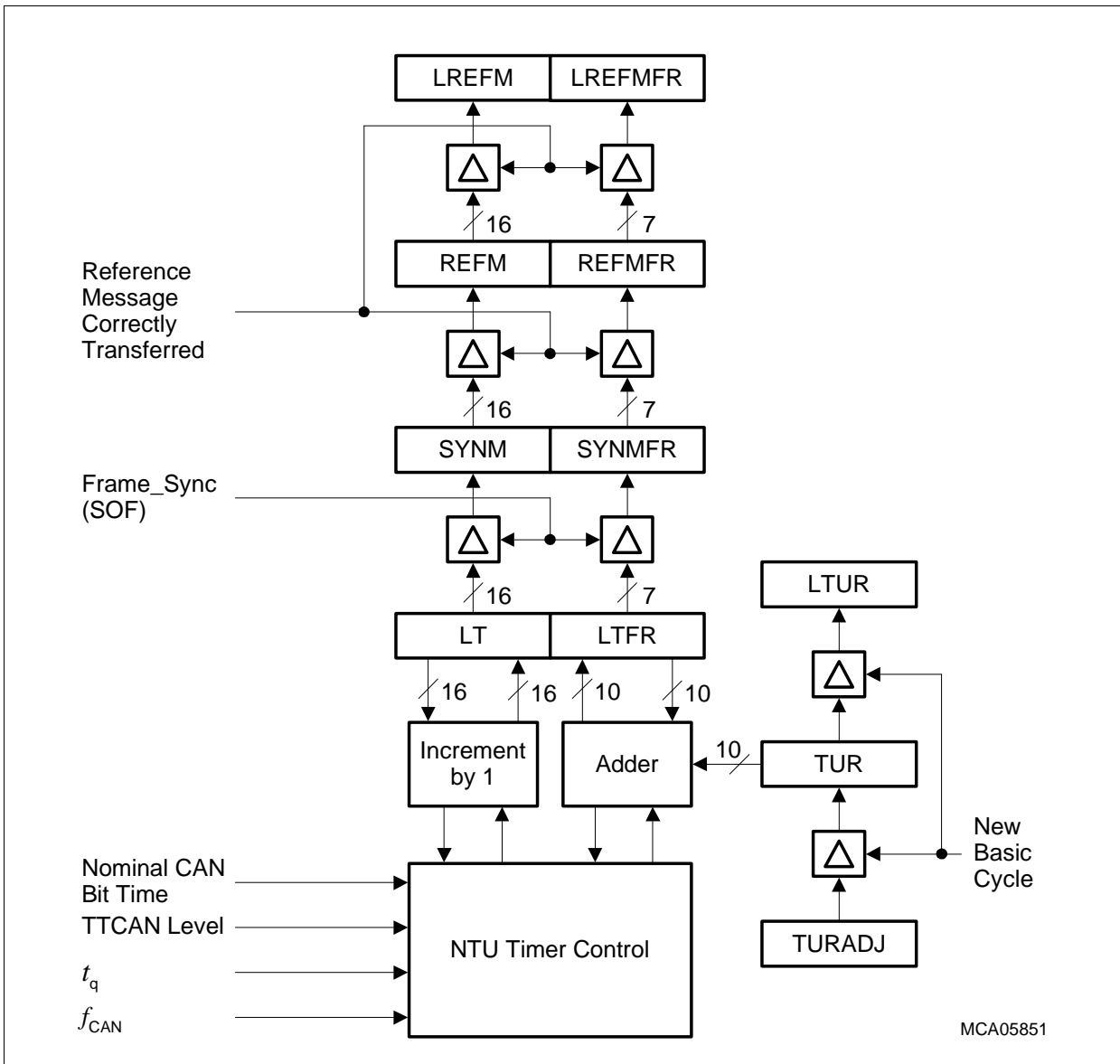


Figure 22-25 Generation of the Local Time

Controller Area Network (MultiCAN) Controller

Note: Register LTUR is used for the internal automatic calculation of the new TURADJ value. It is not visible for the user.

In TTCAN level 1, the LT counter is incremented once per CAN bit time. The fractional part LTFR of the local time and the TUR are not taken into account.

In TTCAN level 2, the LT is extended by its fractional part LTFR. The TUR value is added to the LTFR value with the update period t_{upd} in order to generate a new LT value. If an overflow occurs (carry active) after an addition of LTFR and TUR, the value of LT is incremented by 1. As a result, the value of LT can change only in steps of 1, whereas the value of LTFR changes in steps of TUR.

In order to cover a wide range of CAN baud rates with the same counting range of the local timer, the counting can take place every CAN time quantum t_q ($t_{\text{upd}} = t_q$). If another resolution of the NTU is desired, the local time generation can also be based on the clock f_{CAN} , divided by a programmable factor that is independent from the CAN bit timing.

After the update period t_{upd} has elapsed N times, the difference in local time dt has elapsed to: $dt = \text{TURR.TUR}/1024 \times N \times t_{\text{upd}}$.

22.5.2 Automatic TUR Adjust

It is possible to automatically calculate the new value written to TURADJ for adjusting the correct value for the local time on TTCAN level 2. Each time a new reference message is correctly received, the difference is calculated between the time values in the reference message (GMR) and the previous reference message (LGMR). This difference, divided by the amount of own NTUs for a complete basic cycle multiplied by LTUR, provides the TURADJ value (including fractional parts).

$$\text{TURADJ} = \text{LTUR} \times (\text{GM} - \text{LGM}) / (\text{REFM} - \text{LREFM})$$

The automatic TUR adjust is not made when a discontinuity has been signaled between the reference messages. The automatic calculation of the new value for TURADJ takes place during the basic cycle.

The value of TUR is updated at the beginning of each basic cycle. The new value that is written to TUR to adjust the timing is stored in the bit field TURADJ.

22.5.3 Cycle Time

The cycle time is always positive and represents the time elapsed in the current basic cycle, starting from 0.

22.5.3.1 Local Time and Synchronization Marks

The cycle time is the difference between the local time (stored in bit field LT) and its last reference mark (stored in bit field REFM). The result is available in the bit field CYCTM. The value of the local time LT is captured at the start of each message as a synchronization mark SYN. If the correctly received message has been a reference

Controller Area Network (MultiCAN) Controller

message, the captured value becomes the reference mark REFM. The REFM value of the previous reference mark is stored in LREFM.

The cycle time is still incremented while the reference message is received. After the end of the correctly received reference message, the cycle time takes the value corresponding to the length of this reference message. As a result, the cycle time seems to start from zero each time a reference message starts (= with each new basic cycle).

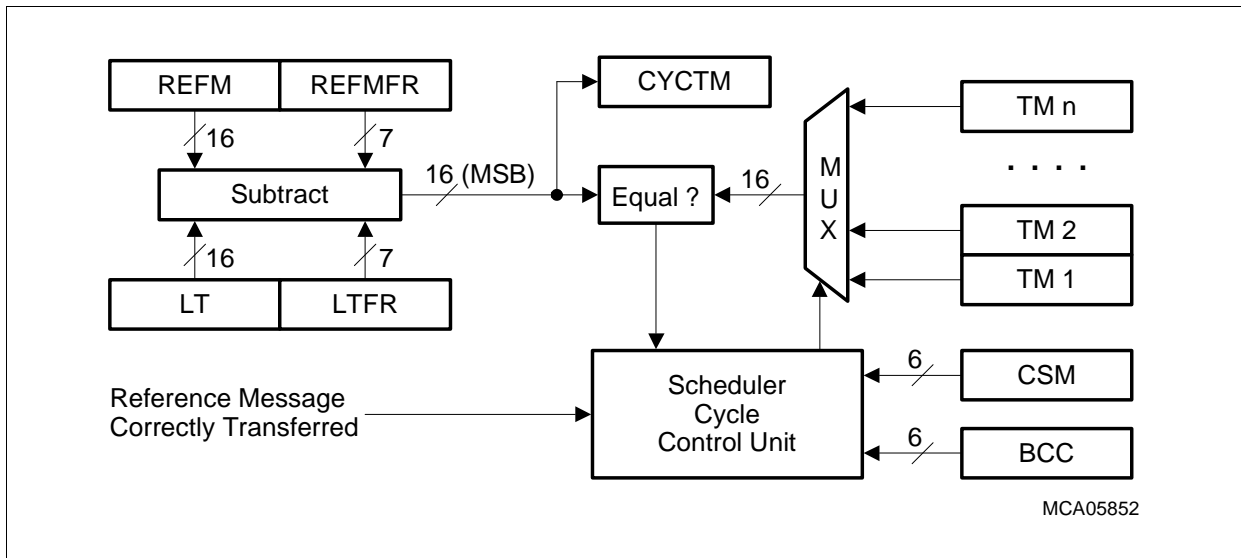


Figure 22-26 Generation of the Cycle Time

The cycle control unit also delivers the values for the basic cycle and matrix cycle handling, and provides the possibility to generate an interrupt if a new basic cycle has started or a new matrix cycle has started. If the TTCAN node receives a reference message, the value for BCC is taken from the reference message.

22.5.3.2 Time Marks

Each time a reference message is received correctly, the cycle control unit starts again comparing the cycle time to the first time mark (TM 1). If one time mark is reached, it continues with the next one. In order to be able to do an “equal-to” comparison, the time marks have to be programmed in increasing order and the first time mark must not be lower than the length of the reference mark.

The cycle time is used to determine the different columns of the system matrix (CSM = column of the system matrix). These columns are defined by the time marks. These time marks can be programmed and are valid for all columns of the system matrix.

The cycle control unit also checks the number of the current basic cycle that is indicated by the basic cycle count BCC. The value of BCC and the value of CSM, taken together, clearly identify each time window in the matrix cycle.

In master mode, the value BCC is transmitted as part of the reference message.

Controller Area Network (MultiCAN) Controller**22.5.3.3 Watch Trigger**

The watch trigger is used to check if the elapsed time since the last reference message was received is too long. This is done by comparing the programmed watch trigger value to the cycle time. This information can generate an interrupt to the system. The watch trigger value is defined by the scheduler entry for the end of the basic cycle.

22.5.4 Master Reference Mark (Level 2 only)

The time master sends the sum of its reference mark SYNM (integer part) and SYNMF (fractional part) and its local offset LOF (integer part) and LOFFR (fractional part) as master reference mark (MRM), which is stored in the global mark register GMR.

If the master's clock or its local time has been corrected and a discontinuity has been introduced, the discontinuity bit DISC is automatically set until the next reference message is started. The discontinuity bit is set if a write access to LOF or LOFFR occurs. As a result, the reference message will contain the new reference mark and the discontinuity bit DISC set for the current time master.

22.5.5 Transmit Enable Window

The transmit enable window determines the number of CAN bit times that may elapse between the transmit trigger event (time marks in the scheduler) and the real start of the message on the bus. If this time expires without starting the transmission of the triggered message, no new transmission attempt will be made in this time window (if it was an exclusive one or a short arbitration one). The transmit trigger is deleted and an interrupt may be generated. If the time window is a long, merged arbitration window, the transmit enable window is not taken into account.

Controller Area Network (MultiCAN) Controller

22.5.6 Local Offset and Global Time

On TTCAN level 2, the local offset of a TTCAN node (not of the time master) is defined as the difference between the Master_Ref_Mark (also named Global_Ref_Mark) and the node's reference mark Ref_Mark. For the time master, the sum of the local time and the local offset is stored as Global_Sync_Mark when the frame synchronization is reached. The Global_Sync_Mark (for the time master) and the Global_Ref_Mark (for all other nodes) are stored in the Global Mark Register GMR. The Local_Offset is stored in the local offset register LOR.

When a TTCAN node receives a reference message, the global mark register is updated by the received value. The value that has been received in the previous reference message is automatically transferred from the global mark register to the last global mark register. The difference between these two values represents the time elapsed between the last two reference messages from the time master's point of view.

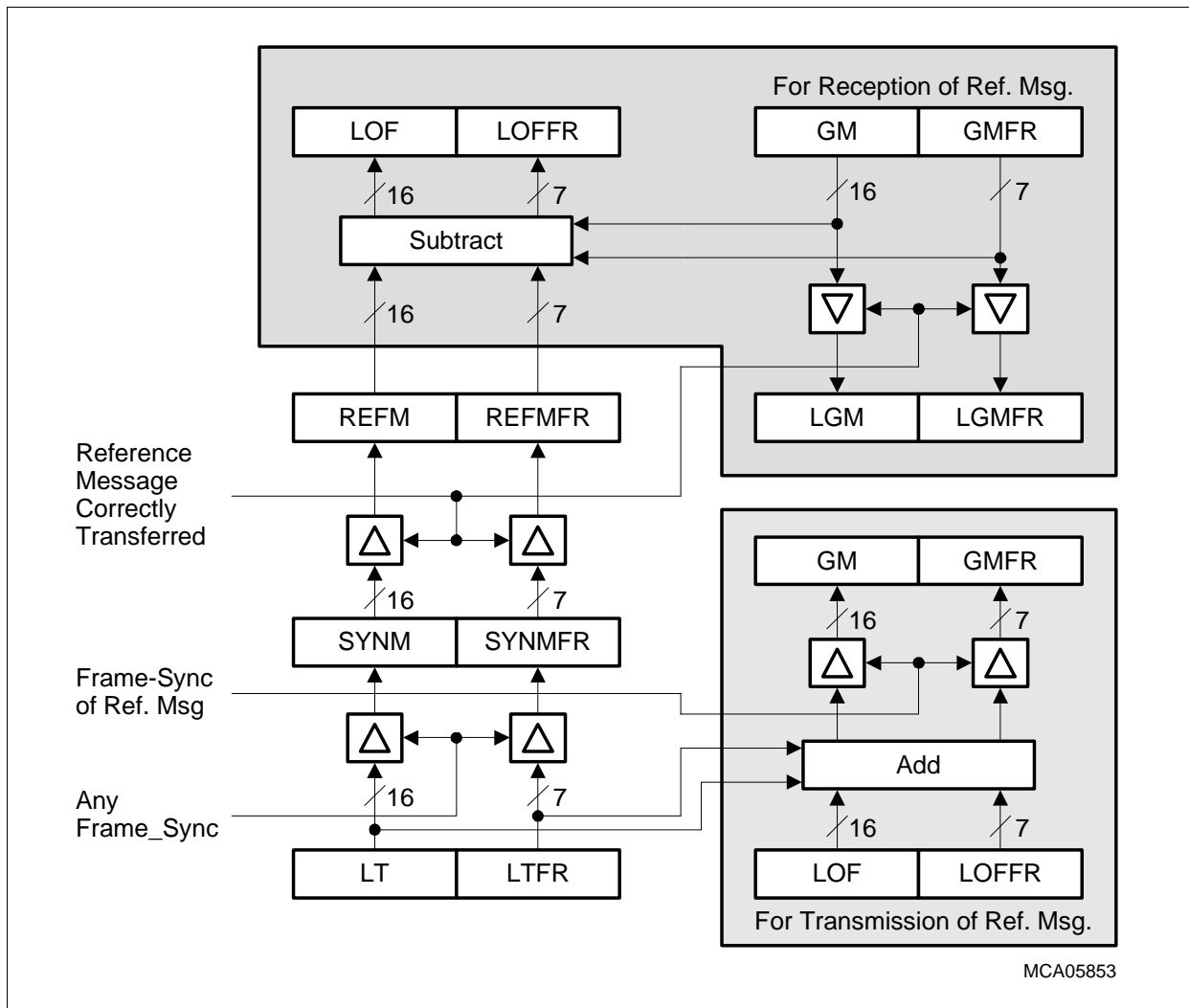


Figure 22-27 Local Offset and Global Time

Controller Area Network (MultiCAN) Controller

Due to possible variations of the LOF values from one reference mark to the other, the current global time may contain discontinuities. In order to get a smooth (continuous) behavior, a SW filter can be used for receiving nodes. The global time can be calculated as shown below:

- Local Time + Local Offset or
- Cycle Time + Reference Mark + Local Offset

22.5.7 Transmit Trigger

The TTCAN feature of a CAN node implies the counting of transmission requests for messages to be sent in exclusive time windows. The number of messages to be transmitted in exclusive time windows is defined by system application design for the entire system matrix (expected transmit triggers EXPTT). This theoretical value is used to verify if the real TTCAN node also requests the same amount of transmissions. If fewer or more transmit triggers have been counted by the transmit trigger counter TTCNT at the end of a system matrix, a transmit trigger error interrupt can be generated (if enabled by TTIEN). This is indicated by the bits transmit trigger underflow TTUF and transmit trigger overflow TTOF.

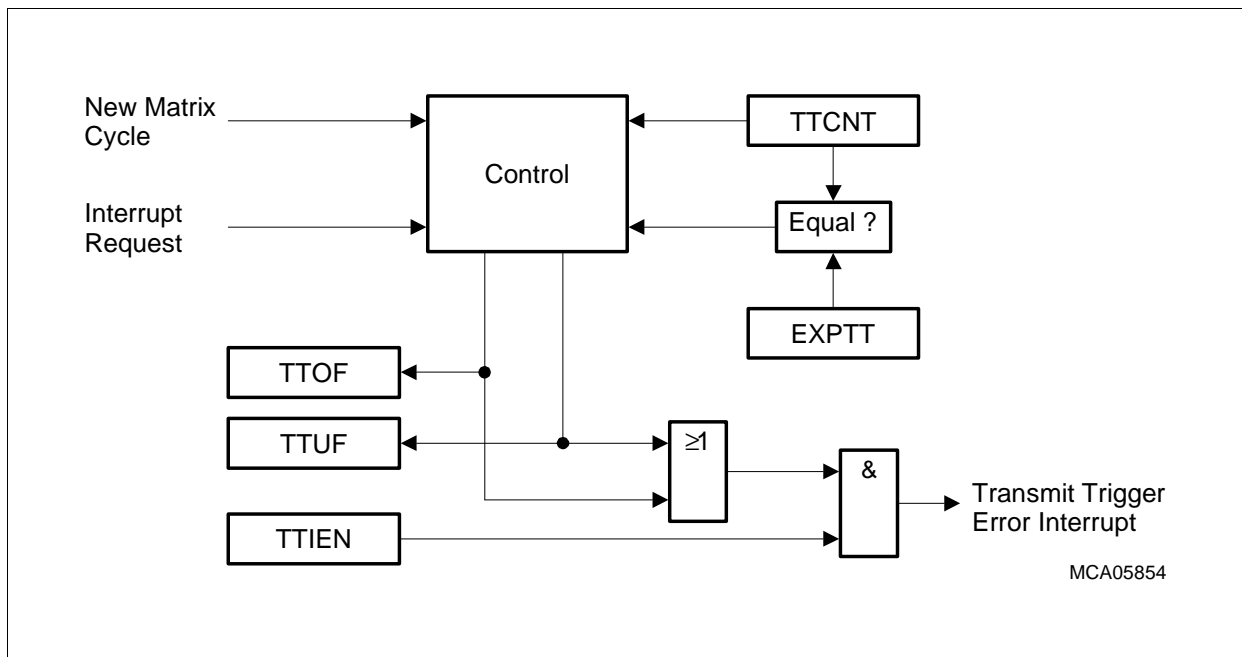


Figure 22-28 Transmit Trigger Monitoring

Controller Area Network (MultiCAN) Controller**22.5.8 Reference Message**

The reference message object has a special role in the TTCAN system. It is the first entry in the message object list for the TTCAN node (node 0 in the TC1796).

22.5.8.1 Differences to Normal CAN Messages

There are several differences between the behavior of a reference message and a normal CAN message:

- If a reference message is disturbed by an error during transmission, it is retransmitted immediately.
- If a reference message loses arbitration, it is not retransmitted, but the message on the bus is received (arbitration of potential time masters).
- For reception, no identifier bits (except the 3 least significant bits ID[2:0]) use an acceptance mask (must be set by software). The entire identifier of the latest correct reference message on the bus is stored in the reference message object.
- For reception, the 3 least significant bits of the identifier always use the acceptance mask 000_B (must be set by software).
- Its transmission request, indicated by the flag TTSR.REFTRG, is related only to a reference time mark or to an external event (for synchronization). If a correct reference message is received while REFTRG is set, REFTRG is automatically cleared.
- For transmission, the identifier bits of the reference message object are taken, except the 3 least significant bits (ID[2:0]). They are taken from the bit field TMCR.TMPRIO.
- The corresponding parts of its eight data bytes are directly connected to the TTCAN internal values (cycle_count, master_ref_mark, etc.) according to the selected level of the TTCAN protocol. The remaining bytes are available for free use.
- The DLC of a reference message to be sent out is determined by the bit field TTCR.RMDLC. This bit field must be at least 1 in TTCAN level 1 and at least 4 in TTCAN level 2.

A reference message object can be selected to be a time master (a potential time master) or a receiving device (no time master).

The reference message object must be initialized as a receive object (but the transmission of this object leads to the reference frame). This is due to the fact that the reception of a reference message is determined by the normal receive acceptance filtering, whereas the transmission is a specific action, triggered by the TTCAN extension.

Reference messages can also be transmitted by time masters that are in S2 error state or after the Init_Watch_Trigger has been reached.

22.5.8.2 Transmit Trigger for a Reference Message

The transmission of a reference message can be triggered by a time mark or by an external event. If it is triggered by a time mark, the “Next is Gap” bit (TTSR.NIG) of the previous reference message has been 0. If it is triggered by an external event, the “Next is Gap” bit of the previous reference message has been 1.

The software can set bit TTSR.NIG in order to initiate the synchronization of a reference message to an event. This event can be an edge at an external input or a software action (write bit TTFMR.STE = 1, software trigger event).

When TTSR.NIG is set, it will be transmitted with the next reference message in order to indicate the following synchronization gap to the other CAN nodes. The value of TTSR.NIG is copied to bit TTSR.ETR (external trigger request) and cleared automatically when the reference message has been correctly transmitted. Bit TTSR.ETR = 1 indicates that the next reference message will be transferred as soon as the corresponding time mark is reached (respecting the gap), or when the selected trigger event (ETREV) is pending. The evaluation of ETREV is not started before 1 CAN bit time has elapsed after the end of the transmit enable window of the last transfer window.

The transmit trigger generation logic for the reference message is shown in [Figure 22-29](#). The edge detection for ECTTx contains a synchronization stage.

The trigger event can be selected by bit field ETESEL (external trigger event selection). Only trigger events are taken into account for the transfer of a reference message that have been detected after the start of the current basic cycle.

The edge detection for ECTTx contains a synchronization stage.

The synchronization of more than one TTCAN node to other TTCAN nodes can be achieved by setting ETESEL = 00_B. In this case, the trigger of the reference message of TTCAN nodes 0-7 (TRMCx) is used to trigger the transmission of the reference message. As a result, the reference messages can be synchronized between several TTCAN nodes of the MultiCAN/TTCAN. If not all possible TTCAN nodes are implemented, the TRMCx of not implemented nodes are 0.

Controller Area Network (MultiCAN) Controller

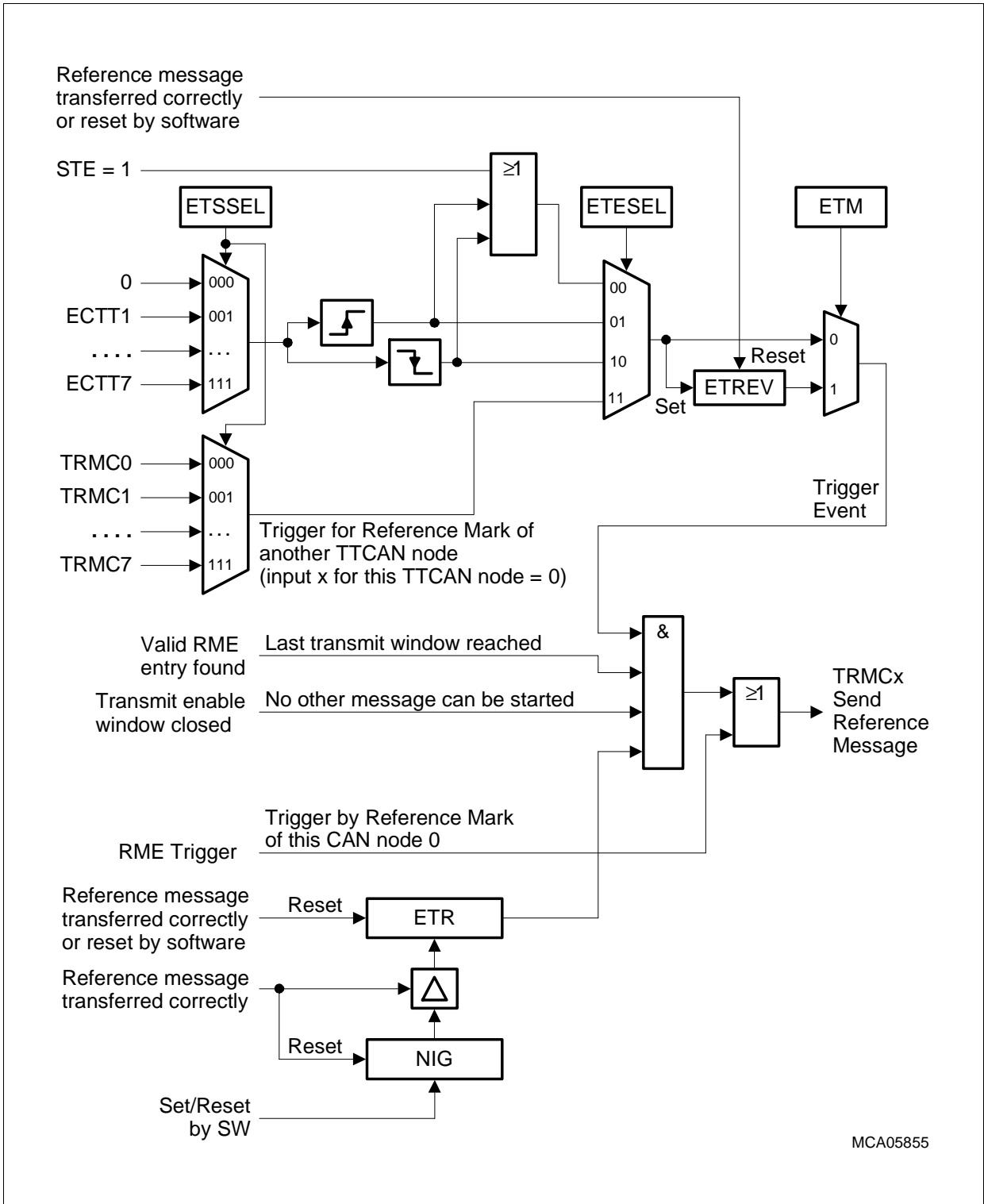


Figure 22-29 External Trigger Generation and Transmit Trigger for the Reference Message

22.6 TTCAN Scheduler

22.6.1 Overview

The message transmission and reception of the time-triggered part TTCAN is controlled by a scheduler mechanism. This scheduler is based on the cycle time and delivers the time marks. The time marks are defined by the time mark entries TME_x. Whenever a time mark is reached, programmable actions (defined by scheduler memory instruction entries INSTR_x_y) can take place. Typical instructions include, for example, starting the transmission of a message, checking if a message has been received, opening or closing arbitration windows, or generating interrupts.

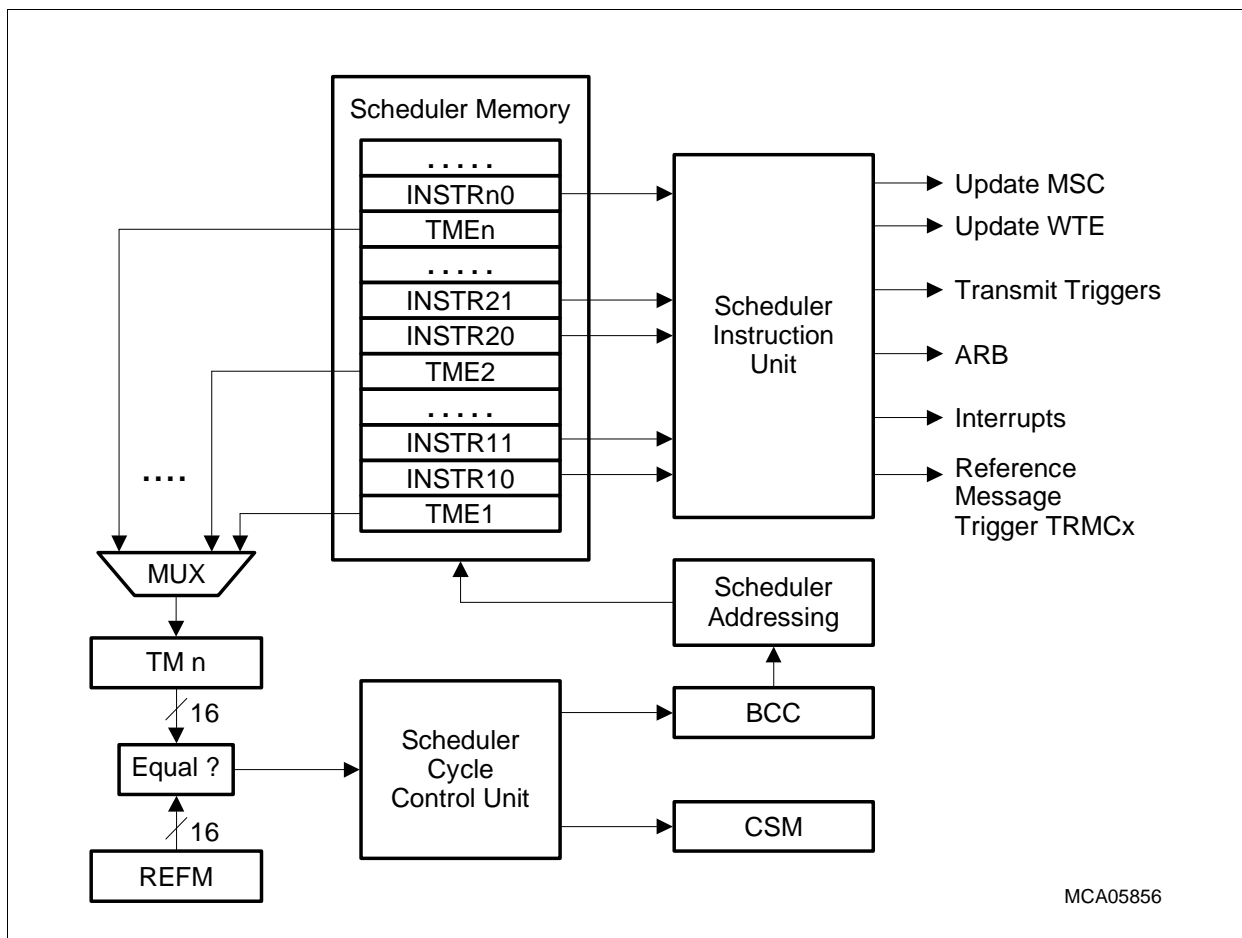


Figure 22-30 Scheduler Overview

The instructions following a time mark are read by the scheduler until the next time mark entry is found. Then, the instruction collection process is stopped until the next time mark is reached.

Note: The time mark values in the scheduler entries must be in increasing order.

Controller Area Network (MultiCAN) Controller

22.6.2 Scheduler Memory

The scheduler contains a memory block in which the time mark entries (TME_x) and the scheduler instruction entries (INSTR_n_x) are stored. Each time mark entry can be followed by a number of scheduler instructions. The scheduler memory is organized as a 32-bit wide memory holding the 32-bit wide time mark entries and the 32-bit wide scheduler instruction entries.

The total number of time mark entries and scheduler instruction entries that can be stored is limited by the size of the scheduler memory. In the TC1796, the scheduler memory has a size of 128 words (32-bits).

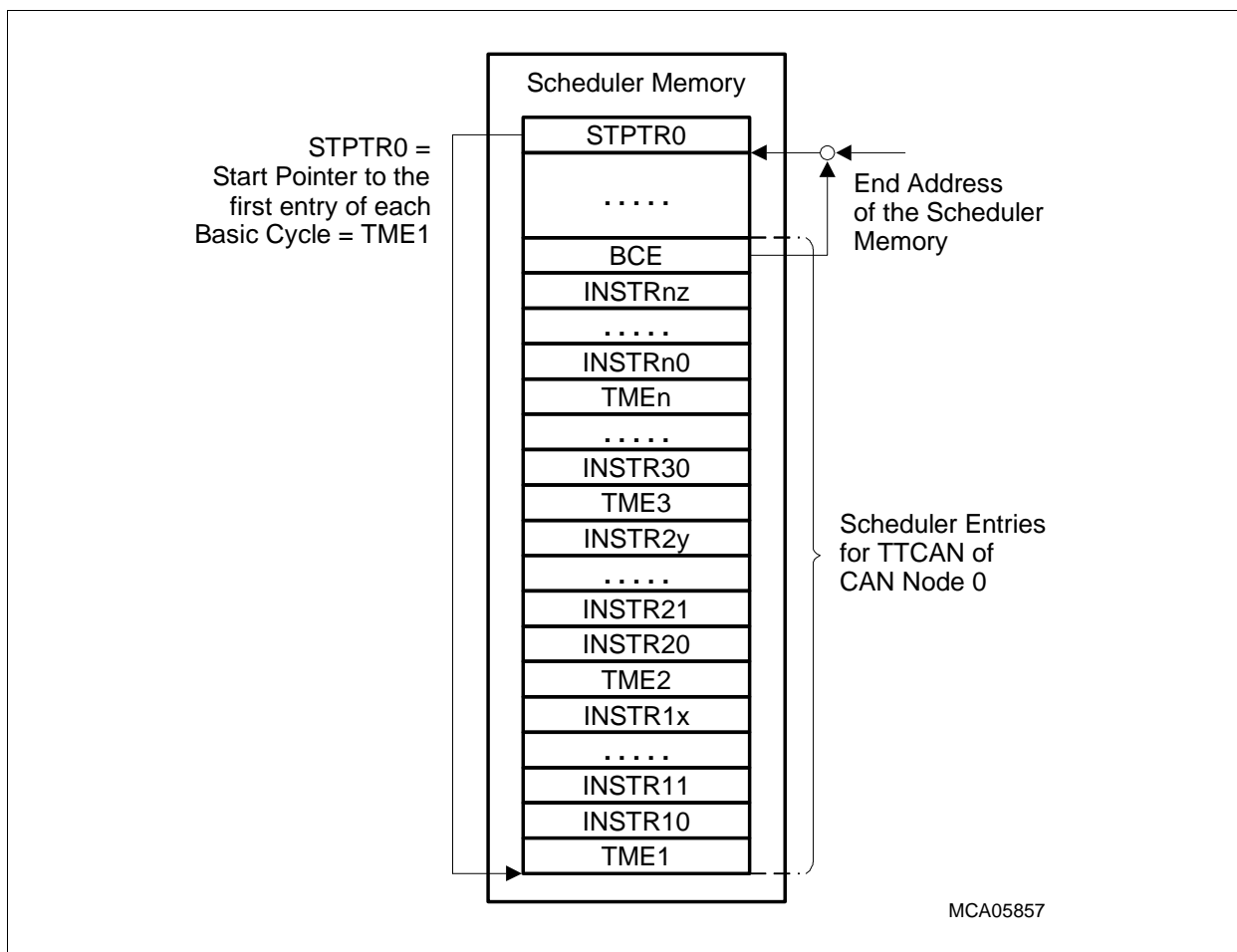


Figure 22-31 TTCAN Scheduler Memory

The last word address of the scheduler memory is reserved for the start pointer STPTR0. The value written at this address determines the start location of the first entry for the TTCAN node (= CAN node 0 in the TC1796). STPTR0 indicates how many 32-bit (word) entries below SPTR0 the first time mark entry (TME1) is located.

When the basic cycle end entry BCE (with GM = 0) is read by the scheduler, it prepares the next scheduler instructions (read from scheduler memory) starting again with TME1.

Controller Area Network (MultiCAN) Controller

Therefore, the first entry read after BCE is STPTR0 in order to obtain the address where TME1 is located.

22.6.2.1 Scheduler Entry Types

The scheduler instruction entries are located after the time mark entry. The time mark entry determines the behavior of the TTCAN when the next time mark is reached. The scheduler instructions after the time mark entry are valid for the time mark they follow. The scheduler control reads the time mark entry and sets up the compare action (compare between the new time mark with the cycle time). Then, the scheduler reads the following instructions and sets up the transfer behavior of the TTCAN node. The scheduler stops reading from the scheduler memory when a new time mark entry is found.

The transfer behavior set up can change with each scheduler instruction. There is no built-in consistency check for scheduler instructions. Previously read scheduler instructions can be overwritten by subsequent ones. With each time mark, the settings are cleared and must be set up by new scheduler instructions (if desired).

The complete scheduler information must be finished with a basic cycle end entry that fixes the value for the watch trigger event.

Each entry in the scheduler memory contains a 4-bit wide code field that determines the type of the entry. The possible scheduler memory entry types are listed in [Table 22-13](#).

Table 22-13 Scheduler Memory Entry Types

Entry Code EC Bits [31:28]	Short Name	Entry Type	Defined at
0001 _B	TME	Time mark entry	see Page 22-126
0010 _B	ICE	Interrupt control entry	see Page 22-129
0011 _B	ARBE	Arbitration entry	see Page 22-131
0100 _B	TCE	Transmit control entry	see Page 22-133
0101 _B	RCE	Receive control entry	see Page 22-136
0110 _B	RME	Reference message entry	see Page 22-138
0111 _B	BCE	Basic cycle end entry	see Page 22-140
other combinations	EOS	End of scheduler memory entry	see Page 22-141

The general transmit trigger control for the message objects to be transmitted when a time mark is reached is performed by the message objects themselves. In order to increase the flexibility, the result of the acceptance filtering done with the message objects can be overruled by control entries. A reference message can be transmitted only when the CAN node is enabled as time master.

Controller Area Network (MultiCAN) Controller

Note: The number of scheduler entries following a time mark should not exceed the number of 10. In order to minimize the required accesses to the scheduler memory by the scheduler control part, the user should set up the system carefully.

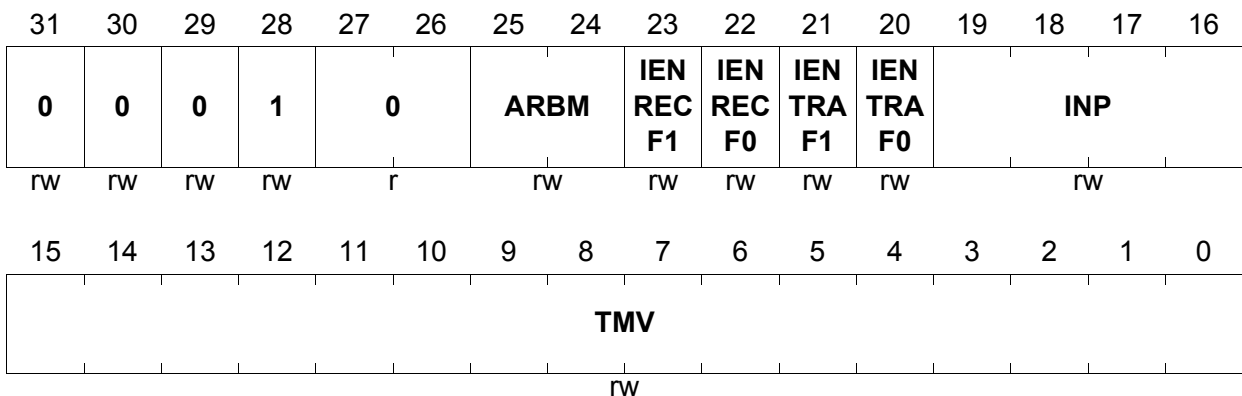
22.6.2.2 Scheduler Entry Type Description

Time Mark Entry

A time mark entry is defined as follows:

TME

Time Mark Entry



Field	Bits	Type	Description
TMV ¹⁾	[15:0]	rw	<p>Time Mark Value</p> <p>This bit field determines the compare value used for the next compare action with the cycle time.</p>
INP	[19:16]	rw	<p>Interrupt Node Pointer</p> <p>INP selects the interrupt output line INT_Om (m = 0-15) that will be activated when a match is detected between the cycle time and the time mark value defined by TMV and if at least one of the four interrupt conditions is met and enabled.</p> <p>0000_B Interrupt output line INT_O0 is selected. 0001_B Interrupt output line INT_O1 is selected. ..._B ... 1110_B Interrupt output line INT_O14 is selected. 1111_B Interrupt output line INT_O15 is selected.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
IENTRAF0	20	rw	<p>Interrupt Enable if TRAF = 0 This bit enables the generation of an interrupt when bit TTSR.TRAF = 0 and a match is detected between the cycle time and the time mark value defined by TMV.</p> <p>0_B Interrupt generation is disabled. 1_B Interrupt generation is enabled.</p>
IENTRAF1	21	rw	<p>Interrupt Enable if TRAF = 1 This bit enables the generation of an interrupt when bit TTSR.TRAF = 1 and a match is detected between the cycle time and the time mark value defined by TMV.</p> <p>0_B Interrupt generation is disabled. 1_B Interrupt generation is enabled.</p>
IENRECF0	22	rw	<p>Interrupt Enable if RECF = 0 This bit enables the generation of an interrupt when when bit TTSR.RECF = 0 and a match is detected between the cycle time and the time mark value defined by TMV.</p> <p>0_B Interrupt generation is disabled. 1_B Interrupt generation is enabled.</p>
IENRECF1	23	rw	<p>Interrupt Enable if RECF = 1 This bit enables the generation of an interrupt when when bit TTSR.RECF = 1 and a match is detected between the cycle time and the time mark value defined by TMV.</p> <p>0_B Interrupt generation is disabled. 1_B Interrupt generation is enabled.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
ARBM	[25:24]	rw	<p>Arbitration Mode This bit field determines the action that is taken with an arbitration window.</p> <p>00_B No action taken; the status of the arbitration window is not changed.</p> <p>01_B A merged (long) arbitration window will be opened. If it is already opened, it is kept open.</p> <p>10_B A merged (long) transfer window will be closed. Closing a merged transfer window leads to a single (short) arbitration window (transmission possible only during the transmit enable window). The short arbitration window is automatically closed after one time window. If there is no arbitration window open, this entry will be ignored.</p> <p>11_B A single (short) arbitration window (transmission possible only during the transmit enable window) is opened. A short arbitration window is automatically closed after one time window.</p>
EC	[31:28]	rw	<p>Entry Code The entry code EC = 0001_B defines this scheduler memory entry as a time mark entry.</p>
0	[27:26]	r	<p>Reserved; read as 0; should be written with 0. Bits are “don’t care” for scheduler operation when EC = 0001_B.</p>

1) In order to be able to detect each time mark correctly, the first time mark of a basic cycle must not be set to a value below the length of the reference message under worst case conditions.

Note: The interrupt settings given by the time mark entry can be overruled by a subsequent valid interrupt control entry. The settings are valid until the next time mark is reached. For the time after the next time mark, the settings read from the next time mark entry become valid.

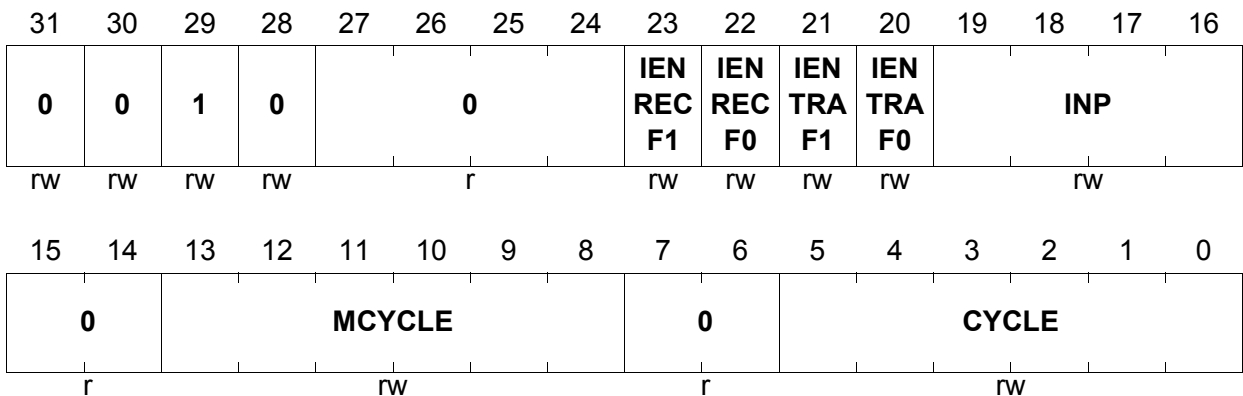
Controller Area Network (MultiCAN) Controller

Interrupt Control Entry

An interrupt control entry is defined as follows:

ICE

Interrupt Control Entry



Field	Bits	Type	Description
CYCLE	[5:0]	rw	<p>Basic Cycle Number</p> <p>This bit field determines the number of the basic cycle during which this interrupt control entry is valid. The value of CYCLE is compared (bit-wise) to the current value of the bit field CYCTMR.BCC. The result is then masked with the value given by the bit field MCYCLE in order to determine the repetition rate for this scheduler entry inside the matrix cycle. This bit field is equivalent to the corresponding part of the MOAMRn.AM bit field (see Page 22-108).</p>
MCYCLE	[13:8]	rw	<p>Mask for Cycle Comparison</p> <p>This bit field determines the mask that is used to determine the repetition rate for this scheduler entry. This bit field is equivalent to the corresponding part of the MOAMRn.AM bit field (see Page 22-108).</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
INP	[19:16]	rw	<p>Interrupt Node Pointer INP selects the interrupt output line INT_Om (m = 0-15) that will be activated when a match is detected between the cycle time and the time mark value defined by TMV and if at least one of the four interrupt conditions is met and enabled.</p> <p>0000_B Interrupt output line INT_O0 is selected. 0001_B Interrupt output line INT_O1 is selected. 1110_B Interrupt output line INT_O14 is selected. 1111_B Interrupt output line INT_O15 is selected.</p>
IENTRAF0	20	rw	<p>Interrupt Enable if TRAF = 0 This bit enables the generation of an interrupt when a match is detected between the cycle time and the time mark value defined by TMV and bit TRAF = 0.</p> <p>0_B An interrupt will not be generated. 1_B An interrupt will be generated.</p>
IENTRAF1	21	rw	<p>Interrupt Enable if TRAF = 1 This bit enables the generation of an interrupt when a match is detected between the cycle time and the time mark value defined by TMV and bit TRAF = 1.</p> <p>0_B An interrupt will not be generated. 1_B An interrupt will be generated.</p>
IENRECF0	22	rw	<p>Interrupt Enable if RECF = 0 This bit enables the generation of an interrupt when a match is detected between the cycle time and the time mark value defined by TMV and bit RECF = 0.</p> <p>0_B An interrupt will not be generated. 1_B An interrupt will be generated.</p>
IENRECF1	23	rw	<p>Interrupt Enable if RECF = 1 This bit enables the generation of an interrupt when a match is detected between the cycle time and the time mark value defined by TMV and bit RECF = 1.</p> <p>0_B An interrupt will not be generated. 1_B An interrupt will be generated.</p>
EC	[31:28]	rw	<p>Entry Code The entry code EC = 0010_B defines this scheduler memory entry as an interrupt control entry.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
0	[7:6], [15:14], [27:24]	r	Reserved ; read as 0; should be written with 0. Bits are “don’t care” for scheduler operation when EC = 0011 _B .

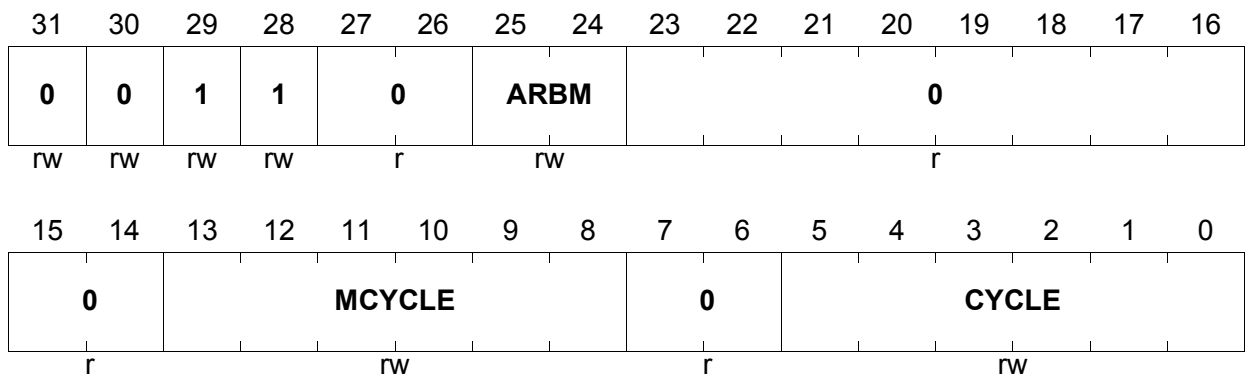
Note: The interrupt settings of a valid interrupt control entry overrule the previously stored interrupt settings for the next time mark. As a result, the use of more than one valid interrupt control entry between two time mark entries should be avoided.

Arbitration Entry

The arbitration entry is defined as follows:

ARBE

Arbitration Entry



Field	Bits	Type	Description
CYCLE	[5:0]	rw	<p>Basic Cycle Number</p> <p>This bit field determines the number of the basic cycle during which this arbitration entry is valid. The value of CYCLE is compared (bit-wise) with the current value of bit field CYCTMR.BCC. The result is then masked with the value given by the bit field MCYCLE in order to determine the repetition rate for this scheduler entry inside the matrix cycle.</p> <p>This bit field is equivalent to the corresponding part of the MOAMRn.AM bit field (see Page 22-108).</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
MCYCLE	[13:8]	rw	<p>Mask for Cycle Comparison</p> <p>This bit field determines the mask used to determine the repetition rate for this arbitration entry. This bit field is equivalent to the corresponding part of the MOAMRn.AM bit field (see Page 22-108).</p>
ARBM	[25:24]	rw	<p>Arbitration Mode</p> <p>This bit field determines the action that is taken with an arbitration window.</p> <p>00_B No action is taken; the status of the window is not changed.</p> <p>01_B A merged (long) arbitration window will be opened. If it is already opened, it is kept open.</p> <p>10_B A merged (long) transfer window will be closed. Closing a merged transfer window leads to a single (short) arbitration window (transmission possible only during the transmit enable window). The short arbitration window is automatically closed after one time window.</p> <p>11_B A single (short) arbitration window (transmission possible only during the transmit enable window) is opened. A short arbitration window is automatically closed after one time window.</p>
EC	[31:28]	rw	<p>Entry Code</p> <p>The entry code EC = 0011_B defines this scheduler memory entry as an arbitration entry.</p>
0	[7:6], [23:14], [27:26]	r	<p>Reserved; read as 0; should be written with 0. Bits are “don’t care” for scheduler operation when EC = 0011_B.</p>

Note: The arbitration settings of a valid arbitration entry overrule the previously stored arbitration settings for the next time mark. As a result, the use of more than one valid arbitration entry between two time mark entries should be avoided.

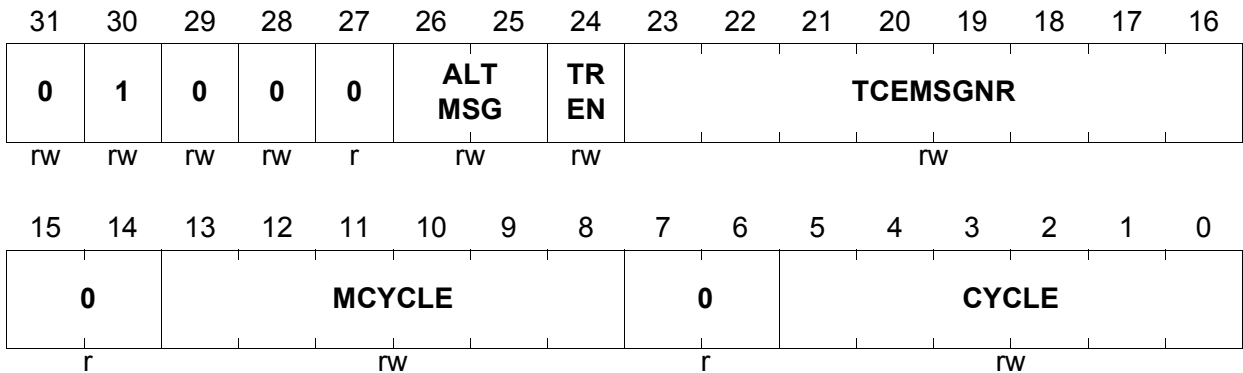
Controller Area Network (MultiCAN) Controller

Transmit Control Entry

The transmit control entry is defined as follows:

TCE

Transmit Control Entry



Field	Bits	Type	Description
CYCLE	[5:0]	rw	<p>Basic Cycle Number</p> <p>This bit field determines the number of the basic cycle during which this transmit control entry is valid. The value of CYCLE is compared (bit-wise) to the current value of the bit field CYCTMR.BCC. The result is then masked with the value given by the bit field MCYCLE in order to determine the repetition rate for this transmit control entry inside the matrix cycle.</p> <p>This bit field is equivalent to the corresponding part of the MOAMRn.AM bit field (see Page 22-108).</p>
MCYCLE	[13:8]	rw	<p>Mask for Cycle Comparison</p> <p>This bit field determines the mask that is used to determine the repetition rate for this transmit control entry.</p> <p>This bit field is equivalent to the corresponding part of the MOAMRn.AM bit field (see Page 22-108).</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
TCEMSGNR	[23:16]	rw	<p>TCE Message Number</p> <p>This bit field determines the number of the message object to be transmitted when the next time mark is reached. The object will be transferred only if it is tagged valid for transmission. With this scheduler instruction, the number of the message object is defined directly, without taking into account to which CAN node it is currently assigned. Even if it is assigned to another CAN node, the selected message object will be transmitted.</p> <p>If the CAN message transmit acceptance filtering delivers an object number to be transmitted, it is overruled by a valid transmit control entry according to ALTMSG.</p> <p>If the message object delivered by the transmit control entry is not valid for transmission, no transmission will be started.</p> <p>If the implementation of the CAN controller supports fewer than 256 message objects, higher values for MSGNR than the number of actually supported message objects will be treated with a modulo operation (e.g. if 128 objects are supported, the value of MSGNR = 135 is equal to MSGNR = 7).</p>
TREN	24	rw	<p>Transmission Enable</p> <p>This bit enables transmission of a message object in the next time window due to the TTCAN transmit trigger.</p> <p>0_B Neither a TTCAN transmit trigger nor the message object transmit acceptance will lead to a transmission of a message object in the next time window.</p> <p>1_B If a TTCAN transmit trigger occurs or the message object transmit acceptance filtering delivers a valid message object, the transmission of a message object in the next time window is enabled.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
ALTMSG	[26:25]	rw	<p>Alternative Message This bit field determines which message object will be transmitted.</p> <p>00_B If the message object number delivered by this entry is not valid for transmission, no message will be sent.</p> <p>01_B The message object number delivered by this entry is not taken into account for transmission, the message object found by the transmit acceptance filtering will be sent if it is valid.</p> <p>10_B If the message object delivered by this entry is not valid for transmission, while the message object transmit acceptance filtering delivered a valid object, the latter one will be sent out.</p> <p>11_B Reserved</p>
EC	[31:28]	rw	<p>Entry Code The entry code EC = 0100_B defines this scheduler memory entry as a transmit control entry.</p>
0	[7:6], [15:14], 27	r	<p>Reserved; read as 0; should be written with 0. Bits are “don’t care” for scheduler operation when EC = 0100_B.</p>

Note: The message transmit settings of a valid transmit control entry overrule the previously stored transmit settings and may also overrule the transmit acceptance filtering of the message objects for the next time mark (depending on bit ALTMSG). As a result, the use of more than one valid transmit control entry between two time mark entries should be avoided.

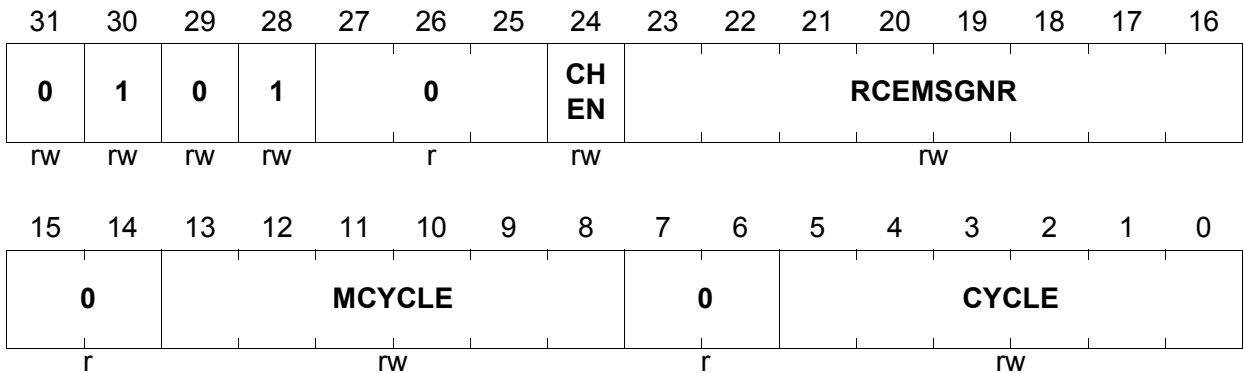
Controller Area Network (MultiCAN) Controller

Receive Control Entry

The receive control entry is defined as follows:

RCE

Receive Control Entry



Field	Bits	Type	Description
CYCLE	[5:0]	rw	<p>Basic Cycle Number</p> <p>This bit field determines the number of the basic cycle during which this receive control entry is valid. The value of CYCLE is compared (bit-wise) to the current value of the bit field CYCTMR.BCC. The result is then masked with the value given by the bit field MCYCLE in order to determine the repetition rate for this scheduler entry inside the matrix cycle. This bit field is equivalent to the corresponding part of the MOAMRn.AM bit field (see Page 22-108).</p>
MCYCLE	[13:8]	rw	<p>Mask for Cycle Comparison</p> <p>This bit field determines the mask that is used to determine the repetition rate for this scheduler entry. This bit field is equivalent to the corresponding part of the MOAMRn.AM bit field (see Page 22-108).</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
RCEMSGNR	[23:16]	rw	<p>RCE Message Number</p> <p>This bit field determines the number of the message object that is checked for correct reception of a message during the last transfer window. A received message is always stored in the message object that is determined by acceptance filtering. When reaching a new time mark, an RCE can be used to check if a desired message has actually been received in the desired message object.</p>
CHEN	24	rw	<p>Check Enable</p> <p>Only time windows with an active receive control entry with CHEN=1 will handle the update of the MSC bit field in the message object indicated by MSGNR. If a frame has been received correctly and also stored in the indicated message object, then the reception is considered as correct; otherwise, it is an error.</p> <p>0_B The MSC bit field of the indicated message object is not updated.</p> <p>1_B The MSC bit field of the indicated message object is updated.</p>
EC	[31:28]	rw	<p>Entry Code</p> <p>The entry code EC = 0101_B defines this scheduler memory entry as a receive control entry.</p>
0	[7:6], [15:14], [27:25]	r	<p>Reserved; read as 0; should be written with 0. Bits are “don’t care” for scheduler operation when EC = 0101_B.</p>

Note: The settings of a valid receive control entry overrule the previously stored information. As a result, the use of more than one valid receive control entry between two time mark entries should be avoided.

- If no receive control entry has been found valid for the current time window, the reception and storage of messages depends only on the message object receive acceptance filtering. Only message objects receiving messages with a valid receive control entry modify their MSC bit field accordingly. The MSC bit field of a receiving object is not modified without a valid receive control entry.*

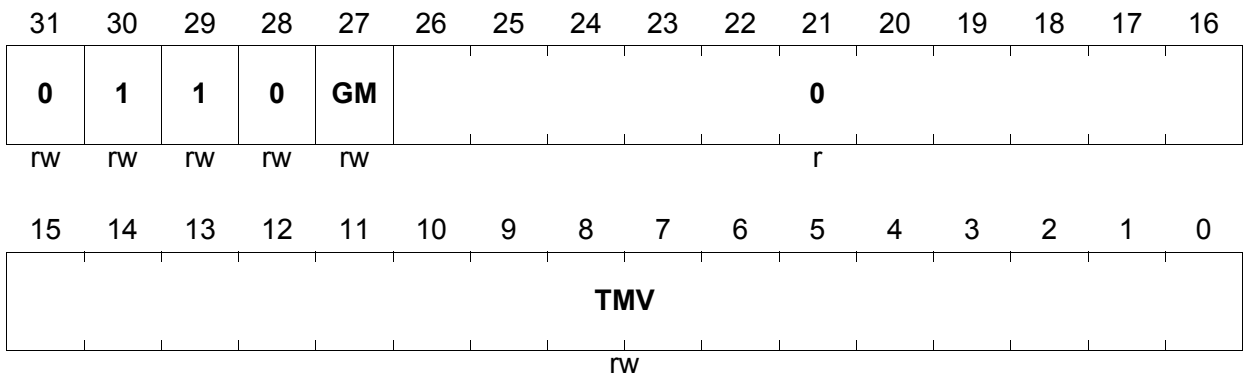
Controller Area Network (MultiCAN) Controller

Reference Message Entry

The reference message is defined as follows:

RME

Reference Message Entry



Field	Bits	Type	Description
TMV	[15:0]	rw	<p>Time Mark Value</p> <p>These bits determine the compare value for the next time mark. This value is used for the next compare action with the cycle time.</p> <p>If the CAN node is a time master, the reference message will be sent out when the time mark is reached (see also TMR.RTO).</p>
GM	27	rw	<p>Gap Mode</p> <p>This bit determines how the scheduler of a time master proceeds if the TTCAN node is “in a gap”.</p> <p>0_B The reference message will be sent according to this RME entry (without respecting a possible gap).</p> <p>1_B If the TTCAN node is “in a gap”, this RME entry is discarded and the following entries are read.</p>
EC	[31:28]	rw	<p>Entry Code</p> <p>The entry code EC = 0110_B defines this scheduler memory entry as a reference message entry.</p>
0	[26:16]	r	<p>Reserved; read as 0; should be written with 0. Bits are “don’t care” for scheduler operation when EC = 0110_B.</p>

Controller Area Network (MultiCAN) Controller

Note: This entry is taken into account for a time master. In the case that more valid reference message entries are detected, the settings of the previously found reference message entries are overruled by a subsequent one.

If the TTCAN node is not configured as time master and the scheduler reads an RME, a configuration error is generated.

While the system is in the synchronization state, an RME entry with GM = 1 is taken into account.

Controller Area Network (MultiCAN) Controller

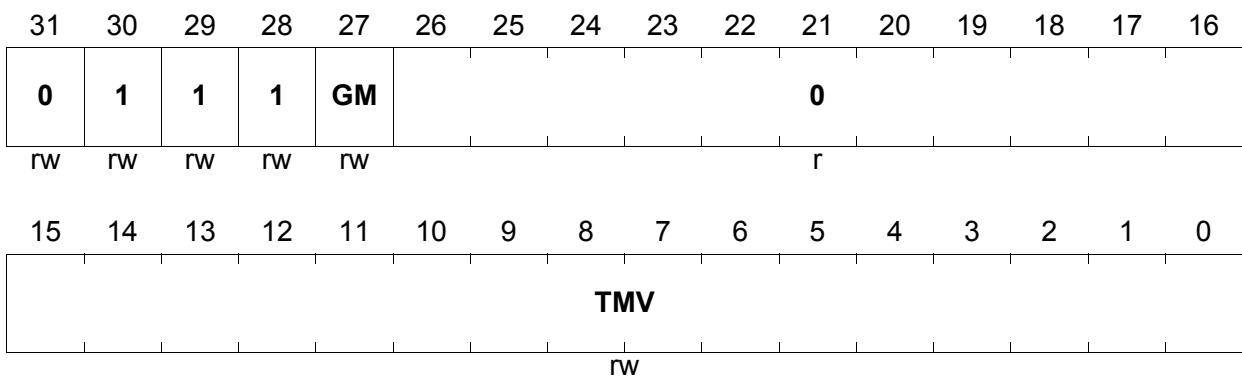
Basic Cycle End Entry

The basic cycle end entry defines the time mark value that is used as a watch trigger time mark. In the case that the TTCAN system is waiting for a reference message while the system is normally synchronized (not in a gap = not waiting for another event to synchronize). If the system is in a gap with GM set, the following entries can be read.

The basic cycle end is defined as follows:

BCE

Basic Cycle End Entry



Field	Bits	Type	Description
TMV	[15:0]	rw	<p>Time Mark Value</p> <p>This bit field determines the compare value for the next time mark that is used for the next compare action with the cycle time.</p> <p>TMV determines the watch trigger that is used to generate a watch trigger event WTE when the cycle time reaches TMV.</p> <p>The value written into TMV must meet the condition that a minimum of one CAN frame length is in between the last time mark and the basic cycle end entry. For this calculation, the reference trigger offset TMR.RTO for the reference message entry must also be taken into account.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
GM	27	rw	<p>Gap Mode This bit determines how the scheduler proceeds when the TTCAN node is “in a gap” and waits for a trigger event to send the reference message (for the current time master) or when it waits for the reception of a reference message.</p> <p>0_B The watch trigger event will be generated according to this BCE entry (without respecting a possible gap).</p> <p>1_B If the TTCAN node is “in a gap”, this basic cycle end entry is discarded and the following entries are read.</p>
EC	[31:28]	rw	<p>Entry Code The entry code EC = 0111_B defines this scheduler memory entry as a basic cycle end entry.</p>
0	[26:16]	r	<p>Reserved; read as 0; should be written with 0. Bits are “don’t care” for scheduler operation when EC = 0111_B.</p>

Note: The BCE determines the end of the scheduler entries in the scheduler memory when the system is normally synchronized. When the time mark of this entry is reached, the TTCAN node will automatically enter the Configuration Mode. It will continue with the first time mark when a reference message has been correctly transferred on the CAN bus.

While the system is in the synchronization state, a BCE entry with GM = 1 is also taken into account.

22.6.2.3 End of Scheduler Memory Entry

The end of scheduler memory entry EOS is defined by entry codes EC = 0000_B or EC = 1XXX_B (see [Table 22-13](#)). An EOS immediately stops the reading of entries in the scheduler memory and sets the TTCAN node into Configuration Mode.

EOS can be regarded as a kind of emergency stop entry. It should not be used to control the scheduler itself but represents a security mechanism for the case that a scheduler entry does not show a valid EC bit field.

Controller Area Network (MultiCAN) Controller

22.6.3 Setup of the Scheduler Entries

The entries in the scheduler memory can be set up only when all bits NCRx.CCE of the TTCAN nodes sharing the scheduler memory are set at the same time. Write actions to the scheduler memory while not all corresponding bits NCRx.CCE = 1 are not taken into account. The scheduler entries can always be read out for verification purposes.

22.6.4 Reading the Scheduler Entries

After a time mark has been reached, the scheduler instructions for the following time window are read by the scheduler. This “collected” information can be read out from the scheduler timing status and from the scheduler instruction status register. The information becomes valid when the next time mark is reached. The information collected between time mark n-1 and time mark n becomes valid when the time mark n is reached.

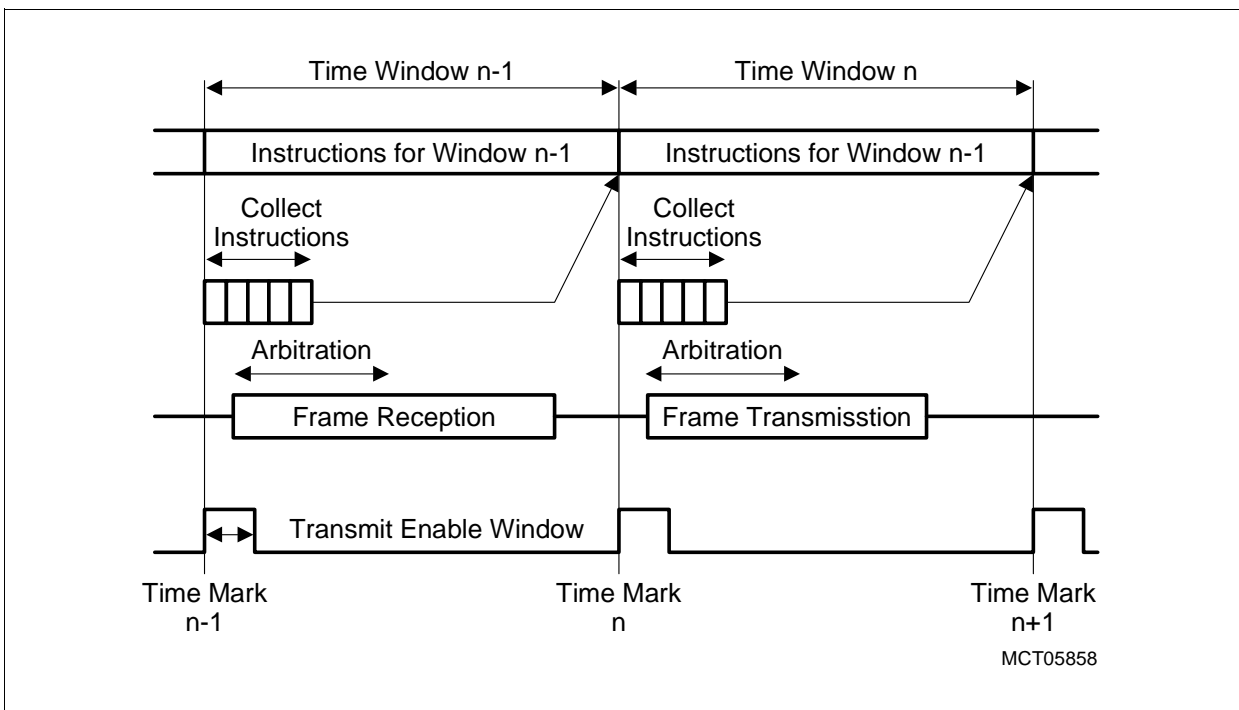


Figure 22-32 Collecting the Instructions

22.6.4.1 Instructions During a Basic Cycle

The handling of entries collected between the time marks n-1 and n is defined as follows:

- Time Mark Entry:
The time mark value is defining the compare value for the time mark n.
- Interrupt Entry:
Valid interrupt information (INP+ 4 enable bits) can generate an interrupt when the time mark n is reached, depending on the flags RECF and TRAF (they are

Controller Area Network (MultiCAN) Controller

automatically cleared when the time mark is reached). As a result, this interrupt indicates that a (no) message has been sent /received in the time window between the time marks n-1 and n.

- **Receive Control Entry:**
Valid receive control information (CHEN, RCEMSGNR) controls the storage of a message in the time window between the time marks n-1 and n.
- **Arbitration Entry:**
Valid arbitration information (ARBM) determines the behavior of the time window starting with the time mark n.
- **Transmit Control Entry:**
Valid transmit control information (TREN, ALTMSG, TCEMSGNR) defines a message to be transmitted after the time mark n. The transmission start is possible while the transmit enable window is active for an exclusive window or for a short (single) arbitration window or for the end of a merged arbitration window. For a merged (long) arbitration window, the transmission can start during the complete time window.
- **Reference Message Entry:**
not used in this time window
- **Basic Cycle End Entry:**
not used in this time window

The TMV value collected after time mark n-1 will become the new compare value for the time mark n.

The value of CYCTMR.CSM equals the time mark number of the last time mark reached (CSM = n after time mark n has been reached). The value of CYCTMR.BCC is number of the current basic cycle. This information is needed to correctly set up the scheduler entries (bit fields CYCLE, MCYCLE) and the message objects (bit fields CYCLE, MCYCLE, COLUMN, MCOLUMN).

22.6.4.2 Instructions at the End of a Basic Cycle

The collection of instructions for the last time window of a basic cycle starts when the time mark before is reached and stops when the entry TME2 is found. Handling of the entries collected during the last time window of a basic cycle:

- **Time Mark Entry:**
After a valid BCE has been read, the first time mark TME1 is read and its value is defining the compare value for the time mark 1 (first time mark after the reference message = start of a new basic cycle).
- **Interrupt Entry:**
Valid interrupt information (INP+ 4 enable bits) collected during the last time window of a basic cycle can generate an interrupt when the time mark 1 is reached, depending on the flags RECF and TRAF (they are automatically cleared when the time mark is reached). These flags do not apply to the reference message (the

Controller Area Network (MultiCAN) Controller

transfer of a reference message does not modify these flags). As a result, this interrupt indicates that a (no) message has been sent /received in the last time window of the basic cycle. The ICE for the last time window of a basic cycle must be written into the scheduler memory after the entry TME1.

- **Receive Control Entry:**
Valid receive control information (CHEN, RCEMSGNR) collected during the last time window of a basic cycle controls the storage of a message in the last time window (not applicable to reference message). The RCE for the last time window of a basic cycle must be written into the scheduler memory after the entry TME1.
- **Arbitration Entry:**
Valid arbitration information (ARBM) determines the behavior of the time window starting with the time mark 1. This entry must be written into the scheduler memory after the entry TME1.
- **Transmit Control Entry:**
Valid transmit control information (TREN, ALTMSG, TCEMSGNR) defines a message to be transmitted after the time mark 1. The transmission start is possible while the transmit enable window is active for an exclusive window or for a short (single) arbitration window. For a merged (long) arbitration window, the transmission can start during the complete time window. This entry must be written into the scheduler memory after the entry TME1.
- **Reference Message Entry:**
The time mark value of a valid RME is defining the compare value for the reference time mark. When this reference time mark plus the reference trigger offset is reached, a reference message will be sent out (depending on the gap state of the system). This entry must be written into the scheduler memory before a valid entry BCE.
- **Basic Cycle End Entry:**
The time mark value of a valid BCE is defining the compare value for the watch trigger event. When this time value is reached, a watch trigger event can be generated (depending on the gap state of the system). A valid BCE always finishes the scheduler entries for each TTCAN node.

Before the reference message has been correctly received:

The value of CYCTMR.CSM equals the time mark number of the last time mark reached (CSM = n after time mark n has been reached). The value of CYCTMR.BCC is number of the current basic cycle. This information is needed to correctly set up the scheduler entries RCE and ICE (bit fields CYCLE, MCYCLE).

After the reference message has been correctly received, but the time mark 1 has not yet been reached:

The value of CYCTMR.CSM is 0. The value of CYCTMR.BCC is number of the new basic cycle. This information is needed to correctly set up the scheduler entries ARBE and TCE (bit fields CYCLE, MCYCLE) and the message objects (bit fields CYCLE, MCYCLE, COLUMN, MCOLUMN).

Controller Area Network (MultiCAN) Controller

As a result, the ICE and the RCE describing the reception / transmission of a message in the last time window of a basic cycle lead to actions (interrupts, etc.) when time mark 1 is reached.

22.6.5 Scheduler Instruction Sequence

22.6.5.1 BCC and CSM

In order to set up the scheduler instructions correctly, the values of CSM and BCC in register CYCTMR must be respected carefully, especially for the last time window of a basic cycle.

In order to set up the scheduler instructions correctly, the values of CSM and BCC in register CYCTMR must be respected carefully, especially for the last time window of a basic cycle. When leaving the Configuration Mode, BCC and CSM are 0 and the scheduler starts with time mark 1. The values for BCC and CSM in [Figure 22-23](#) represent the internal values that are updated after the correct transfer of a reference message.

The message triggered by the last time mark in a basic cycle is the reference mark (see RME description). In [Figure 22-33](#), this corresponds to the message after the time window n. The reference time mark follows after the time window n and the reference message starts a new basic cycle.

Controller Area Network (MultiCAN) Controller

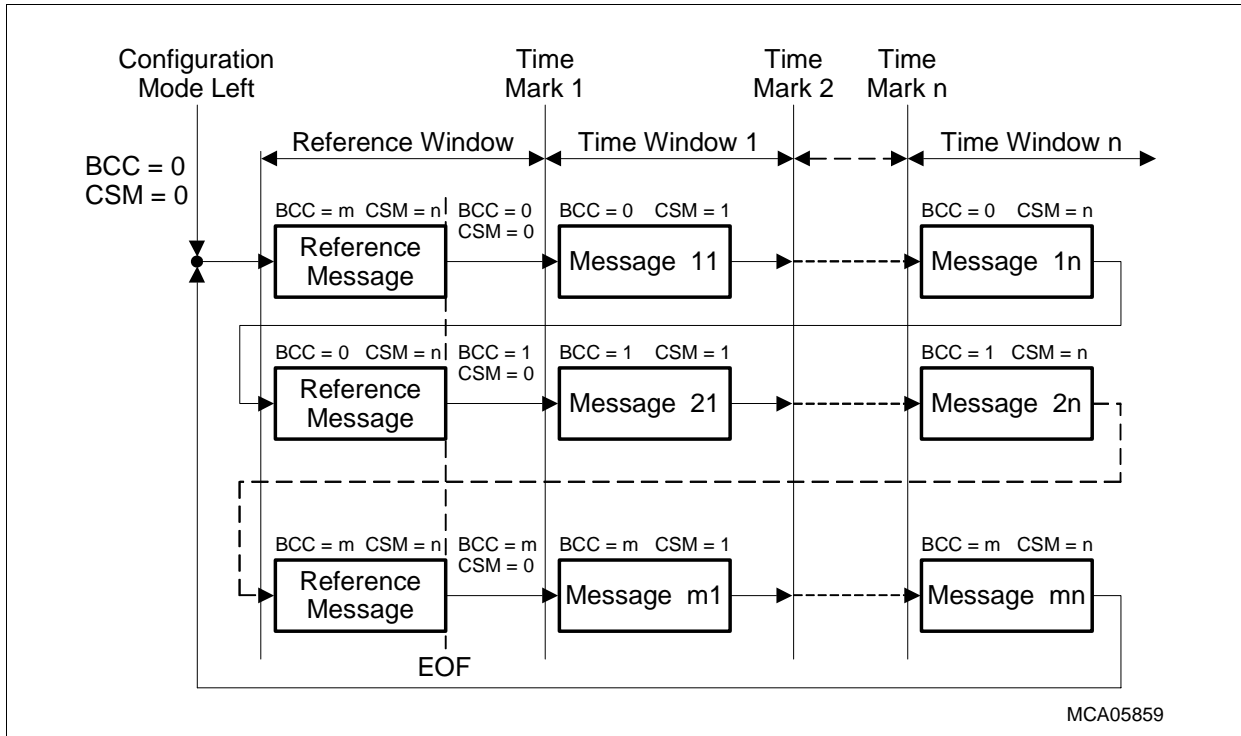


Figure 22-33 BCC and CSM in a Matrix Cycle

The cycle time of each basic cycle starts with 0 (virtual value). The start of a new basic cycle can only be detected after the correct transfer of a reference message. As a result, the cycle time value can be considered as valid after the reference message has been correctly transferred.

22.6.5.2 General Instruction Sequence Rules

The settings of already read scheduler instructions can be overwritten by following valid scheduler instructions. The following order of scheduler entry types must be respected to set up the scheduler correctly (starting with the first):

- TME then RCE or ICE or TCE or ARBE
- RME then BCE (for time masters)
- BCE (for slave devices)

The scheduler entries must always be closed with a BCE with GM = 0.

For time masters, the following sequence can be set up to close the scheduler entries:

- RME (GM = 1) then BCE (GM = 1) then RME (GM = 0) then BCE (GM = 0)
(the entries RME(GM = 0) and BCE(GM = 0) are mandatory)

If the system is in-a-gap, the first RME and BCE entries (both with GM = 1) are not taken into account. With these entries, the standard (not in-a-gap) timing values can be adjusted. A second RME (with GM = 0) can be set up to send an emergency reference message while the system is in-a-gap and the synchronization event takes too long. The

Controller Area Network (MultiCAN) Controller

scheduler entries are then closed again with a BCE with GM = 0 (e.g. with a longer timing value as a time-out criteria for a missing synchronization event).

22.6.5.3 Scheduler Sequence Example

The values given for CSM and BCC (both bit fields are located in register CYCTMR) lead to valid scheduler entries RCE, TCE, ICE and ARBE (bit fields CYCLE, MCYCLE) and the transmit control of the message objects (bit fields CYCLE, MCYCLE, COLUMN, MCOLUMN). In the case that the programmed values for CYCLE, MCYCLE, etc. do not match the given values of CSM and BCC, the respective entry is considered invalid and its information is not taken into account for the corresponding time window. The following example shows a scheduler instruction sequence for the basic cycle m with n time mark entries:

- TME1
- RCE, ICE (CSM = n , BCC = $m-1$ or CSM = BCC = 0 when the Configuration Mode is left)
- TCE, ARBE (CSM = 0, BCC = m)

- TME2
- RCE, ICE (CSM = 1, BCC = m)
- TCE, ARBE (CSM = 1, BCC = m)
- TME3
- RCE, ICE (CSM = 2, BCC = m)
- TCE, ARBE (CSM = 2, BCC = m)

- other time marks and instructions

- TME n
- RCE, ICE (CSM = $n-1$, BCC = m)
- TCE, ARBE (CSM = $n-1$, BCC = m)

- RME (GM = 1, only for time masters)
- BCE (GM = 1, for time masters and slaves)

- RME (GM = 0, only for time masters)
- BCE (GM = 0, for time masters and slaves)

22.7 TTCAN Operation

22.7.1 Configuration

After a reset operation, the TTCAN extension must be configured. The Configuration Mode is entered automatically after reset or can be initiated through software by writing $TTFMR.CFGMEL = 01_B$. The Configuration Mode can be left only by software by writing $TTFMR.CFGMEL = 10_B$. The status flag $TTSR.CFGM$ indicates whether or not the Configuration Mode is active. This flag is automatically set when the corresponding CAN node turns off its clock (disable request).

In Configuration Mode of the TTCAN node (CAN node 0 in TC1796), the following actions must be performed, as indicated, by software or hardware:

For all nodes:

- The local time, the global time, and the cycle time are set to 0 (hardware).
- Transmission or reception of messages of TTCAN node is not possible, because the results of the acceptance filtering are not enabled (hardware).
- An appropriate TUR value must be written to bit field $TURR.TURADJ$ (software). This value is automatically transferred to $TURR.TUR$ (hardware).
- The scheduler memory entries must be initialized (software).
- The TTCAN control information (ID of reference message, etc.) and the TTCAN node itself must be set up completely and enabled for CAN message transfer (software).
- After the complete configuration (software), $TTFMR.CFGMEL$ must be set to 10_B (software).
- The local time starts after leaving the Configuration Mode (hardware).
- The synchronization phase is entered automatically when the Configuration Mode is left (hardware). This is indicated by bit field $TTSR.SYNCS = 01_B$ (hardware).
- For time masters, the transmission of the reference messages is scheduled as in a gap while the TTCAN node is in the “synchronizing” state. The scheduler entries RME with $GM = 1$ are taken into account only while the TTCAN node is in the “in schedule” state.
- For time masters and for time slaves, the scheduler entries BCE with $GM = 1$ are taken into account only while the TTCAN node is in the “in schedule” state.

22.7.2 Configuration Error

During the scheduler actions, some conditions lead to a configuration error:

- At the end of a basic cycle, a merged arbitration window is still open.
- An RME entry is found for a slave device.
- No time mark is found during instruction collection.
- An RCE, ICE, TCE or ARBE has been found before TME1 (after start or at the end of a basic cycle).

Controller Area Network (MultiCAN) Controller

- Another entry (except EOS) than an RME or a BCE has been found after the first RME.
- A reference message object is not valid when it is requested for transmission.

22.7.3 Synchronization Phase

When starting a TTCAN system (e.g. after reset), the entire system must be synchronized after completing the configuration phase. During the synchronization phase (indicated by bit field TTSR.SYNCS = 01_B), the following actions must be taken (or are taken automatically):

The Init_Watch_Trigger is taken into account until the first message is correctly received or transmitted. An Init_Watch_Trigger event is detected when the cycle time reaches the value of $2^{16} - 1$. This event is indicated by IWTE = 1 and it only leads to an interrupt.

The watch trigger value given by the basic cycle end entry is taken into account only after the first correct transfer of a message. When a watch trigger event occurs after transferring a message correctly on the bus, the transmission and the reception of Data Frames or Remote Frames are disabled (the CAN node is set to Configuration Mode).

During the synchronization phase, the scheduler entries RME or BCE with GM = 1 are not taken into account.

22.7.4 Time Masters

22.7.4.1 State of a Time Master

A potential time master is a device that can transmit a reference message. A backup time master is a potential time master that has received a reference message that was not its own. The current time master is the device that has successfully sent its own reference message. There is only one current time master in a TTCAN system.

If the current time master receives a reference message that is not its own, then it becomes a backup time master. If a backup time master has successfully transmitted its own reference message, it becomes the new current time master.

When a priority conflict between the TTCAN node and the actual time master is detected, the value of RTO is automatically modified (decrement by 1). This means that the priority of the actual time master (given by bits ID[2:0]) of the latest received reference message is lower than the programmed TMPRIO value.

22.7.4.2 Strictly Time-Triggered Behavior

A strictly time-triggered behavior (gaps are not allowed) can be achieved if the scheduler entries RME and BCE with GM = 0 are only used. In this case, the gap condition is “ignored” for the transfer of a reference message. In order to avoid unintentional

Controller Area Network (MultiCAN) Controller

transfers of reference messages, the external trigger generation should be switched off in this case.

22.7.5 Error Handling

The TTCAN error handling distinguishes among four levels of error severity:

- S0:
No Error: Normal functionality
- S1:
Warning: Only notification of the application by an interrupt (ERRS1).
Source: Scheduling Error 1 ($MSC_{max} - MSC_{min} > 2$ or when MSC of a receive object equal 7), Tx_Underflow
- S2:
Error: Notification of the application by an interrupt (ERRS2). All transmissions are disabled (except for reference messages), TTCFGR.RTO is set to 127.
Source: Scheduling Error 2 (MSC of a transmit object reaches 7), Tx_overflow
Impact: The transmit enable window will not be opened (for exclusive and for arbitration windows).
- S3:
Severe Error: Notification of the application by an interrupt (ERRS3). All CAN bus actions are stopped (no dominant values are transmitted on the bus). The configuration phase is entered automatically (CFGM is set).
Source: Application Watchdog, Bus Off, Config Error, Watch Trigger Event
Impact: Bit INIT of the CAN node will be set (the CAN node stops actions on the bus).

Note: Any change of the error state can generate an interrupt.

22.7.6 Application Watchdog

The application watchdog provides the possibility of checking for the TTCAN module if the main system is still running. The application watchdog counter counts in steps of 256 NTUs.

Each time 256 NTUs have elapsed, the value AWDR.AWDV is decremented by one. If the value 0 is reached after decrementing, an S3 error is signaled by bit AWDERR. The falling edge of bit LTR.LT.7 (transition from 1 to 0) indicates that the time of 256 NTUs has elapsed and the application watch dog value is decremented.

The application watchdog is serviced by the program by writing a new value to the bit field AWDR.AWDE.

Controller Area Network (MultiCAN) Controller**22.7.7 MSC Handling**

The MSC bit field is updated only for those message objects that are indicated by the valid RCE or TCE entry while the corresponding bit MSGVAL is set. The message number taken into account for an update is defined in the respective bit fields RCMSGNR or TCEMSGNR.

Checking for the receive/transmit condition takes place when the end of the transmit enable window of the following time marks is reached. Because the transmit enable window length is known after each time mark (even if no message shall be sent out), this point in time can be used for the receive and the transmit check.

This feature makes it possible to have a positive check result even if the message in the preceding time window finishes during the transmit enable window.

A transmission attempt is defined as failed if a message object valid for transmission could not be transferred on the CAN bus. A transmit trigger being active without a valid message object being found is not considered as a transmission attempt.

Controller Area Network (MultiCAN) Controller

22.7.8 TTCAN Interrupt Control

Figure 22-34 and Figure 22-35 show the configurations of the three TTCAN interrupts.

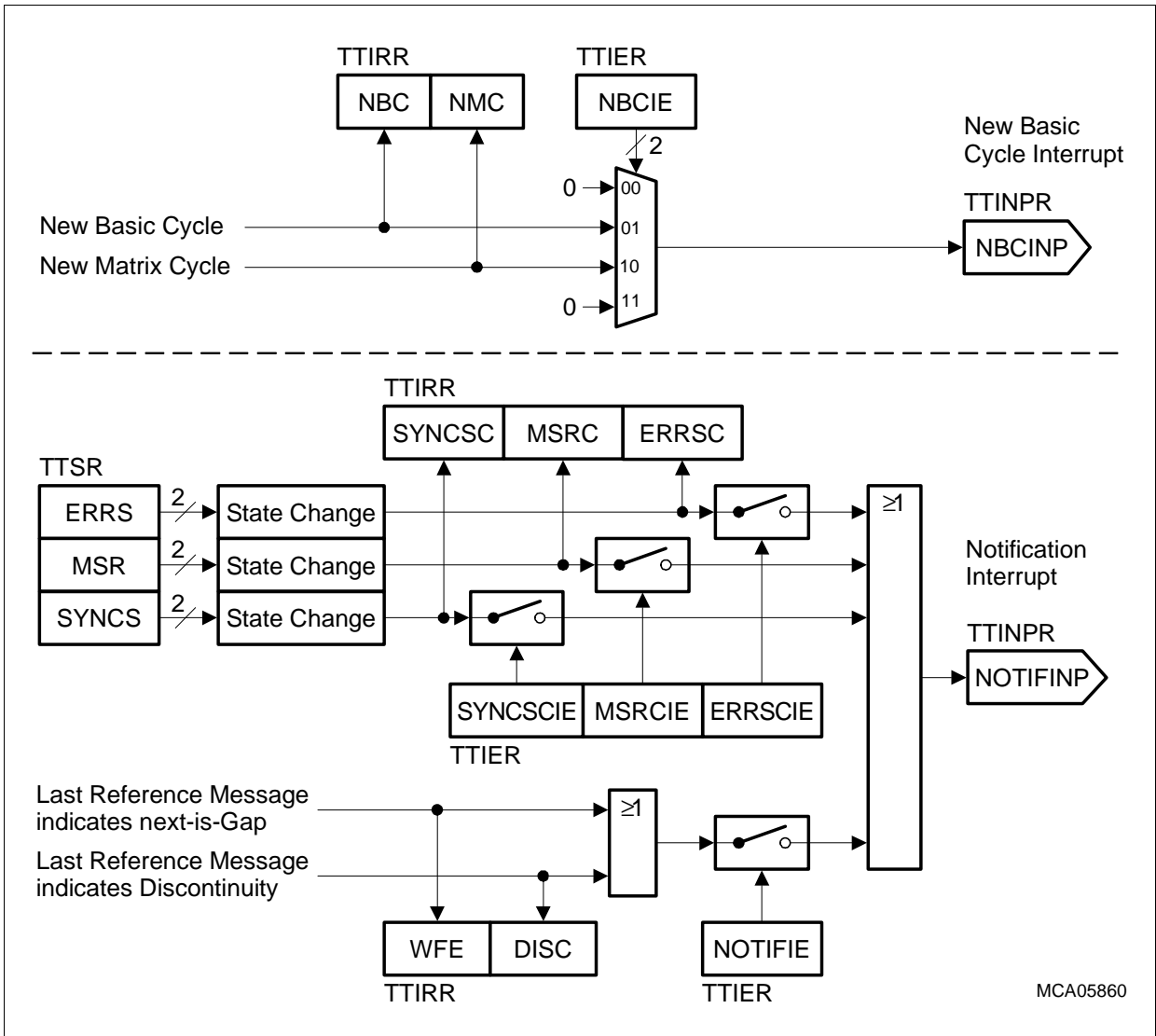


Figure 22-34 New Basic Cycle and Notification Interrupt Structure

Controller Area Network (MultiCAN) Controller

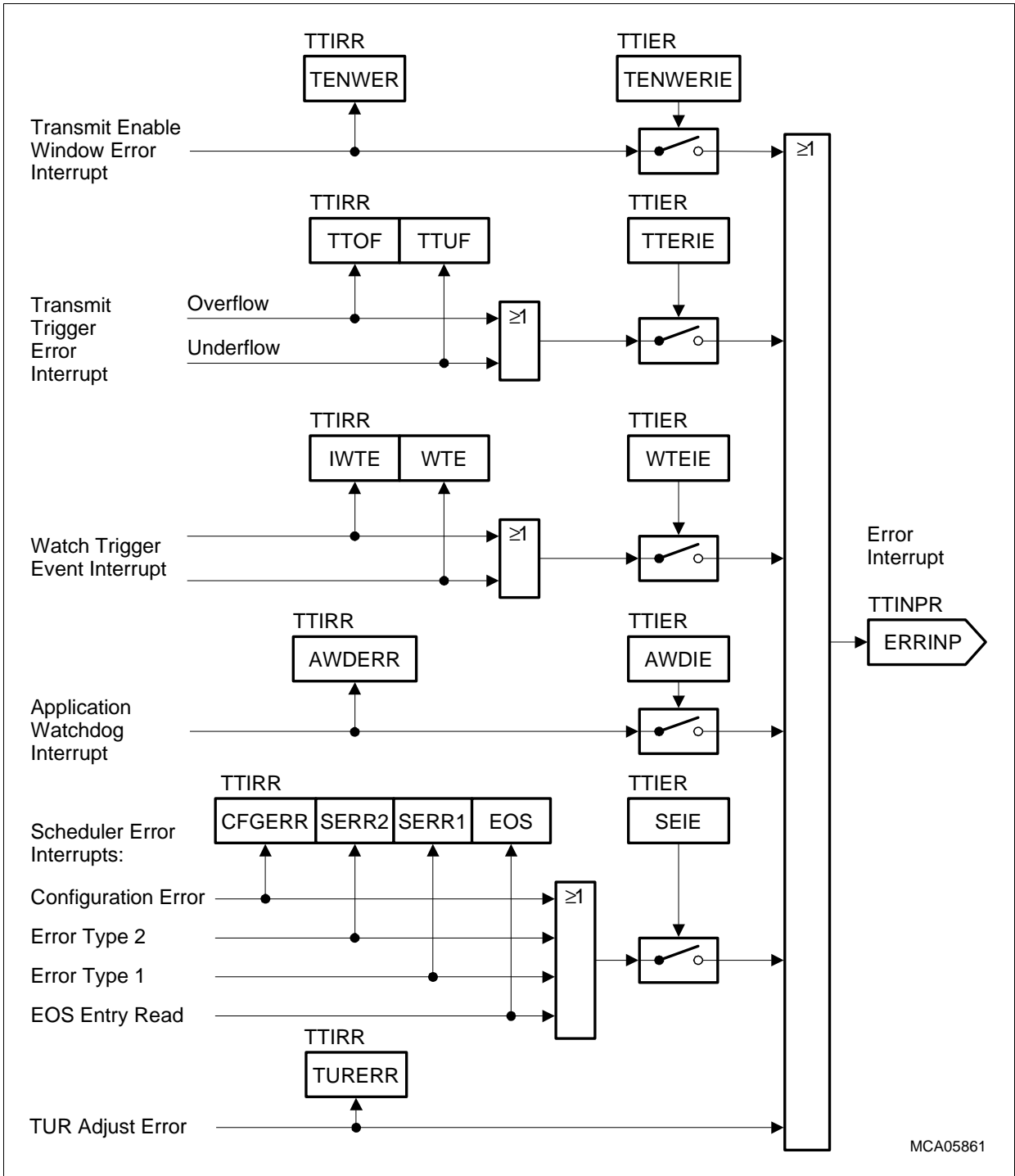


Figure 22-35 Error Interrupt Structure

See also **“Interrupt Control”** on Page 22-210 for further processing of the TTCAN interrupts.

Controller Area Network (MultiCAN) Controller

22.8 TTCAN Registers

All TTCAN register names described in this section are referenced by the module name prefix “CAN_” in other parts of the TC1796 User’s Manual.

TTCAN Register Overview

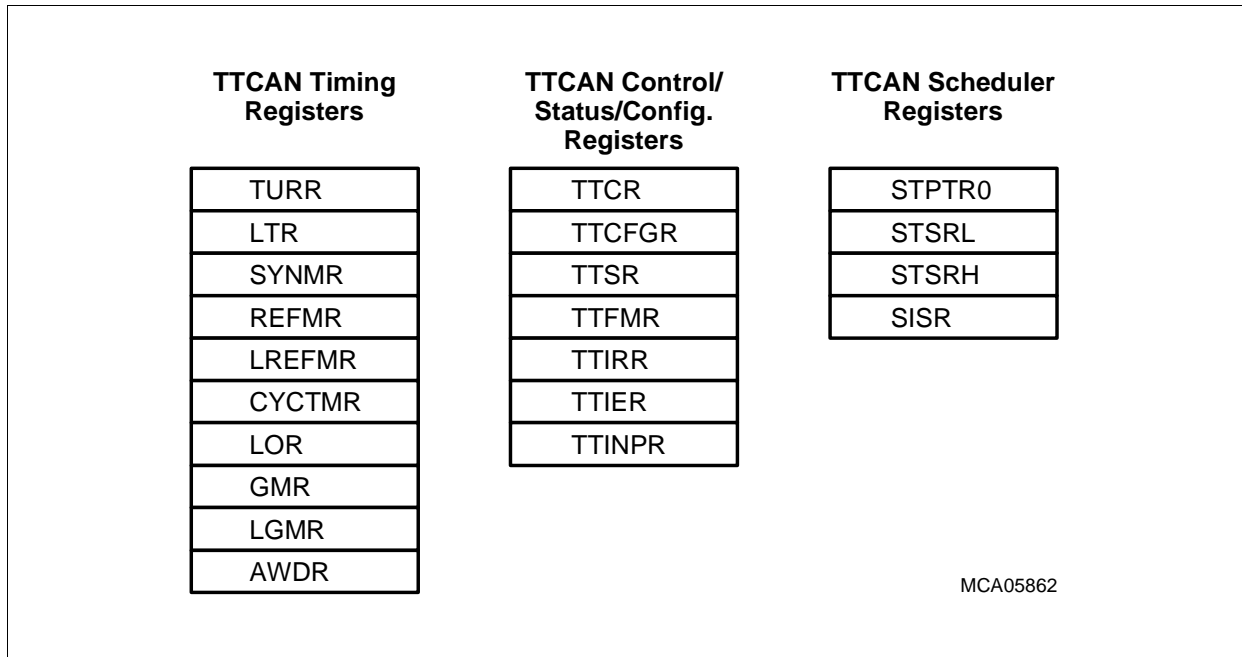


Figure 22-36 TTCAN Registers

Table 22-14 TTCAN Registers

Register Short Name	Register Long Name	Offset Address	Description see
LTR	Local Time Register	280 _H	Page 22-159
SYNMR	Synchronization Mark Register	284 _H	Page 22-160
REFMR	Reference Mark Register	288 _H	Page 22-161
LREFMR	Last Reference Mark Register	28C _H	Page 22-162
TURR	Time Unit Ratio Register	290 _H	Page 22-157
CYCTMR	Cycle Time Register	294 _H	Page 22-163
LOR	Local Offset Register	298 _H	Page 22-164
GMR	Global Mark Register	29C _H	Page 22-166
LGMR	Last Global Mark Register	2A0 _H	Page 22-167
AWDR	Application Watchdog Register	2A4 _H	Page 22-168

Controller Area Network (MultiCAN) Controller

Table 22-14 TTCAN Registers (cont'd)

Register Short Name	Register Long Name	Offset Address	Description see
TTCR	Time Trigger Control Register	2C0 _H	Page 22-169
TTCFGR	Time Trigger Configuration Register	2C4 _H	Page 22-173
TTSR	Time Trigger Status Register	2C8 _H	Page 22-175
TTFMR	Time Trigger Flag Modification Register	2CC _H	Page 22-180
TTIRR	Time Trigger Interrupt Request Register	2D0 _H	Page 22-182
TTIER	Time Trigger Interrupt Enable Register	2D4 _H	Page 22-186
TTINPR	Time Trigger Interrupt Node Pointer Register	2D8 _H	Page 22-190
STSRL	Scheduler Timing Status Register Low	2F0 _H	Page 22-193
STSRH	Scheduler Timing Status Register High	2F4 _H	Page 22-194
SISR	Scheduler Instruction Status Register	2F8 _H	Page 22-195
STPTR0	Scheduler Start Pointer Node 0 Register	17FC _H	Page 22-192

Controller Area Network (MultiCAN) Controller

Figure 22-37 shows the TTCAN register address map.

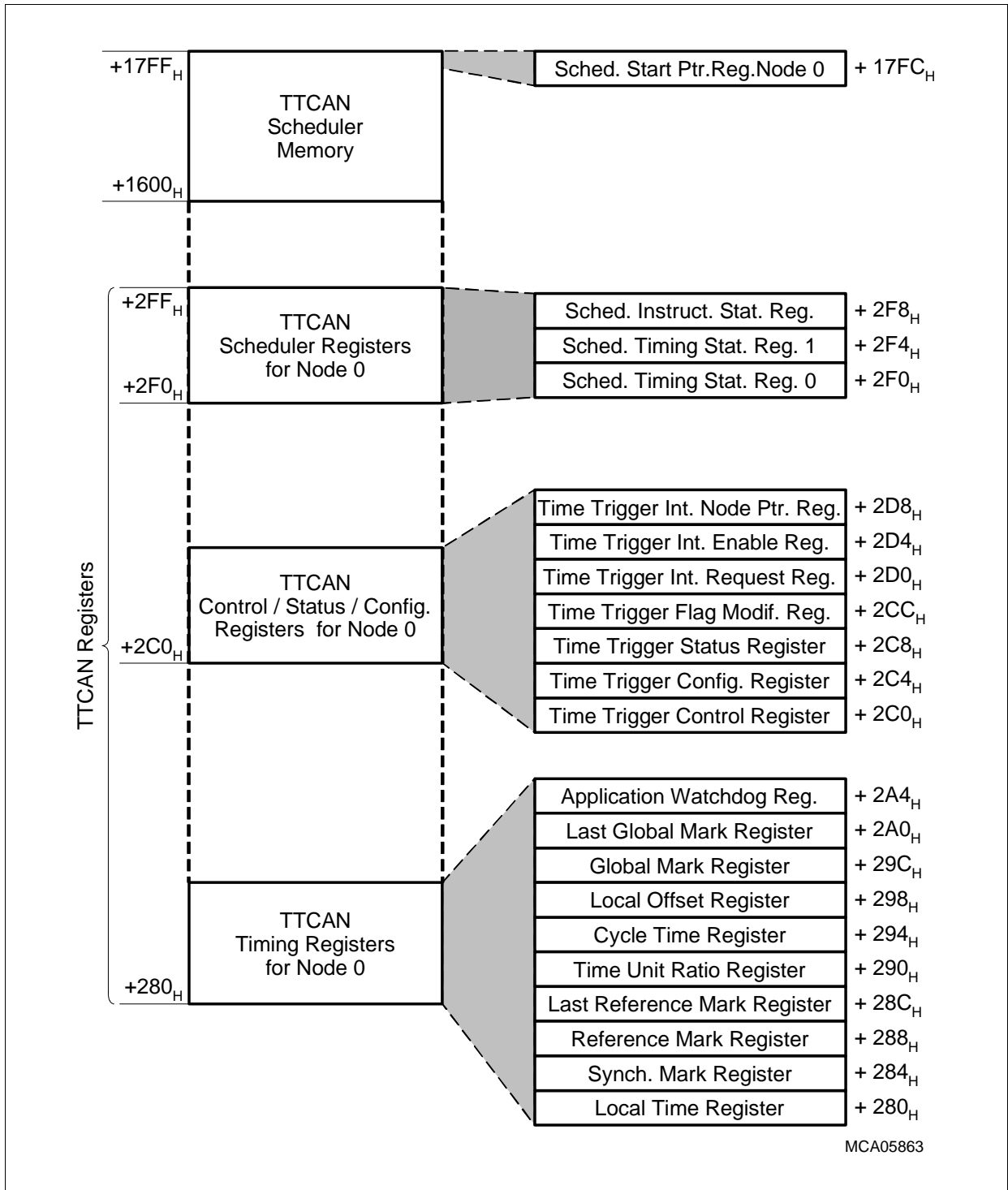


Figure 22-37 TTCAN Register Address Map

Controller Area Network (MultiCAN) Controller

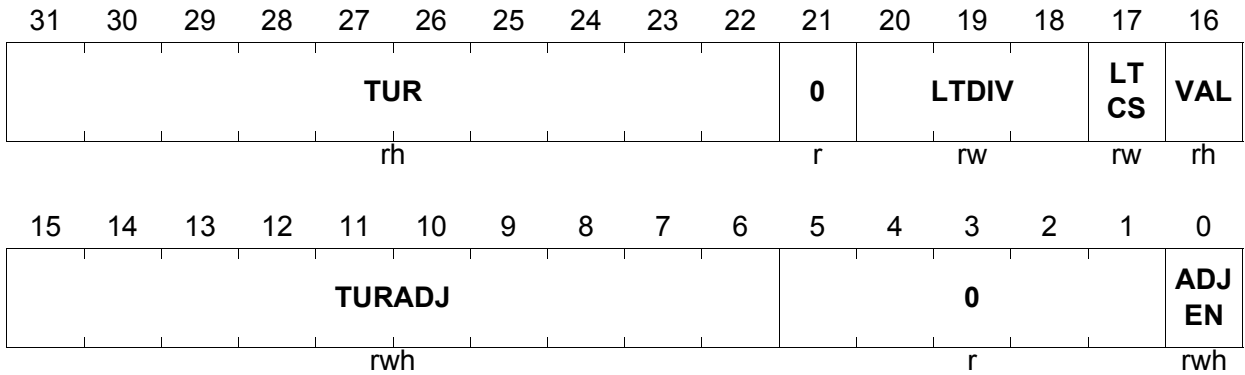
22.8.1 TTCAN Timing Registers

TURR

Time Unit Ratio Register

(290_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
ADJEN	0	rwh	<p>Adjust Enable</p> <p>This bit enables automatic calculation of a new TURADJ value when a new reference message has been received (not for the current time master). A time unit ratio adjust error interrupt will be generated and bit ADJEN will be automatically cleared when an overflow or an underflow of the automatically calculated value occurs.</p> <p>0_B The automatic TUR calculation is disabled. The new value for time unit ratio for TURADJ must be calculated and written by software.</p> <p>1_B The automatic TUR calculation is enabled. After receiving a reference message, a new value for TURADJ is calculated by hardware. The calculated value is checked for overflow or underflow.</p>
TURADJ	[15:6]	rwh	<p>Time Unit Ratio Adjust</p> <p>This bit field holds the time unit ratio value that will be used for the next basic cycle. In the case of an automatic time unit ratio calculation with an underflow (overflow), TURADJ is written with all 0s (1s), ADJEN is cleared and a time unit ratio adjust error interrupt is generated. A hardware calculated TURADJ value can be overwritten by software.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
VAL	16	rh	<p>Valid</p> <p>This bit indicates that a new value of TURADJ is available. VAL becomes set when the hardware has finished its automatic calculation (with a correct result without overflow or underflow) or when the software writes to TURADJ. An automatic calculation does not take place while VAL = 1. VAL is cleared when the value of TURADJ is copied to TUR.</p> <p>0_B The value of TURADJ has not been updated and no automatic update of TUR by the value of TURADJ takes place.</p> <p>1_B The value of TURADJ has been updated by software or by hardware and it will be copied to TUR at the start of the next basic cycle.</p>
LTCS	17	rw	<p>Local Time Clock Source</p> <p>This bit determines the clock source for the local time generation (addition of TUR to LT, LTFR) for TTCAN level 2.</p> <p>0_B A new local time value is generated with each time quantum t_q of the corresponding CAN node (depending on the CAN bit timing).</p> <p>1_B The local time generation is based on the CAN module clock f_{CAN} (independent from the CAN bit timing).</p> <p>The update rate t_{upd} is based on the divider factor given by the bit field LTDIV according to:</p> $t_{upd} = 2^{LTDIV} / f_{CAN}$
LTDIV	[20:18]	rw	<p>Local Time Divider</p> <p>This bit field determines the divider factor for the local time generation (if f_{CAN} is selected by LTCS = 1). The divider factor is given by 2^{LTDIV}.</p> <p>000_B Divider factor = 1 selected.</p> <p>001_B Divider factor = 2 selected.</p> <p>..._B ...</p> <p>111_B Divider factor = 128 selected.</p>
TUR	[31:22]	rh	<p>TUR</p> <p>This bit field contains the currently active time unit ratio value.</p>
0	[5:1], 21	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

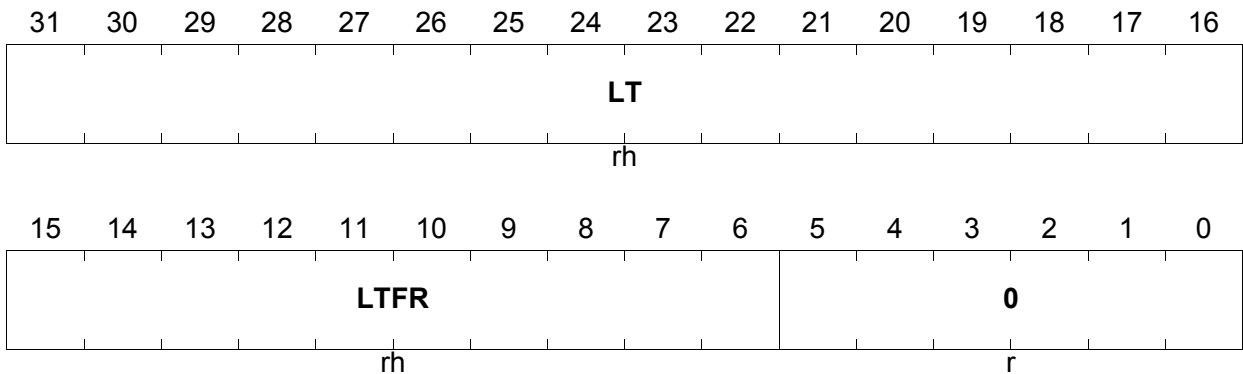
Controller Area Network (MultiCAN) Controller

LTR

Local Time Register

(280_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
LTFR	[15:6]	rh	Local Time Fraction This bit field contains the fractional part of the NTU counter (TTCAN level 2 only). In the case of an overflow after the addition of the TUR value, the value of LT is incremented by one.
LT	[31:16]	rh	Local Time This bit field contains the integer part of the NTU counter. In TTCAN level 1, it is incremented with each CAN bit time, in level 2 it is incremented each time the fractional part LTFR has an overflow.
0	[5:0]	r	Reserved Read as 0; should be written with 0.

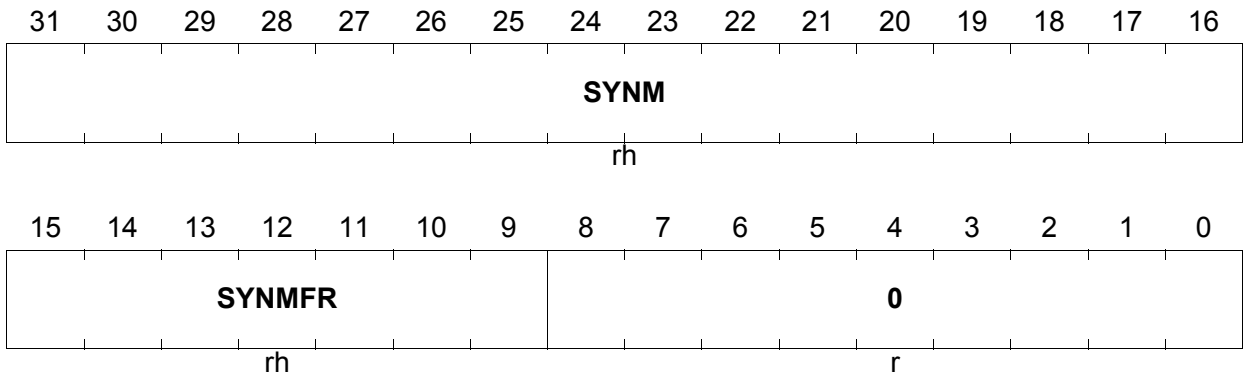
Controller Area Network (MultiCAN) Controller

SYNMR

Synchronization Mark Register

(284_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SYNMFR	[15:9]	rh	Synchronization Mark Fraction This bit field contains the fractional part of the synchronization mark. SYNMFR is the bit field LTR.LTFR captured with the frame synchronization pulse.
SYNM	[31:16]	rh	Synchronization Mark This bit field contains the integer part of the synchronization mark. SYNM is the bit field LTR.LT captured with the frame synchronization pulse.
0	[8:0]	r	Reserved Read as 0; should be written with 0.

Note: If the CAN node is time master, the contents of register SYNMR are used as timing data transferred in the reference message.

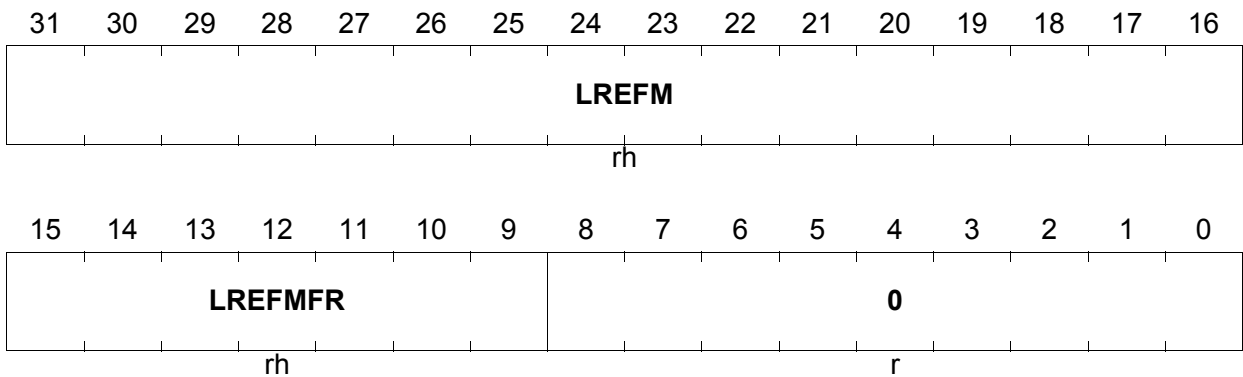
Controller Area Network (MultiCAN) Controller

LREFMR

Last Reference Mark Register

(28C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
LREFMFR	[15:9]	rh	Last Reference Mark Fraction This bit field contains the fractional part of the last reference mark. LREFMFR is the bit field REFMR.REFMFR captured with the correct end of the reference message.
LREFM	[31:16]	rh	Last Reference Mark This bit field contains the integer part of the last reference mark. LREFM is the bit field REFMR.REFM captured with the correct end of the reference message.
0	[8:0]	r	Reserved Read as 0; should be written with 0.

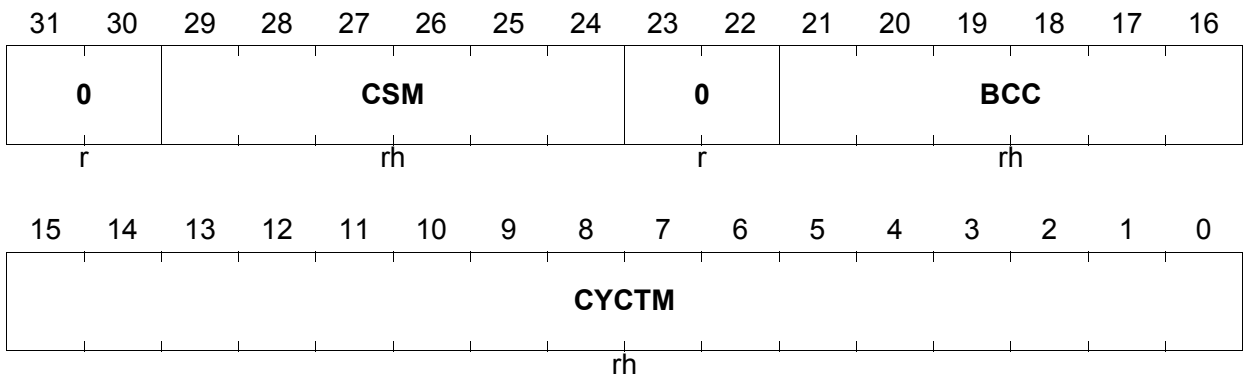
Controller Area Network (MultiCAN) Controller

CYCTMR

Cycle Time Register

(294_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CYCTM	[15:0]	rh	Cycle Time The cycle time indicates the time already elapsed in the current basic cycle. It is calculated by LTR.LT - REFMR.REFM. In the case of a negative result (overflow of LTR.LT), the result is corrected.
BCC	[21:16]	rh	Basic Cycle Count This bit field indicates the number of the current basic cycle. It is incremented after each correctly transferred reference message.
CSM	[29:24]	rh	Column of System Matrix This bit field indicates the number of the current column of the system matrix. It is incremented when the current cycle time becomes equal to the stored time mark (indicating the start of the next column).
0	[23:22], [31:30]	r	Reserved Read as 0; should be written with 0.

Note: With a reference message, the counting of the columns restarts at 0.

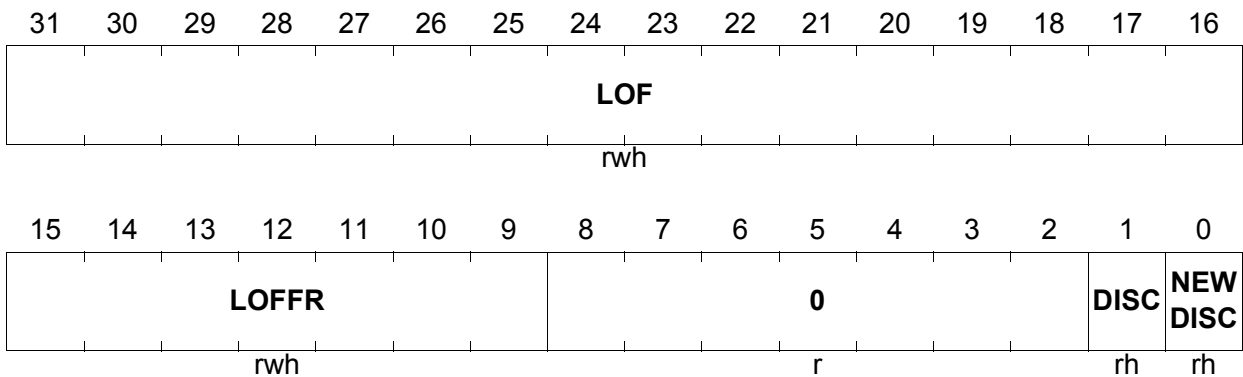
Controller Area Network (MultiCAN) Controller

LOR

Local Offset Register

(298_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
NEWDISC	0	rh	<p>New Discontinuity Bit</p> <p>This bit indicates that the register LOR has been the target of a write access for a time master (and that the contents of LOF or LOFFR could have been changed). This bit indicates that the next reference message will be sent with a discontinuity in the reference mark. It is set automatically when a write action to LOR is detected. It is cleared when the local time is captured (at SOF) for the transmission of the reference message.</p> <p>0_B A write access to LOR has not occurred. 1_B A write access to LOR has occurred.</p>
DISC	1	rh	<p>Discontinuity Bit</p> <p>This bit contains the DISC bit of the reference message that is sent out (taken into account for a time master). This bit is set if the NEWDISC bit is 1 when the global time is captured (at SOF) for the transmission of the reference message. It is cleared when the reference message has been transferred correctly.</p> <p>0_B The DISC bit in the reference message of the current time master is 0. 1_B The DISC bit in the reference message of the current time master is 1.</p>
LOFFR	[15:9]	rwh	<p>Local Offset Fraction</p> <p>This bit field contains the fractional part of the local offset.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
LOF	[31:16]	rwh	<p>Local Offset This bit field contains the integer part of the local offset.</p> <p>Time master (transmitting reference messages): The sum of the local offset and the local time is stored as global time in register GTR.</p> <p>Not time master (receiving reference messages): The local offset is the received global time information minus the local reference mark (in REFMR).</p>
0	[8:2]	r	<p>Reserved Read as 0; should be written with 0.</p>

Note: Register LOR can be written by software only if the CAN node is the current time master or in Configuration Mode; otherwise, the write action is not taken into account. Because the written local offset is only taken into account for the transmission of the reference message, bit DISC is set automatically.

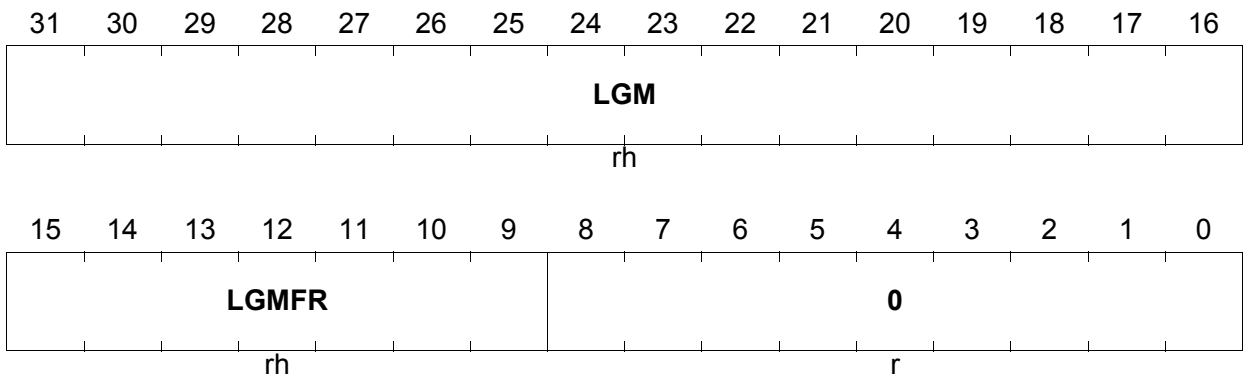
Controller Area Network (MultiCAN) Controller

LGMR

Last Global Mark Register

(2A0_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
LGMFR	[15:9]	rh	Last Global Mark Fraction This bit field contains the value of GMFR of the last reference mark.
LGM	[31:16]	rh	Last Global Mark This bit field contains the value of GM of the last reference mark.
0	[8:0]	r	Reserved Read as 0; should be written with 0.

Note: The difference between the actual global mark and the last one can be used to determine the value required for the T_{URR}.T_{UR} update (not for the actual time master).

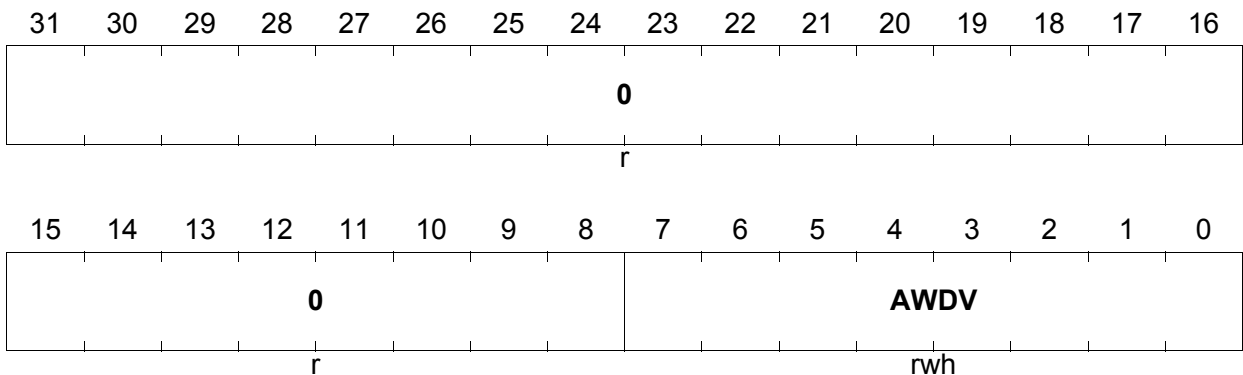
Controller Area Network (MultiCAN) Controller

AWDR

Application Watchdog Register

(2A4_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
AWDV	[7:0]	rwh	<p>Application Watchdog Value</p> <p>This bit field contains the current value of the application watchdog. The falling edge of bit LTR.LT[7] (1-to-0 transition) indicates that the time of 256 NTUs has elapsed and the application watchdog value is automatically decremented by 1. If the value 0 is reached after decrementing, an S3 error is signaled by bit TTIRR.AWDERR. AWDV is not decremented below 0.</p> <p>The application watchdog is serviced by the program by writing a new value to the bit field AWDV. In the case of a collision between the write by software and the automatic decrement, the write is taken into account.</p> <p>Due to the fact that the counter counting the NTU (register LTR) is not reset when the application watchdog is serviced, the total time between the write to AWDV and the signaling of an S3 error has an uncertainty of one step of AWDV.</p>
0	[31:8]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

The application watchdog event occurs when the value of AWDV is decremented and reaches zero. When writing 0 to AWDV by software, the application watchdog is switched off without generating an application watchdog event.

Controller Area Network (MultiCAN) Controller

22.8.2 TTCAN Control / Status / Configuration Registers

TTCR

Time Trigger Control Register (2C0_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RMDLC				TENW				0		CYCLE					
rw				rw				r		rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		TMPRIO			0		TT LVL	ETM	ETSSEL			ETESEL		TTM	
r		rw			r		rw	rw	rw			rw		rw	

Field	Bits	Type	Description
TTM	[1:0]	rw	<p>Time Trigger Mode</p> <p>This bit determines the behavior of the TTCAN node concerning the time trigger and the time master functionality.</p> <p>00_B CAN node is disabled for TTCAN operation and operates in event triggered mode (other settings for TTCAN are inactive). The reference message object operates for reception or transmission as any other message object.</p> <p>01_B CAN node is enabled for TTCAN operation as a receiving device. Reference messages cannot be transmitted by the CAN node.</p> <p>10_B CAN node is enabled for TTCAN operation as the actual or a potential time master. The CAN node is able to transmit the reference message.</p> <p>11_B Reserved; do not use this combination.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
ETESEL	[3:2]	rw	<p>External Trigger Event Selection</p> <p>This bit field determines the source of the external trigger event that can be used to trigger a transmission of the reference message. The transmission of the reference message will be triggered only when the external event is detected and the external trigger request bit TTSR.ETR is set (= 1).</p> <p>00_B Reserved; external trigger does not trigger the transmission of the reference message.</p> <p>01_B A negative edge at an external trigger input ECTTx (as selected by ETSSSEL) triggers the transmission of the reference message.</p> <p>10_B A positive edge at an external trigger input ECTTx (as selected by ETSSSEL) triggers the transmission of the reference message.</p> <p>11_B A negative or positive edge at an external trigger input ECTTx (as selected by ETSSSEL) triggers the transmission of the reference message.</p>
ETSSSEL	[6:4]	rw	<p>External Trigger Source Selection</p> <p>This bit fields selects the input source for the external reference message trigger.</p> <p>000_B No external trigger possible.</p> <p>001_B External trigger input line ECTT1 selected.</p> <p>010_B External trigger input line ECTT2 selected.</p> <p>010_B External trigger input line ECTT3 selected.</p> <p>100_B External trigger input line ECTT4 selected.</p> <p>101_B External trigger input line ECTT5 selected.</p> <p>110_B External trigger input line ECTT6 selected.</p> <p>111_B External trigger input line ECTT7 selected.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
ETM	7	rw	<p>External Trigger Mode</p> <p>This bit determines the external trigger mode. The transfer of a reference message is started if the selected trigger condition is met when the bus is idle after the end of the transmit enable window of the last time window of a basic cycle. The trigger event is stored in ETREV.</p> <p>0_B The trigger event itself is taken into account if it occurs after the end of the transmit enable window of the last time window of a basic cycle.</p> <p>1_B The trigger event stored in ETREV is taken into account after the end of the transmit enable window of the last time window of a basic cycle.</p>
TTLVL	8	rw	<p>Time Trigger Level</p> <p>This bit determines the level of the TTCAN functionality.</p> <p>0_B TTCAN level 1 is selected.</p> <p>1_B TTCAN level 2 is selected.</p>
TMPRIO	[14:12]	rw	<p>Time Master Priority</p> <p>This bit field determines the priority of the potential time master. This value will be used for the transmission of the ID bits [2:0] of the reference message. In the case that the TTCAN node loses arbitration against another reference mark on the bus or receives a reference mark while its own reference trigger is set, the reference trigger is reset and the received ID is stored in the reference message object's ID bit field.</p>
CYCLE	[21:16]	rw	<p>Basic Cycle Number</p> <p>This bit field determines the number of the last basic cycle in the matrix cycle.</p> <p>For time masters, the value of CYCLE is compared to the current value of the bit field CYCTMR.BCC. If the comparison detects a match, the reference message is sent out with a new basic cycle count of 0.</p> <p>For slave devices, the value is used to detect the expected end of a matrix cycle.</p>

Controller Area Network (MultiCAN) Controller

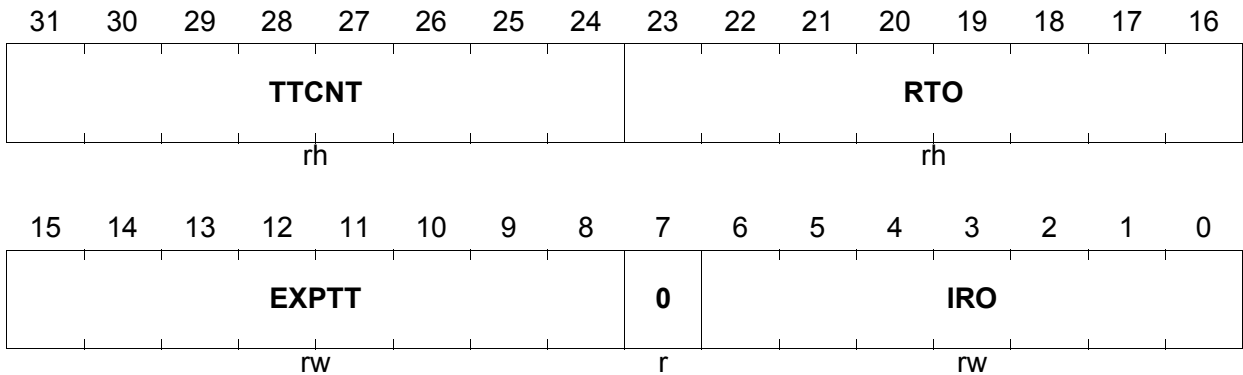
Field	Bits	Type	Description
TENW	[27:24]	rw	<p>Transmit Enable Window</p> <p>This bit field determines how many CAN bit times can elapse before a pending transmit trigger is discarded.</p> <p>0000_B One CAN bit time can elapse. 0001_B Two CAN bit times can elapse. ..._B ... 1110_B Fifteen CAN bit times can elapse. 1111_B Sixteen CAN bit times can elapse.</p>
RMDLC	[31:28]	rw	<p>Reference Message DLC</p> <p>This bit field determines the data length code DLC of reference messages sent out by this CAN node if it is a time master.</p> <p>The DLC bit field in the reference message object contains the DLC of the previously received reference message.</p> <p>Note that RMDLC must be programmed only to values from 1 to 8 for TTCAN level 1 and to values from 4 to 8 for TTCAN level 2. Other values must not be programmed.</p>
0	[11:9], 15, [23:22]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Controller Area Network (MultiCAN) Controller

TTCFGR

Time Trigger Configuration Register (2C4_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
IRO	[6:0]	rw	Initial Reference Offset This bit field determines the initial reference trigger offset. The value is considered as 2's complement and can reach values between 0 and 127.
EXPTT	[15:8]	rw	Expected Transmit Triggers This bit field determines how many transmit requests are expected in a matrix cycle.
RTO	[23:16]	rh	Reference Trigger Offset This bit field indicates the actual reference trigger offset. This value is considered as 2's complement and can reach values between -127 and 127. It is added to the time mark (given by the basic cycle end entry in the scheduler memory) for the trigger of the reference message. The modification and the corresponding conditions are listed in Table 22-15 .
TTCNT	[31:24]	rh	Transmit Trigger Counter The transmit trigger counter is incremented by 1 each time a transmit trigger is requested for a TTCAN node. When reaching the value of EXPTT, it is no longer incremented and further transmit requests are not serviced (messages are not transferred). It is reset at the beginning of each new matrix cycle (after correct transfer on the reference message of basic cycle 0).

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
0	7	r	Reserved Read as 0; should be written with 0.

Table 22-15 Modification by Hardware of Bit Field RTO

Modification	Condition
Set to TTCFGR.IRO	During reset (TTCFGR.IRO = 0) and Configuration Mode (write of TTCFGR.IRO by software)
Set to TTCFGR.IRO	Each time a potential time master receives a reference message with a higher priority than its own
Decrement by 1	Each time a potential time master receives a reference message with a lower priority than its own until -127 is reached
Set to 127	When the S2 state is entered
Set to 0	When the TTCAN node has correctly transmitted its reference message (it is the current time master) or when a potential master receives a reference message with a priority lower than its own and its RTO is positive

Controller Area Network (MultiCAN) Controller

The Time Trigger Status Register TTSR contains the time trigger status information that does not lead directly to interrupt events.

TTSR

Time Trigger Status Register

(2C8_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					ETR EV	ETR	NIG	0	MSCMAX			0	MSCMIN		
r					rh	rh	rh	r	rh			r	rh		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REC F	TRA F	TM PC	CFG M	ARB	REF TRG	EFF	EFI	0		SYNCS		MSR		ERRS	
rh	rh	rh	rh	rh	rh	rh	rh	r		rh		rh		rh	

Field	Bits	Type	Description
ERRS	[1:0]	rh	Error State This bit field indicates the current error severity level. 00 _B No error 01 _B Warning 10 _B Error 11 _B Severe error
MSR	[3:2]	rh	Master-Slave Relation This bit field indicates the current master to slave relation of the TTCAN node. 00 _B Master off 01 _B Slave (receiving device) 10 _B Potential time master 11 _B Current time master
SYNCS	[5:4]	rh	Synchronization State This bit field indicates the current synchronization state of the TTCAN node. 00 _B Sync off 01 _B Synchronizing 10 _B In gap 11 _B In schedule

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
EFI	8	rh	<p>Error Frame Indication</p> <p>This bit is set when an error condition in the bitstream of a CAN frame is detected and an Error Frame will be sent out. It is automatically cleared when the next time mark is reached and its status at this moment is monitored by bit EFF.</p> <p>0_B Since the last time mark, the CAN node has detected no error condition in the CAN bitstream.</p> <p>1_B Since the last time mark, the CAN node has detected an error condition in the CAN bitstream. When EFI becomes set, a LEC interrupt can be generated by the CAN node.</p>
EFF	9	rh	<p>Error Frame Flag</p> <p>This bit monitors the Error Frame indication EFI when a time mark is reached.</p> <p>0_B No CAN error has been detected in the last time window.</p> <p>1_B A CAN error has been detected in the last time window.</p>
REFTRG	10	rh	<p>Reference Trigger Flag</p> <p>This bit is set when the reference message of a TTCAN node is intended to be sent. It is cleared when the reference message has been sent out correctly or a reference message has been received correctly (sent by another time master).</p>
ARB	11	rh	<p>Arbitration Window Flag</p> <p>This bit is set when an arbitration window is opened. It is cleared when the arbitration window is closed.</p> <p>0_B The current time window is a time-triggered window.</p> <p>1_B The current time window is an arbitration window.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
CFGM	12	rh	<p>Configuration Mode Flag This bit indicates whether or not Configuration Mode is active.</p> <p>0_B The TTCAN node is not in Configuration Mode.</p> <p>1_B The TTCAN node is in Configuration Mode. It will not transmit messages and it will not update the MSC bit fields.</p>
TMPC	13	rh	<p>Time Master Priority Conflict This bit indicates if there is a priority conflict between the current time master and this TTCAN node. A priority conflict occurs if the current time master has a lower priority than the TTCAN node itself (potential time master).</p> <p>0_B A priority conflict has not been detected for the last reference message.</p> <p>1_B The TTCAN node is a potential time master with a priority conflict (the last reference message has been received with a lower priority than TTCR.TMPRIO).</p>
TRAF	14	rh	<p>Transmission Finished Flag This bit is set when the CAN node correctly terminates a transmission of a message. It is cleared when the next time mark is reached. This flag is used for the interrupt generation of the time mark entries and the interrupt entries.</p> <p>0_B Since the last time mark has been reached, the CAN node has not yet correctly terminated a transmission.</p> <p>1_B Since the last time mark has been reached, the CAN node has correctly terminated a transmission.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
RECF	15	rh	<p>Reception Finished Flag</p> <p>This bit is set when the CAN node correctly terminates a reception of a message (= a correct message has been seen on the bus). It is cleared when the next time mark is reached.</p> <p>This flag is used for the interrupt generation of the time mark entries and the interrupt entries.</p> <p>0_B Since the last time mark has been reached, the CAN node has not yet received a message.</p> <p>1_B Since the last time mark has been reached, the CAN node has received a message.</p>
MSCMIN	[18:16]	rh	<p>Minimum of MSC Bit Fields</p> <p>This bit field indicates the minimum value of the MSC bit fields of the message objects activated in exclusive time windows. This value is set to 7 at the beginning of a new matrix cycle.</p> <p>It is updated according to:</p> <p><i>Note: if MSC_cur < MSCMIN then MSCMIN := MSC_cur</i></p> <p>The value MSC_cur is the MSC value of the currently activated message object.</p>
MSCMAX	[22:20]	rh	<p>Maximum of MSC Bit Fields</p> <p>This bit field indicates the maximum value of the MSC bit fields of the message objects activated in exclusive time windows. This value is set to 0 at the beginning of a new matrix cycle.</p> <p>It is updated according to:</p> <p><i>Note: if MSC_cur > MSCMAX then MSCMAX := MSC_cur</i></p> <p>The value MSC_cur is the MSC value of the currently activated message object.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
NIG	24	rh	<p>Next Is Gap</p> <p>This bit indicates the condition leading to the transmission of the reference message coming after the next one (only for the current time master). This bit will be transmitted with the next reference message.</p> <p>0_B The reference message after the next one will be transmitted when the corresponding time mark is reached.</p> <p>1_B The reference message after the next one will be transmitted when the selected trigger event occurs. Bit NIG will be cleared and bit ETR will be set when the next reference message is correctly transferred (received or transmitted).</p>
ETR	25	rh	<p>External Trigger Request</p> <p>This bit indicates the condition leading to the transmission of the next reference message (only for the current time master). The value of NIG is copied to ETR when a reference message is correctly transmitted. It will be cleared automatically when the next reference message is correctly received.</p> <p>0_B The next reference message will be transmitted when the corresponding time mark is reached.</p> <p>1_B The next reference message will be transmitted when the selected trigger or the RME trigger event occurs.</p>
ETREV	26	rh	<p>External Trigger Event</p> <p>This bit indicates that the external trigger event has been detected. It is automatically cleared when a reference message has been transferred correctly on the bus.</p> <p>0_B The selected external trigger event has not yet been detected.</p> <p>1_B The selected external trigger event has been detected.</p>
0	[7:6], 19, 23, [31:27]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Controller Area Network (MultiCAN) Controller

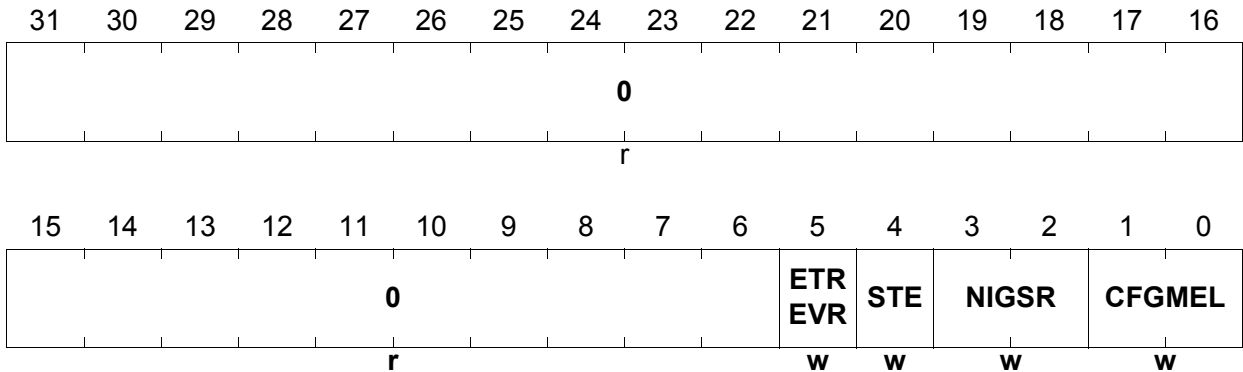
TTCAN status bits/flags can be modified when executing a write operation to the Time Trigger Flag Modification Register TTFMR.

TTFMR

Time Trigger Flag Modification Register

(2CC_H)

Reset Value: 0000 0001_H



Field	Bits	Type	Description
CFGMEL	[1:0]	w	Configuration Mode Enter/Leave This bit field is used to enter/leave the Configuration Mode. 00 _B No action 01 _B The Configuration Mode will be entered (set TTSR.CFGM). 10 _B The Configuration Mode will be left (clear TTSR.CFGM). 11 _B No action
NIGSR	[3:2]	w	Next Is Gap Flag Set/Reset This bit field can set/cleared bit TTSR.NIG and can cleared bit TTSR.ETR. 00 _B No action 01 _B Bit TTSR.NIG will be set. 10 _B Bits TTSR.NIG and TTSR.ETR will be cleared. 11 _B No action

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
STE	4	w	<p>Software Trigger Event This bit can be used to synchronize a TTCAN node by software.</p> <p>0_B No action 1_B The transmission of a reference message is triggered if TTCR.ETESEL = 11 and the system is in-a-gap.</p>
ETREVR	5	w	<p>Reset External Trigger Event This bit clears the external trigger event flag.</p> <p>0_B No action. 1_B The bit TTSR.ETREV is cleared.</p>
0	[31:6]	r	<p>Reserved Read as 0; should be written with 0.</p>

Controller Area Network (MultiCAN) Controller

The Time Trigger Interrupt Request Register TTIRR contains the time trigger status information related to interrupt events. Note that all bits in TTIRR can be cleared by software by writing 0 to it. Writing a 1 to the bits has no effect.

TTIRR

Time Trigger Interrupt Request Register

(2D0_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
0														TUR ERR	CFG ERR	
														r	rwh	rwh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
SER R2	SER R1	DISC	WFE	EOS	SYN CSC	MSR C	ERR SC	AWD ERR	I WTE	WTE	TT OF	TT UF	TEN WER	NBC	NMC	
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	

Field	Bits	Type	Description
NMC	0	rwh	<p>New Matrix Cycle</p> <p>This bit indicates that a new matrix cycle has started. It is set when a reference message with the cycle count = 0 has been transferred correctly.</p> <p>0_B A new matrix cycle has not yet been detected. 1_B A new matrix cycle has been detected.</p>
NBC	1	rwh	<p>New Basic Cycle</p> <p>This bit indicates that a new basic cycle has started.</p> <p>0_B A new basic cycle has not yet been detected. 10_B A new basic cycle has been detected.</p>
TENWER	2	rwh	<p>Transmit Enable Window Error</p> <p>This bit indicates that the specified time elapsed after the transmit trigger without starting the transmission of a message.</p> <p>0_B The triggered messages have been sent out before the transmit enable window elapsed. 1_B A triggered message was not started before the transmit enable window elapsed.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
TTUF	3	rwh	<p>Transmit Trigger Underflow</p> <p>This bit indicates that fewer transmit triggers have been requested during a matrix cycle than specified in TTCFGR.EXPTT.</p> <p>0_B The expected number or more transmit triggers have been requested.</p> <p>1_B Less transmit triggers have been requested.</p>
TTOF	4	rwh	<p>Transmit Trigger Overflow</p> <p>This bit indicates that more transmit triggers have been requested during a matrix cycle than specified in TTCFGR.EXPTT.</p> <p>0_B The expected number or less transmit triggers have been requested.</p> <p>1_B More transmit triggers have been requested.</p>
WTE	5	rwh	<p>Watch Trigger Event¹⁾</p> <p>This bit indicates a watch trigger event WTE. This event is detected when the cycle time in CYCTMR.CYCTM becomes equal to the watch trigger value given by the time mark of the BCE entry.</p> <p>0_B The cycle time has not been equal to WTV.</p> <p>1_B The cycle time has been equal to WTV.</p>
IWTE	6	rwh	<p>Init Watch Trigger Event²⁾</p> <p>This bit indicates a watch trigger event WTE with the value of the Init_Watch_Trigger. This event is detected when the cycle time in CYCTMR.CYCTM becomes equal to the init watch trigger value of $2^{16} - 1$.</p> <p>0_B The cycle time has not been equal to $2^{16} - 1$.</p> <p>1_B The cycle time has been equal to $2^{16} - 1$.</p>
AWDERR	7	rwh	<p>Application Watchdog Error</p> <p>This bit indicates that the application watchdog has been decremented to 0 without being serviced.</p> <p>0_B AWDR.AWDV has not yet reached 00_H.</p> <p>1_B AWDR.AWDV has reached 00_H. This is an S3 error condition.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
ERRSC	8	rwh	<p>Error State Change This bit indicates that the bit field TTSR.ERRS has changed.</p> <p>0_B TTSR.ERRS has not changed. 1_B TTSR.ERRS has changed.</p>
MSRC	9	rwh	<p>Master-Slave Relation Change This bit indicates that the bit field MSR has changed.</p> <p>0_B TTSR.MSR has not changed. 1_B TTSR.MSR has changed.</p>
SYNCS	10	rwh	<p>Synchronization State Change This bit indicates that the bit field TTSR.SYNCS has changed.</p> <p>0_B TTSR.SYNCS has not changed. 1_B TTSR.SYNCS has changed.</p>
EOS	11	rwh	<p>End Of Scheduler Entry Flag This bit is set when the TTCAN scheduler reads an EOS entry. In this case, bit TTSR.CFGM is automatically set.</p> <p>0_B The TTCAN scheduler has not yet read an EOS entry. 1_B The TTCAN scheduler has read an EOS entry.</p>
WFE	12	rwh	<p>Wait For Event Flag This bit is set when a reference message is received indicating a next-is-gap (not for the current time master). This event can generate a notification interrupt.</p> <p>0_B The last reference message received was not indicating next-is-gap. 1_B The last reference message received was indicating next-is-gap.</p>
DISC	13	rwh	<p>Discontinuity Flag This bit is set when a reference message is received indicating a discontinuity (not for the current time master). This event can generate a notification interrupt.</p> <p>0_B The last reference message received was not indicating a discontinuity. 1_B The last reference message received was indicating a discontinuity.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
SERR1	14	rwh	<p>Scheduler Error 1 Flag</p> <p>This bit is set when a scheduler error type 1 is detected. The scheduler error can change the TTCAN error state and, as a result, can be indicated by an error state change interrupt.</p> <p>0_B A scheduler error type 1 has not been detected.</p> <p>1_B A scheduler error type 1 has been detected.</p>
SERR2	15	rwh	<p>Scheduler Error 2 Flag</p> <p>This bit is set when a scheduler error type 2 is detected. The scheduler error can change the TTCAN error state and, as a result, can be indicated by an error state change interrupt.</p> <p>0_B A scheduler error type 2 has not been detected.</p> <p>1_B A scheduler error type 2 has been detected.</p>
CFGERR	16	rwh	<p>Configuration Error</p> <p>This bit indicates that a configuration error has been detected by the scheduler. A configuration error is detected when an arbitration window is not closed when the BCE is reached.</p> <p>When RME.TMV is reached, no other scheduler entries as RME and BCE are allowed. See also EOSERR.</p> <p>0_B A configuration error has not been detected.</p> <p>1_B A configuration error has been detected (S3 error condition).</p>
TURERR	17	rwh	<p>TUR Adjust Error</p> <p>This bit indicates that a TUR adjust error has been detected. A TUR adjust error is detected when the automatic TUR adjustment is enabled and an overflow or an underflow of the calculated TURADJ value occurs.</p> <p>0_B A TUR adjust error has not been detected.</p> <p>1_B A TUR adjust error has been detected.</p>
0	[31:18]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

- 1) The WTE can be generated only after the first message has been transferred on the bus.
- 2) The IWTE can be generated only until the first message has been transferred on the bus.

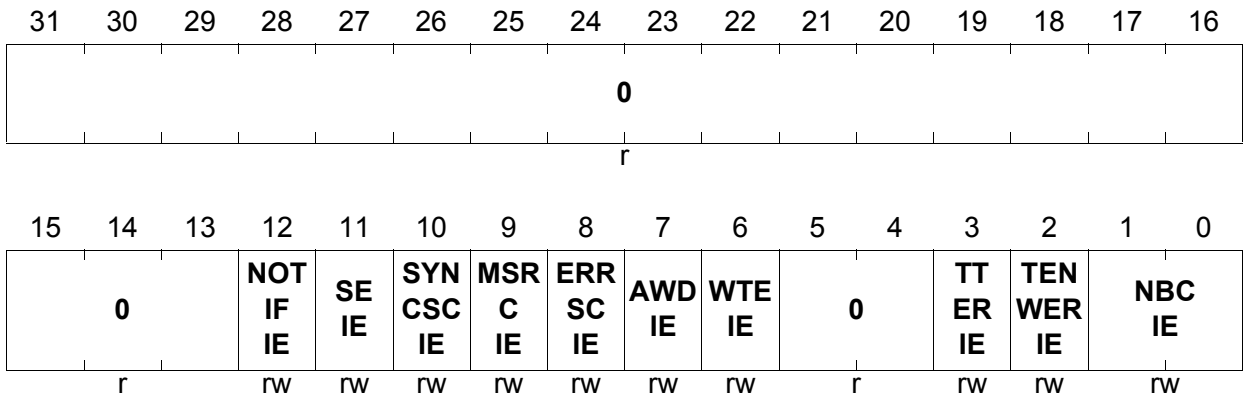
Controller Area Network (MultiCAN) Controller

TTIER

Time Trigger Interrupt Enable Register

(2D4_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
NBCIE	[1:0]	rw	<p>New Basic Cycle Interrupt Enable NBCIE enables the new basic or matrix cycle interrupt. This interrupt is generated when either bit TTIRR.NBC or bit TTIRR.NMC become set (independent of its current state).</p> <p>00_B A new basic or matrix cycle interrupt is disabled.</p> <p>01_B A basic cycle interrupt is generated whenever TTIRR.NBC becomes set.</p> <p>10_B A matrix cycle interrupt is generated whenever TTIRR.NMC becomes set.</p> <p>11_B Reserved Bit field TTINPR.NBCINP selects the interrupt output line that becomes activated at this type of interrupt.</p>
TENWERIE	2	rw	<p>Transmit Enable Window Error Interrupt Enable TENWERIE enables the transmit enable window error interrupt. This interrupt is generated when bit TTIRR.TENWER is set.</p> <p>0_B Transmit enable window error interrupt is disabled.</p> <p>1_B Transmit enable window error interrupt is enabled.</p> <p>Bit field TTINPR.ERRINP selects the interrupt output line that becomes activated at this type of interrupt.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
TTERIE	3	rw	<p>Transmit Trigger Error Interrupt Enable TTERIE enables the transmit trigger error interrupt. This interrupt is generated when TTIRR.TTOF or TTIRR.TTUF become set (independent of its current state).</p> <p>0_B Transmit trigger error interrupt is disabled. 1_B Transmit trigger error interrupt is enabled. Bit field TTINPR.ERRINP selects the interrupt output line that becomes activated at this type of interrupt.</p>
WTEIE	6	rw	<p>Watch Trigger Event Interrupt Enable WTEIE enables the watch trigger event interrupt. This interrupt is generated when TTIRR.IWTE or TTIRR.WTE become set (independent of its current state).</p> <p>0_B Watch trigger event interrupt is disabled. 1_B Watch trigger event interrupt is enabled. Bit field TTINPR.ERRINP selects the interrupt output line that becomes activated at this type of interrupt.</p>
AWDIE	7	rw	<p>Application Watchdog Interrupt Enable AWDIE enables the application watchdog interrupt. This interrupt is generated when TTIRR.AWDERR becomes set (independent of its current state).</p> <p>0_B Application watchdog interrupt is disabled. 1_B Application watchdog interrupt is enabled. Bit field TTINPR.ERRINP selects the interrupt output line that becomes activated at this type of interrupt.</p>
ERRSCIE	8	rw	<p>Error State Change Interrupt Enable ERRSCIE enables the error state change interrupt. This interrupt is generated when bit field TTSR.ERRSC changes its state.</p> <p>0_B Error state change interrupt is disabled. 1_B Error state change interrupt is enabled. Bit field TTINPR.NOTIFINP selects the interrupt output line that becomes activated at this type of interrupt.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
MSRCIE	9	rw	<p>Master Slave Relation Change Interrupt Enable MSRCIE enables the master slave relation change interrupt. This interrupt is generated when bit field TTSR.MSRC changes its state.</p> <p>0_B Master slave relation change interrupt is disabled.</p> <p>1_B Master slave relation change interrupt is enabled.</p> <p>Bit field TTINPR.NOTIFINP selects the interrupt output line that becomes activated at this type of interrupt.</p>
SYNCSCIE	10	rw	<p>Synchronization State Change Interrupt Enable SYNCSCIE enables the synchronization state change interrupt. This interrupt is generated when bit field TTSR.SYNCSC changes its state.</p> <p>0_B Synchronization state change interrupt is disabled.</p> <p>1_B Synchronization state change interrupt is enabled.</p> <p>Bit field TTINPR.NOTIFINP selects the interrupt output line that becomes activated at this type of interrupt.</p>
SEIE	11	rw	<p>Scheduler Error Interrupt Enable SEIE enables the scheduler error interrupt. This interrupt is generated whenever bits EOS or SERR1 or SERR2 or CFGERR of register TTSR are set by hardware.</p> <p>0_B Scheduler error interrupt generation is disabled.</p> <p>1_B Scheduler error interrupt generation is enabled.</p> <p>Bit field TTINPR.ERRINP selects the interrupt output line that becomes activated at this type of interrupt.</p>

Controller Area Network (MultiCAN) Controller

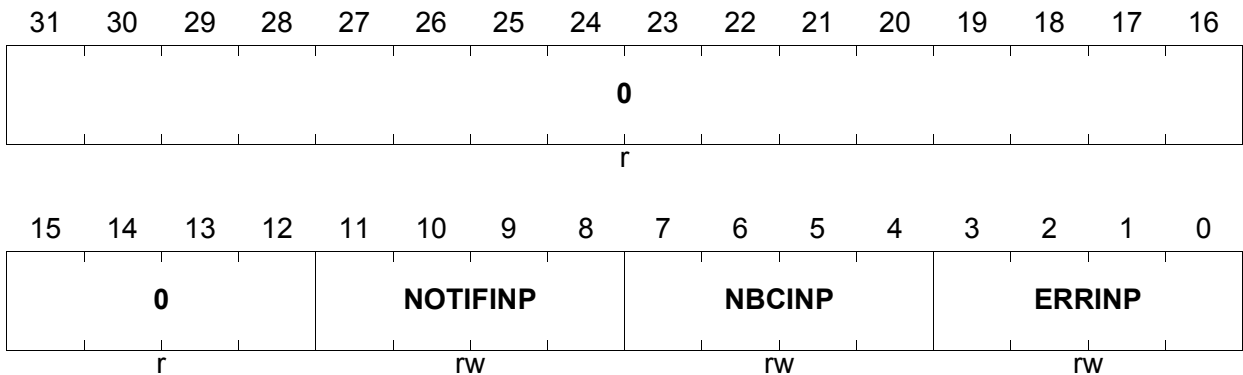
Field	Bits	Type	Description
NOTIFIE	12	rw	<p>Notification Interrupt Enable NOTIFIE enables the notification interrupt. This interrupt is generated whenever bits TTIRR.WFE or TTIRR.DISC are set by hardware.</p> <p>0_B Notification interrupt is disabled. 1_B Notification interrupt is enabled. Bit field TTINPR.NOTIFINP selects the interrupt output line which becomes activated at this type of interrupt.</p>
0	[5:4], [31:13]	r	<p>Reserved Read as 0; should be written with 0.</p>

Controller Area Network (MultiCAN) Controller

TTINPR

**Time Trigger Interrupt Node Pointer Register
(2D8_H)**

Reset Value: 0000 0000_H



Field	Bits	Type	Description
ERRINP	[3:0]	rw	<p>Error Interrupt Node Pointer ERRINP selects the interrupt output line INT_O_m (m = 0-15) for an error interrupt. Possible error events for this interrupt node pointer are:</p> <ul style="list-style-type: none"> • Transmit enable window error event • Transmit trigger error event • (Initial) Watch trigger event • Application watchdog event • TUR adjust error • Scheduler error event <p>0000_B Interrupt output line INT_O0 is selected. 0001_B Interrupt output line INT_O1 is selected. ..._B ... 1110_B Interrupt output line INT_O14 is selected. 1111_B Interrupt output line INT_O15 is selected.</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
NBCINP	[7:4]	rw	<p>New Basic Cycle Interrupt Node Pointer NBCINP selects the interrupt output line INT_Om (m = 0-15) for a new basic or matrix cycle interrupt. This interrupt can be generated only when the reference message is received. The interrupt for transmitted reference messages is controlled by the reference message entry in the scheduler.</p> <p>0000_B Interrupt output line INT_O0 is selected. 0001_B Interrupt output line INT_O1 is selected. ..._B ... 1110_B Interrupt output line INT_O14 is selected. 1111_B Interrupt output line INT_O15 is selected.</p>
NOTIFINP	[11:8]	rw	<p>Notification Interrupt Node Pointer NOTIFINP selects the interrupt output line INT_Om (m = 0-15) for a notification interrupt. Possible notification events for this interrupt node pointer are:</p> <ul style="list-style-type: none"> • bit field TTSR.ERRS changes • bit field TTSR.MSR changes • bit field TTSR.SYNCS changes • bit TTIRR.WFE is set • bit TTIRR.DISC is set <p>0000_B Interrupt output line INT_O0 is selected. 0001_B Interrupt output line INT_O1 is selected. ..._B ... 1110_B Interrupt output line INT_O14 is selected. 1111_B Interrupt output line INT_O15 is selected.</p>
0	[31:12]	r	<p>Reserved Read as 0; should be written with 0.</p>

Controller Area Network (MultiCAN) Controller

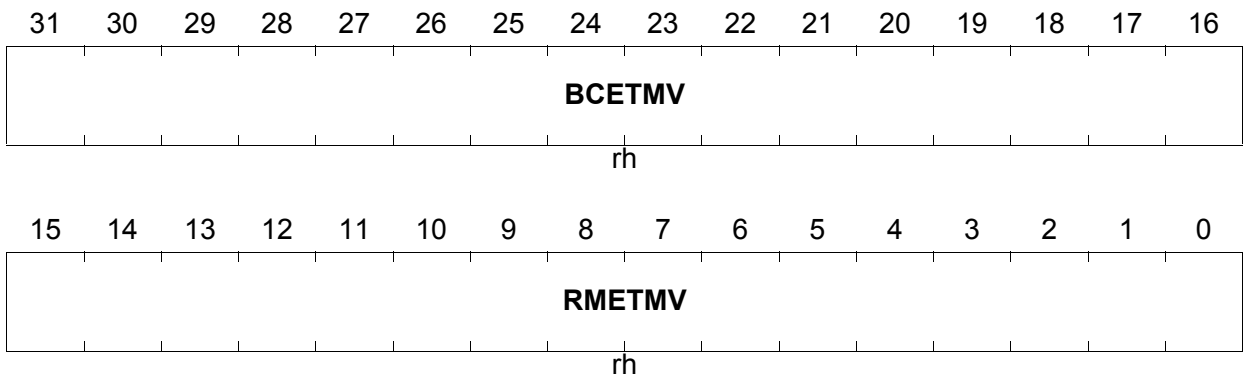
The TMV bit fields in the scheduler timing status registers monitor the time mark information for the start of the next time window after the instruction collection phase of the scheduler (collected from the TME, RME or BCE entries).

STSRL

Scheduler Timing Status Register Low

(2F0_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
RMETMV	[15:0]	rh	Time Mark Value from RME This bit field indicates the compare value for the next time mark defined by an RME. This value is valid only if SISR.ICF = 1 and SISR.RMEV = 1.
BCETMV	[31:16]	rh	Time Mark Value from BCE This bit field indicates the compare value for the next time mark defined by an BCE. This value is valid only if SISR.ICF = 1.

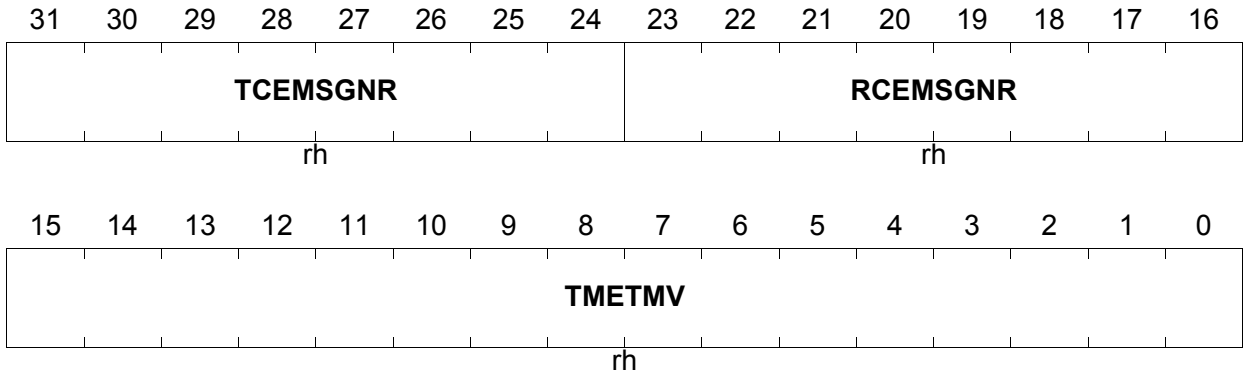
Controller Area Network (MultiCAN) Controller

STSRH

Scheduler Timing Status Register High

(2F4_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
TMETMV	[15:0]	rh	Time Mark Value from TME This bit field indicates the compare value for the next time mark defined by an TME. This value is valid only if SISR.ICF = 1 and SISR.TMEV = 1.
RCEMSGNR	[23:16]	rh	Receive Control Entry Message Number This bit field indicates the collected RCEMSGNR information from an RCE. This value is taken into account only if SISR.ICF = 1 and SISR.RCEV = 1.
TCEMSGNR	[31:24]	rh	Transmit Control Entry Message Number This bit field indicates the collected TCEMSGNR information from an TCE. This value is taken into account only if SISR.ICF = 1 and SISR.TCEV = 1.

Controller Area Network (MultiCAN) Controller

The bits in the scheduler instruction status register monitor the information during and after the instruction collection phase of the scheduler. These values will become valid when the next time mark is reached.

SISR

Scheduler Instruction Status Register

(2F8_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0									BCE	RME	TME	ARB	ICE	TCE	RCE
									V	V	V	V	V	V	V
r									rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ICF	GM	ARBM	ALTMSG	TR EN	CH EN	IEN REC F1	IEN REC F0	IEN TRA F1	IEN TRA F0	INP					
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh					

Field	Bits	Type	Description
INP	[3:0]	rh	Interrupt Node Pointer This bit field indicates the collected INP information. This value is taken into account only when at least one of the four interrupt requests is enabled.
IENTRAF0	4	rh	Interrupt Enable if TRAF = 0 This bit field indicates the collected IENTRAF0 information.
IENTRAF1	5	rh	Interrupt Enable if TRAF = 1 This bit field indicates the collected IENTRAF1 information.
IENRECF0	6	rh	Interrupt Enable if RECF = 0 This bit field indicates the collected IENRECF0 information.
IENRECF1	7	rh	Interrupt Enable if RECF = 1 This bit field indicates the collected IENRECF1 information.

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
CHEN	8	rh	Check Enable This bit field indicates the collected CHEN information from an RCE. CHEN is taken into account only if EC indicates an RCE.
TREN	9	rh	Transmit Enable This bit field indicates the collected TREN information from an TCE.
ALTMSG	[11:10]	rh	Alternative Message This bit field indicates the collected ALTMSG information from an TCE.
ARBM	[13:12]	rh	Arbitration Mode This bit field indicates the collected ARBM information from an TME or an ARBE.
GM	14	rh	Gap Mode This bit field indicates the collected GM information from an RME or an BCE only valid if RME or a BCE has been found).
ICF	15	rh	Instruction Collection Finished This bit field indicates that the instruction collection is finished for a time window. It is automatically cleared when a time mark is reached. It is set when the instruction collection is terminated. 0 _B The instruction collection is not yet terminated. All other values in registers SISR, STSR0 and STSR4 are invalid. 1 _B The instruction collection is terminated. All other values in registers SISR, STSR0 and STSR4 are valid.

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
RCEV	16	rh	<p>Receive Control Entry Valid This bit indicates that a valid receive control information has been found during the instruction collection for this time window. It is automatically cleared when a time mark is reached.</p> <p>0_B The bit fields CHEN and RCMSGNR are invalid. They are not taken into account for the next time window.</p> <p>1_B The bit fields CHEN and RCMSGNR are valid. They are taken into account for the next time window.</p>
TCEV	17	rh	<p>Transmit Control Entry Valid This bit indicates that valid transmit control information has been found during the instruction collection for this time window. It is automatically cleared when a time mark is reached.</p> <p>0_B The bit fields TREN, ALTMSG and TCMSGNR are invalid. They are not taken into account for the next time window.</p> <p>1_B The bit fields TREN, ALTMSG and TCMSGNR are valid. They are taken into account for the next time window.</p>
ICEV	18	rh	<p>Interrupt Control Entry Valid This bit indicates that a valid interrupt control entry has been found during the instruction collection for this time window. It is automatically cleared when a time mark is reached.</p> <p>0_B No valid ICE has been found</p> <p>1_B A valid ICE has been found</p>
ARBV	19	rh	<p>Arbitration Entry Valid This bit indicates that a valid arbitration entry has been found during the instruction collection for this time window. It is automatically cleared when a time mark is reached.</p> <p>0_B No valid ARBE has been found</p> <p>1_B A valid ARBE has been found</p>

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
TMEV	20	rh	<p>Time Mark Entry Valid This bit indicates that a valid time mark entry has been found during the instruction collection for this time window. It is automatically cleared when a time mark is reached.</p> <p>0_B No valid TME has been found 1_B A valid TME has been found</p>
RMEV	21	rh	<p>Reference Mark Entry Valid This bit indicates that a valid reference mark entry has been found during the instruction collection for this time window. It is automatically cleared when a time mark is reached.</p> <p>0_B No valid RME has been found 1_B A valid RME has been found</p>
BCEV	22	rh	<p>Basic Cycle End Entry Valid This bit indicates that a valid basic cycle end entry has been found during the instruction collection for this time window. It is automatically cleared when a time mark is reached.</p> <p>0_B No valid BCE has been found 1_B A valid BCE has been found</p>
0	[31:23]	r	<p>Reserved Read as 0; should be written with 0.</p>

Note: This register is reset at the start of a new instruction collection phase.

Controller Area Network (MultiCAN) Controller

22.9 MultiCAN Module Implementation

This section describes CAN module interfaces with the clock control, port connections, interrupt control, and address decoding.

22.9.1 Interfaces of the MultiCAN Module

Figure 22-38 shows the TC1796 specific implementation details and interconnections of the MultiCAN module. The eight I/O lines of the MultiCAN module (two I/O lines of each CAN node) are connected to I/O lines of Port 6. The MultiCAN module is also supplied by clock control, interrupt control, and address decoding logic. MultiCAN interrupts can be directed to the DMA controller and the GPTA modules. External trigger events for the TTCAN can be generated by the GPTA0, the External Request Unit located in the SCU, or directly through to pins. CAN interrupts are able to trigger DMA transfers and GPTA operations.

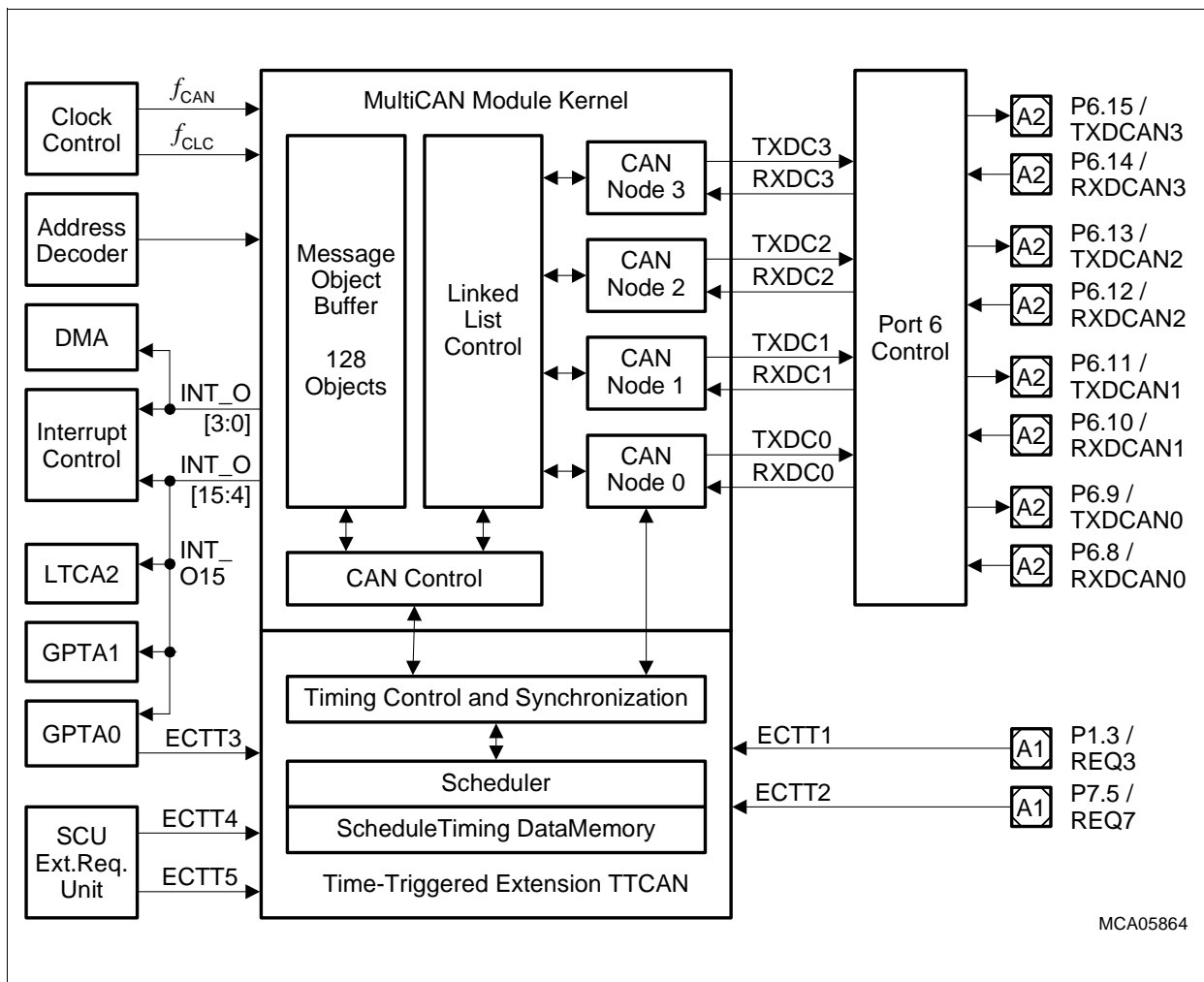


Figure 22-38 CAN Module Implementation and Interconnections

Controller Area Network (MultiCAN) Controller

22.9.2 MultiCAN Module External Registers

The registers listed in [Figure 22-39](#) are not included in the MultiCAN module kernel but must be programmed for proper operation of the MultiCAN module.

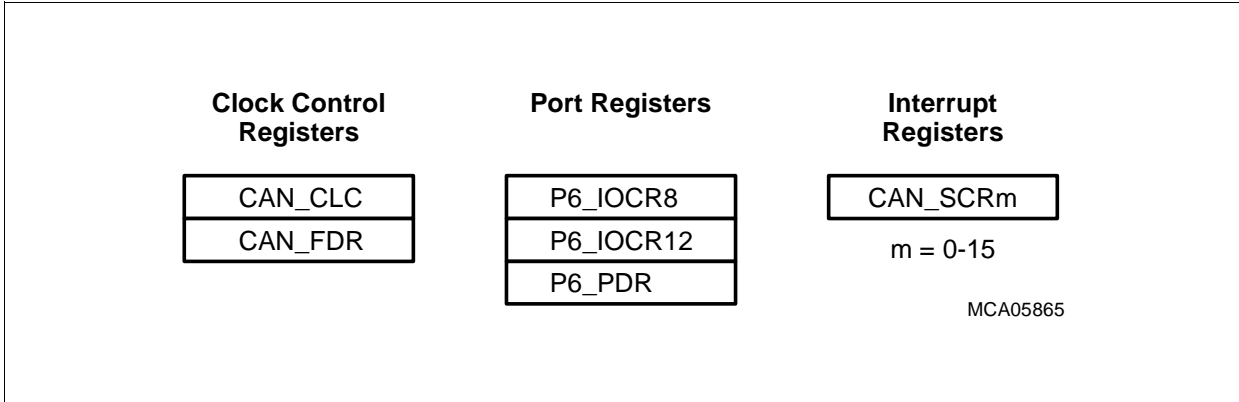


Figure 22-39 CAN Implementation-specific Special Function Registers

Controller Area Network (MultiCAN) Controller

22.9.3 Module Clock Generation

As shown in **Figure 22-40**, the clock signals for the MultiCAN module are generated and controlled by a clock control unit. This clock generation unit is responsible for the enable/disable control, the clock frequency adjustment, and the debug clock control. This unit includes two registers:

- CAN_CLC: generation of the module control clock f_{CLC}
- CAN_FDR: frequency control of the module timer clock f_{CAN}

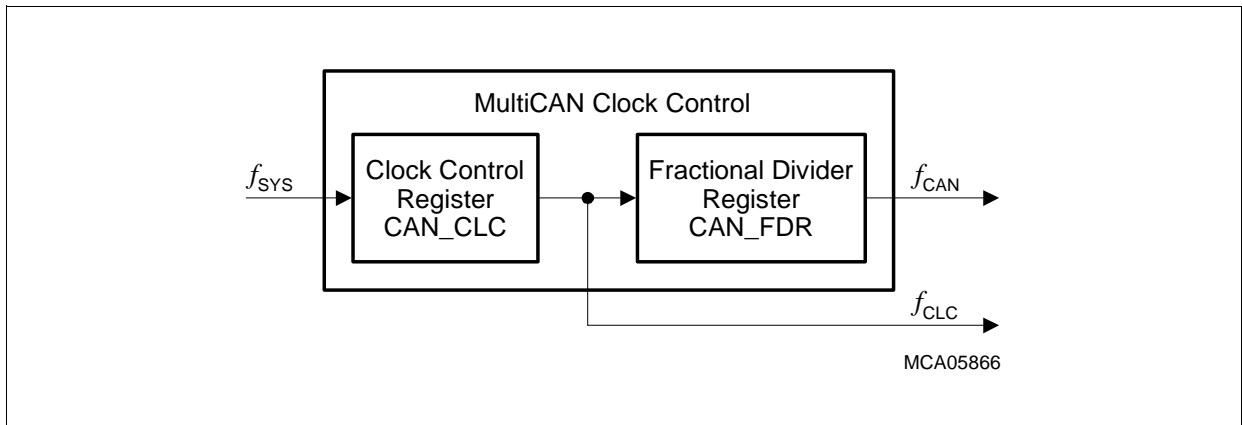


Figure 22-40 MultiCAN Module Clock Generation

The module control clock f_{CLC} is used inside the MultiCAN module for control purposes such as clocking of control logic and register operations. The frequency of f_{CLC} is identical to the system clock frequency f_{SYS} . The clock control register CAN_CLC makes it possible to enable/disable f_{CLC} under certain conditions.

The module timer clock f_{CAN} is used inside the MultiCAN module as input clock for all timing relevant operations (e.g. bit timing). The settings in the CAN_FDR register determine the frequency of the module timer clock f_{CAN} according the following two formulas:

$$f_{CAN} = f_{SYS} \times \frac{1}{n} \quad \text{with } n = 1024 - \text{CAN_FDR.STEP} \quad (22.1)$$

$$f_{CAN} = f_{SYS} \times \frac{n}{1024} \quad \text{with } n = 0-1023 \quad (22.2)$$

Equation (22.1) applies to normal divider mode (CAN_FDR.DM = 01_B) of the fractional divider. **Equation (22.2)** applies to fractional divider mode (CAN_FDR.DM = 10_B).

Note: The CAN module is disabled after reset. In general, after reset, the module control clock f_{CLC} must be switched on (writing to register CAN_CLC) before the frequency of the module timer clock f_{CAN} is defined (writing to register CAN_FDR).

Controller Area Network (MultiCAN) Controller

22.9.3.1 Clock Control Registers

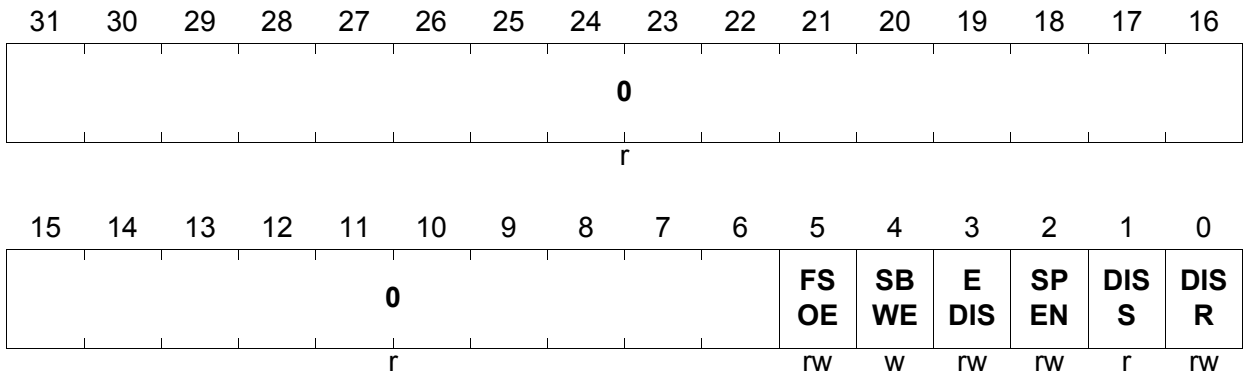
The clock control register makes it possible to control (enable/disable) the module control clock f_{CLC} .

CAN_CLC

CAN Clock Control Register

(000_H)

Reset Value: 0000 0003_H



Field	Bits	Type	Description
DISR	0	rw	Module Disable Request Bit Used for enable/disable control of the module.
DISS	1	r	Module Disable Status Bit Bit indicates the current status of the module.
SPEN	2	rw	Module Suspend Enable for OCDS Used to enable the Suspend Mode.
EDIS	3	rw	External Request Disable Used to control external clock disable request.
SBWE	4	w	Module Suspend Bit Write Enable for OCDS Determines whether SPEN and FSOE are write-protected.
FSOE	5	rw	Fast Switch Off Enable Used for fast clock switch off in Suspend Mode.
0	[31:6]	r	Reserved Read as 0; should be written with 0.

Note: Additional details on the clock control register functionality are described in section “Clock Control Register CLC” on Page 3-24 of the TC1796 User’s Manual System Units part (Volume 1).

Controller Area Network (MultiCAN) Controller

Field	Bits	Type	Description
DISCLK	31	rwh	Disable Clock Hardware controlled disable for f_{OUT} signal.
0	10, [27:26]	rw	Reserved Read as 0; should be written with 0.

*Note: Additional details on the fractional divider register functionality are described in section **“Fractional Divider Operation”** on **Page 3-29** of the TC1796 User’s Manual System Units part (Volume 1).*

Controller Area Network (MultiCAN) Controller

22.9.4 Port and I/O Line Control

The interconnections between the MultiCAN module and the port I/O lines are controlled in the port logic. Additionally to the port input selection, the following port control operations must be executed:

- Input/output function selection (IOCR registers)
- Pad driver characteristics selection for the outputs (PDR registers)

22.9.4.1 Input/Output Function Selection in Ports

The port input/output control registers contain the bit fields that select the digital output and input driver characteristics such as pull-up/down devices, port direction (input/output), open-drain, and alternate output selections. The I/O lines for the MultiCAN module are controlled by the port input/output control registers P6_IOCR8 and P6_OCR12.

Table 22-16 shows how bits and bit fields must be programmed for the required I/O functionality of the CAN I/O lines.

Table 22-16 MultiCAN I/O Control Selection and Setup

Module	Port Lines	Input/Output Control Register Bits ¹⁾	I/O
CAN	P6.8 / RXDCAN0	P6_IOCR8.PC8 = 0XXX _B	Input
	P6.9 / TXDCAN0	P6_IOCR8.PC9 = 1X01 _B	Output
	P6.10 / RXDCAN1	P6_IOCR8.PC10 = 0XXX _B	Input
	P6.11 / TXDCAN1	P6_IOCR8.PC11 = 1X01 _B	Output
	P6.12 / RXDCAN2	P6_IOCR12.PC12 = 0XXX _B	Input
	P6.13 / TXDCAN2	P6_IOCR12.PC13 = 1X01 _B	Output
	P6.14 / RXDCAN3	P6_IOCR12.PC14 = 0XXX _B	Input
	P6.15 / TXDCAN3	P6_IOCR12.PC15 = 1X01 _B	Output

1) Possible PCx bit field combinations see [Table 22-17](#).

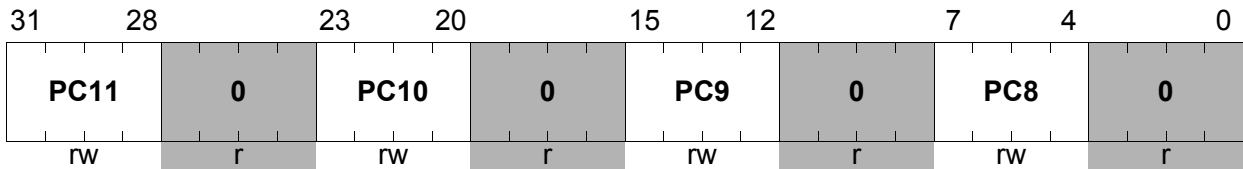
Controller Area Network (MultiCAN) Controller

22.9.4.2 CAN Related Input/Output Control Registers

P6_IOC8

Port 6 Input/Output Control Register 8(18_H)

Reset Value: 2020 2020_H



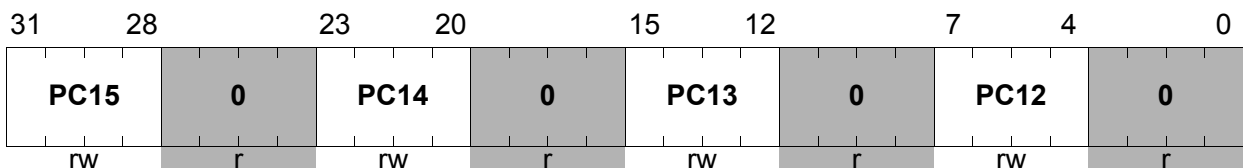
Field	Bits	Type	Description
PC8, PC9, PC10, PC11	[7:4], [15:12], [23:20], [31:28]	rw	Port Output Control for Port 6 Pins 8-11¹⁾ These bit fields determine the output port functionality: Port input/output control for P6.8/RXDCAN0 Port input/output control for P6.9/TXDCAN0 Port input/output control for P6.10/RXDCAN1 Port input/output control for P6.11/TXDCAN1

1) Coding of bit fields see [Table 22-17](#). Shaded bits and bit fields are “don’t care” for CAN I/O port control.

P6_IOC12

Port 6 Input/Output Control Register 12(1C_H)

Reset Value: 2020 2020_H



Field	Bits	Type	Description
PC12, PC13, PC14, PC15	[7:4], [15:12], [23:20], [31:28]	rw	Port Output Control for Port 6 Pins 12-15¹⁾ These bit fields determine the output port functionality: Port input/output control for P6.12/RXDCAN2 Port input/output control for P6.13/TXDCAN2 Port input/output control for P6.14/RXDCAN3 Port input/output control for P6.15/TXDCAN3

1) Coding of bit fields see [Table 22-17](#). Shaded bits and bit fields are “don’t care” for CAN I/O port control.

Controller Area Network (MultiCAN) Controller

PCx Coding

Table 22-17 PCx Coding for MultiCAN I/O Lines

PCx[3:0]	I/O	Output Characteristics	Selected Pull-up/Pull-down/ Selected Output Function
0X00 _B	Input	–	No pull device connected
0X01 _B			Pull-down device connected
0X10 _B ¹⁾			Pull-up device connected
0X11 _B			No pull device connected
1001 _B	Output	Push-pull	Output function ALT1
1101 _B		Open-drain	Output function ALT1

1) This bit field value is the default after reset.

22.9.4.3 Node Receive Input Selection

Additionally to the I/O control selection, as defined in [Table 22-16](#), one of the four CAN node's receive input lines must be selected using bit field RXSEL in its node port control register NPCRx. Two of the four CAN node's receive input lines are connected with GPIO lines according to [Table 22-18](#).

This feature allows, for example, a CAN node which operates in analyzer mode to monitor the receive operations of its neighbor CAN node. The default setting after reset of a node's NPCRx.RXSEL bit field connect node x with RXDCANx I/O line (x = 0-3).

Table 22-18 Receive Input Selection

Receive Input of	Connected to Port Pin	Selected by ¹⁾
CAN Node 0	P6.8 / RXDCAN0	NPCR0.RXSEL = 000 _B
	P6.10 / RXDCAN1	NPCR0.RXSEL = 001 _B
CAN Node 1	P6.10 / RXDCAN1	NPCR1.RXSEL = 000 _B
	P6.12 / RXDCAN2	NPCR1.RXSEL = 001 _B
CAN Node 2	P6.12 / RXDCAN2	NPCR2.RXSEL = 000 _B
	P6.14 / RXDCAN3	NPCR2.RXSEL = 001 _B
CAN Node 3	P6.14 / RXDCAN3	NPCR3.RXSEL = 000 _B
	P6.8 / RXDCAN0	NPCR3.RXSEL = 001 _B

1) Other values for NPCRx.RXSEL than 000_B and 001_B result in the following behavior for a CANx node:
 Recessive receive input level for RXSEL = 010_B to 110_B.
 Dominant receive input level for RXSEL = 111_B.

Controller Area Network (MultiCAN) Controller

22.9.4.4 Pad Output Driver Register

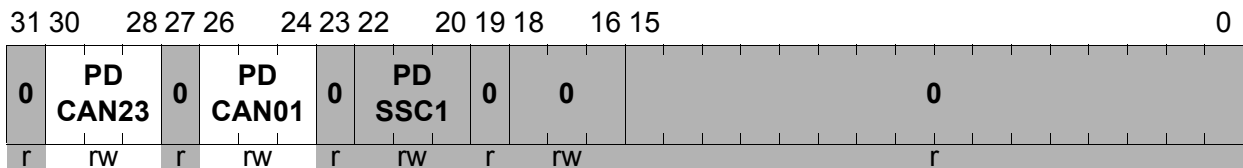
The Port 6 pad driver mode register contains bit fields that determine the output driver strength and the slew rate of MultiCAN output lines. The coding of the PDx bit field combinations is shown in [Table 22-19](#).

P6_PDR

Port 6 Pad Driver Mode Register

(40_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PDCAN01	[26:24]	rw	Pad Driver Mode for P6.9/TXDCAN0 and P6.11/TXDCAN1 ¹⁾
PDCAN23	[30:28]	rw	Pad Driver Mode for P6.13/TXDCAN2 and P6.15 ¹⁾ /TXDCAN3

1) Coding of bit field see [Table 22-19](#). Shaded bits and bit fields are “don’t care” for CAN I/O port control.

PDx Selection Table

Table 22-19 Pad Driver Mode Mode Selection (Class A2 Pads)

PDx Bit Field	Driver Strength	Signal Transitions
000 _B	Strong driver	Sharp edge ¹⁾
001 _B		Medium edge ¹⁾
010 _B		Soft edge ¹⁾
011 _B	Weak driver	–
100 _B	Medium driver	Sharp edge
101 _B		Medium edge
110 _B		Soft edge
111 _B	Weak driver	–

1) In strong driver mode, the output driver characteristics of class A2 pads can be additionally controlled by the temperature compensation logic.

Controller Area Network (MultiCAN) Controller

22.9.4.5 External CAN Time Trigger Inputs

The external CAN time trigger inputs ECTT[7:1] can be used as a transmit trigger for a reference message. In the TC1796, these input lines are connected as shown in [Table 22-20](#).

Table 22-20 External CAN Time Trigger Inputs

Receive Input	Connected to	From Module
ECTT1	P1.3 / REQ3 Input	Port 1
ECTT2	P7.5 / REQ7 Input	Port 7
ECTT3	Output OUT5	GPTA0
ECTT4	Output IOOUT2	External Request Unit (SCU)
ECTT5	Output IOOUT3	External Request Unit (SCU)
ECTT6	0	–
ECTT7	0	–

22.9.4.6 DMA Request Outputs

The interrupt output lines INT_O0 to INT_O2 of the MultiCAN module can be used as a DMA requestor and are able to trigger DMA transfers. INT_O[2:0] are connected to the DMA controller as shown in [Table 22-21](#).

Table 22-21 CAN-to-DMA Request Connections

DMA Channel	Connected to CAN Interrupt Output	Selected in DMA Controller by Programming
04	INT_O0	CHCR04.PRSEL = 101 _B
05	INT_O0	CHCR05.PRSEL = 101 _B
06	INT_O1	CHCR06.PRSEL = 110 _B
07	INT_O2	CHCR07.PRSEL = 110 _B
15	INT_O3	CHCR15.PRSEL = 101 _B

Controller Area Network (MultiCAN) Controller**22.9.5 Interrupt Control**

The interrupt control logic in the MultiCAN module uses an interrupt compressing scheme that allows high flexibility in interrupt processing. There are 274 hardware interrupt sources and one software interrupt source available:

- CAN node interrupts:
 - Four different interrupt sources for each of the four CAN nodes = 16 interrupt sources
- Message object interrupts:
 - Two interrupt sources for each message object = 256 interrupt sources
- TTCAN interrupts:
 - 3 interrupt sources
- One software initiated interrupt (register MITR)

Each of the 274 hardware initiated interrupt sources is controlled by a 4-bit interrupt pointer that directs the interrupt source to one of the sixteen interrupt outputs INT_Om (m = 0-15). This makes it possible to connect more than one interrupt source to one interrupt output line. The interrupt wiring matrix shown in [Figure 22-41](#) is built up according to the following rules:

- Each output of the 4-bit interrupt pointer demultiplexer is connected to exactly one OR-gate input of the INT_Om line. The number “m” of the corresponding selected INT_Om interrupt output line is defined by the interrupt pointer value.
- Each INT_Om output line has a 147-input OR-gate which is connected to all interrupt pointer demultiplexer outputs which are selected by an identical 4-bit pointer value.

Controller Area Network (MultiCAN) Controller

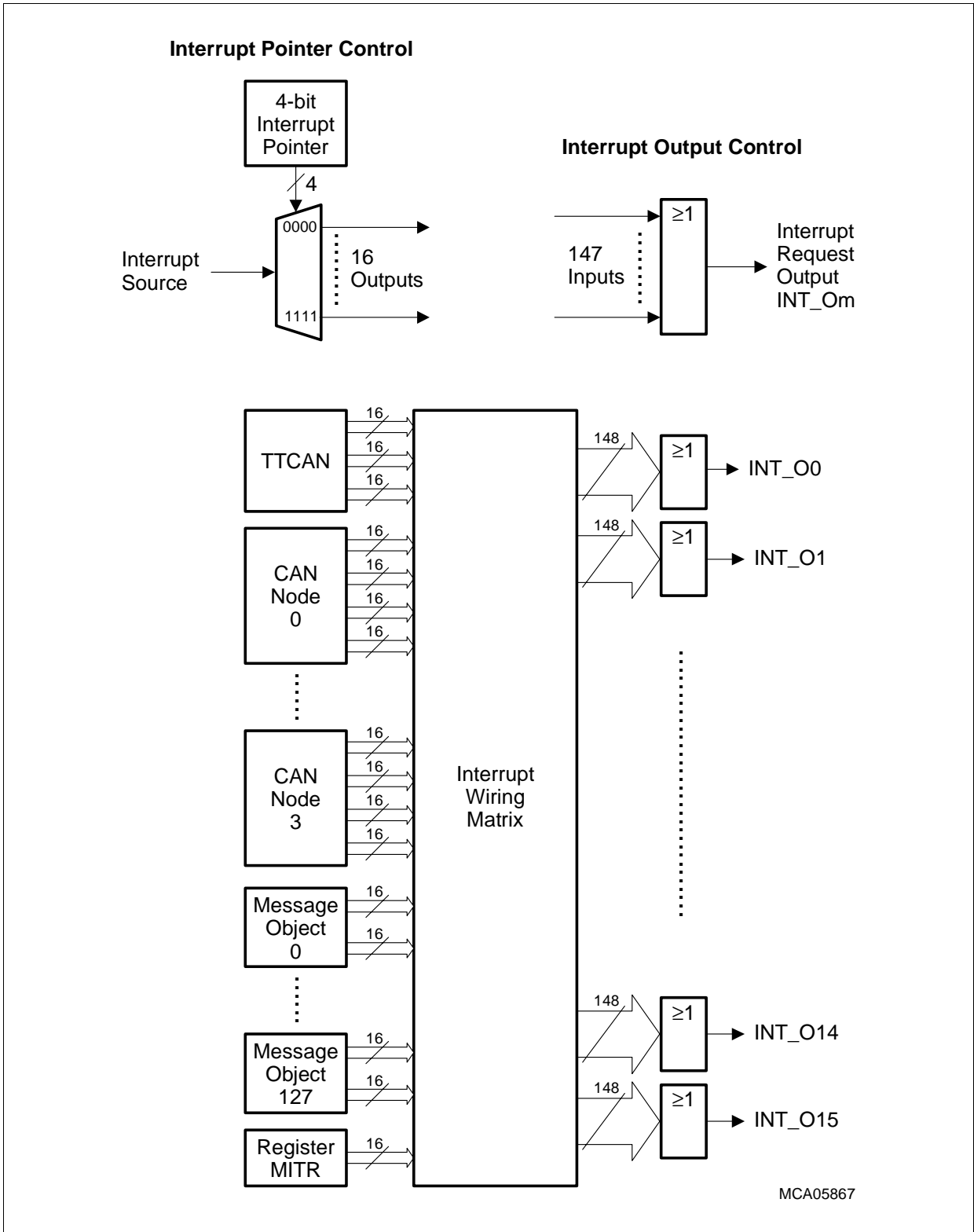


Figure 22-41 Interrupt Compressor

Controller Area Network (MultiCAN) Controller

22.9.5.1 Service Request Control Registers

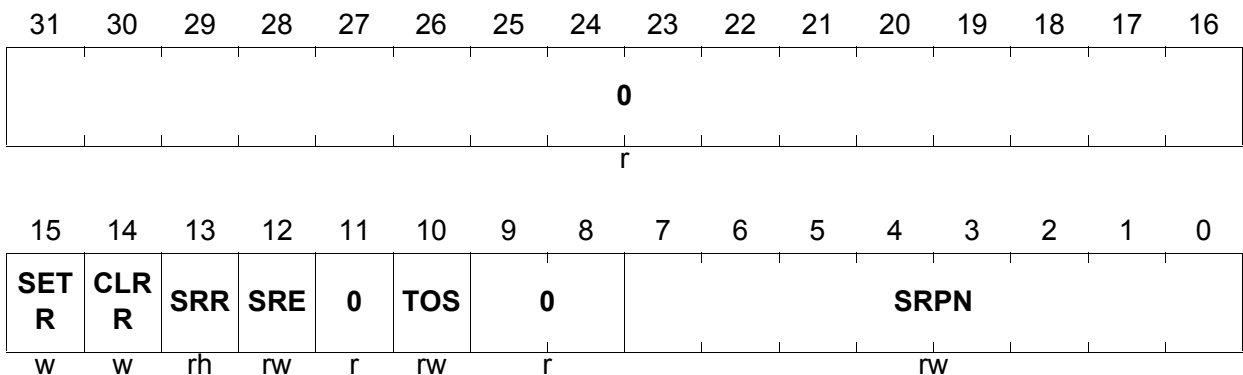
Each of the sixteen interrupt outputs INT_{Om} of the MultiCAN module is controlled by one service request control register.

CAN_SRC_m (m = 0-15)

CAN Service Request Control Register m

(0FC_H-m*4_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SRPN	[7:0]	rw	Service Request Priority Number
TOS	10	rw	Type of Service Control
SRE	12	rw	Service Request Enable
SRR	13	rh	Service Request Flag
CLRR	14	w	Request Clear Bit
SETR	15	w	Request Set Bit
0	[9:8], 11, [31:16]	r	Reserved Read as 0; should be written with 0.

*Note: Additional details on service request nodes and the service request control registers are described in section **“Service Request Nodes”** on Page 14-3 of the TC1796 User’s Manual System Units part (Volume 1).*

Some of the sixteen interrupt outputs of the MultiCAN module can be used to trigger operations in the DMA controller and the GPTA module.

22.9.6 Parity Protection for CAN Memories

In the TC1796, the static RAM memories in the MultiCAN module which include message objects and TTCAN scheduler entries, are equipped with a parity error detection logic that makes it possible to detect parity errors. In case of a parity error a NMI is generated.

At a power-on reset operation the corresponding MultiCAN module memories are initialized automatically. Therefore, unlike other TC1796 on-chip memories the MultiCAN module memories must not be initialized by a user program before it can be enabled for parity error detection.

More details on the parity control for on-chip memories is described in [Section 5.5](#) on [Page 5-37](#).

Controller Area Network (MultiCAN) Controller

22.9.6.1 CAN Module Register Map

The complete MultiCAN module register address map as shown in [Figure 22-42](#) combines the MultiCAN register address map from [Page 22-60](#) and the TTCAN register address map from [Page 22-156](#). It additionally shows the general implementation-specific registers for clock control, module identification, and interrupt service request control and adds the absolute address information.

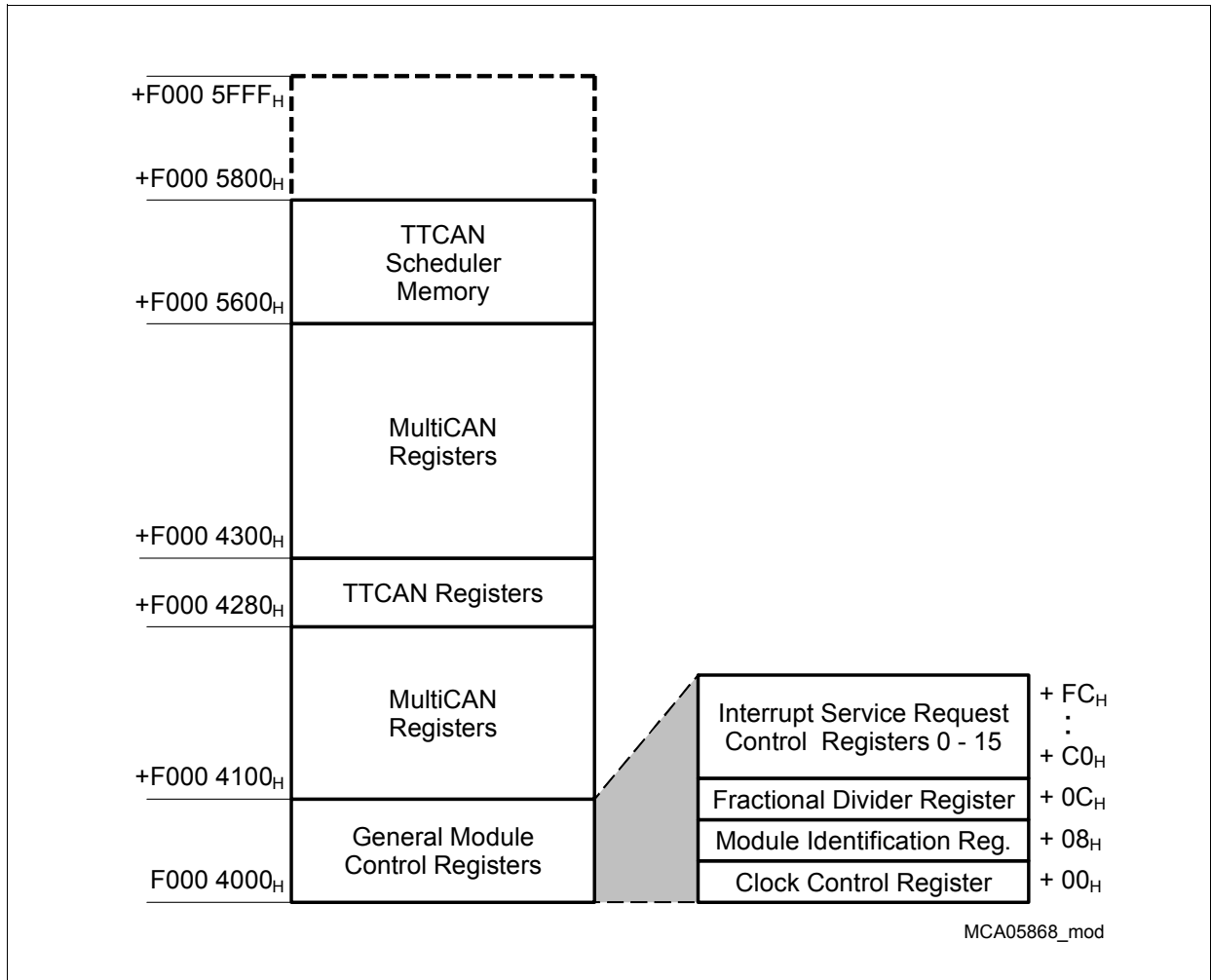


Figure 22-42 Complete MultiCAN Module Register Address Map with TTCAN

23 Micro Link Interface (MLI)

This chapter describes the Micro Link Interface module and the MLI protocol. It contains the following sections:

- Functional description of the MLI (see [Page 23-2](#))
- Module kernel description (see [Page 23-25](#))
- Operation the MLI module (see [Page 23-67](#))
- MLI kernel register descriptions (see [Page 23-75](#))
- Device implementation-specific descriptions and details (see [Page 23-124](#))

Note: The MLI kernel register names described in [Section 23.3](#) are referenced in the TC1796 User's Manual by the module name prefix "MLI0_" for the MLI0 interface and "MLI1_" for the MLI1 interface.

23.1 Functional Description

This chapter describes the functionality of the MLI interface.

- A general introduction to the interface (see [Page 23-2](#))
- The MLI frame structure for data exchange (see [Page 23-10](#))

23.1.1 General Introduction

The introduction comprises:

- An overview about the MLI (see [Page 23-2](#))
- Naming conventions (see [Page 23-3](#))
- A description of the MLI communication principles (see [Page 23-6](#))

23.1.1.1 MLI Overview

The Micro Link Interface (MLI) is a fast synchronous serial interface to exchange data between microcontrollers or other devices, such as stand-alone peripheral components. [Figure 23-1](#) shows how two microcontrollers are typically connected together via their MLI interfaces.

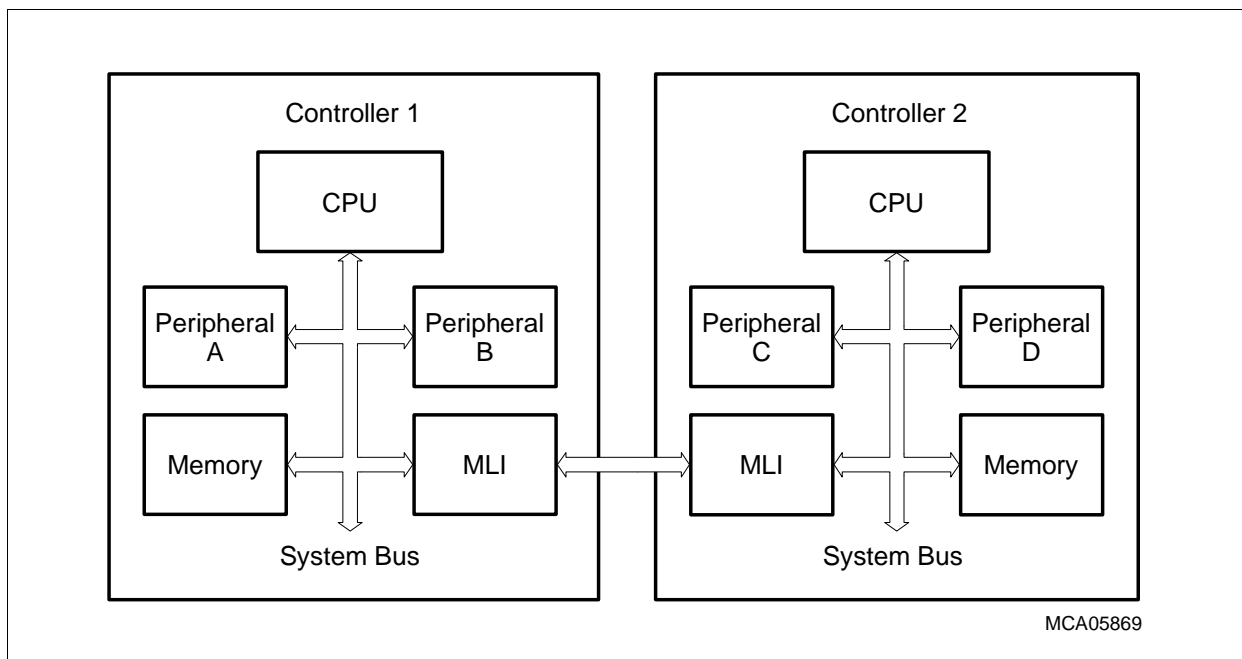


Figure 23-1 Typical Micro Link Interface Connection

Features

- Synchronous serial communication between an MLI transmitter and an MLI receiver
- Different system clock speeds supported in MLI transmitter and MLI receiver due to full handshake protocol (4 lines between a transmitter and a receiver)
- Fully transparent read/write access supported (= remote programming)

Micro Link Interface (MLI)

- Complete address range of target device available
- Specific frame protocol to transfer commands, addresses and data
- Error detection by parity bit
- 32-bit, 16-bit, or 8-bit data transfers supported
- Programmable baud rates
 - MLI transmitter baud rate: max. $f_{MLI}/2$ (= 37.5 Mbit/s @ 75 MHz module clock)
 - MLI receiver baud rate: max. f_{MLI}
- Address range protection scheme to block unauthorized accesses
- Multiple receiving devices supported

23.1.1.2 Naming Conventions**Local and Remote Controller**

The terms “Local” and “Remote” Controller are assigned to the two partners (microcontrollers or other devices with MLI modules) of a serial MLI connection. The controller with an MLI module initiating a data exchange or a control task is defined as Local Controller. Each data exchange and control task starts with a frame transmission of the Local Controller. The controller with an MLI module reacting on received data exchange requests or executing control tasks is defined as Remote Controller. The terms “Local” and “Remote” are independent of the direction of the information flow (transmission or reception), except for Read Frames (always transmitted by the Local Controller) and Answer Frames (always transmitted by the Remote Controller).

Due to the full duplex operation capability of an MLI module (independent transmitter and receiver), each microcontroller with an MLI module is able to operate as a Local Controller (e.g. for data transmission) as well as a Remote Controller (e.g. for data reception) at the same time.

Transmitting and Receiving Controller

The terms “transmitting” and “receiving” controller are referring to the direction of the information flow. These terms are independent from the terms “Local” and “Remote”. For example, the initialization of a bi-directional MLI connection between two controllers (or between a controller and a stand-alone device) is always controlled and initiated by one controller (named Local), although during this phase, both MLI participants can transmit and receive frames.

Due to the full duplex operation capability of the MLI module (independent transmitter and receiver), each microcontroller with an MLI module is able to operate as a transmitting controller as well as a receiving controller at the same time.

Transfer Window

A Transfer Window is an address space in the address map of the transmitting controller. Transfer Windows are typically assigned to a fixed address space (base address and

Micro Link Interface (MLI)

size). The Transfer Windows are the logical data inputs for the MLI transmitter. Data write actions via MLI are initiated by a write access to a Transfer Window, whereas data read actions are started by a read access from a Transfer Window.

Each MLI module supports up to four independent Transfer Windows, one for each pipe. In the implementation of a specific device, a Transfer Window can appear at several locations in the address map. Here, each Transfer Window can be accessed at two different address ranges with two different window sizes (one 64 Kbyte and one 8 Kbyte area for each Transfer Window), leading to:

- Four Small Transfer Windows STW with 8 Kbyte address range each and
- Four Large Transfer Windows LTW with 64 Kbyte address range each

If the address areas of the four small transfer windows together form a 64 Kbyte address range (aligned to its size), then a single MLI pipe can pass through a device with several MLI modules before being split up in a target device into 4 independent Remote Windows.

Remote Window

A Remote Window is an area in the address space of the receiving controller. Remote Window parameters (base address and size) of the receiving controller are programmable by the transmitting microcontroller by MLI transfers, independently for each pipe. Each Remote Window of a receiving controller is related to specific Transfer Window of the transmitting controller.

The Remote Windows are the logical data outputs of the MLI receiver. If enabled, the MLI module can automatically execute the requested data transfer to/from the defined address location in the Remote Window. If the automatic data handling is disabled, the offset and the data are available in the MLI receiver registers and have to be handled by software. Remote windows can not be accessed by read or write accesses by software of the Remote Controller (either the data is automatically transferred or it is located in receiver registers).

Pipe

A pipe defines the logical connection between a Transfer Window in the transmitting controller and the associated Remote Window in the receiving controller. The MLI protocol supports four independent pipes.

Frame

A frame is a contiguous set of bits forming a message sent by an MLI transmitter to an MLI receiver.

A **Normal Frame** is a frame used for data exchange between a transmitting and a receiving controller (read request and write data from a Local Controller to a Remote

Micro Link Interface (MLI)

Controller, as well as the answer to a read request back to the Local Controller). Base address copy frames are also considered as Normal Frames.

A **Command Frame** contains information about the receiver setting or triggers actions in the MLI receiver.

Offset

The offset is an address distance relative to the base address of the Transfer Window in the transmitting controller and the base address of the Remote Window in the receiving controller. For example, a write access to the 10th byte of the Transfer Window is transferred to a write to the 10th byte of the Remote Window.

The offset of a write access to a Transfer Window is also called write offset, whereas a read offset is related to a read access from a Transfer Window.

23.1.1.3 MLI Communication Principles

The communication principle of the MLI modules allows data to be transferred between a Local and a Remote Controller without intervention of a CPU in the Remote Controller. Data transfers are always triggered in the Local Controller by read or write operations to an address location in a Transfer Window. All control tasks, address and data transmissions that are required for the data transfer/request between Local and Remote Controller can be handled autonomously by the two connected MLI modules.

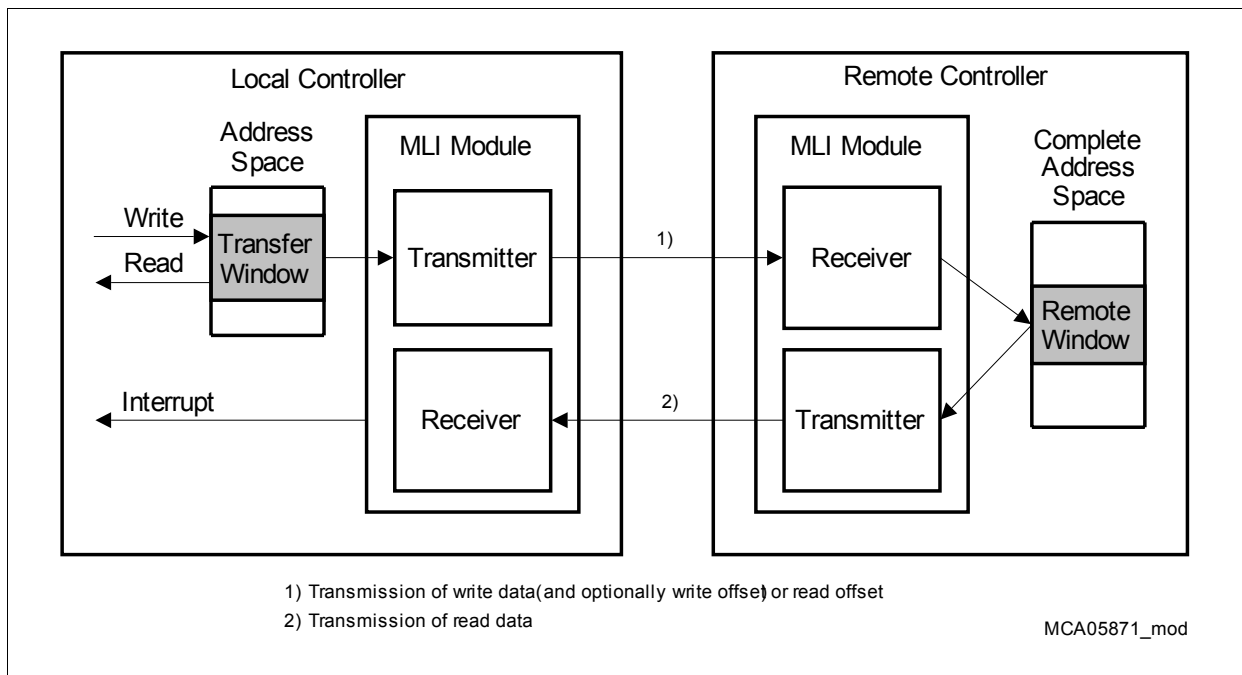


Figure 23-2 MLI Communication Principles

Write Access to a Transfer Window

A write access to a location within a Transfer Window of the transmitting (Local) controller is detected by the MLI transmitter. This detection initiates a transfer of the data that has been written to the Transfer Window together with the write offset to the MLI of the receiving controller. The receiving controller stores the data internally and can also automatically place the data in the Remote Window of the receiving controller (at the address location defined by the write offset plus the base address).

Read Access from a Transfer Window

A read access from a location of a Transfer Window in the Local Controller is detected by the MLI transmitter and delivers dummy data. This detection initiates a transfer of the read offset from the Local microcontroller to the MLI receiver to request data from the Remote Controller. This data can be automatically read or prepared by a CPU in the Remote Controller. When the requested data is available in the Remote Controller, it is

introduced into the data stream back to the Local Controller (Answer Frame). Then, the CPU in the Local Controller is informed by an MLI event that the requested data is now available and can be read.

Transfer Window Organization

Figure 23-3 shows an example of the organization of Transfer Windows and Remote Windows with a possible assignment in Local and Remote Controller. Each of the four pipes assigns one Transfer Window to one Remote Window with its base address and window size. For reasons of simplicity, a pipe to a Remote Window is only shown either from a LTW or from a STW, although each Transfer Window can be accessed at both address locations, its LTW and its STW.

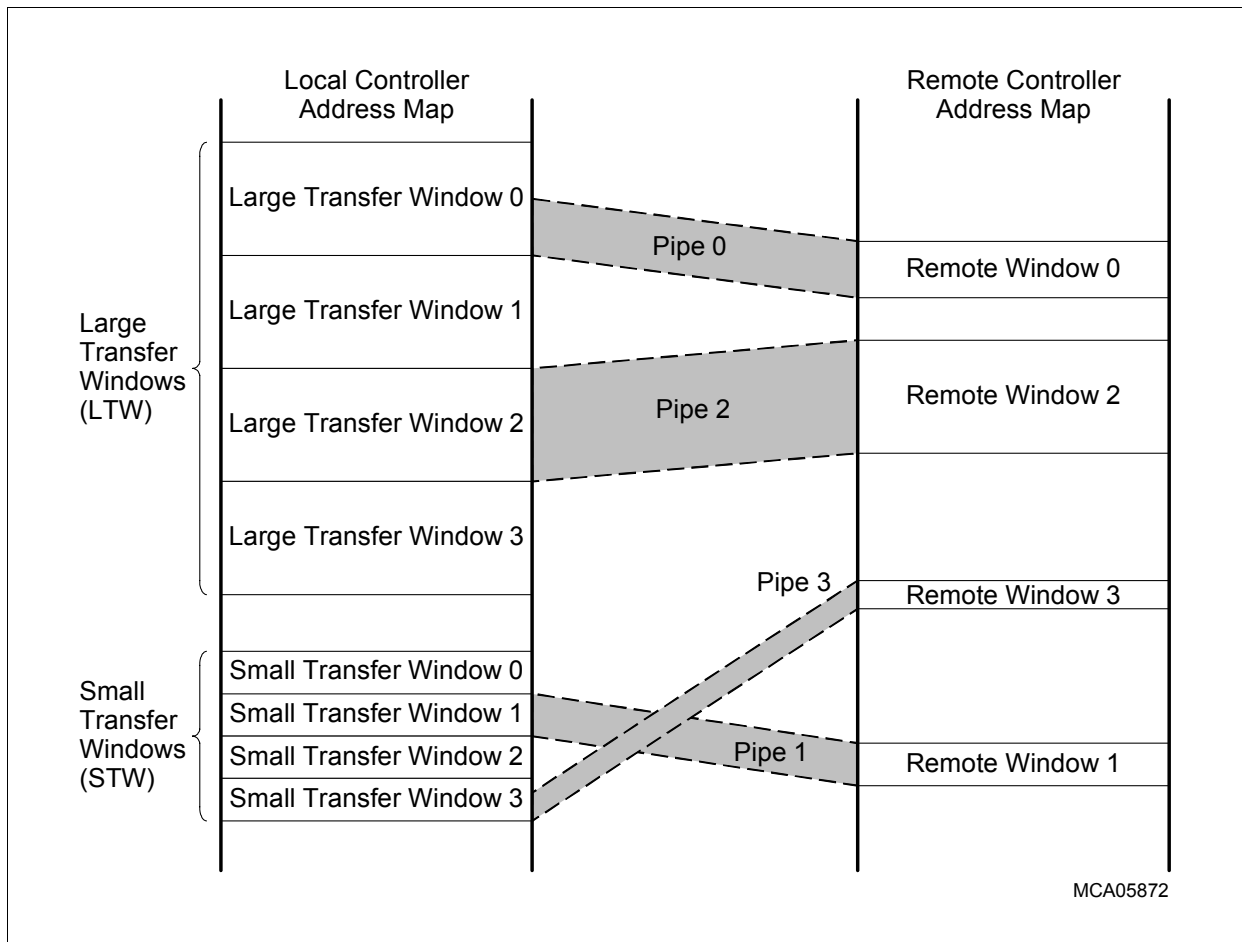


Figure 23-3 Transfer/Remote Window Assignment Example

During initialization of the pipes, base addresses and sizes of the Remote Windows are transmitted from the Local Controller to the Remote Controller. In the example of **Figure 23-3**, pipe 1 and pipe 2 cover the full range of their Transfer and Remote Windows. The ranges of the Remote Windows of pipe 0 and pipe 3 are sub-ranges of the related Transfer Windows.

Micro Link Interface (MLI)

The location of a Transfer Window (base address and size) in the Local Controller is always fixed in a specific product device. Remote windows can be freely moved and located within the address space of the receiving controller. They are used to overlay address ranges of peripheral modules or internal memories.

Remote Window Address Generation

Figure 23-4 shows the generation of the Remote Window address ranges, with fixed base address part and additional variable address part. The variable address part is determined by the available address area for each Remote Window (also named buffer size, value of BSx = buffer size for Remote Window x indicates how many address bits are variable, defining the available address range).

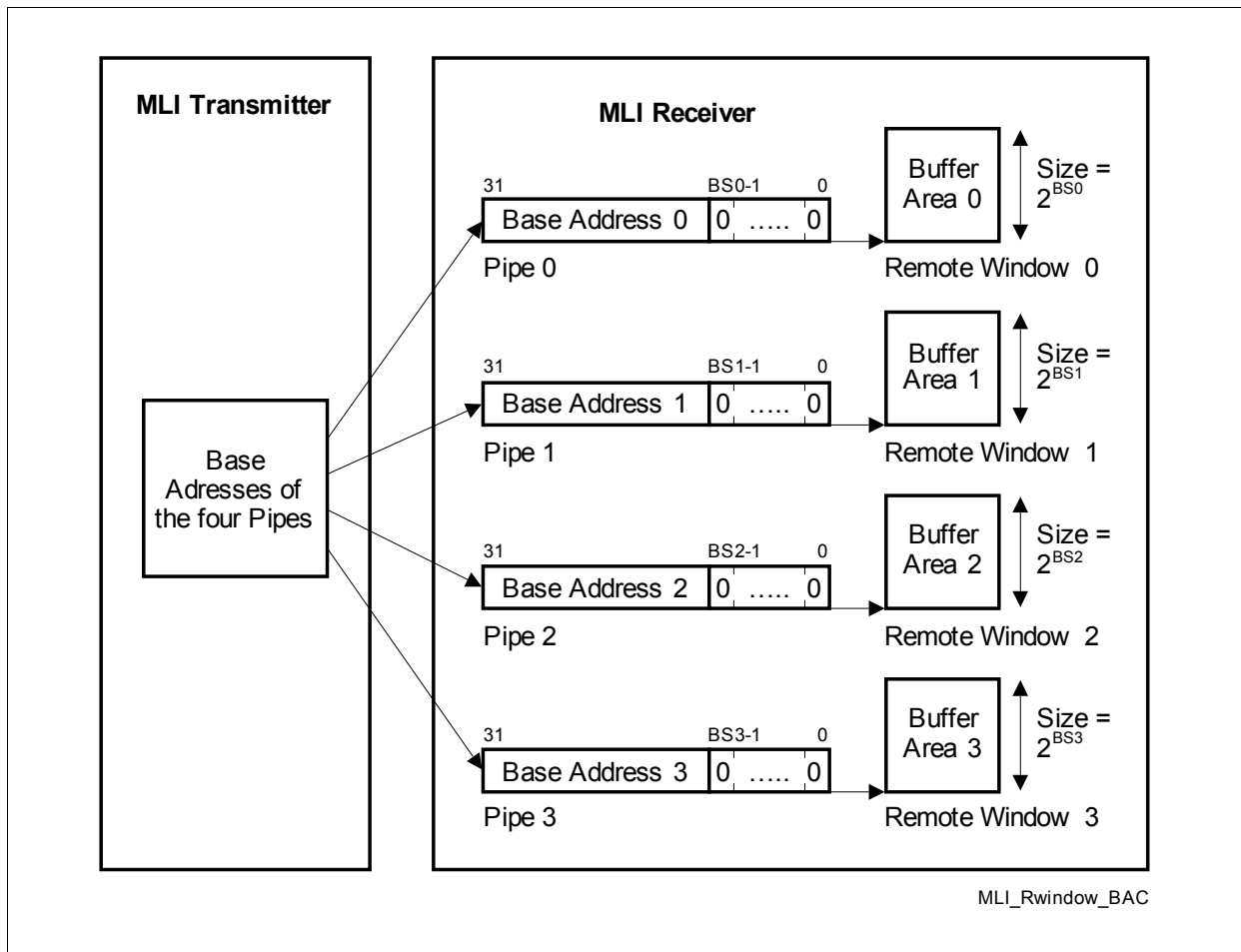


Figure 23-4 Base Address Definition of Remote Windows

23.1.2 MLI Frame Structure

A frame is a message sent by an MLI transmitter to an MLI receiver. Depending on the desired behavior, different frame types exist:

- Copy Base Address Frame to define location and size of a Remote Window (see [Page 23-12](#))
- Write Offset and Data Frame to transmit the write offset and the write data (see [Page 23-13](#))
- Optimized Write Frame to transmit write data without write offset in case of an address prediction match (see [Page 23-14](#))
- Discrete Read Frame to transmit read request with the read offset (see [Page 23-15](#))
- Optimized Read Frame to transmit the read request without read offset in case of an address prediction match (see [Page 23-16](#))
- Command Frame to transmit a command, e.g. setup information or MLI service request generation (see [Page 23-17](#))
- Answer Frame to transmit the data previously requested by a Read Frame (see [Page 23-18](#))

The local/remote structure of an MLI connection between two microcontrollers requires a transmitter unit and a receiver unit in both MLI modules (local and remote) for communication.

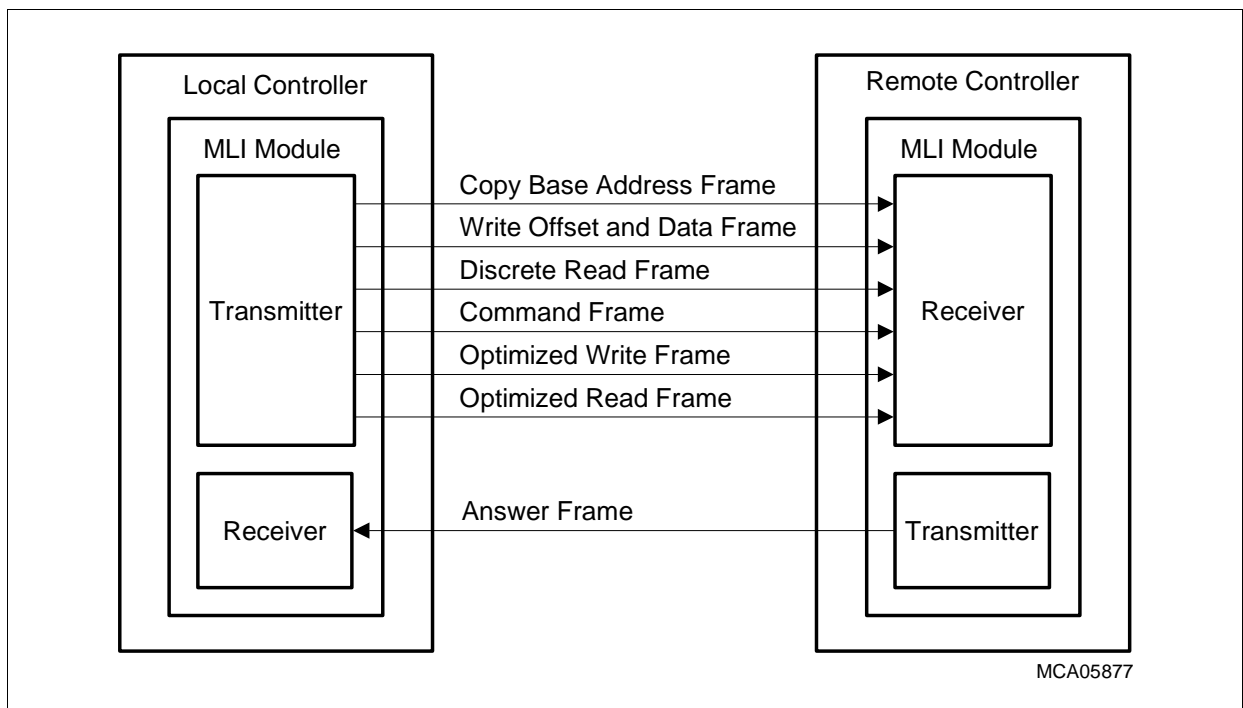


Figure 23-7 Logic Frame Assignment to Local/Remote Controller

23.1.2.1 General Frame Layout

The general layout of a frame is shown in [Figure 23-8](#). It contains the following parts:

- A frame starts with a 4-bit header field that contains a 2-bit frame code (FC) and a 2-bit pipe number (PN).
- The data field can contain address, data, or control information. The width of the data field depends on the frame type.
- The frame is terminated by a parity bit (P) with even parity (see [Page 23-24](#)), calculated over header and data field bits.

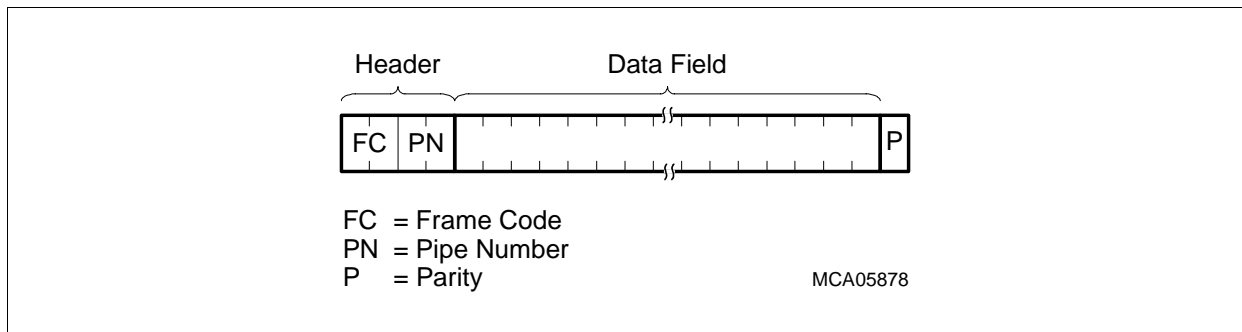


Figure 23-8 General Frame Layout

The frame code (FC) determines the frame type of the transmitted frame. The pipe number (PN) indicates the pipe that is related to the frame content (the value of PN is defined as 00_B for pipe 0, 01_B for pipe 1, 10_B for pipe 2, and 11_B for pipe 3).

The FC parameter is coded according to [Table 23-1](#). If more than one frame type is defined with the same frame code value (see FC = 01_H , 10_H or 11_H), the width of the received frame defines the type. The value given by m in the table below represents the number of address bits transferred as offset (defined by the buffer size BS_x of the Remote Window x).

Table 23-1 Frame Code Definition

Frame Code FC	Frame Type	Data Field Width [bit]	Description see
00_B	Copy Base Address Frame	32	Page 23-12
01_B	Write Offset and Data Frame	$8+m$, $16+m$, or $32+m$	Page 23-13
	Discrete Read Frame	$2+m$	Page 23-15
10_B	Command Frame	4	Page 23-17
	Answer Frame	8, 16, or 32	Page 23-18
11_B	Optimized Write Frame	8, 16, or 32	Page 23-14
	Optimized Read Frame	2	Page 23-16

23.1.2.2 Copy Base Address Frame

With a Copy Base Address Frame, the two parameters of a Remote Window are transferred from the transmitting controller to the receiving controller to initialize or to redirect the Remote Window.

The Copy Base Address Frame contains the following parts:

- Header:
The header starts with frame code FC = 00_B followed by the pipe number PN of the pipe targeted by the transmitted base address bits and the size code.
- Remote Window address location:
The 28 most significant bits of the 32-bit base address bits can be programmed by the transmitting controller (the 4 LSBs are considered as 0). The base address of a Remote Window has to be aligned to its size, e.g. a window of 1 Kbyte has to start at 1Kbyte address boundaries.
- Remote Window size:
The size is defined by the 4-bit coded buffer size BS. The maximum size is 64 Kbytes.
- Parity bit P

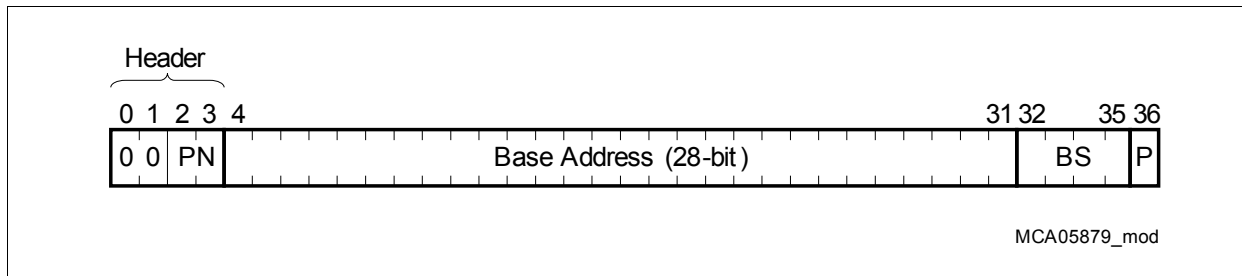


Figure 23-9 Copy Base Address Frame

Table 23-2 BS Coding

Buffer Size Code BS	Remote Window Size (also named Buffer Size)	Number m of Offset Bits
0000 _B	2 bytes	m = 1
0001 _B	4 bytes	m = 2
...
1110 _B	32 Kbytes	m = 15
1111 _B	64 Kbytes	m = 16

More details on the Copy Base Address Frame handling of the MLI module are described on [Page 23-26](#).

23.1.2.3 Write Offset and Data Frame

A Write Offset and Data Frame is used by the transmitting controller to send an address offset and data to the receiving controller. This frame is initiated by a write operation to one of the Transfer Windows in the transmitting controller.

The Write Offset and Data Frame contains the following parts:

- Header:
The header starts with frame code FC = 01_B followed by the pipe number PN of the Transfer Window that has been the target of the write operation.
- m-Bits of write offset:
These bits define the write offset. The value of m depends on the size of the Remote Window, defined by the Copy Base Address Frame (m = 1-16).
- Write data field:
The write data field can be 8-bit, 16-bit, or 32-bit wide, depending on the data width of the write access to the Transfer Window.
- Parity bit P

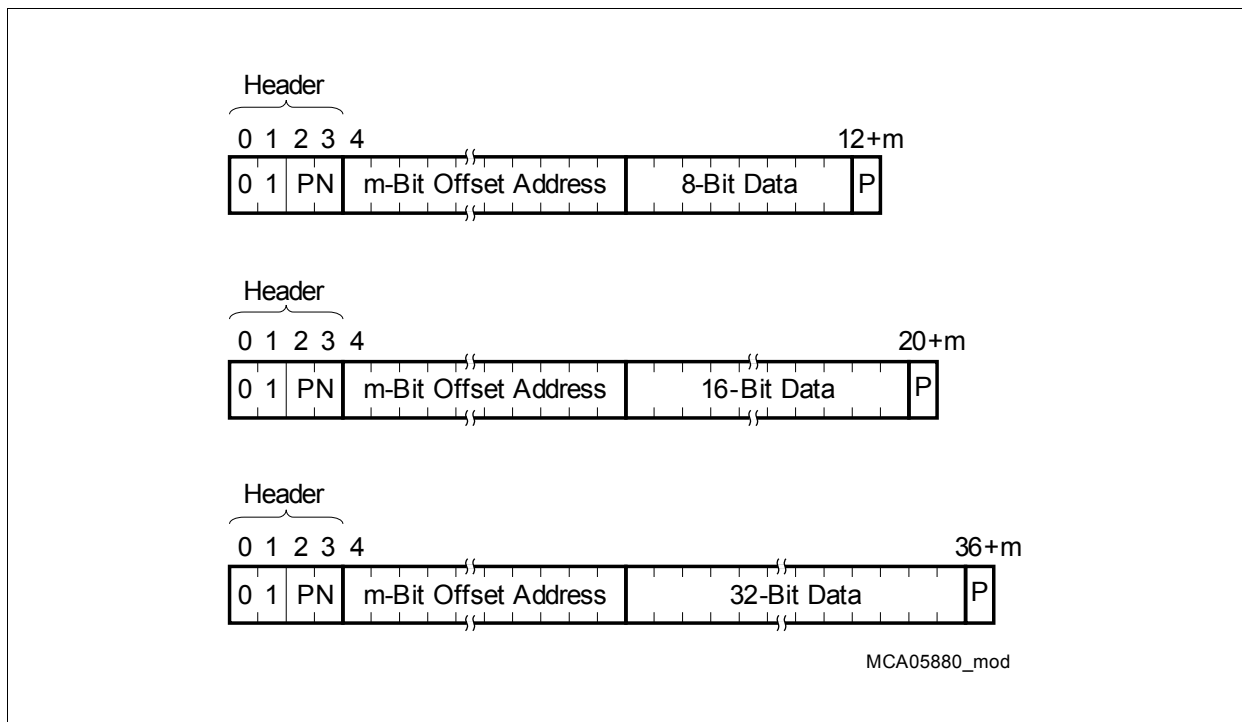


Figure 23-10 Write Offset and Data Frame

More details on the Write Offset and Data Frame handling of the MLI module are described on [Page 23-28](#).

23.1.2.4 Optimized Write Frame

An Optimized Write Frame is used by the transmitting controller to send 8-bit, 16-bit, or 32-bit wide data to the receiving controller. This frame is initiated by a write operation to one of the Transfer Windows in the transmitting controller. In contrast to a Write Offset and Data Frame, no write offset is transmitted because the offset address for the write data can be predicted and calculated by the receiving controller. An Optimized Write Frame allows a higher data bandwidth than Write Offset and Data Frames, because they are shorter. An optimized frame is only possible if the predicted address matches with the actually written one.

The Optimized Write Frame contains the following parts:

- Header:
The header starts with frame code FC = 11_B followed by the pipe number PN of the Transfer Window that has been the target of the write operation.
- Write data field:
The write data field can be 8-bit, 16-bit, or 32-bit wide, depending on the data width of the write access to the Transfer Window.
- Parity bit P

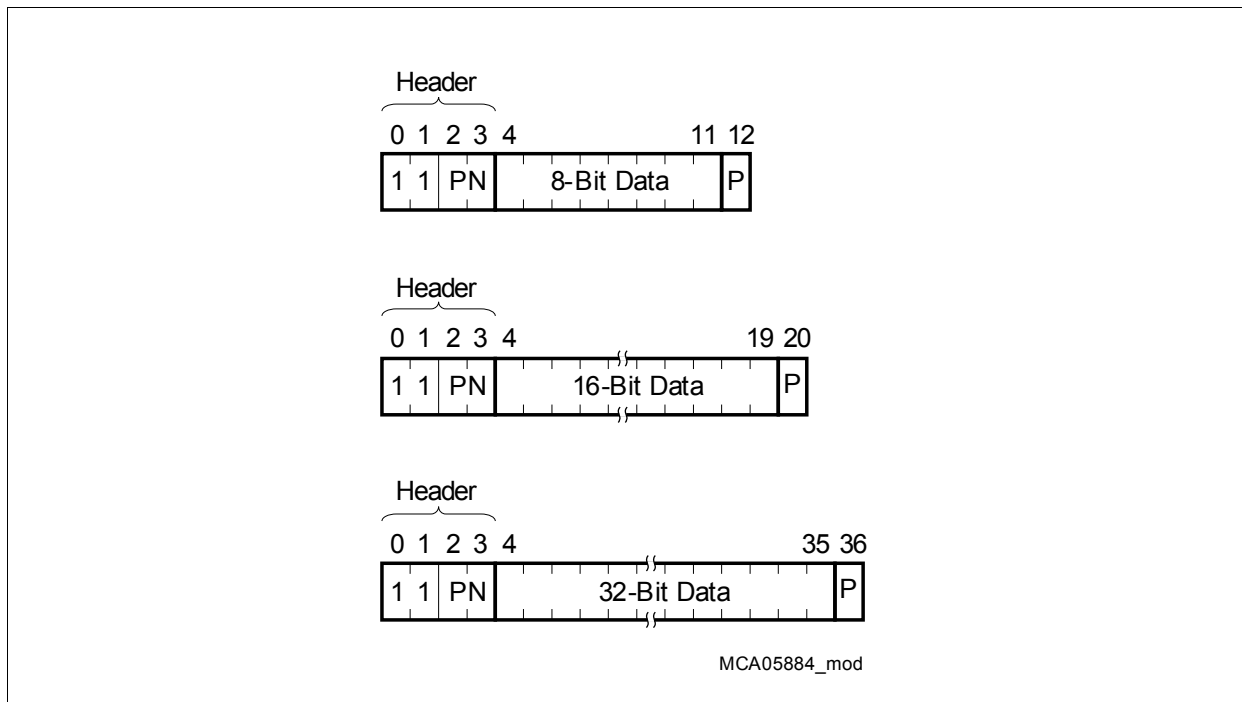


Figure 23-11 Optimized Write Frame

More details on the Optimized Write Frame handling of the MLI module are described on [Page 23-28](#).

23.1.2.5 Discrete Read Frame

A Discrete Read Frame is used by the Local Controller to request data to be read from the Remote Window in the Remote Controller. If the data is available, the Remote Controller typically responds to this request by sending an Answer Frame with the requested read data back to the Local Controller.

The Discrete Read Frame contains the following parts:

- Header:
The header starts with frame code FC = 01_B followed by the pipe number PN of the Transfer Window that has been the target of the read operation.
- m-Bits of write offset:
These bits define the read offset. The value of m depends on the size of the Remote Window, defined by the Copy Base Address Frame (m = 1-16).
- Data Width DW:
The data width DW indicates if the read from the Transfer Window was a 8-bit, 16-bit, or 32-bit read action. It defines how many bytes have to be delivered to the Local Controller by the Answer Frame.
- Parity bit P

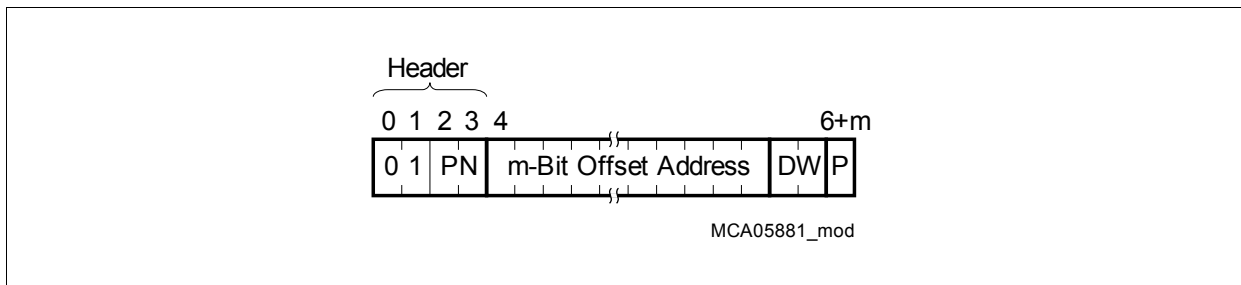


Figure 23-12 Discrete Read Frame

Table 23-3 Data Width DW Coding

Data Width DW	Number of Data Bits to be transferred
00 _B	8-bit read access
01 _B	16-bit read access
10 _B	32-bit read access
11 _B	reserved for future use

More details on the Discrete Read Frame handling of the MLI module are described on [Page 23-32](#).

23.1.2.6 Optimized Read Frame

An Optimized Read Frame is used by the Local Controller to request 8-bit, 16-bit, or 32-bit wide data from the Remote Controller without sending any offset address. The address for the requested data can be predicted and calculated by the MLI receiver of the Remote Controller.

The Optimized Read Frame contains the following parts:

- Header:
The header starts with frame code FC = 11_B followed by the pipe number PN of the Transfer Window that has been the target of the read operation.
- Data Width DW:
The data width DW indicates if the read from the Transfer Window was a 8-bit, 16-bit, or 32-bit read action. It defines how many bytes have to be delivered to the Local Controller by the Answer Frame. Same coding as for the Discrete Read Frame.
- Parity bit P

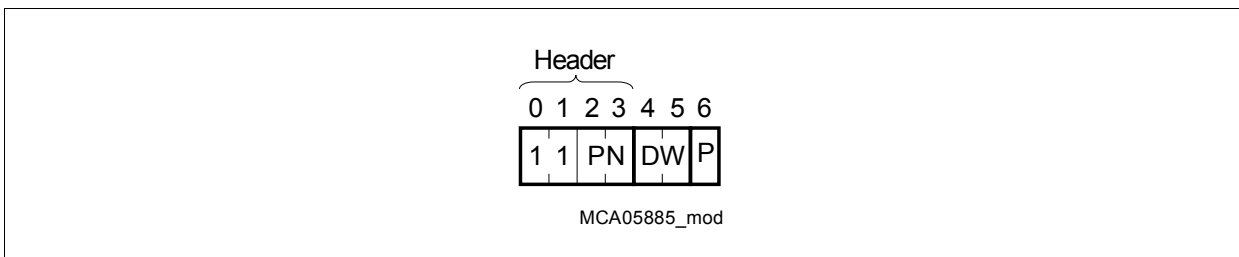


Figure 23-13 Optimized Read Frame

More details on the Optimized Read Frame handling of the MLI module are described on [Page 23-28](#).

23.1.2.7 Command Frame

The transmitting controller is able to initiate control actions to be executed by the receiving controller by sending a Command Frame.

The Command Frame contains the following parts:

- Header:
The header starts with frame code FC = 10_B followed by the pipe number PN. The pipe number defines the type of command to be executed.
- Command Code CMD:
Pipe number PN and a 4-bit CMD field are used for command coding. The command coding of some control actions is fixed, but free programmable software commands can also be defined (with PN = 11_B). The coding of the command bit field is pipe-specific and depends on the transmitted pipe number x.
- Parity bit P

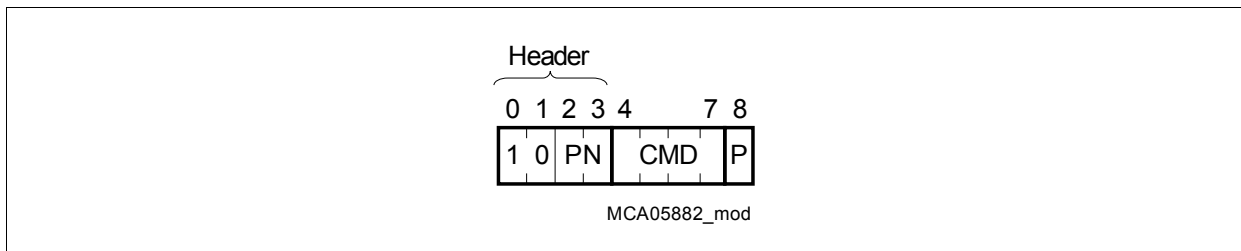


Figure 23-14 Command Frame

Table 23-4 PN for Command Coding

Pipe Number PN	Command Type
00 _B	Activate MLI service request or other control signal(s) of the receiving controller. The definition which signal becomes activated is defined by CMD. The usage of these lines depends on the implementation.
01 _B	Define delay for parity error indication in the receiving controller. The delay in RCLK cycles is defined by the value of CMD.
10 _B	Control of internal functions of the receiving controller. The value of CMD indicates which function is controlled. The coding of CMD and the control mechanisms depend on the implementation.
11 _B	Freely programmable software command.

More details on the Command Frame handling of the MLI module are described on [Page 23-39](#).

23.1.2.8 Answer Frame

An Answer Frame is used by the Remote Controller to send 8-bit, 16-bit, or 32-bit wide data to the Local Controller. The Answer Frame is the only frame that is transmitted within a logic Local/Remote Controller assignment from the Remote Controller to the Local Controller. It is the answer to a Discrete Read Frame or an Optimized Read Frame that has been sent by the Local Controller to request data from the Remote Controller.

The Answer Frame contains the following parts:

- Header:
The header starts with frame code FC = 10_B followed by the pipe number PN. The value of PN is taken from the Read Frame that has triggered the Answer Frame.
- Read data field:
The read data field can be 8-bit, 16-bit, or 32-bit wide, depending on the data width requested by the Read Frame that triggered the Answer Frame.
- Parity bit P

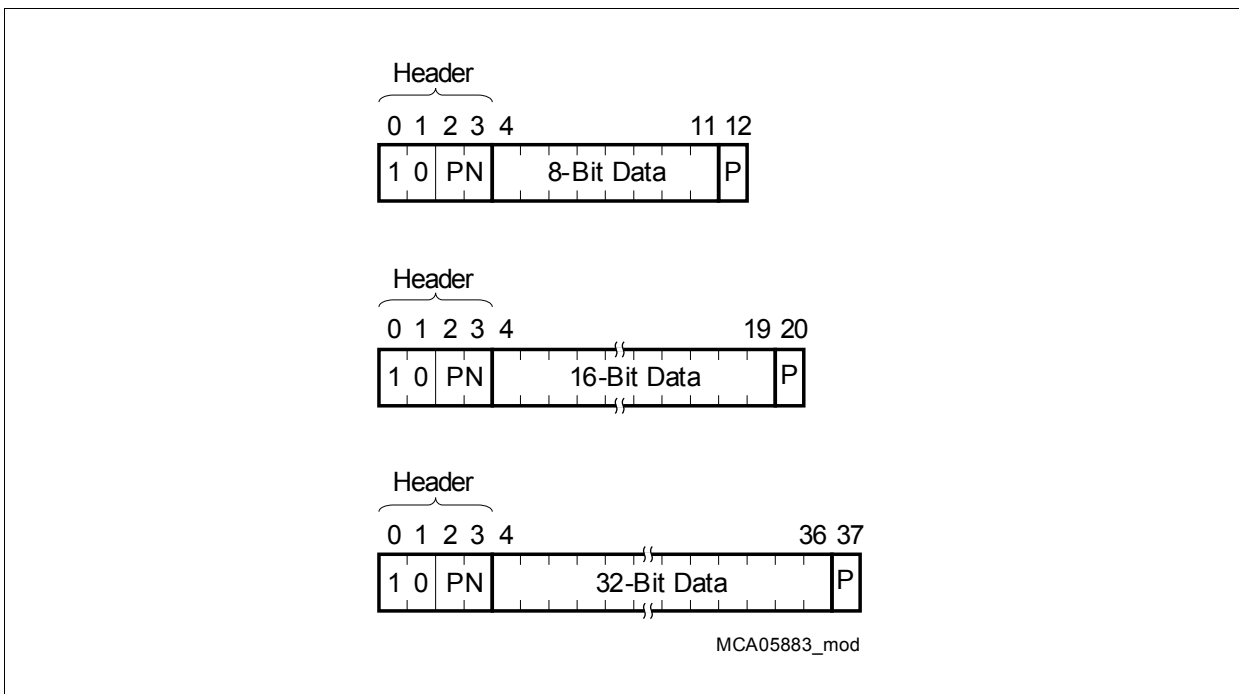


Figure 23-15 Answer Frame

More details on the Answer Frame handling of the MLI module are described on [Page 23-37](#).

23.1.3 Handshake Description

The description of the transmitter/receiver signal handshaking refers to an MLI connection between an MLI transmitter and an MLI receiver. MLI module transmitter I/O signals are indicated with prefix “T” and MLI receiver I/O signals are indicated with the prefix “R”. The 4-line MLI bus between a transmitter and a receiver outside the controllers uses signal names without any prefix.

In order to lay emphasis where a signal is generated or sampled, actions taken by the transmitter are described referring to signals with the prefix “T”, whereas receiver actions are referring to signals with the prefix “R”.

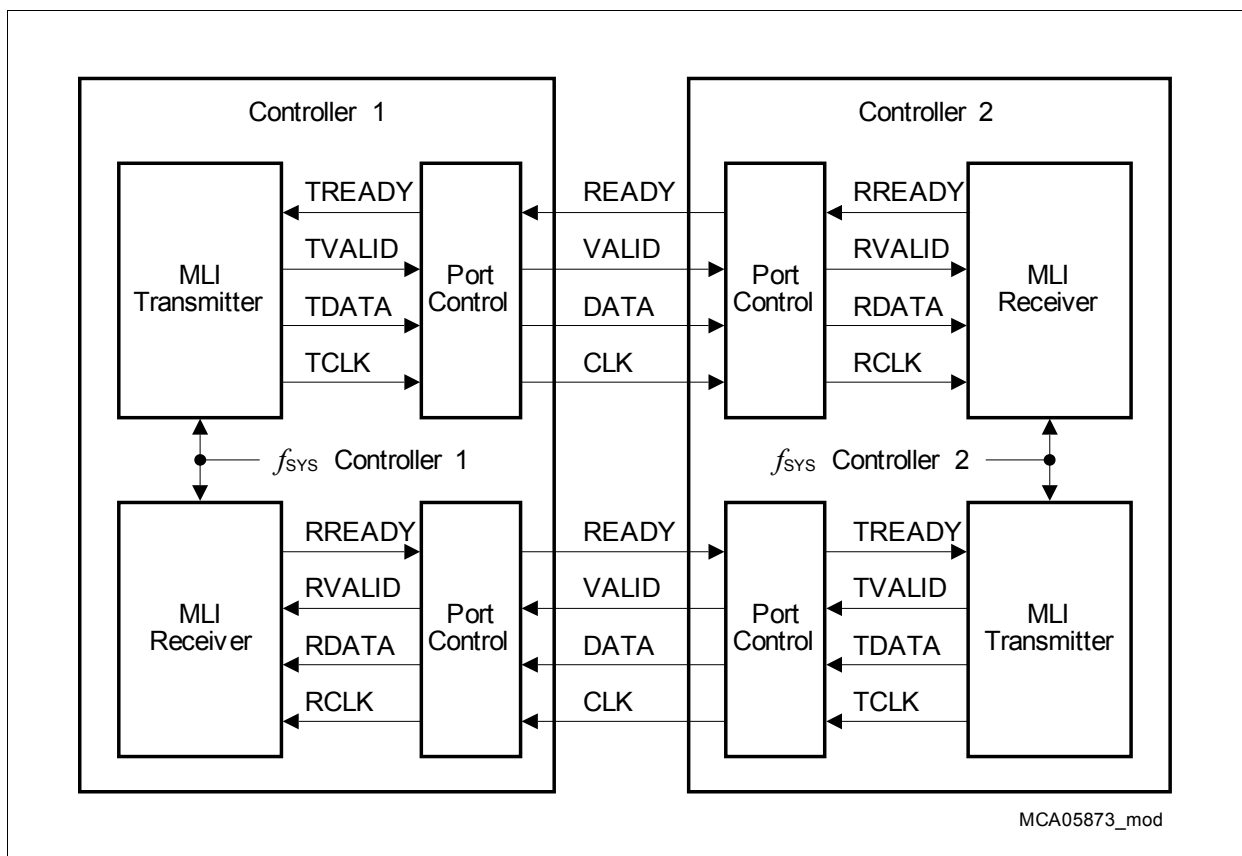


Figure 23-16 Transmitter/Receiver Signal Definitions

The MLI connection allows high data rates and, at the same time, supports significant signal propagation delays between the transmitter and the receiver. As shown in [Figure 23-16](#), each output signal passes through the port stage, reaches the physical interface line between the MLI modules, enters via an input stage and can be finally evaluated. All these steps introduce an accumulating propagation delay. In standard synchronous serial connections (such as SPI), this delay limits the reachable baud rate to a few Mbit/s (closed-loop delay problem). In order to support higher baud rates than a standard SPI, the MLI protocol is based on a full handshake (READY-VALID) to deal

Micro Link Interface (MLI)

with propagation delays in the range of some shift clock cycles and to avoid the closed-loop delay limitations of an SPI connection.

In a full handshake, each edge of the handshake signals has a defined meaning and the sequence of edges is clearly specified.

As a result, the propagation delays do not directly limit the MLI baud rate. Therefore, the points in time when a signal is generated, when it is visible on the physical interface line, or when it is evaluated have to be considered independently. This is done by defining 3 different names for a signal, referring to the 3 significant locations:

- The place where it is generated, also in relation to the generation clock edge
- The physical interface line where it can be observed
- The place where it is evaluated, also in relation to the evaluation clock edge

If a Local Controller should be connected to more than one Remote Controller, the transmitter signals CLK and DATA can be used as broadcast signals (parallel connection to the Remote Controllers), whereas the handshake signals VALID and READY have to be established as independent signal pairs for each device. As a result, a Local Controller only needs one CLK and one DATA output, but an individual set of READY and VALID handshake signals for each Remote Controller. Please note that Read Frames and Answer Frames are based on an established connection between a Local and a Remote Controller (because the Answer Frame is the only frame sent back to the Local Controller). Therefore, switching between several Remote Controllers can only be done while no read request is pending in the Local Controller. If no Read Frames are used by the Local Controller, frames can be sent out in parallel to all Remote Controllers if their READY signals are all respected.

If a Remote Controller should be connected to several Local Controllers, it may have several DATA and CLK inputs in addition to the READY-VALID signal sets. Please note that an active switching of a Remote Controller between several Local Controllers requires that all Local Controllers have the information which connection is active.

In any case, switching between Local and Remote Controllers is not allowed while frame transmission is in progress.

23.1.3.1 Handshake Signals

The synchronous serial frame transfer from an MLI transmitter to an MLI receiver is based on the following 4 signals (the MLI protocol only defines the signal transitions, but neither the signaling level nor the driver characteristics):

- Shift clock CLK:
This signal is used as serial shift clock that is generated by the transmitter during the complete frame transfer (TVALID is active) and until the end of ready delay time. Signal TCLK can also be generated while no frame is transferred. In this case, the receiving controller can use the incoming RCLK receiver signal as base for its internal clock generation.
The transmitter signals are always referring to the rising edge of TCLK, so TREADY is sampled and the output signals TDATA and TVALID are changing with the rising edge of TCLK.
The MLI receiver actions refer to the falling edges of its RCLK input. The receiver samples the RVALID and RDATA signals and outputs its RREADY line with the falling edges of RCLK.
- Shift data DATA:
This signal represents the transmit data TDATA transferred from the MLI transmitter to the MLI receiver input RDATA. Changes on transmitter side take place with rising edges of TCLK, whereas sampling on the receiver side takes place with falling edges of RCLK.
- Transmitter valid handshake VALID:
This signal indicates the start and the end of each frame. It is active (1-level) during a frame transmission and passive (0-level) while no frame is transferred. Changes of TVALID on transmitter side take place with rising edges of TCLK, whereas sampling of RVALID on the receiver side takes place with falling edges of RCLK.
An activation of TVALID to start a new frame can only take place if TREADY is 1.
- Receiver ready handshake READY:
This signal indicates that the receiver is ready for a data transfer. Additionally, this line is used to indicate reception errors (parity error indication). Changes of RREADY on receiver side take place with falling edges of RCLK, whereas sampling of TREADY on the transmitter side takes place with rising edges of TCLK.

23.1.3.2 Error-free Handshake

A transmission can be started by an MLI transmitter when the MLI receiver is ready to receive data indicated by RREADY = 1 by the receiver. When the MLI transmitter detects TREADY = 1 and starts its transmission, TVALID is asserted to 1 level while a frame transfer is in progress. When the MLI receiver has detected the 0-to-1 transition of the RVALID signal it will de-assert RREADY back to 0 (transmission start acknowledged by receiver). At the end of the frame transmission, the MLI transmitter also de-asserts signal TVALID back to 0 and checks if the TREADY signal is at 0 level,

too. This check is used as life-sign of the receiver and the MLI transmitter can detect whether the receiver is able to react in-time to the transmitter actions (see also [Chapter 23.1.3.4](#)).

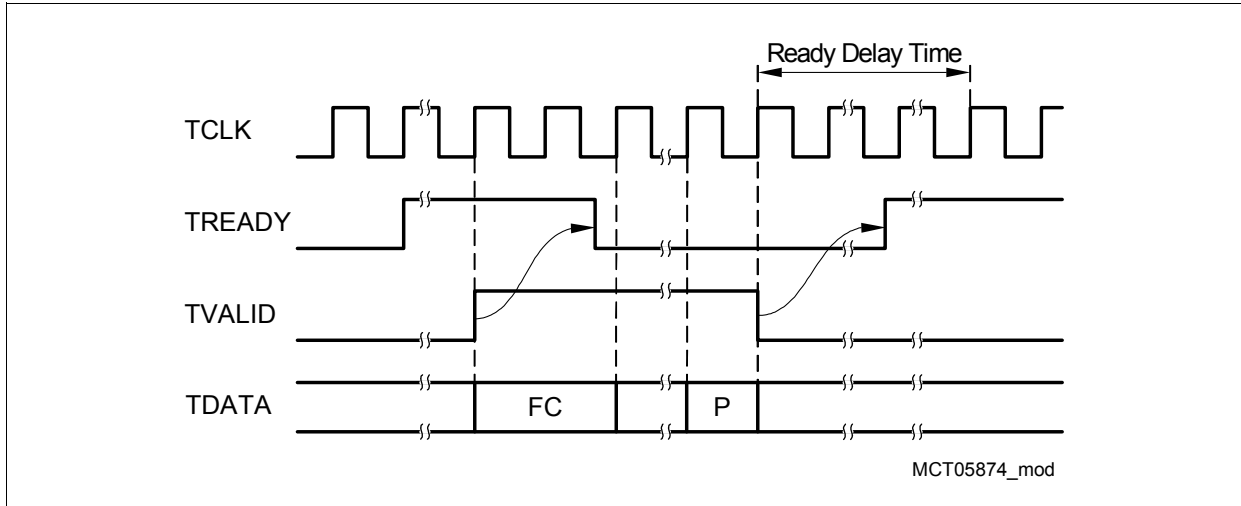


Figure 23-17 MLI Handshake without Error Indication

23.1.3.3 Ready Delay Time

In order to support significant propagation delays, the handshake signal TREADY is evaluated with respect to TVALID and TCLK in an time interval called Ready Delay Time after the end of the frame (see [Figure 23-17](#)). The length of the Ready Delay Time is programmable, defining the size of the time interval.

When a transmission is finished (RVALID becomes 0), the MLI receiver checks the received frame for correct reception (parity error). If no parity error has been detected, the MLI receiver asserts its RREADY signal again to 1 to indicate the correct reception with the next falling edge of RCLK. The MLI transmitter checks its TREADY input with each rising edge of TCLK after TVALID has become 0 and increments a counter. This counter is started from 0 at the end of a frame transmission (TVALID becomes 0) and counts TCLK periods (Ready Delay Time Counter). If the condition TREADY = 1 is detected before the programmed Ready Delay Time has elapsed, the MLI receiver has indicated a frame reception without parity error to the MLI transmitter. In this case, a new frame transmission can be started. The transfer handshake signalling without a parity error indication is shown in [Figure 23-17](#).

[Figure 23-18](#) shows the transfer handshake if a parity error condition has been detected by the MLI receiver and indicated to the MLI transmitter. In this case, the receiver waits a programmable number of RCLK clock cycles before setting RREADY to 1. If the TREADY = 1 condition is detected by the transmitter after the ready delay has elapsed, a parity error has been indicated by the MLI receiver. In this case, it is assumed that the MLI receiver has detected a frame with a parity error and has discarded the frame. The

transmitter automatically sends the last frame again after a parity error indication. Optionally, this MLI event can activate a service request output.

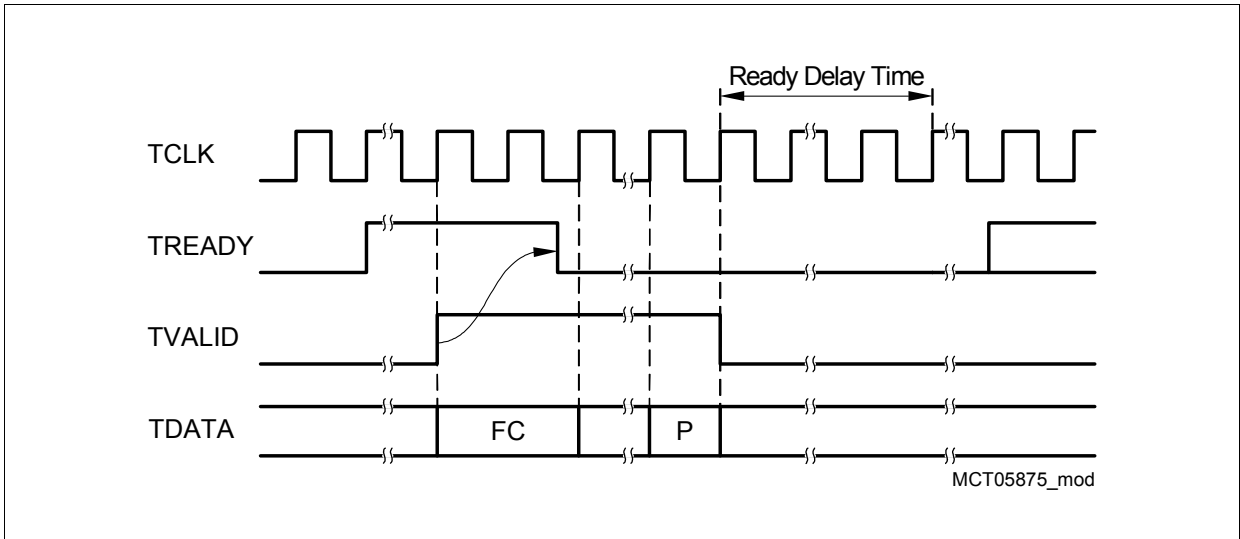


Figure 23-18 MLI Handshake with Parity Error Indication

23.1.3.4 Non-Acknowledge Error

A transmitter of an MLI module is able to detect an inoperable receiver by analyzing the handshake signal TREADY. After TVALID has been asserted to 1, the transmitter checks the receiver's acknowledge (TREADY becoming 0). A Non-Acknowledge error condition is detected by the transmitter when at the end of a frame transmission the TREADY signal is still at high level (TREADY = 1 when TVALID becomes 0). **Figure 23-19** shows the Non-Acknowledge error case. In this case, the transmitter automatically sends the last frame again. Optionally, this MLI event can activate a service request output.

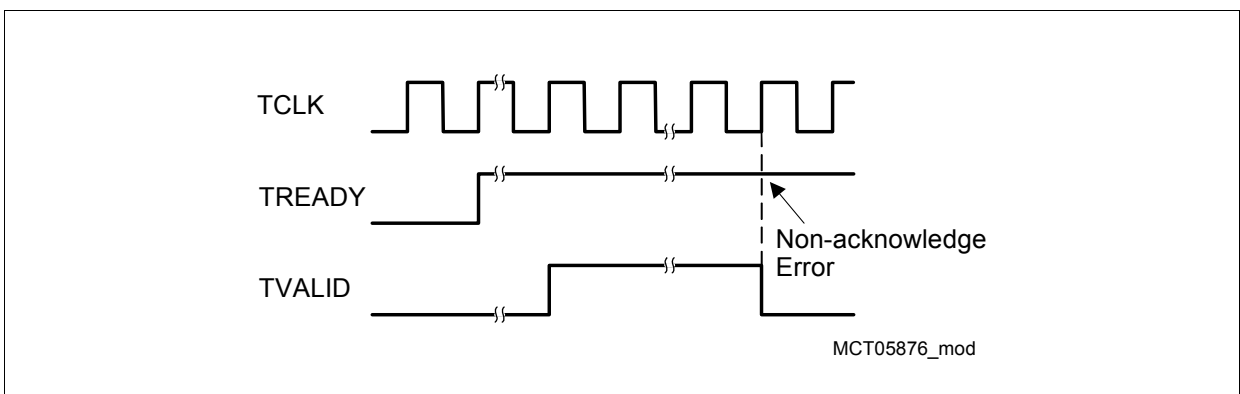


Figure 23-19 Non-Acknowledge Error

23.1.4 Parity Generation

For parity generation, the number of transmitted bits with the value of 1 is counted over the header and the complete data field of a frame. For even parity, the parity bit is set if the result of a modulo-2 division of the elaborated number is 1. For error-free MLI traffic, even parity generation and checking is defined.

More details on the parity handling of the MLI module are provided on [Page 23-42](#).

23.1.5 Address Prediction

An address prediction method can be enabled to support communication between MLI transmitter and MLI receiver without sending address offset information in the frames. This feature reduces the required bandwidth for MLI communication. Both of the communication partners, MLI transmitter and receiver are able to detect regular offset differences of consecutive window accesses to the same window. The address prediction mechanism working independently for each pipe, different prediction values can be handled in parallel for the different pipes.

The MLI transmitter can compare the offset of each Transfer Window read or write access with the offset of the previous access to the same Transfer Window. Between the accesses to a specific window, other windows can be accessed without disturbing the prediction. Bigger offset differences than 512 bytes are not supported by the address prediction.

If the offset differences are identical in at least two accesses to the same Transfer Window, an address prediction is possible and Optimized Write Frames or Optimized Read Frames can be sent to the receiving controller for this pipe. If the offset difference of a next access to this Transfer Window does not match the former ones (predicted offset), address prediction is not possible. In this case, a Normal Frame for writing or reading (Write Offset and Data Frame or Discrete Read Frame) is started.

The identical address prediction mechanism is built in the receiver. As a result, the receiver can elaborate the original offset value in the transmitter when receiving an optimized frame for any pipe.

More details on the address prediction mechanism of the MLI module are provided on [Page 23-45](#).

23.2 Module Kernel Description

This chapter describes how the MLI protocol is implemented in the MLI module and how frame handling can be done by software, comprising:

- The frame handling (see [Page 23-25](#))
- The general MLI features (see [Page 23-42](#))
- The interface signals (see [Page 23-49](#))
- The general MLI service request structure (see [Page 23-55](#))
- The MLI transmitter events (see [Page 23-57](#))
- The MLI receiver events (see [Page 23-60](#))
- The baud rate generation (see [Page 23-65](#))

23.2.1 Frame Handling

The frame handling is based on receiver and transmitter registers and the Transfer Windows. Depending on the type of access to the Transfer Windows, different actions take place inside the MLI module. Please refer to the following pages for the handling of:

- Copy Base Address Frame (see [Page 23-26](#))
- Data Frames (see [Page 23-28](#))
- Read Frames (see [Page 23-32](#))
- Answer Frame (see [Page 23-37](#))
- Command Frame (see [Page 23-39](#))

23.2.1.1 Copy Base Address Frame

A Copy Base Address Frame defines the location and the size of a Remote Window.

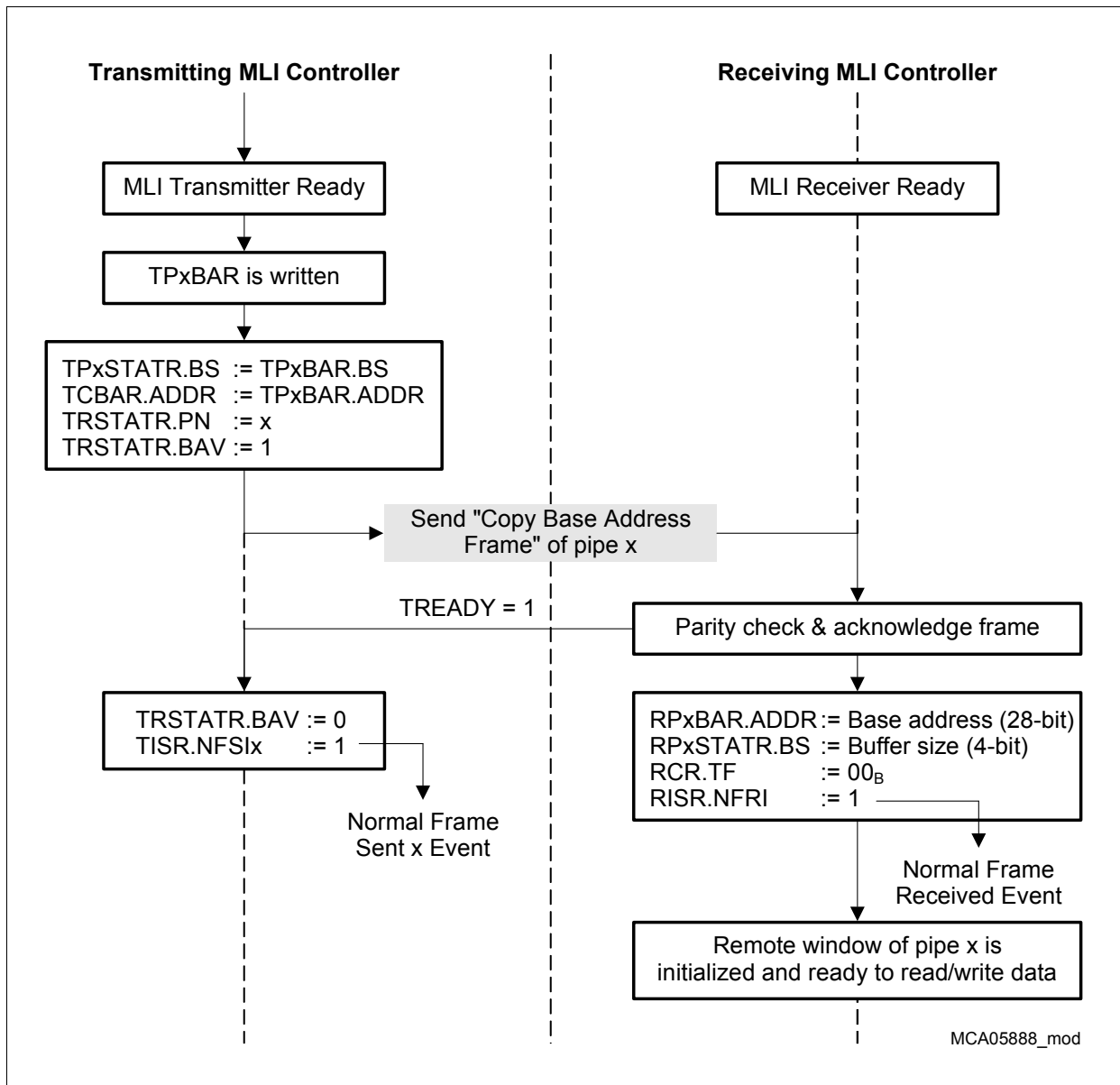


Figure 23-20 Copy Base Address Frame Flow

Transmitting Controller

The transmission of a Copy Base Address Frame is started after a transmitter pipe x base address registers TPxBAR has been written, triggering the following actions for pipe x.

- Bit field TPxBAR.BS (4-bit coded buffer size) is loaded into bit field TPxSTATR.BS

Micro Link Interface (MLI)

- Bit field TPxBAR.ADDR (28 most significant base address bits) is loaded into bit field TCBAR.ADDR.
- Status bit field TRSTATR.PN is updated with the pipe number x (for example x = 2 when TP2BAR has been written).
- Status flag TRSTATR.BAV (base address valid) becomes set.
- The transmission of a Copy Base Address Frame with the two buffered parameters TCBAR.ADDR and TPxSTATR.BS is started for pipe x (if the corresponding pipe is idle and TREADY = 1).
- Status flag TRSTATR.BAV (in the transmitting controller) is cleared after the Copy Base Address Frame has been finished and correctly acknowledged by the MLI receiver of the receiving controller.
- MLI event status flag TISR.NFSIx (Normal Frame Sent event in pipe x) is set and a service request output is activated if enabled by TIER.NFSIEx = 1.

Note: After the transfer of a Copy Base Address Frame the optimized mode will be suppressed automatically by hardware for the next two data frames. This ensures a correct offset prediction afterwards.

Receiving Controller

When a Copy Base Address Frame for pipe x has been received correctly and acknowledged, the following actions are executed in the MLI receiver.

- The received 28 most significant address bits are written into the receiver pipe x base address register bit field RPxBAR.ADDR. This bit field determines the base address of the pipe x Remote Window.
- The received 4-bit coded buffer size is stored in the receiver pipe x status register bit field RPxSTATR.BS. This bit field determines the number of variable address bits for the offset (determining the size) of the pipe x Remote Window.
- The information about the received frame type (= 00_B for Copy Base Address Frame) is written into the receiver control register bit field RCR.TF.
- MLI event status flag RISR.NFRI (Normal Frame Received event) is set and a service request output is activated if enabled by RIER.NFRIE = 01_B or 10_B.

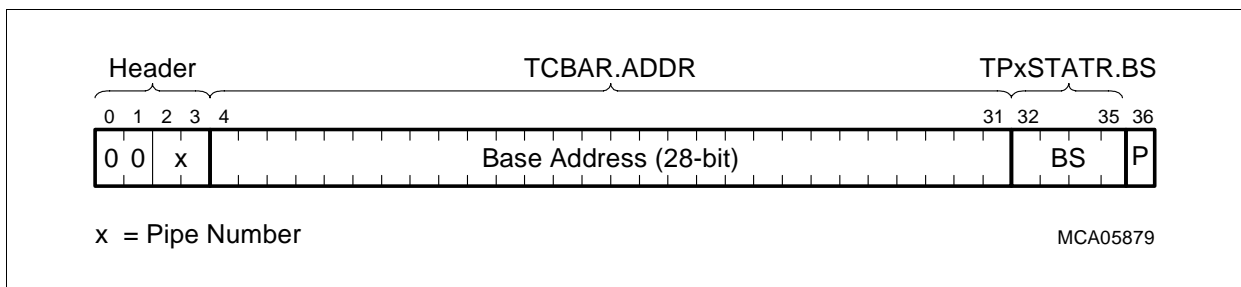


Figure 23-21 Copy Base Address Frame

23.2.1.2 Write/Data Frames

Write Frames (also named Data Frames) transmit the write data and optionally the write offset.

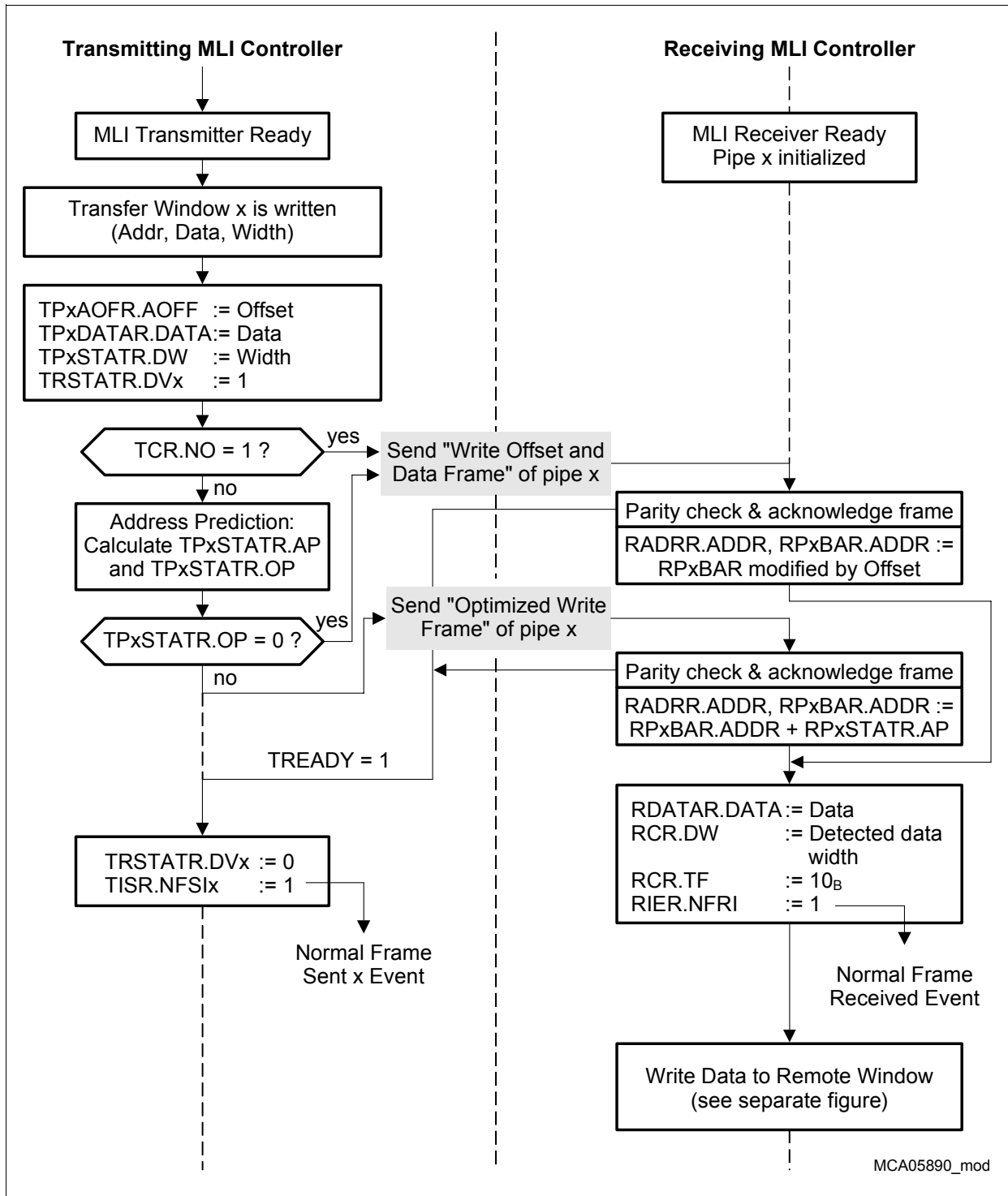


Figure 23-22 Write Frame Flow

Transmitting Controller

In the transmitting controller, a write operation to a location within a Transfer Window delivers the address, the data, and the data size to the transmitter and triggers the following actions in the MLI transmitter.

- The 16 least significant address bits of the Transfer Window write access are stored in TPxAOFR.AOFF as write offset address. In case of a an access to a Small Transfer Window, also 16 bits are stored, but the higher bits are not taken into account.
- The data of the write access to the Transfer Window is stored in TPxDATAR.DATA.
- The data width of the write access to the Transfer Window (8-bit, 16-bit, or 32-bit) is stored in bit field TPxSTATR.DW.
- Status flag TRSTATR.DVx (data valid) is set, indicating that the pipe contains valid data for transmission.
- If the address prediction method is disabled (TCR.NO = 1), the transmission of a Write Offset and Data Frame is started as soon as the MLI transmitter is idle, no higher priority frames are pending, and TREADY = 1.

If the address prediction method is enabled (TCR.NO = 0), a Write Offset and Data Frame is started only if an address prediction is not possible (indicated by TPxSTATR.OP = 0). If TPxSTATR.OP = 1, an address prediction is possible in the MLI transmitter (and the MLI receiver) and an Optimized Write Frame can be started. The address prediction method used is described on [Page 23-45](#).

- Status flag TRSTATR.DVx is cleared by hardware and MLI event status flag TISR.NFSIx (Normal Frame Sent event in pipe x) is set (and a service request output is activated if enabled by TIER.NFSIEx = 1) after the Write Frame has been finished and correctly acknowledged by the MLI receiver.

The number m of offset address bits that are transmitted at a Write Offset and Data Frame is determined by the size of the Remote Window in the receiving controller that has been previously initialized by the transmission of a Copy Base Address Frame. Parameter m is referring to bit field TPxSTATR.BS (and RPxSTATR.BS) and can be in the range of 1 to 16 bits.

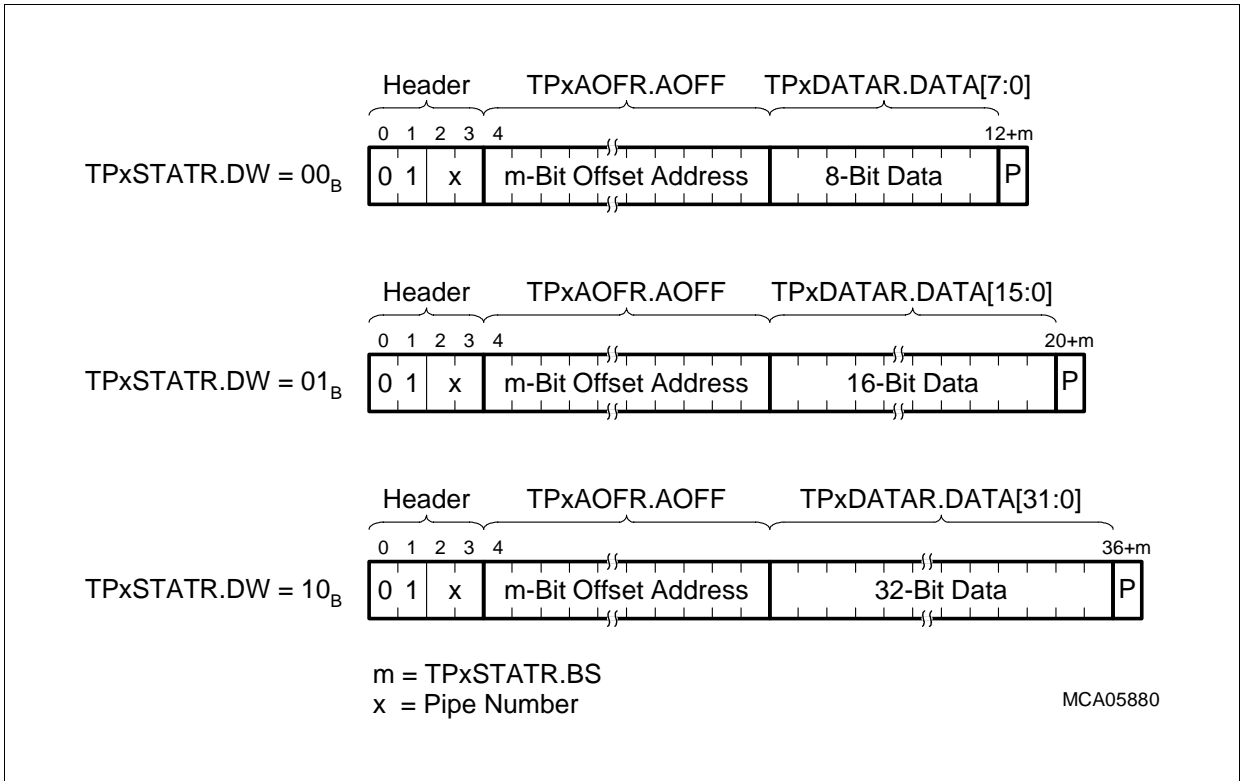


Figure 23-23 Write Offset and Data Frame

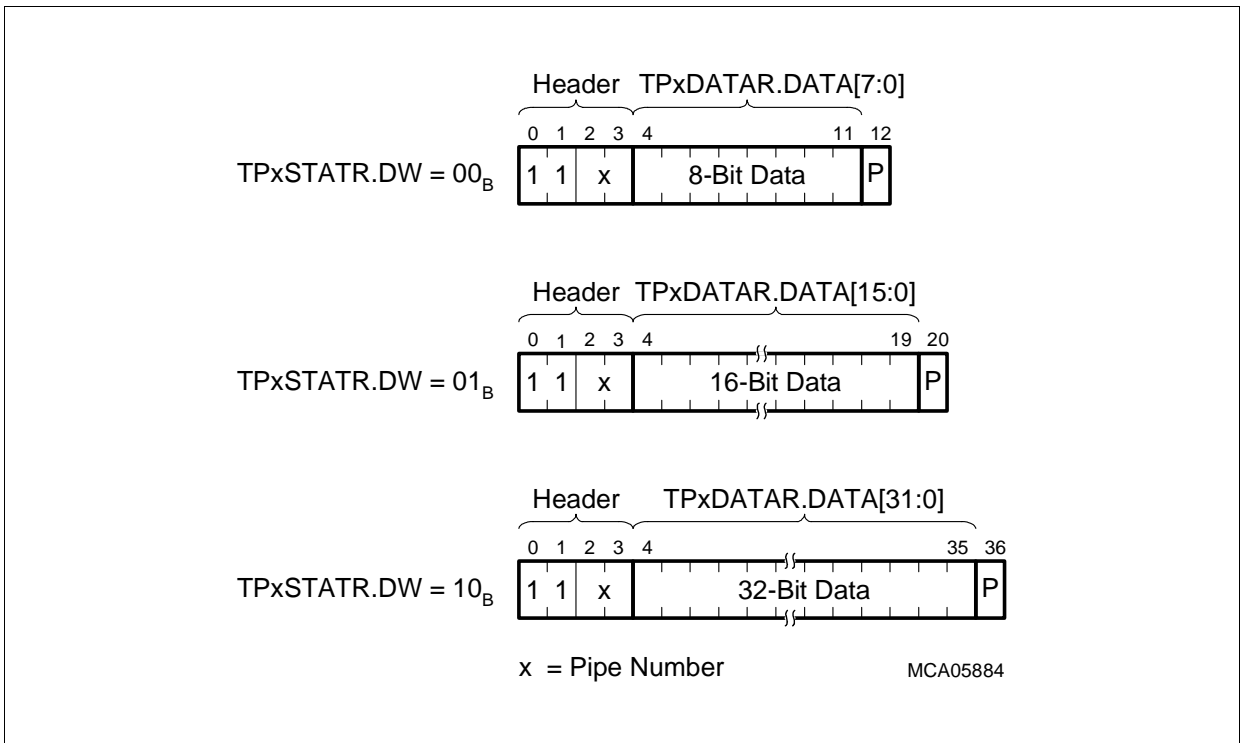


Figure 23-24 Optimized Write Frame

Receiving Controller

After a Write Frame has been received correctly and acknowledged, the following actions are automatically executed in the MLI receiver:

- In the case of a Write Offset and Data Frame:
The result of the internal address prediction is not taken into account. The received offset address is added to the base address of the pipe x Remote Window and the result is stored in RPxBAR.ADDR. It is also stored in RADDR.ADDR and represents the destination address in the receiving controller where data should be written to.
In the case of an Optimized Write Frame:
The result of the internal address prediction is taken into account. The next address in the receiving controller where data should be written to is calculated by adding the detected receiver address prediction value RPxSTATR.AP to the actual address stored in RPxBAR.ADDR and the result is stored in RPxBAR.ADDR and in RADDR.ADDR.
- The received data is written into the receiver data register RDATAR (right aligned, unused bits are 0).
- The detected data width of the received data is written into bit field RCR.DW.
- The information about the received frame type (= 10_B for a Write Frame) is written into bit field RCR.TF.
- MLI event status flag RISR.NFRI (Normal Frame Received event) is set and an SR output line is activated if enabled by RIER.NFRIE = 01_B or 10_B .

After these actions related to the reception of a Write Frame by the receiving controller, the data that has been received from the transmitting controller is ready to be written into the Remote Window related to the receiving pipe.

This write operation can be executed in two ways:

- RCR.MOD = 0: Automatic Data Mode is disabled.
In this mode, a bus master of the receiving controller, typically a CPU, is informed by a Normal Frame received event RISR.NFRI (a service request output is activated if RIER.NFRIE = 10_B) to transfer the received write data from the MLI receiver to the Remote Window. Therefore, it must read the data from RDATAR, together with width RCR.DW and the address stored in RADDR and write it to the indicated address location.
- RCR.MOD = 1: Automatic Data Mode is enabled.
In this mode, the MLI module automatically writes the received write data to the Remote Window. This automatic action is controlled by a move engine block in the MLI receiver. It also sets event status flag RISR.MEI (move engine event when the access is terminated). A service request output is activated if enabled by RIER.MEIE = 1.
The write operation to the Remote Window is executed only if the write address is within an enabled access protection range. If the address range is disabled for the write address, the automatic write action does not take place and event status flag RISR.MPEI (memory protection error) is set and a service request output is activated

Micro Link Interface (MLI)

if enabled by RIER.MPEIE = 1. In this case, the receiving controller software can analyze the values in RDATAR, together with width RCR.DW and the address stored in RADRR.

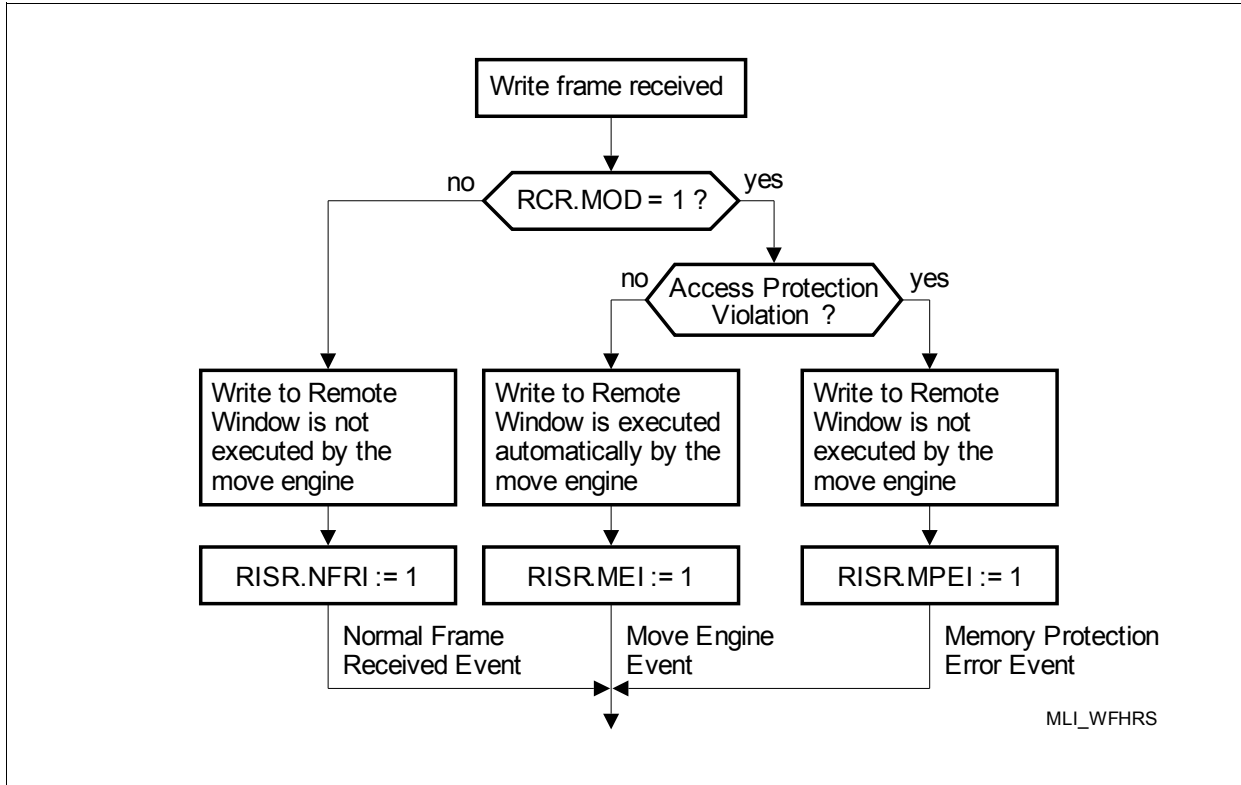


Figure 23-25 Write Frame Handling on Receiving Side

Note: In Automatic Data Mode, Write Frames are leading to a write action executed by the MLI move engine. During the move engine operation, only one more MLI frame can be received (stored in a waiting position to be executed). Then the reception of more frames is blocked by Non-Acknowledge handshake. If the move engine operation is finished, frame execution and reception continue normally. If Automatic Data Mode is disabled, no blocking mechanism has been implemented. The receiving controller software has to take care to deal with the received data before it is overwritten by new incoming frames.

23.2.1.3 Read Frames

Read Frames transmit read request and optionally the read offset from the Local Controller to the Remote Controller.

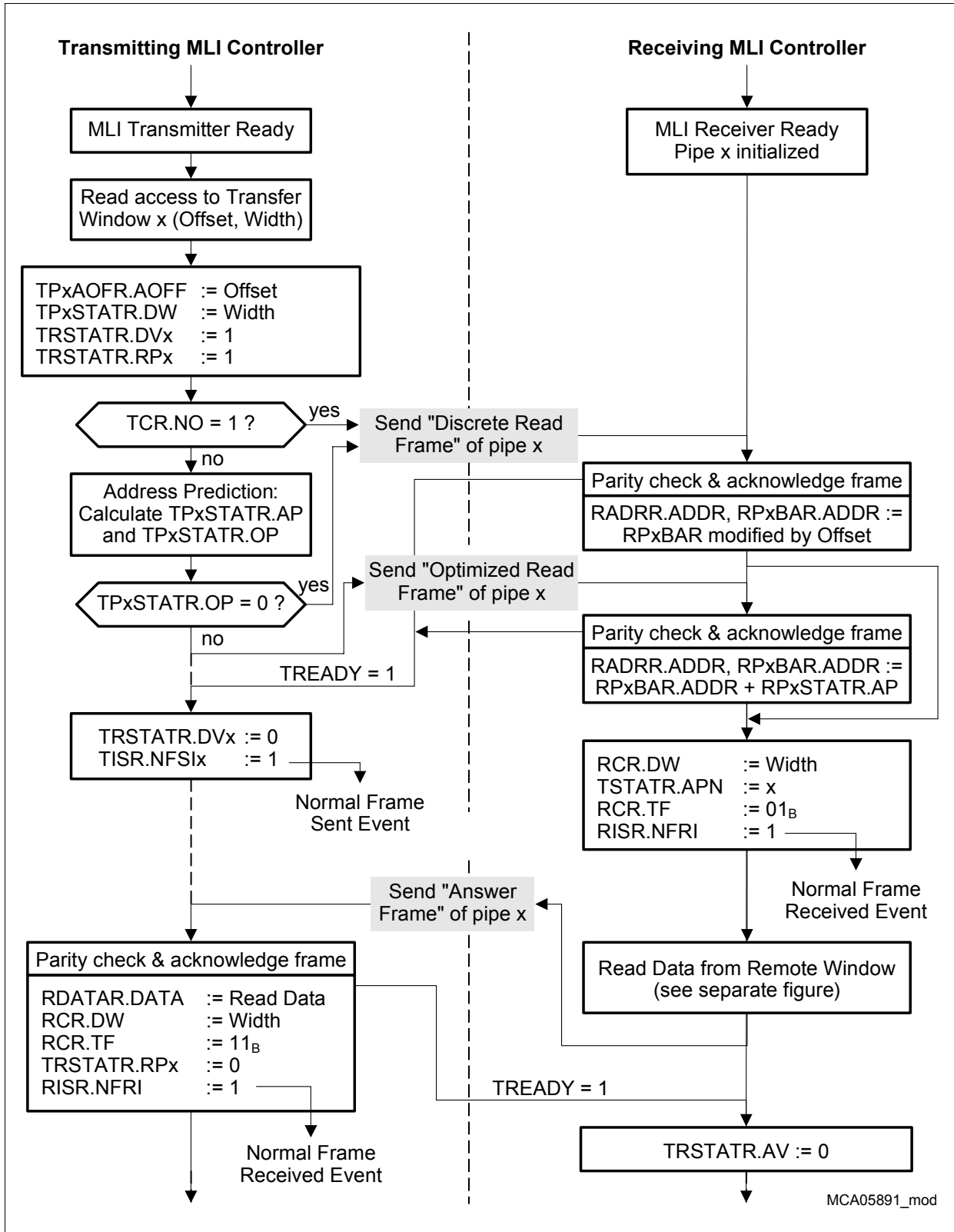


Figure 23-26 Read Frame and Answer Frame Flow

Local Controller

A read operation from a location within a Transfer Window x of the Local Controller delivers a dummy value as result of the read action and triggers the transmission of a Read Frame. The dummy value of the initial read action should be ignored and the software has to wait for the reception of the Answer Frame to get the desired data.

- The 16 least significant address bits of the Transfer Window read access are stored in TPxAOFR.AOFF as read offset address. In case of a an access to a Small Transfer Window, also 16 bits are stored, but the higher bits are not taken into account.
- The data width of the Transfer Window read access (8-bit, 16-bit, or 32-bit) is stored in bit field TPxSTATR.DW.
- Status flag TRSTATR.DV x (data valid) is set.
- Status flag TRSTATR.RPx (read pending) is set. This bit is cleared by hardware when an Answer Frame has been received correctly.
- If the address prediction method is not enabled (TCR.NO = 1), transmission of a Discrete Read Frame is started. If the address prediction method is enabled (TCR.NO = 0), a Discrete Read Frame is started only if an address prediction is not possible (indicated by TPxSTATR.OP = 0). If TPxSTATR.OP = 1, an address prediction is possible and an Optimized Read Frame is started.
- Status flag TRSTATR.DV x is cleared by hardware and MLI event status flag TISR.NFSI x (Normal Frame Sent event in pipe x) is set (and a service request output is activated if enabled by TIER.NFSIEx = 1) after the Read Frame has been finished and correctly acknowledged by the MLI receiver of the Remote Controller.

The number m of offset address bits that are transmitted at a Discrete Read Frame is determined by the (coded) size of the Remote Window in the Remote Controller that has been previously initialized by the transmission of a Copy Base Address Frame. Parameter m is stored in bit field TPxSTATR.BS (and RPxSTATR.BS) and can be in the range of 1 to 16 bits.

After a completed transmission of a Read Frame, the Local Controller expects the reception of an Answer Frame. The Answer Frame is introduced with the highest priority into the data flow of the transmitter of the Remote Controller.

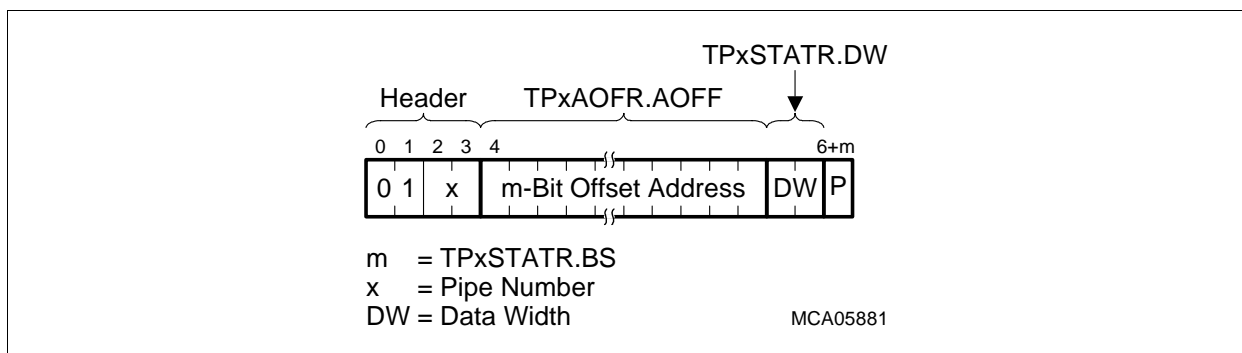


Figure 23-27 Discrete Read Frame

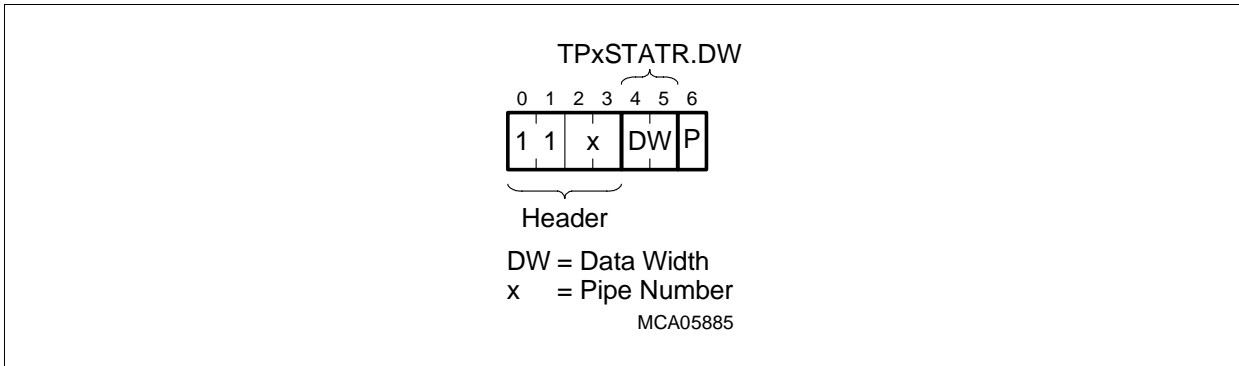


Figure 23-28 Optimized Read Frame

Remote Controller

After a Read Frame has been correctly received and acknowledged, the following actions are executed in the MLI receiver of the Remote Controller:

- In the case of a Discrete Read Frame:
The result of the address prediction is not taken into account. The received offset address is added to the base address of the pipe x Transfer Window (stored in RPxBAR.ADDR). The result of this addition is stored in RADRR.ADDR and also in RPxBAR.ADDR and represents the destination address in the Remote Controller from where data should be read.
- In the case of an Optimized Read Frame:
The result of the address prediction is taken into account. The next address in the Remote Controller where data should be read is calculated by adding the detected receiver address prediction value RPxSTATR.AP to the actual address stored in RPxBAR.ADDR. The result of this addition is stored in RADRR.ADDR and also in RPxBAR.ADDR and represents the destination address in the Remote Controller from where data should be read.
- The transmitted data width DW is written into bit field RCR.DW.
- The information about the received frame type (= 01_B for a Read Frame) is written into bit field RCR.TF.
- MLI event status flag RISR.NFRI (Normal Frame Received event) is set and a service request output is activated if enabled by RIER.NFRIE = 01_B or 10_B.

After correct reception of a Read Frame by the Remote Controller, the data requested by the Local Controller can be read by the Remote Controller and sent back to the Local Controller in form of an Answer Frame.

This read operation can be executed in two ways:

- RCR.MOD = 0:
Automatic Data Mode is disabled. In this mode, a bus master of the Remote Controller, typically a CPU, is informed by a Normal Frame received event to read the requested read data and transfer it to the MLI receiver. Therefore, it must read data

Micro Link Interface (MLI)

with width RCR.DW from the address stored in RADRR and write the data into TDRAR.DATA.

- RCR.MOD = 1:
Automatic Data Mode is enabled. In this mode, the move engine of the MLI automatically reads the data from the Remote Window and sets event status flag RISR.MEI (move engine access terminated). A service request output is activated if enabled by RIER.MEIE = 1.
The read operation from the Remote Window is executed only if the read address is within an enabled access protection range. If no address range is enabled for the actual read address, the automatic read action is not executed by the move engine, event status flag RISR.MPEI (memory protection error) is set and a service request output is activated if enabled by RIER.MPEIE = 1. In the interrupt handler routine, a bus master (e.g. CPU or PCP) must then take care of the remote window read operation and the data transfer to TDRAR.
- After TDRAR.DATA has been updated, status flag TRSTATR.AV of the Remote Controller is set and the transmission of an Answer Frame is started.

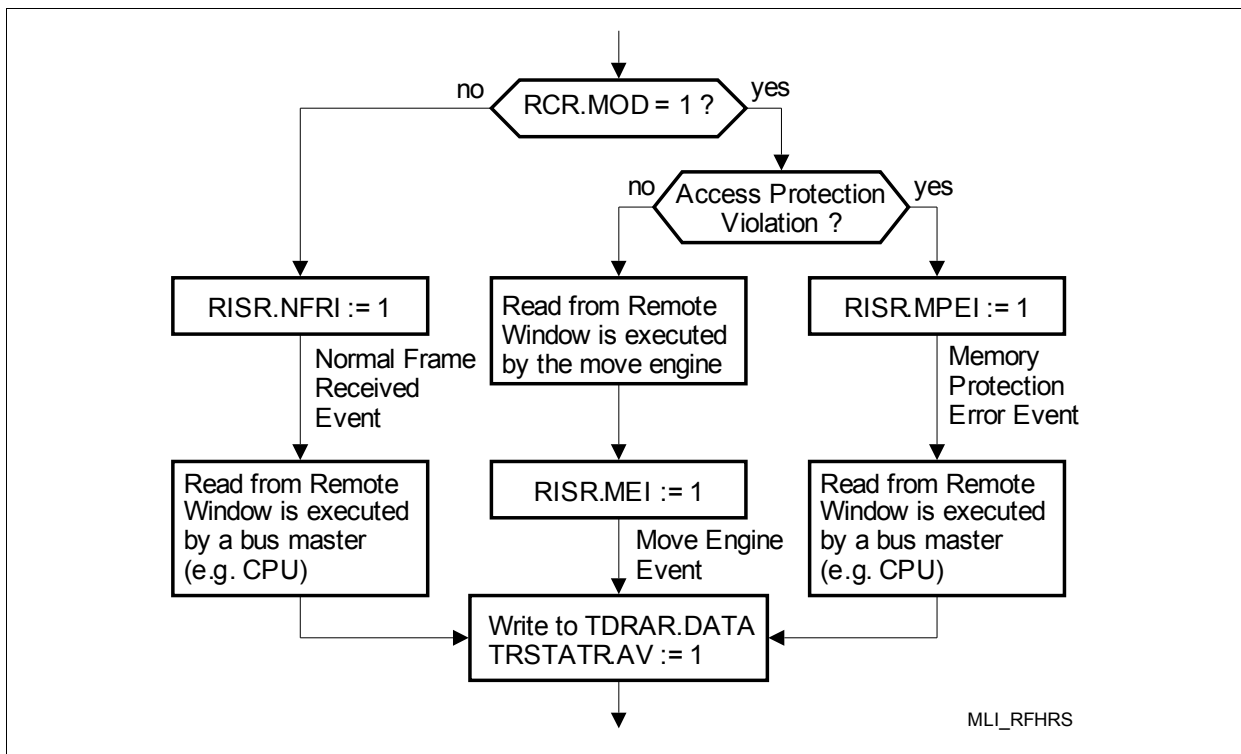


Figure 23-29 Read Frame Handling on Remote Side

Note: In Automatic Data Mode, Read Frames are leading to a read action executed by the MLI move engine. During the move engine operation, only one more MLI frame can be received (stored in a waiting position to be executed). Then the reception of more frames is blocked by a Non-Acknowledge handshake. If the move engine operation is finished, frame execution and reception can continue

normally. If Automatic Data Mode is disabled, no blocking mechanism has been implemented. The Remote Controller software has to take care to read the received data.

23.2.1.4 Answer Frame

Please note that only one Answer Frame can be handled by the system at a time (no Read Frame request while any TRSTATR.RPx is set). Make sure that not more than one Read Frame is pending at a time. If a Read Frame is not answered by an Answer Frame during a certain time interval, a time-out criterion should be handled in software. The Remote Controller has to take care that no Answer Frame is delivered after the time-out criterion has been detected (e.g. by a software-triggered Command Frame). Do not start a new Read Frame while waiting for an Answer Frame if the time-out criterion has not yet been detected and the Answer Frame has not yet been received. The length of the time-out interval depends on the application and has to be defined accordingly on a case by case base (e.g. the transfer rates between MLI modules, bus architecture, etc. have to be considered). In the case a time-out has been detected, the Local Controller software has to clear the TRSTATR.RPx bit by writing 1 to SCR.CDVx and can start a new Read Frame.

Remote Controller (Receiving the read request)

The Answer Frame is the only frame sent from the Remote Controller back to the Local Controller. The transmitter registers of the Remote Controller are used to generate the Answer Frame.

Every time the transmitter data read answer register TDRAR is written in the Remote Controller, the transmission of an Answer Frame is started and the following actions are triggered.

- Status flag TRSTATR.AV is set to trigger the transmission of an Answer Frame.

The following parameter is transmitted in the data field of the Answer Frame:

- Read data: stored in TDRAR.DATA; data width is determined by TRSTATR.DW.
- Status flag TRSTATR.AV is cleared after the Answer Frame has been finished and correctly acknowledged by the MLI receiver of the Local Controller.

An Answer Frame should be sent through the pipe that has received a read request but there must be only one MLI Transfer Window read access pending on any side of a MLI connection at any time, because the answer mechanism does not contain buffers for multiple Answer Frames.

Local Controller (Transmitting the read request)

If an Answer Frame has been received correctly and acknowledged, the following actions are executed in the MLI receiver of the Local Controller:

Micro Link Interface (MLI)

- The TRSTATR.RPx flags are cleared. In future versions, it is intended to clear only the flag belonging to pipe x.
- The received data is written into the receiver data register RDATAR.
If 8 data bits are received, they are duplicated to all 4 bytes in RDATAR.
If 16 data bits are received, they are duplicated to both half-words in RDATAR.
- The detected data width of the received data is written into bit field RCR.DW.
- The received Pipe Number x represents the answer Pipe Number and is written into bit field TSTATR.APN.
- The information about the received frame type (= 11_B for an Answer Frame) is written into bit field RCR.TF.
- MLI event status flag RISR.NFRI (Normal Frame Received event) is set and a service request output is activated if enabled by RIER.NFRIE = 01_B or 10_B.
- The content of RADRR becomes invalid.
- The data that has been previously requested from the Remote Controller by a Read Frame is now available in RDATAR and can be read by a bus master (e.g. the CPU) of the Local Controller.
- If an Answer Frame is received while the corresponding TRSTATR.RPx bit is 0, the reception is declared as unintended and a Discarded Read Answer event is generated (see [Page 23-60](#)).

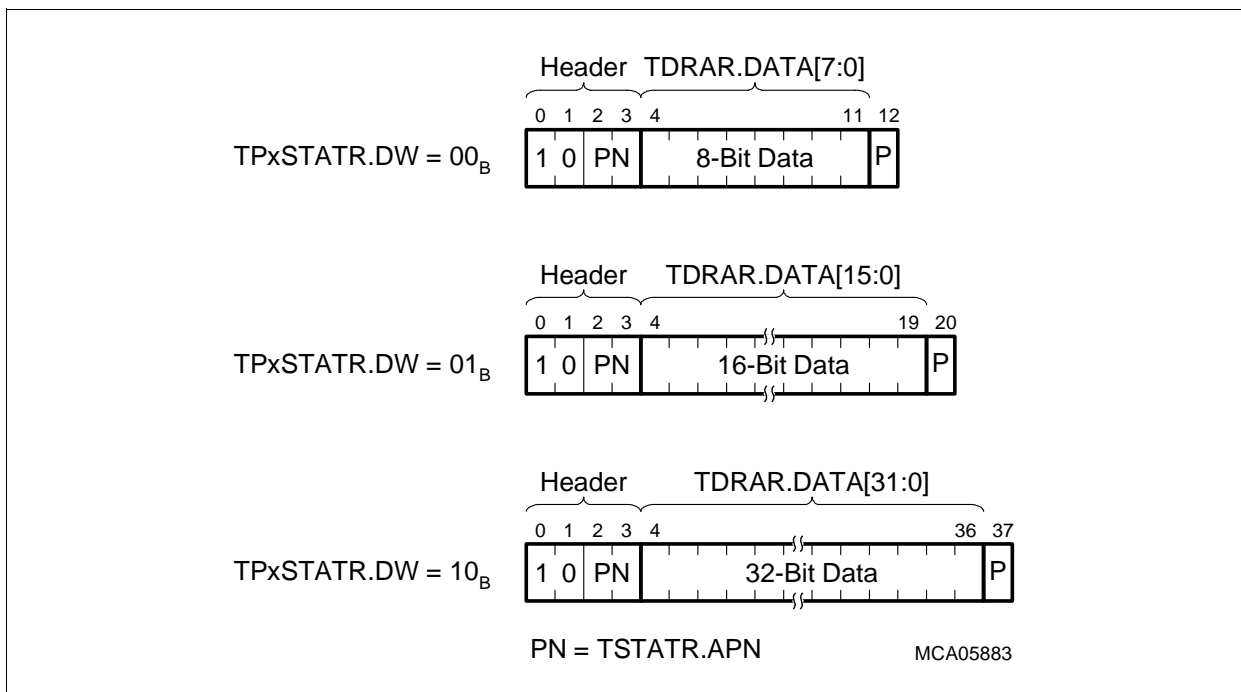


Figure 23-30 Answer Frame

Note: If an Answer Frame has been correctly received in the Local Controller, the Local Controller's software has to read it. As long as at least one byte of this data has not yet been read out, only one more MLI frame can be received (stored in a waiting position to be executed). Then the reception of more frames is blocked by

Non-Acknowledge handshake. If the received data has been read out, frame execution and reception continue normally.

23.2.1.5 Command Frame

Command Frames transmit a command (e.g. setup information or service request) from a transmitting controller to a receiving controller.

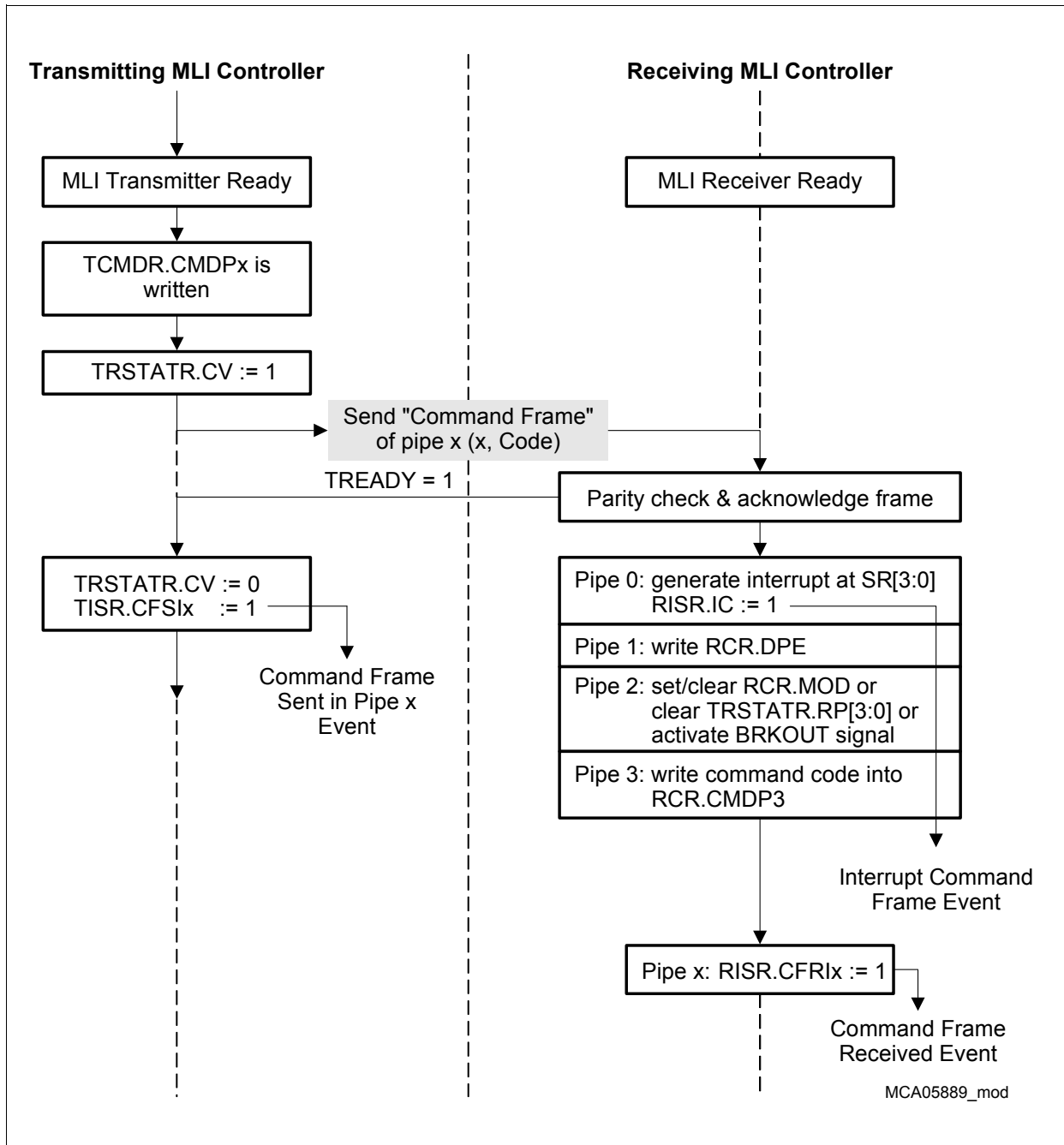


Figure 23-31 Command Frame Transaction Flow

Transmitting Controller

The transmission of a Command Frame is initiated by writing one of the four pipe x related command code bit fields in register TCMDR.CMDPx, triggering the following actions:

- Status flag TPxSTATR.CVx (command valid) is set and the Command Frame transmission is started using x as pipe number PN and the command code stored in TCMDR.CMDPx as parameters.
- TRSTATR.CVx is cleared after the Command Frame has been finished and correctly acknowledged by the MLI receiver of the Remote Controller.
- MLI event status flag TISR.CFSIx (Command Frame Sent event in pipe x) is set and a service request output is activated if enabled by TIER.CFSIEx = 1.

Receiving Controller

Depending on the pipe x related command code that is transmitted by a Command Frame, different actions are triggered in the receiving controller. [Table 23-5](#) describes the actions that are transmitted by a Command Frame and that cause a specific control task in the MLI receiver.

- The received PN value is checked and the corresponding control actions are executed according to [Table 23-5](#).
- Independent of the received Pipe Number, event status flag RISR.CFRIx (Command Frame Received event in pipe x) is set and a service request output is activated if enabled by RIER.CFRIEx = 1.

If a Command Frame is received for pipe 2 with command code 1111_B, the $\overline{\text{BRKOUT}}$ output signal of the MLI module becomes activated if it is enabled by bit RCR.BEN = 1. If disabled by RCR.BEN = 0, signal $\overline{\text{BRKOUT}}$ will not be activated. The usage of $\overline{\text{BRKOUT}}$ is implementation-specific and can be used, for example, to generate a break condition in the on-chip debug support logic or trigger other functions.

Table 23-5 Command Frame Encoding

PN	CMD	Command Description
00 _B	0001 _B	Activate service request output SR0 of receiving MLI module
	0010 _B	Activate service request output SR1 of receiving MLI module
	0011 _B	Activate service request output SR2 of receiving MLI module
	0100 _B	Activate service request output SR3 of receiving MLI module
	Others	no effect, reserved for future use

Table 23-5 Command Frame Encoding (cont'd)

PN	CMD	Command Description
01 _B	0000 _B	Set RCR.DPE (delay for parity error indication) in receiving MLI to 0000 _B
	0001 _B	Set RCR.DPE in receiving MLI to 0001 _B
	0010 _B	Set RCR.DPE in receiving MLI to 0010 _B

	1111 _B	Set RCR.DPE in receiving MLI to 1111 _B
10 _B	0001 _B	Enable Automatic Data Mode in receiving MLI (set RCR.MOD = 1)
	0010 _B	Disable Automatic Data Mode in receiving MLI (set RCR.MOD = 0)
	0100 _B	Clear bit TRSTATR.RP0 in receiving MLI
	0101 _B	Clear bit TRSTATR.RP1 in receiving MLI
	0110 _B	Clear bit TRSTATR.RP2 in receiving MLI
	0111 _B	Clear bit TRSTATR.RP3 in receiving MLI
	1111 _B	Generate break output signal $\overline{\text{BRKOUT}}$ in receiving MLI (if enabled by RCR.BEN = 1)
	others	no effect, reserved for future use
11 _B	Any	Free programmable software command, written into bit field RCR.CMDP3 of receiving MLI

23.2.2 General MLI Features

The general MLI features comprise the:

- Parity generation and checking (see [Page 23-42](#))
- Non-Acknowledge error (see [Page 23-45](#))
- Address prediction (see [Page 23-45](#))
- Automatic data transfers (see [Page 23-46](#))
- Access protection (see [Page 23-47](#))
- Transmit priority (see [Page 23-48](#))
- Transmission delay (see [Page 23-48](#))

23.2.2.1 Parity Check and Parity Error Indication

For parity generation, the number of transmitted bits with the value of 1 is counted over the header and the complete data field of a frame. For even parity, the parity bit is set if the result of a modulo-2 division of the elaborated number is 1. For odd parity, the parity bit is set if the result of a modulo-2 division of the elaborated number is 0.

For a parity error-free MLI connection, even parity must be selected in the transmitter because the receiver operates only with even parity detection. The capability to select odd parity can be used by the transmitter to force a parity error reply from the receiver during the startup procedure of the MLI connection. This can be used to measure the propagation delay and to optimize the ready delay time (see [Page 23-71](#)).

Note: There is no protection against frames where more than one bit is corrupted (e.g. shortened frames). In such a case, an unpredicted behavior of the MLI module may occur.

Transmitting Controller

The MLI transmitter counts the detected parity error conditions and generates a parity error event if a programmable number (max. 16) of parity error conditions has occurred. A parity error condition is indicated to the transmitter by the receiver after the transmission of a frame (see [Page 23-23](#)). The transmitter parity error condition is detected when the TREADY signal is sampled at low level within a programmable number (TCR.MDP = maximum delay for parity errors) of TCLK clock cycles after TVALID has been de-asserted to low.

If a transmitter parity error condition is detected, the MLI transmitter sets the parity error flag TSTATR.PE and also decreases the maximum parity error counter TCR.MPE by 1. The maximum parity error counter of the transmitter TCR.MPE determines the number of transmit parity error conditions that can be still detected until a transmitter parity error event is generated. If a transmitter parity error condition is detected and TCR.MPE is becoming 0 or while it is 0, a transmitter parity error event is generated by setting bit TISR.PEI (see [Figure 23-39](#) on [Page 23-58](#)) and an SRx output line is activated if enabled by TIER.PEIE = 1. After a transmitter parity error event occurred, TCR.MPE can

be set again by software to a value greater 0001_B . Otherwise, each additional transmitter parity error condition will generate a parity error event.

The transmitter parity error flag TSTATR.PE is cleared by hardware when a correct frame transmission and TREADY has been sampled with 1 within the ready delay time. It can be cleared by software by writing a 1 to bit SCR.CTPE. If for example, each transmitter parity error condition should generate a transmitter parity error event, TCR.MPE should be set to 0000_B . The software can check for accumulated parity error conditions by reading TCR.MPE or TISR.PEI, for the status of the latest received frame, it can check TSTATR.PE.

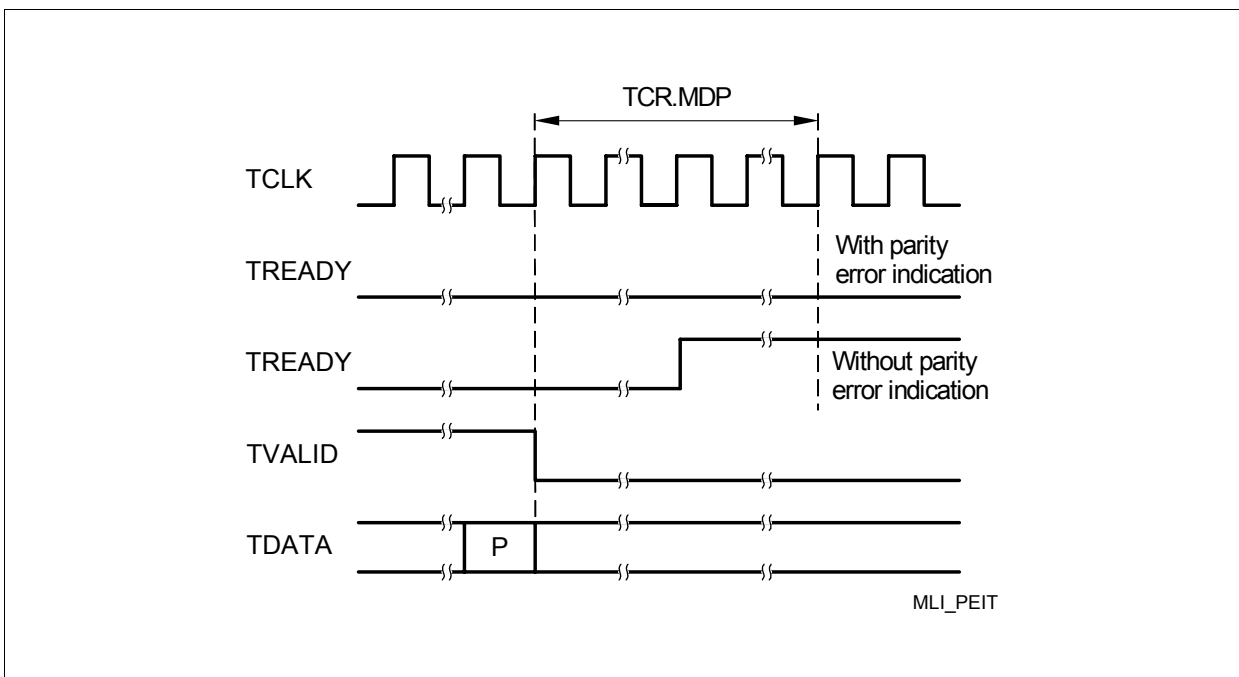


Figure 23-32 Parity Error Indication for the Transmitter

Receiving Controller

The receiver always checks the parity bit of a received frame for even parity. A receiver parity error condition is detected if the received parity bit does not match with the internally calculated one. If no receiver parity error condition is found after the reception of a frame, RREADY is immediately set to 1, otherwise RREADY is kept at 0 until a defined number of RCLK cycles (determined by bit field RCR.DPE = delay for parity error) has been elapsed. Then, RREADY is asserted high.

If a receiver parity error condition is found, the MLI receiver sets the parity error flag RCR.PE and additionally decreases the maximum parity error counter of the receiver RCR.MPE by 1. The maximum parity error counter RCR.MPE determines the number of receiver parity error conditions that can be still detected until a receiver parity error event is generated. If a receiver parity error condition is detected and RCR.MPE is becoming 0 or while it is already 0, a receiver parity error event is generated by setting bit RISR.PEI

Micro Link Interface (MLI)

(see [Figure 23-43](#) on [Page 23-61](#)) and a service request output is activated if enabled by $RIER.PEIE = 1$. After a receiver parity error event has occurred, $RCR.MPE$ can be set again by software to a value greater 0001_B . If, for example, each receiver parity error condition should generate a receiver parity error event, $RCR.MPE$ can be programmed to 0000_B or 0001_B .

The receiver parity error flag $RCR.PE$ is cleared by hardware if a correct frame transmission has occurred. $RCR.PE$ can be cleared by software by writing a 1 to bit $SCR.CRPE$.

The receiver parity error flag $RCR.PE$ is cleared by hardware after a correct frame reception. It can be cleared by software by writing a 1 to bit $SCR.CRPE$. The software can check for accumulated parity error conditions by reading $RCR.MPE$ or $RISR.PEI$, for the status of the latest received frame, it can check $RCR.PE$.

The delay for parity error bit field $RCR.DPE$ is a read-only bit field in the receiver that is updated by hardware if a Command Frame for pipe 1 is received. With this frame type, the transmitting controller transfers a value for $RCR.DPE$ to the receiving controller during the setup phase of the MLI connection.

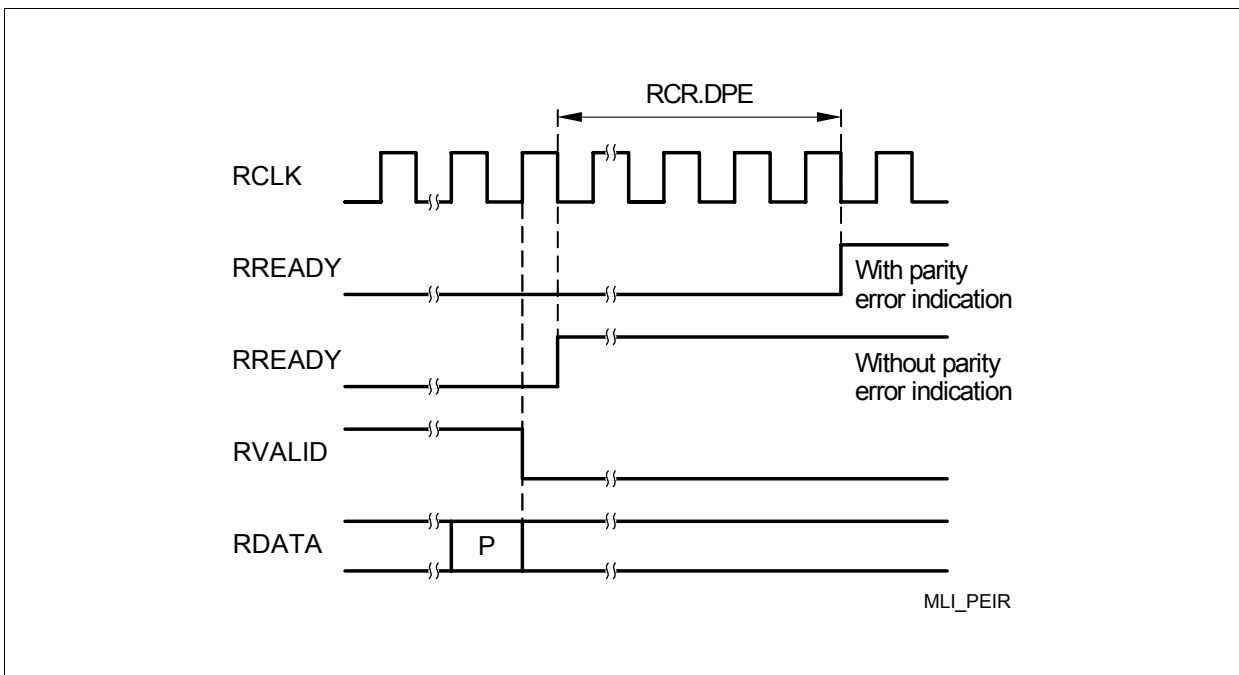


Figure 23-33 Parity Error Indication by the Receiver

23.2.2.2 Non-Acknowledge Error

A Non-Acknowledge error condition is detected by the transmitter when at the end of a frame transmission, the TREADY signal is still at high level (TREADY = 1 when TVALID becomes 0). In this case, the error flag TSTATR.NAE is set and the maximum Non-Acknowledge error counter TCR.MNAE is decremented by 1. If a Non-Acknowledge error condition is detected and TCR.MNAE is becoming 0 or while it is already 0, a time-out event is generated by setting bit TISR.TEI (see [Figure 23-39](#) on [Page 23-58](#)) and an MLI service request is generated if enabled by TIER.TEIE = 1. The Non-Acknowledge error flag TSTATR.NAE is cleared by hardware when a frame transmission has been acknowledged correctly. It can also be cleared by software when writing a 1 to bit SCR.CNAE.

The Non-Acknowledge error counter TCR.MNAE is automatically set to 11_B when a frame has been acknowledged correctly. It can be read and written by software, allowing a limited number of consecutive Non-Acknowledge errors to be defined that can be detected until a time-out event is generated. If, for example, the first occurrence of a Non-Acknowledge error should lead to a time-out event, bit TCR.MNAE has to be written by software with 00_B or 01_B after each correctly received frame.

23.2.2.3 Address Prediction

An address prediction method can be enabled to support communication between MLI transmitter and MLI receiver without sending address offset information in the frames to optimize the required MLI bandwidth. This feature reduces the required bandwidth for MLI communication. Both communication partners, MLI transmitter and the MLI receiver are able to detect regular offset differences of consecutive window accesses to the same window. The address prediction mechanism working independently for each pipe, different prediction values can be handled in parallel for the different pipes.

Transmitting Controller

If the address prediction method is enabled (TCR.NO = 0), the MLI transmitter compares the offset of each Transfer Window read or write access with the offset of the previous access to the same Transfer Window (stored in TPxAOFR.AOFF). The result of this comparison is stored in two's complement representation in TPxSTATR.AP (limited to 9 bits, otherwise prediction is not possible). Between the accesses to a specific window, other windows can be accessed without disturbing the prediction.

If the offset differences are identical in at least two consecutive accesses to the same Transfer Window, an address prediction is possible (flag TPxSTATR.OP becomes set) and optimized frames can be sent to the receiving controller for this pipe. If the offset difference of a next access to the same Transfer Window does not match the calculated value in TPxSTATR.AP, flag TPxSTATR.OP is cleared and address prediction is not possible. In this case, a Normal Frame for writing or reading (Write Offset and Data Frame or Discrete Read Frame) is started.

Receiving Controller

The MLI receiver operates with an address prediction method equivalent to the MLI transmitter. This means that after receiving at least two consecutive Write Offset and Data Frames and/or Discrete Read Frames that include address information, the MLI receiver is able to follow the address prediction method used by the MLI transmitter. Each received offset is compared in the MLI receiver with the offset of the previously received frame of the same pipe. The result of this comparison is stored in two's complement representation in RPxSTATR.AP (limited 9 bits).

If an optimized frame is received by the MLI receiver, it calculates the next address by adding the value stored in RPxSTATR.AP to the contents of the receiver address register RADRR.

In case of a Write Offset and Data Frame or a Discrete Read Frame (m offset bits), the receiver address registers RADRR and RPxBAR are always loaded with an updated address. This address is calculated by replacing the lowest m bit positions in RPxBAR with the received offset value. In this case, the address delta value stored in RPxSTATR.AP is not taken into account. The programmed size of the Remote Window and the number m of offset bits are given by RPxSTATR.BS. The bit positions RPxBAR[31: m] are kept constant, whereas the bit positions RPxBAR[m -1:0] are replaced.

23.2.2.4 Automatic Data Mode

The MLI module supports automatic data transfers for read or Write Frames without any CPU load in the receiving controller. This feature is based on a move engine block providing the data, the complete address and the data width to an associated bus master on the system bus (see [Figure 23-1](#)). Depending on the implementation, this bus master can be capable of executing the requested data move operations autonomously. The Automatic Data Mode in the receiving controller can be enabled (RCR.MOD = 1) or disabled (RCR.MOD = 0) by software on receiving side or a Command Frame sent by the transmitting controller.

If the Automatic Data Mode is disabled, the receiving controller software has to execute the requested data transfers.

Additionally to the global enable/disable of the automatic mode by RCR.MOD, it is possible to individually exclude address ranges from automatic data transfer by an access protection scheme. The definition of the address ranges depend on the product and has been introduced to support the protection of critical data or modules.

Note: If a device contains the MLI move engine block as the only bus master, automatic mode has to be selected to allow transfers. This could be the case for external peripheral devices without own CPU.

23.2.2.5 Memory Access Protection

The MLI receiver provides a memory access protection logic allowing to exclude read and write accesses of the MLI move engine to specific parts of the memory map from automatic mode. Each address of a data move (read or write) is always checked if it targets an address range that is enabled for read/write access. If a requested data move is targeting an excluded address range, a memory access protection error event is generated and the receiving controller's software can take care of the service request.

The memory access protection logic handles two levels of address range definitions:

- Fixed address ranges (for complete modules or memory areas)
- Programmable address sub-ranges (to limit accesses to specific parts of bigger memory areas)

There is a maximum of 32 fixed address ranges available that can be individually enabled/disabled by the address range enable bits AER.AENx (x = 0-31). If bit AER.AENx is set, read/write accesses to the associated address range x are supported in automatic mode. If bit AENx is cleared, read/write accesses to the associated address range x are not automatically executed, a memory protection error event is generated, and SRx output line is activated if enabled by RISR.MPEI.

The MLI module supports a definition of up to four programmable address sub-ranges (with index n) within fixed address ranges. The parameters for the sub-ranges are stored in the access range register ARR, comprising:

- The size of an address slice defined as sub-range (ARR.SIZE_n)
- The location of an address slice defined as sub-range (ARR.SLICE_n)

Note: The definition of the fixed address ranges and the sub-ranges is product-specific and given in [Section 23.5.6](#) on [Page 23-141](#).

23.2.2.6 Transmit Priority

In the case that several requests for frame transmission are pending at the same time in a MLI transmitter, the following priority scheme is applied, starting with the highest priority.

For the Answer Frame, only one frame can be pending at a time in the transmitter. So the user has to take care that an older Answer Frame is completely handled before requesting a new one. The same applies for the base address copy frame.

- Answer Frame (only one frame pending allowed at a time)
- Software driven Command Frames (CV0 before CV1 before CV2 before CV3)
- Read or Write Frames (DV0 before DV1 before DV2 before DV3)
- Base Address Copy Frame (only one frame pending allowed at a time)

23.2.2.7 Transmission Delay

A transmission delay can be introduced in the transmitter between the detection of the rising edge of the RREADY input signal and the next possible frame start. This delay represents the minimum time between the acknowledge of a former frame by RREADY and a new frame (if a request is pending). The delay is defined by bit field TCR.TDEL in cycles of the transmitter system clock f_{SYS} .

23.2.3 Interface Description

The MLI transmitter and MLI receiver communicate with other MLI receivers and MLI transmitters via a four-line serial connection each. Several I/O lines of these connections are available outside the MLI module kernel as a four-line output or input vector with index numbering A, B, C and D. The MLI module internal I/O control blocks define which signal of a vector is actually taken into account and also allow polarity inversions (to adapt to different physical interconnection means).

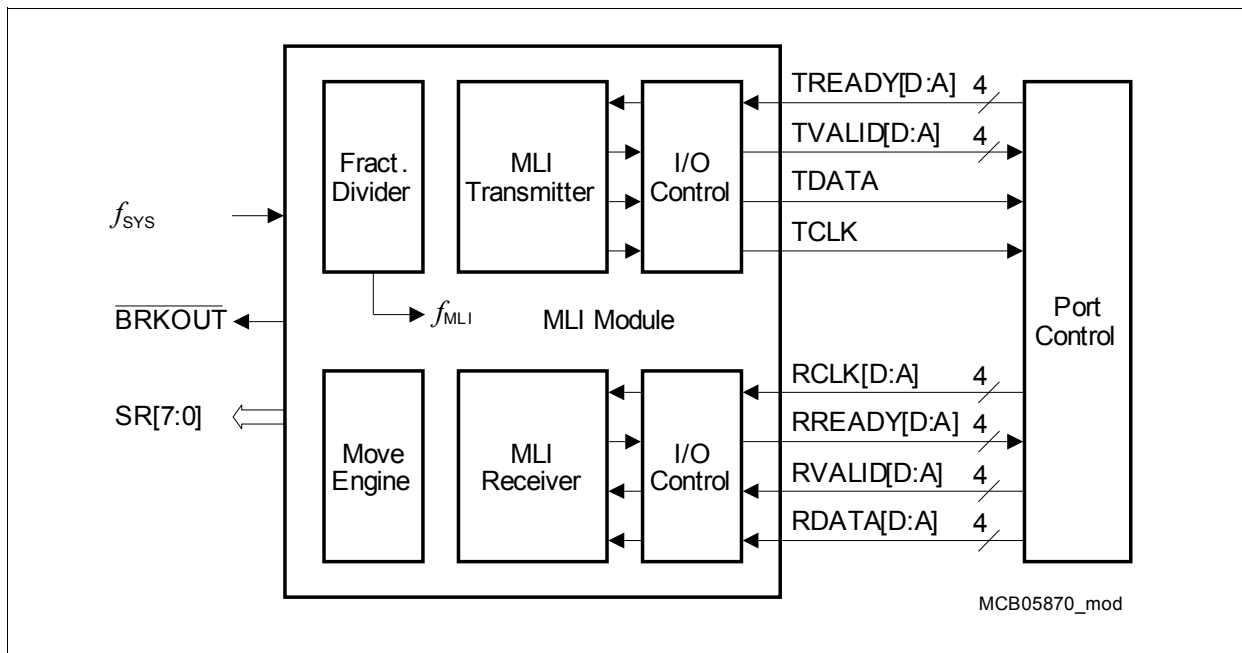


Figure 23-34 General Block Diagram of the MLI Module

Each input/output signal used for MLI communication between a transmitter and a receiver can be disabled and inverted in its polarity. Please note that all waveform diagrams in the MLI chapter refer to non-inverted signals. If polarity inversions are programmed, the waveform diagrams have to be interpreted accordingly. In order to avoid naming mismatches, the signals keep their names, although a polarity inversion might have been programmed. If desired, polarity inversions for the same signal have to be programmed in the transmitter and in the receiver to guaranty signal consistency (there has always to be an even number of inversions between an MLI transmitter and receiver). After reset, the following setting is applied, allowing MLI communication without modification of register OICR¹⁾:

- The signal with the index A is selected from each input/output vector.
- TCLK generation is enabled and RCLK reception is enabled.
- Polarity inversion is disabled for all signals (no inversion).

1) Other services (e.g. an automatic boot sequence or a boot routine) can change the OICR setting. Differing values are then indicated in the corresponding implementation chapter.

Micro Link Interface (MLI)

- Not selected output signals are at low level.

The usage of signal BRKOUT is implementation-specific and can be used, for example, to generate a break condition in the on-chip debug support logic or trigger other functions. This signal is activated (as a pulse) by a Command Frame.

The service request outputs SR[7:0] of the MLI module can be activated (as a pulse) by transmitter or receiver events (for all SRx), as well as by Command Frames (only for SR[3:0]).

23.2.3.1 Transmitter I/O Line Control

Figure 23-35 shows the MLI transmitter I/O control logic.

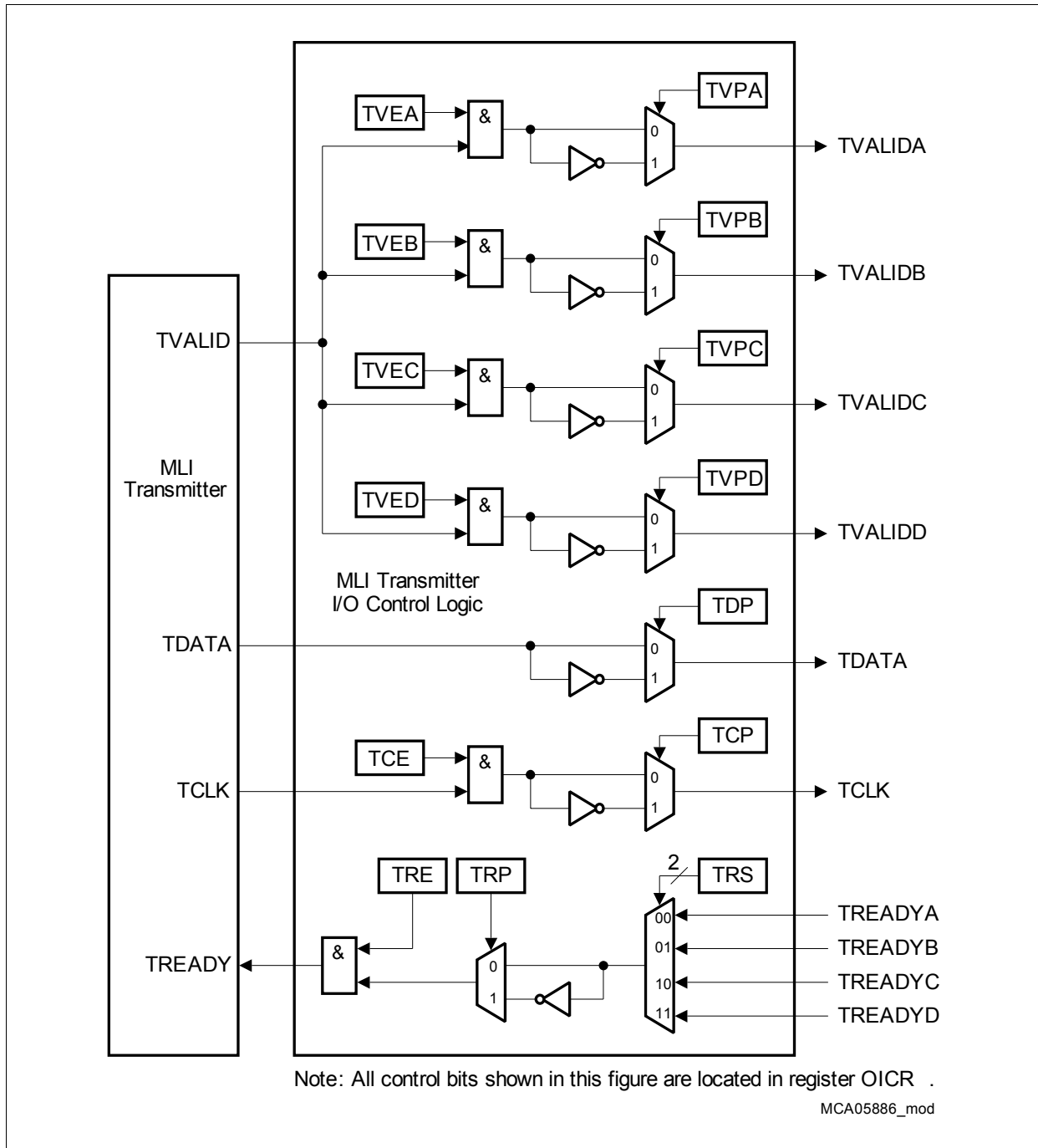


Figure 23-35 Transmitter Input/Output Control Logic

23.2.3.2 Receiver I/O Line Control

Figure 23-36 shows the MLI receiver I/O control logic.

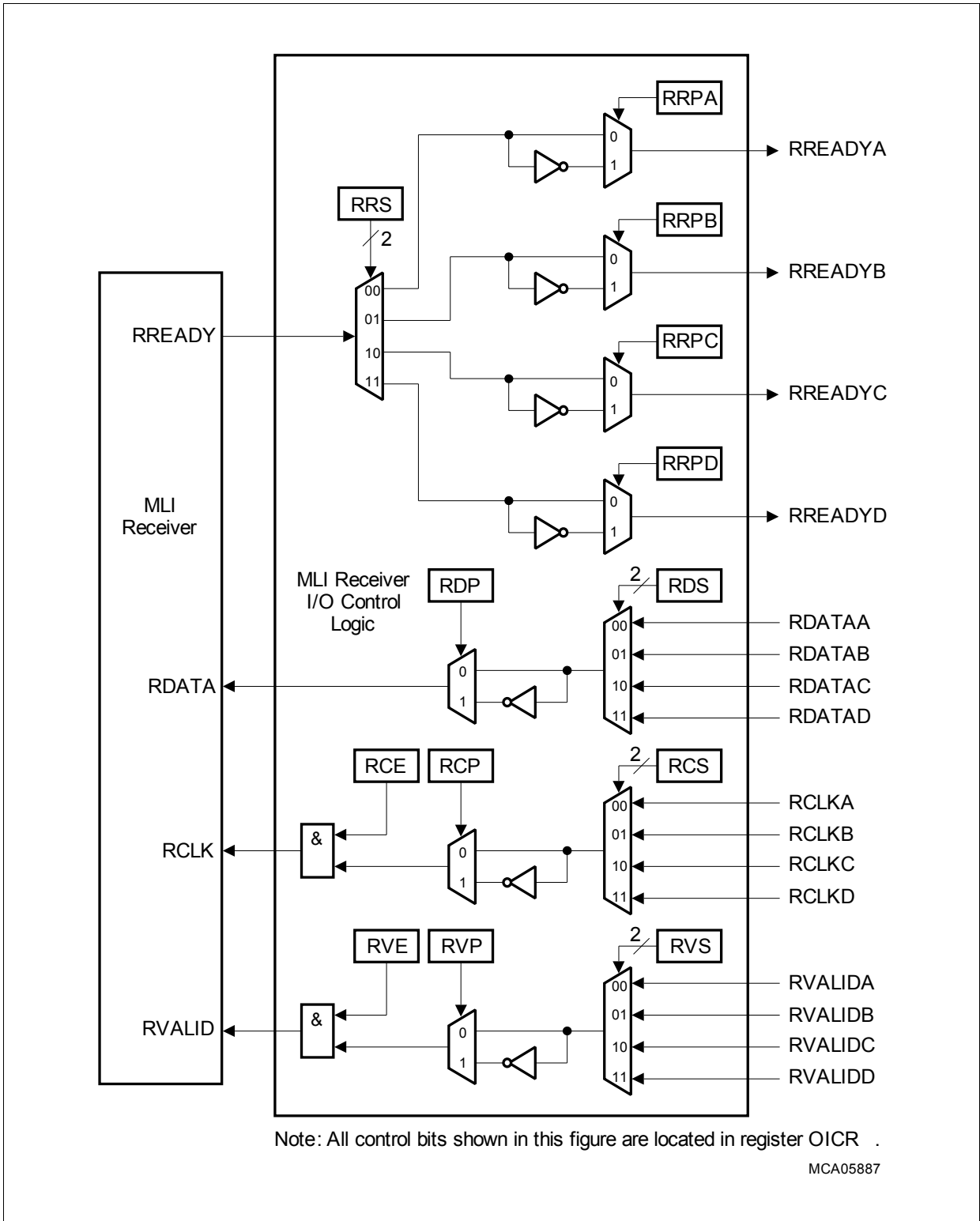


Figure 23-36 Receiver Input/Output Control Logic

23.2.3.3 Connecting Several MLI Modules

The MLI structure also allows to connect several MLI modules together, e.g. two Remote Controllers (X and Y) to one Local Controller. In this case, the Local Controller can send data to either one or the other or to both Remote Controllers in parallel. Each Remote Controller is connected via an own set of READY/VALID signals to the Local Controller, whereas the transmitter DATA and CLK are broadcast signals. The status of the VALID lines defines, which Remote Controller is accessed.

Only one receiver being available in the Local Controller, the reception of data can be handled only either from one or the other Remote Controller. The software has to ensure that only one Remote Controller sends data back to the Local Controller, e.g. by using Read Frames or by enabling/disabling the generation of Write Frames in the Remote Controllers.

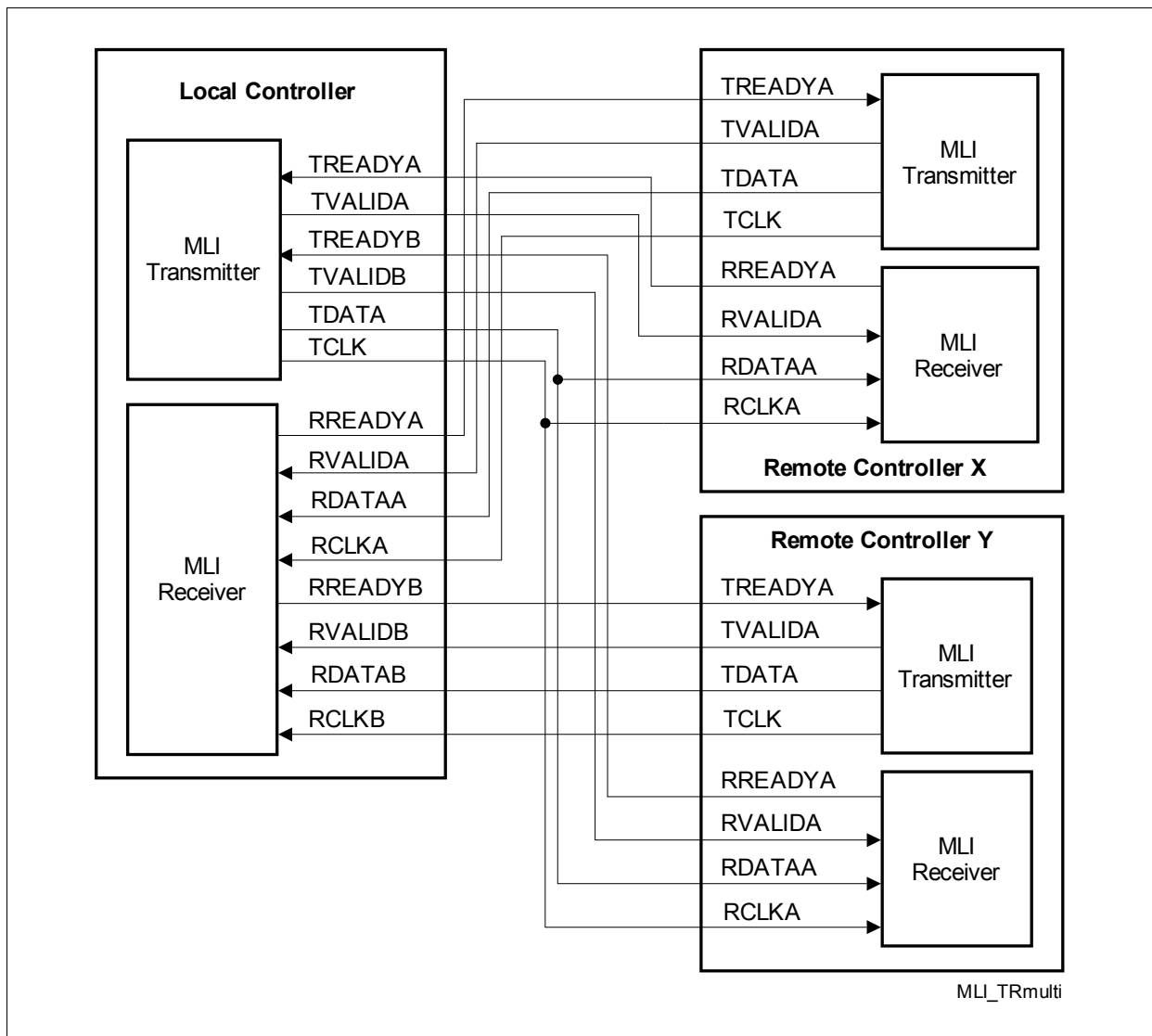


Figure 23-37 Connecting Two Remote Controllers

Micro Link Interface (MLI)

Another possibility to connect several MLI modules is a ring structure, with (at least) one dedicated pipe per device. This leads to a structure where the Local Controller's transmitter is connected to the receiver of Remote Controller X, the transmitter of Remote Controller X to the receiver of Remote Controller Y, and the transmitter of Remote Controller Y to the Local Controller's receiver.

This structure supports autonomous data generation and transfer in both Remote Controllers, for example to transfer data generated in a Remote Controller to the Local Controller without using Read Frames. In a ring structure, the Read Frame handling should be avoided. It is possible for the Local Controller to access both Remote Controllers independently. For example, the Remote Window of pipe x covers the address range of Remote Controller X, whereas pipe y targets the Transfer Window y of Remote Controller X. In Remote Controller Y, the pipe y targets the available address range. If the Local Controller issues a Write Frame on pipe x, the Remote Controller X is addressed. In case of a Write Frame on pipe y, the Remote Controller Y is targeted, passing through a Transfer Window of Remote Controller X. The two remaining pipes could be used for Write Frames issued by Remote Controller X (passing through a Transfer Window of Remote Controller Y) and by Remote Controller Y.

23.2.4 MLI Service Request Generation

The MLI module's service request outputs SRx are used to indicate module internal MLI events to other modules or devices outside the MLI module, depending on the device implementation. They can trigger interrupts of a CPU (if available), can be used as DMA request lines (if available), or for other trigger purposes. The MLI events being able to trigger interrupts or other service requests, names of some flags and control registers refer to interrupt generation.

MLI module events are generated by event sources in the transmitter and in the receiver. Each event source provides a status flag and an enable bit with software clear capability. In some cases, several event sources are combined to a common event. An MLI event, internally generated by an event source as a request pulse, is stored in a status flag that is located in the interrupt status registers TISR (for transmitter events) or RISR (for receiver events). All event flags can be cleared individually by software write actions to bits located in the interrupt enable registers TIER (for transmitter events) or RIER (for receiver events). These two registers also contain the enable control bits that allow each event source to be enabled/disabled individually for service request activation. Each event can be connected to exactly one of the eight service request outputs SR[7:0] by a 3-bit interrupt node pointer.

One additional register, the Global Interrupt Set Register GINTR, allows each service request output to be activated separately without setting the status flags of the event sources (see [Page 23-56](#)). This feature is sometimes helpful for software test purposes or to trigger MLI external actions.

Interrupt Registers

The MLI event sources are controlled by several registers (for transmitter see [Page 23-105](#), for receiver see [Page 23-117](#)). The register name prefixes "T" and "R" indicate if a register is assigned to the MLI transmitter or to the MLI receiver.

Table 23-6 Interrupt Registers

Unit	Registers with		
	Request Flags	Enable Bits/ Req. Flag Clear Bits	Node Pointer
MLI Transmitter	TISR	TIER	TINPR
MLI Receiver	RISR	RIER	RINPR

Service Request Compressor

The MLI event logic uses a compressing scheme for flexible service request processing. Eleven MLI events (six transmitter events and four of the five receiver events) are directed via a 3-bit interrupt node pointer to one of the eight service request outputs

Micro Link Interface (MLI)

SR[7:0]. Each demultiplexer output selected by its Node Pointer = x ($x = 0-7$) is connected to one input of the SR x OR-Gate. This wiring scheme also supports the connection of more than one event source to an service request output SR x . One receiver event, the interrupt Command Frame event, has a special characteristic: its node pointer is controlled by the received CMD value directly and only SR[3:0] OR-Gates are selectable.

Figure 23-38 shows the service request compressing logic. For reasons of simplicity, not all MLI events, connections, and OR-Gates are explicitly shown. The OR-Gate inputs are connected to the demultiplexers of the MLI event specific lines. Furthermore, a service request output SR x can be triggered by software if the corresponding interrupt set bit in register GINTR is written with a 1.

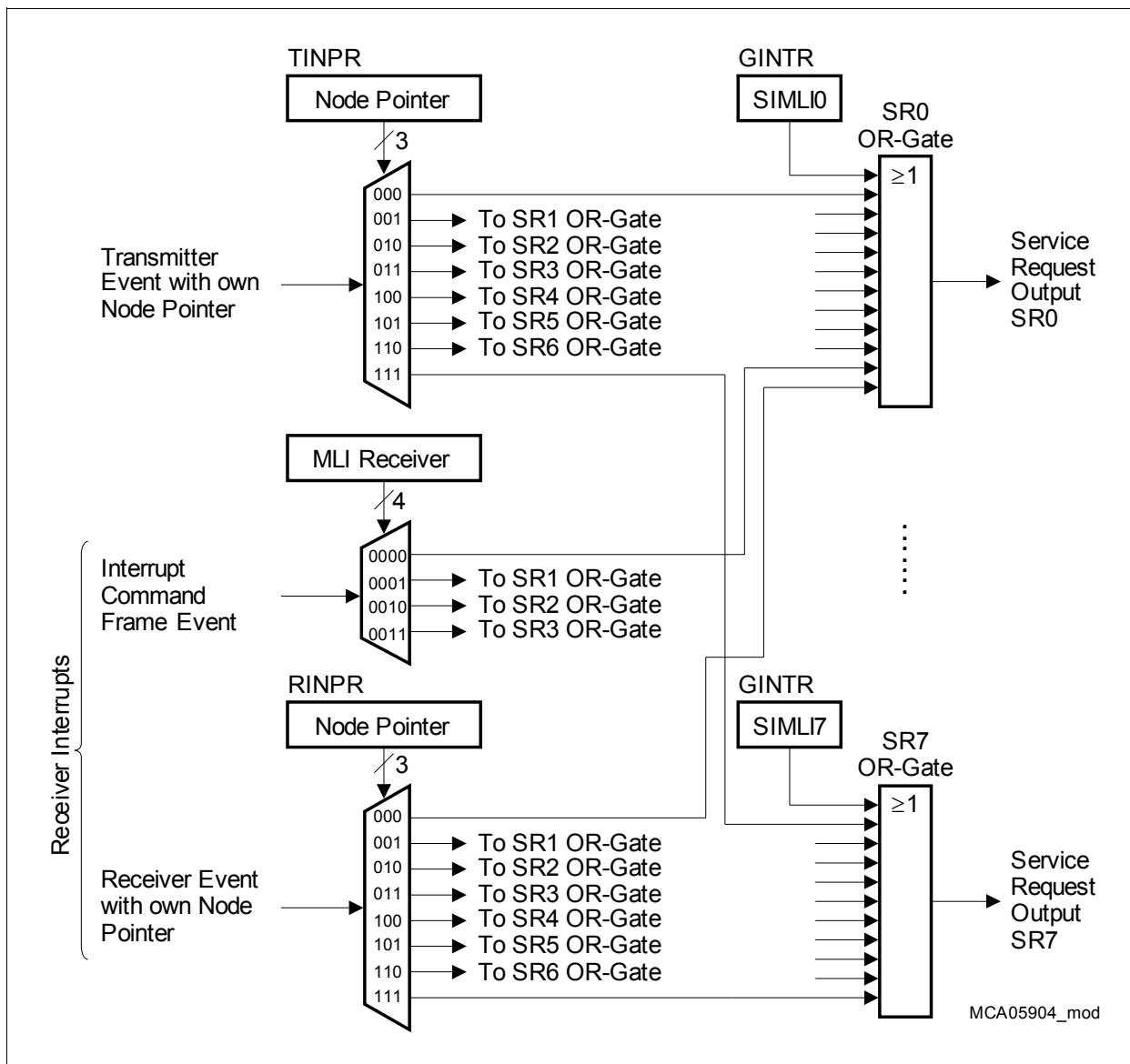


Figure 23-38 Service Request Compressor

Note: The number of SRx outputs of an MLI module and their connection to other modules depends on the implementation of the MLI module in the specific product.

23.2.5 Transmitter Events

The MLI transmitter can generate the following MLI events:

Table 23-7 MLI Transmitter Events

Events	Events combined to	See
Parity Error	Parity/Time-out Error	Page 23-58
Time-out Error		
Normal Frame Sent in Pipe 0	Normal Frame Sent in Pipe 0	Page 23-58
Normal Frame Sent in Pipe 1	Normal Frame Sent in Pipe 1	
Normal Frame Sent in Pipe 2	Normal Frame Sent in Pipe 2	
Normal Frame Sent in Pipe 3	Normal Frame Sent in Pipe 3	
Command Frame Sent in Pipe 0	Command Frame Sent	Page 23-59
Command Frame Sent in Pipe 1		
Command Frame Sent in Pipe 2		
Command Frame Sent in Pipe 3		

23.2.5.1 Parity/Time-out Error Event

A parity/time-out error event is generated when a programmable maximum number of parity errors or a programmable maximum number of Non-Acknowledge errors have been reached. Both events have separate status/control bits but are concatenated to one common error event.

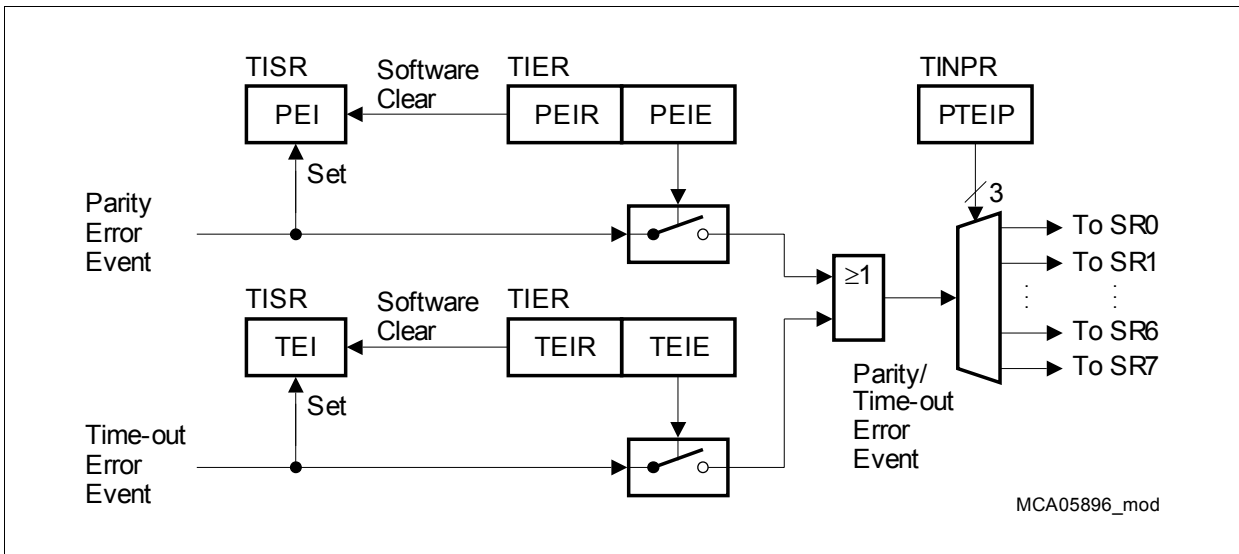


Figure 23-39 Parity/Time-out Error Event Logic

23.2.5.2 Normal Frame Sent x Event

A Normal Frame sent in pipe x (x = 0-3) event is generated when a Normal Frame has been sent and correctly received in pipe x.

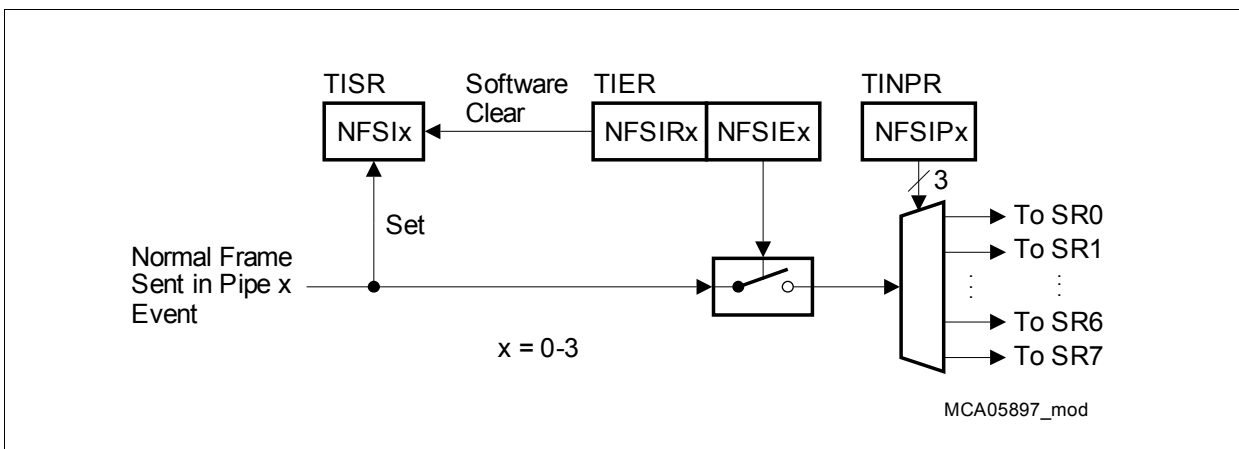


Figure 23-40 Normal Frame Sent in Pipe x Event Logic

23.2.5.3 Command Frame Sent Events

A Command Frame sent event is generated when the MLI transmitter has sent a Command Frame through pipe x (x = 0-3) that has been correctly received. Separate status/control bits are assigned to each pipe. All four pipe related Command Frame sent events are concatenated to one common Command Frame sent event.

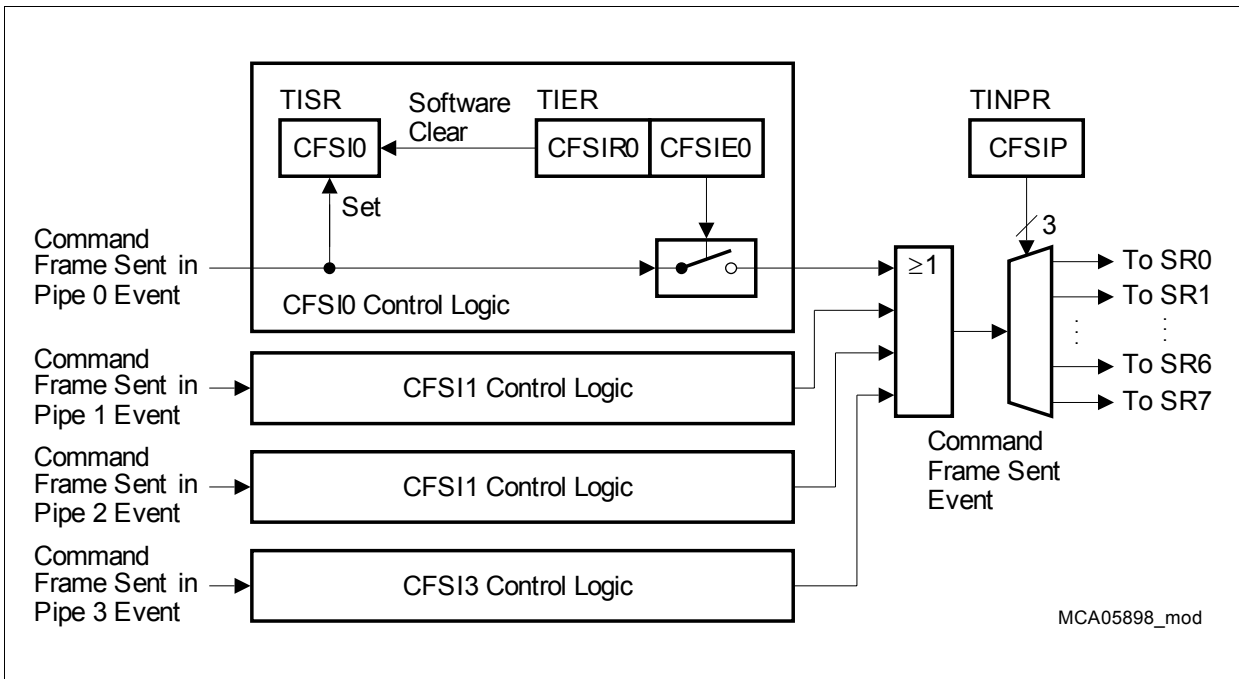


Figure 23-41 Command Frame Sent Event Logic

23.2.6 Receiver Events

The MLI receiver can generate the following MLI events:

Table 23-8 MLI Receiver Interrupts

Events	Events combined to	See
Discarded Read Answer	Discarded Read Answer	Page 23-60
Memory Access Protection Error	Memory Access Protection/ Parity Error	Page 23-61
Parity Error		
Normal Frame Correctly Received	Normal Frame Received	Page 23-62
Move Engine Access Terminated		
Interrupt Command Frame	Interrupt Command Frame	Page 23-63
Command Frame Received on Pipe 0	Command Frame Received	Page 23-64
Command Frame Received on Pipe 1		
Command Frame Received on Pipe 2		
Command Frame Received on Pipe 3		

23.2.6.1 Discarded Read Answer Event

A discarded read answer received event is generated if an Answer Frame has been received and the read pending flag TRSTATR.RPx of its correspondent pipe is 0. Although named “discarded”, the received data is available in the receiver data register until it is overwritten by the next incoming data.

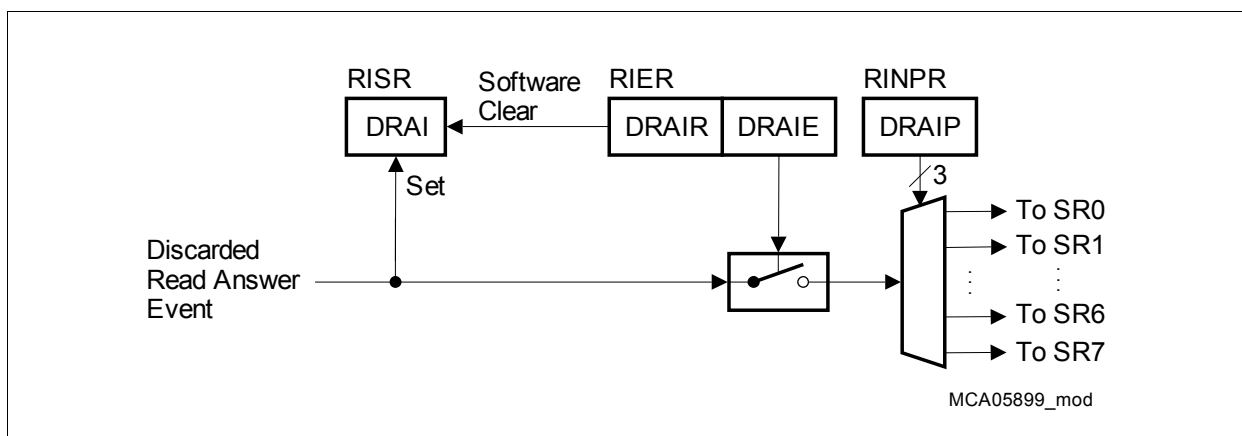


Figure 23-42 Discarded Read Answer Event Logic

23.2.6.2 Memory Access Protection/Parity Error Event

A memory access protection/parity error event is detected if a non allowed read or write access has been detected or if a programmable maximum number of receiver parity errors is reached. Both MLI events have separate status/control bits but are concatenated to one common error event.

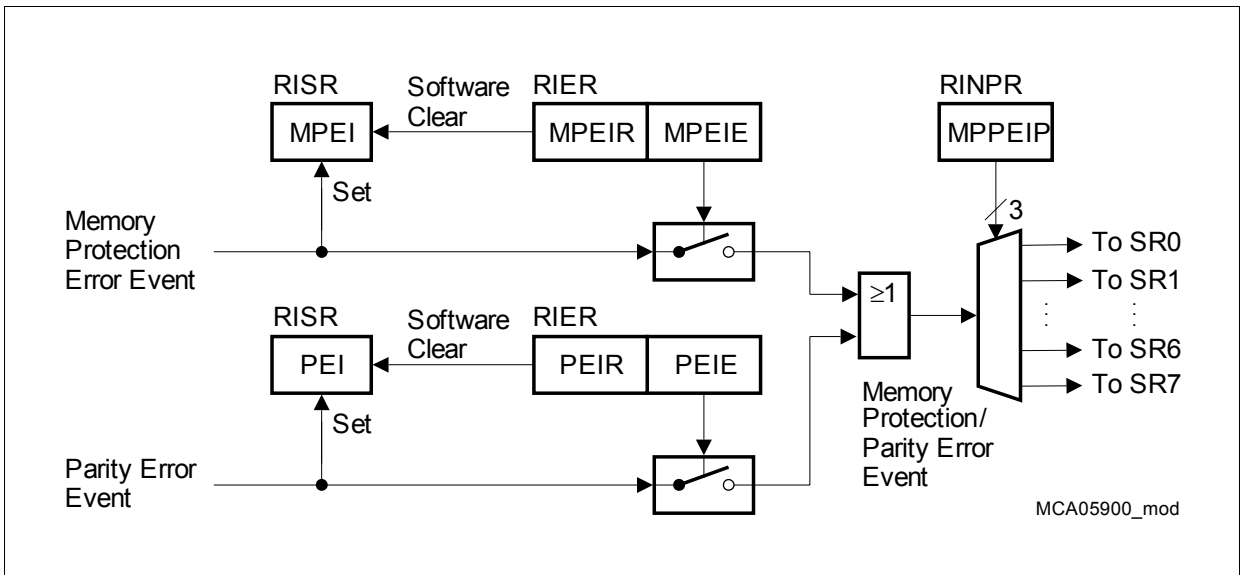


Figure 23-43 Memory Access Protection/Parity Error Event Logic

23.2.6.3 Normal Frame Received/Move Engine Terminated Event

A Normal Frame received event is generated if the MLI receiver has correctly received a Normal Frame (a Copy Base Address Frame, a Read or a Write Frame, an Answer Frame, but not a Command Frame) or if the move engine has terminated its read or write access. Both event sources have separate status/control bits but are concatenated to one common Normal Frame received event.

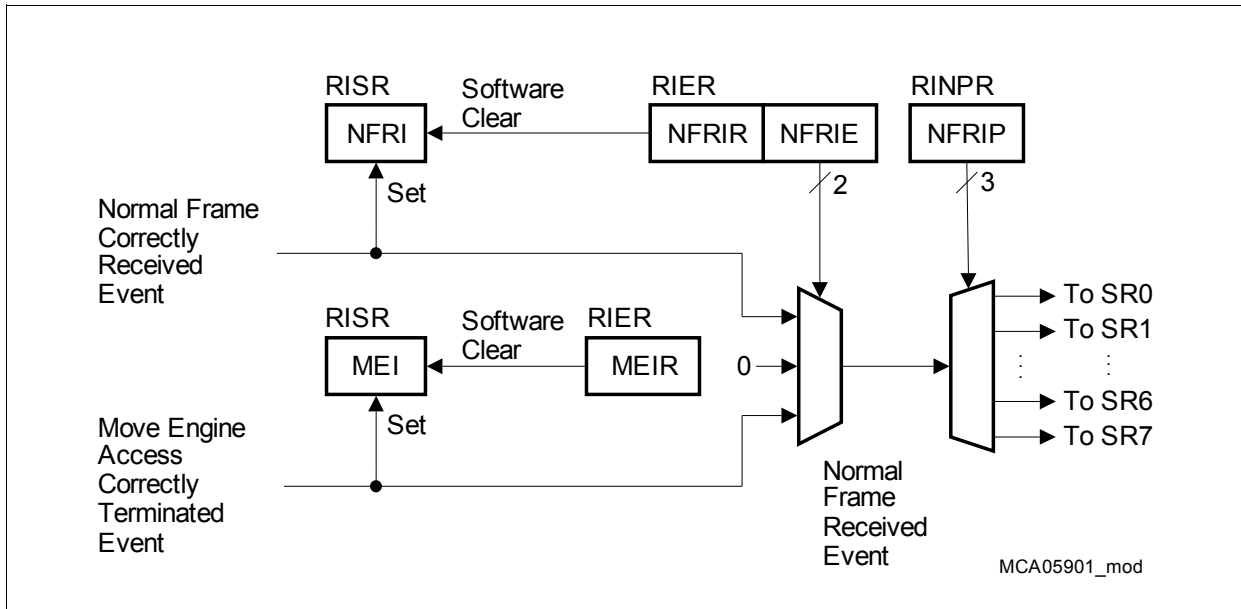


Figure 23-44 Normal Frame Received Event Logic

23.2.6.4 Interrupt Command Frame Event

An interrupt Command Frame event is generated if a Command Frame is received correctly on pipe 0 with a valid command code for service request output activation (CMD = 0000_B to 0011_B). The received command code determines which of the service request outputs SR[3:0] should be activated.

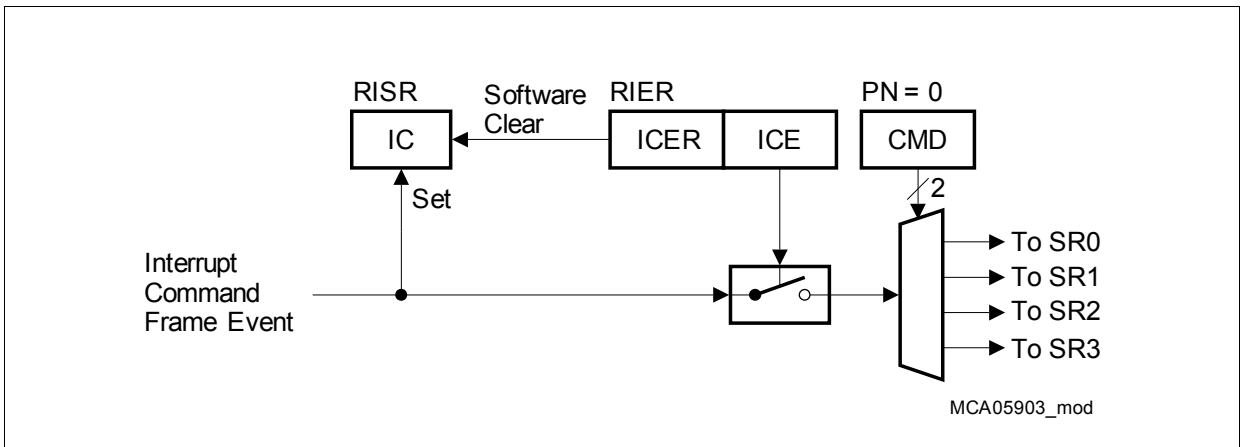


Figure 23-45 Interrupt Command Frame Event Logic

23.2.6.5 Command Frame Received Event

A Command Frame received event is generated if the MLI receiver has correctly received a Command Frame through Pipe Number x (x = 0-3). Separate status/control bits are assigned to each pipe. All four pipe related Command Frame received in pipe x events are concatenated to one common Command Frame received event.

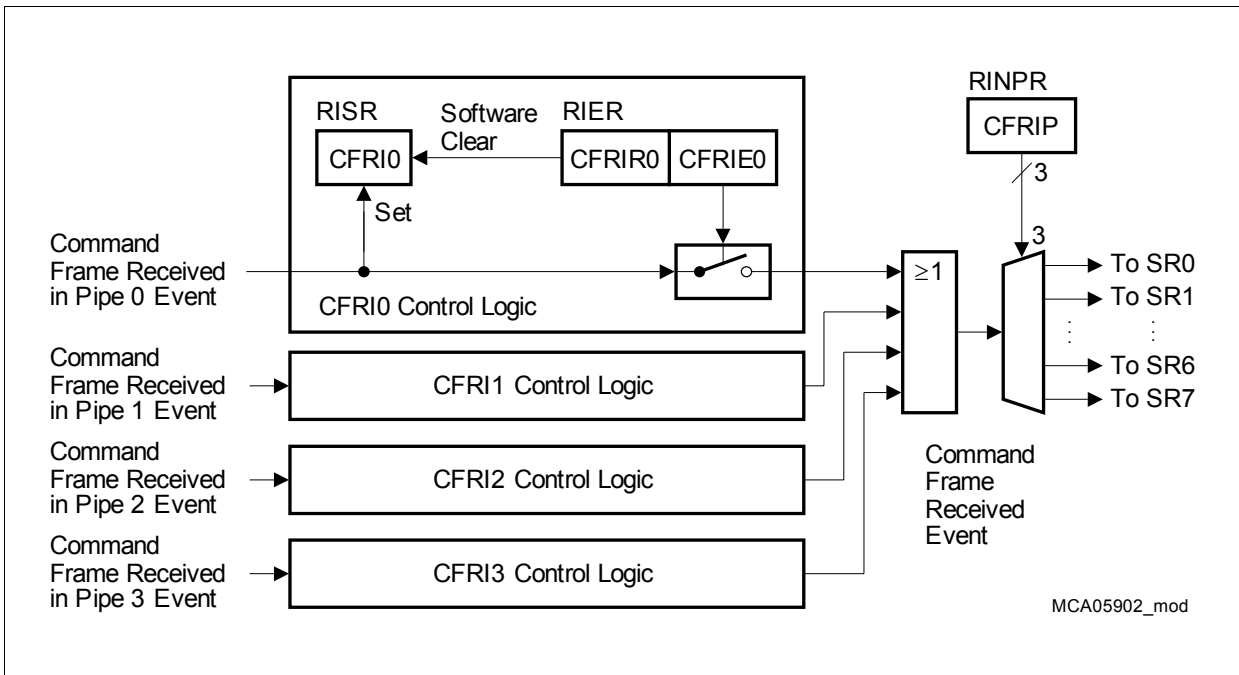


Figure 23-46 Command Frame Received Event Logic

23.2.7 Baud Rate Generation

The MLI transmitter baud rate is given by $f_{MLI}/2$. The MLI shift clock output signal TCLK of the transmitter toggles with each clock cycle of f_{MLI} in order to obtain a 50% duty cycle (the 50% duty cycle can vary up to one clock cycle of f_{SYS} in fractional divider mode). The MLI receiver automatically adapts to the incoming receive shift clock signal RCLK. The received baud rate is determined by the connected transmitter and has no direct relation to f_{SYS} except that it should not exceed f_{SYS} .

The frequency f_{MLI} is generated by the fractional divider FDIV.

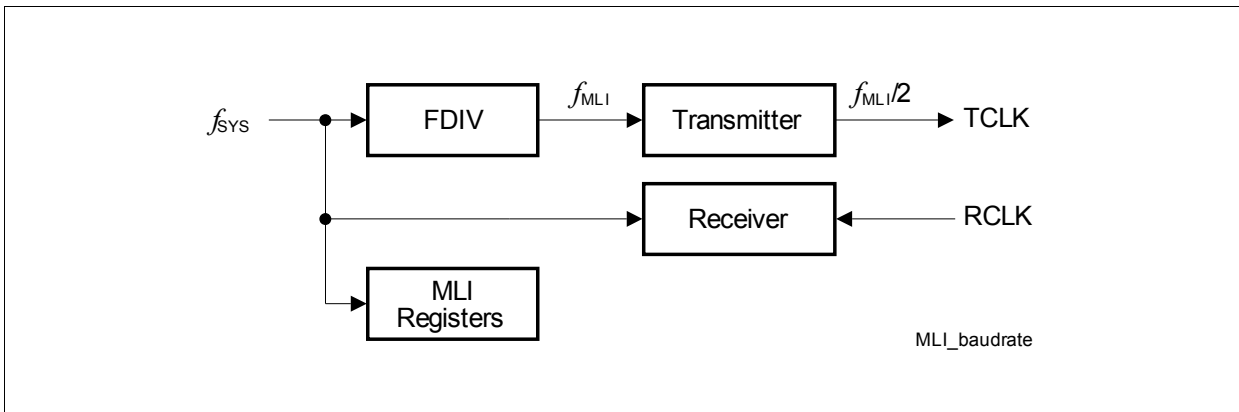


Figure 23-47 MLI Baud Rate Generation

Normal Divider Mode

In normal divider mode ($FDR.DM = 01_B$) the fractional divider behaves like a reload counter (addition of +1) that generates a clock f_{MLI} on the transition from $3FF_H$ to 000_H . $FDR.RESULT$ represents the counter value and $FDR.STEP$ defines the reload value. In order to achieve $f_{MLI} = f_{SYS}$, $FDR.STEP$ must be programmed with $3FF_H$. The output frequency in normal divider mode is defined according the following equation:

$$f_{MLI} = f_{SYS} \times \frac{1}{1024 - FDR.STEP} \quad (23.1)$$

Fractional Divider Mode

If the fractional divider mode is selected ($FDR.DM = 10_B$), the clock f_{MLI} is derived from the input clock f_{SYS} by division of a fraction of $STEP/1024$ for any value of $STEP$ from 0 to 1023. In general, the fractional divider mode allows to program the average clock frequency with a higher accuracy than in normal divider mode. In fractional divider mode a clock pulse f_{MLI} is generated depending on the result of the addition $FDR.RESULT + FDR.STEP$. The frequency f_{MLI} corresponds to the overflows over $3FF_H$. Note that in fractional divider mode the clock f_{MLI} can have a maximum period jitter of one f_{SYS} clock period. This jitter is not accumulated over several cycles and does not

exceed one cycle of f_{SYS} .

The frequency in fractional divider mode is defined according the following equation:

$$f_{\text{MLI}} = f_{\text{SYS}} \times \frac{\text{STEP}}{1024} \quad (23.2)$$

The baud rate of MLI transmissions equals f_{TCLK} , that is defined by the frequency of clock signal f_{MLI} divided by 2 to create the 50% duty cycle of the shift clock signal TCLK. The signal TCLK toggling with each period of f_{MLI} , a jitter due to fractional dividing is propagated to TCLK.

$$f_{\text{TCLK}} = \frac{f_{\text{MLI}}}{2} \quad (23.3)$$

23.2.8 Automatic Register Overwrite

The value of register OICR and bit RCR.RCVRST is overwritten by hardware in the next two clock cycles after a reset (first OICR, followed by RCR). The value applied during reset is given in the register description. This automatic overwrite allows adapting the module to different application requirements without changing the module itself. For example, during reset the receiver is set to a defined state and can be used afterwards for reception without the need to modify it by a write action (if the bit RCVRST is modified to 0).

The values applied after the overwrite can be identical to the indicated reset values. Please refer to the implementation chapter for the modified values (see [Page 23-127](#)).

23.3 Operating the MLI

Data transfer via MLI between a Local Controller and a Remote Controller is only possible if both are initialized correctly by following sequence of 4 steps. Steps 3 and 4 are necessary if the initialization sequence is exclusively controlled by the Local Controller. If both communication partners are able to run initialization software, steps 1 and 2 can be executed separately by both controllers to initialize both transmitters and both receivers.

1. The transmitter of the Local Controller has to be initialized by write actions to the transmitter registers.
2. The pipes from the Local Controller's transmitter to the Remote Controller's receiver and the Remote Controller's receiver have to be initialized.
3. The Remote Controller's transmitter has to be initialized by data write actions from the Local Controller via the Remote Controller's receiver to the Remote Controller's transmitter registers.
4. The pipes from the Remote Controller's transmitter back to the Local Controller's receiver and the Local Controller's receiver have to be initialized. This is done by frames from the Remote Controller's transmitter. These frames are the result of data write actions of the Local Controller to the Remote Controller's transmitter registers.

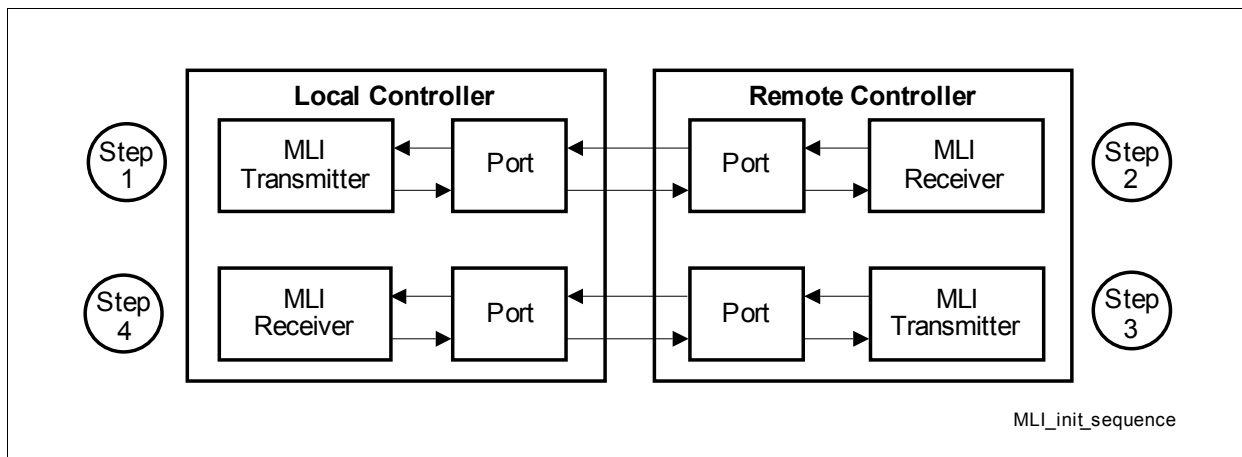


Figure 23-48 Initialization Sequence for an MLI Connection

To initialize and to operate the MLI, the following items should be taken into account:

- Connection setup (see [Page 23-68](#))
- Local Controller transmitter and pipe setup (see [Page 23-69](#))
- Remote Controller receiver setup (see [Page 23-69](#))
- Remote Controller transmitter and Local Controller receiver setup (see [Page 23-70](#))
- Delay adjustment (see [Page 23-71](#))
- Connection to DMA mechanism (see [Page 23-73](#))
- Connection of MLI to SPI (see [Page 23-73](#))

23.3.1 Connection Setup

For the general setup of an MLI connection, several steps have to be respected.

- There is the possibility to change the signal routing to adapt to different applications. If another connection than the default one from an input/output vector of the MLI signals is desired, register OICR has to be programmed (see also [Section 23.2.8](#)).
- In some devices (mainly stand-alone peripheral devices without CPU, where the MLI module is a possible communication channel), the setting “A” can be modified by hardware to another setting (e.g. to setting “B”) during the boot phase. In this case, the initial setting “A” can correspond to an inactive setting (MLI not used for communication), whereas the setting “B” is used for MLI communication.
- In the case a memory access protection is implemented in the receiver and automatic handling of data is desired, the user has to enable the corresponding address range in registers AER and ARR. After a reset, in most microcontrollers, the access protection is generally disabled to avoid access to safety-critical data. Depending on the device, some specific address ranges can already be enabled for automatic access by default.
- In devices with explicit port control (such as microcontrollers), the port pins are generally set to input after a reset. In order to allow MLI communication, the MLI-related port pins have to be configured to make the MLI signals externally available and to adapt the driver setting (refer to port chapter).

The MLI module should not be enabled for reception ($RCR.RCVRST = 1$) before programming the desired port setting, because changing the port setting can lead to unintended edges at the module inputs due to setting changes. If the MLI module is already enabled for reception, unintended edges are interpreted as communication signals, so the receiver might deliver wrong results. If this has happened unintentionally, the receiver can be reset by $RCR.RCVRST=1$.

23.3.2 Local Transmitter and Pipe Setup

The initialization of the transmitter of the Local Controller is done by writing to the transmitter registers. The Remote Controller's MLI receiver can then be initialized by the Local Controller's transmitter.

- After a hardware reset operation, the MLI transmitter is disabled (TCR.MOD = 0). In disabled mode, no frame transmission can take place. After writing TCR.MOD = 1, the transmitter is enabled to send frames.
- The desired transmitter baud rate can be adjusted by the fractional divider FDIV. It has to be ensured that the fractional divider is set to a value that is supported by the port structures of the Local and the Remote Controllers (rise/fall times) and the physical layer. For example, if a division by 1,5 is selected, the fractional divider will deliver count pulses for f_{MLI} with a sequence of 1-2-1-2-1-2- clock cycles of f_{SYS} . The shortest interval between two count pulses in a sequence (given by the truncated divider factor, so 1 cycle of f_{SYS} in this example) has to be handled by the communicating devices.
- Depending on the application requirements, a desired service request output SRx can be activated if a transmitter event is detected.
- The maximum delay for parity error detection in the transmitter has to be programmed. There are two possibilities to get the MLI communication started. First (easier) possibility is to write TCR.MDP to 14 and to set RCR.DPE to 15. The second possibility could be used to optimize the bandwidth of the MLI connection. It is described in [Section 23.3.5](#) on [Page 23-71](#).

23.3.3 Remote Receiver Setup

The initialization of the Remote Controller's receiver is done by frames sent by the Local transmitter. Therefore, the Remote Controller's receiver has to be able to receive frames.

- In order to allow communication, the Remote Controller's MLI signals have to be connected to the Local Controller's transmitter signals (see register OICR and port settings).
- The Remote Controller's bit RCR.RCVRST has to be 0 to enable frame reception.
- The buffer area size and the base address of the Remote Window for pipe x are defined by the data written to registers TPxBAR. Bit TRSTATR.BAV has to be 0 before each write action to one of these registers. With this information, the buffer area sizes (defining the number of address bits in data frames or Read Frames) are known in the transmitter and in the receiver for each pipe.
The base addresses for the Remote Windows have to be selected to cover the target address ranges in the Remote Controller. It is recommended to use the minimum buffer size required by the application in order to minimize the bandwidth taken by the transfer of the address bits. The base address of a Remote Window has to be set to a value aligned to its size, e.g. a Remote Window of 8 Kbytes must start at an 8 Kbyte address boundary.

Micro Link Interface (MLI)

- In devices with access protection mechanism against unauthorized accesses via MLI, the Remote Controller has to enable the desired address range(s) to support automatic mode. If automatic mode is not desired, the Remote Controller has to handle the complete data traffic by software.
- A possibility to test the setup in devices with the capability to run own test software is the local loop back (the transmitter is connected locally to the receiver of the same MLI module). In devices without this capability, the module loop back can be hardly used (or it is even not implemented, refer the connection table in the implementation chapter).

In local loop back mode (see [Section 23.5.1](#) on [Page 23-124](#)), the signal connections have to be programmed to setting “D”, leading to the local receiver being connected directly to the local transmitter (without using a port structure). In this case, the local receiver seems to be the remote receiver. Data written to a local Transfer Window are received and handled by the local receiver. Test software in the Local Controller can check for correct setup, data consistency, MLI event handling, and correct address handling in the Local Controller.

- If automatic data handling is desired (necessary for devices without the capability to handle data traffic by its CPU), the Automatic Data Mode has to be enabled by sending a Command Frame in pipe 2 with $CM = 0001_B$ to set $RCR.MOD = 1$ in the Remote Controller.

23.3.4 Remote Transmitter and Local Receiver Setup

The initialization of the Remote Controller’s transmitter and the Local Controller’s receiver can be done by data frames sent by the local transmitter. Therefore, the Remote Controller’s receiver has to be able to receive frames (the port structure has to be set up accordingly).

- The Remote Window of pipe x (x can be freely chosen) has to be set to the MLI register address range in the Remote Controller. The initialization by data frames is then done via pipe x.
- The automatic mode has to be enabled in the Remote Controller (Command Frame in pipe 2 with $CM = 0001_B$).
- The connections between the remote transmitter and the local receiver have to be established (if not already done by the default setting), similar to [Section 23.3.2](#).
- The remote transmitter has to be enabled, similar procedure as for the local transmitter. The data word to be written to the Remote Controller’s MLI registers have to be written to the corresponding address in the local Transfer Window of pipe x.
- The local receiver can then be configured by writing the appropriate data (similar scenario as for the remote receiver) to the local Transfer Window of pipe x.
- A possibility to test the complete setup is the remote loop back. In this case another Remote Window is overlaid directly to a Transfer Window in the Remote Controller. Writing data to the corresponding Transfer Window in the Local Controller leads to a data frame sent to the Remote Controller. There, the received data is written to the

Transfer Window and a new data frame is sent back to the Local Controller. The MLI move engine in the Local Controller's receiver can be used to write the received data to a defined location, e.g. to a memory location. Test software in the Local Controller can check for correct setup, data consistency, MLI event handling, and correct address handling in the Local and the Remote Controllers.

23.3.5 Delay Adjustment

The local MLI transmitter is measuring the number of TCLK clock cycles between TVALID becoming 0 after a transmission and TREADY becoming 1 again. This time represents the overall loop delay of the MLI connection. The loop delay is the time used for signal propagation, input/output driver delay and remote receiver reaction. For example, with slow drivers and a high load (due to long wires, etc.), the signals take a longer time to propagate from the local transmitter to the remote receiver and back again (READY-VALID control handshake). This delay (also visible when TVALID becomes 1 at the beginning of a frame) limits the maximum baud rate of an MLI connection, because the answer of the receiver has to be detected by the transmitter with TREADY = 0 at the end of the frame. The value measured after the end of the frame is indicated in bit field TSTATR.RDC.

The receiver participates in the control handshake by changing its RREADY output as a reaction to an incoming RVALID signal. For the transmitter, the TREADY input delivers the information that a receiver is connected and that it is ready for reception (transfer only starts if TREADY = 1). If a receiver is not able to handle the data or is not connected, the TREADY line will not become low after TVALID becomes 1 (Non-Acknowledge).

In addition to this information, the MLI protocol offers the possibility to use the control handshake also to indicate that the receiver has detected a parity error in the received frame. If a correct frame has been received, the receiver immediately asserts RREADY = 1 after the reception of a frame when detecting RVALID = 0. If the receiver has detected a parity error, it waits for a programmable number of RCLK cycles before setting RREADY = 1 again. This additional delay is defined by bit field RCR.DPE.

The transmitter measuring the delay and comparing it to a programmed value, it can detect that the receiver has signaled a parity error by introducing the additional delay. The compare value for the transmitter is programmed by bit field TCR.MDP. A measured value of TSTATR.RDC above TCR.MDP is interpreted as parity error by the transmitter (for parity error handling refer to [Page 23-42](#)).

In the receiver, frames with parity error are ignored for data transfers and don't lead to internal move actions.

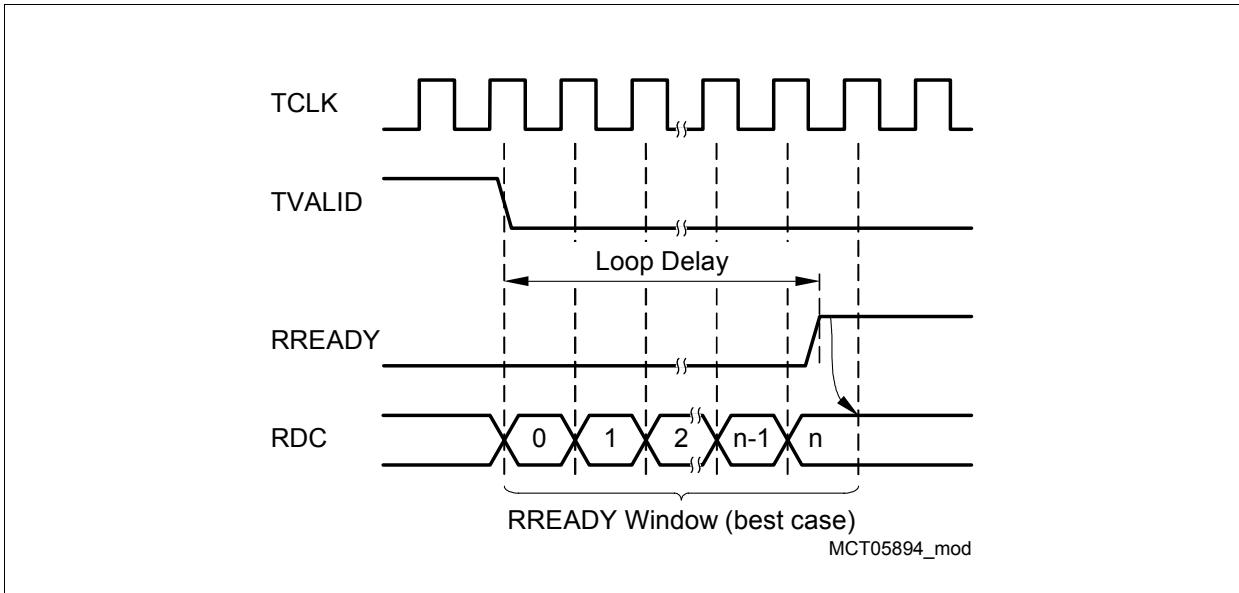


Figure 23-49 Loop Delay Measurement

To adjust the generated parity delay in the local transmitter and in the remote receiver, the following steps are necessary:

- Send a dummy frame to the receiver for measuring the loop delay. This frame should not lead to internal data move actions in the receiver, so a parity error can be simulated in the transmitter. The receiver has a fixed even parity scheme, whereas the transmitter can be programmed either for even or for odd parity. Programming odd parity before sending a frame will generate a (dummy) frame that will be discarded by the receiver (assuming a correct transfer). For a dummy frame, it is recommended to use a data frame with disabled Automatic Data Mode in the receiver ($RCR.MOD = 0$).
- The receiver delay $RCR.DPE$ being 0 after a module reset, the transmitter can measure the loop delay and the receiver discards the frame (without modification of DPE , there is no difference in time between a frame with or without a parity error having been detected). The value given by $TSTATR.RDC$ indicates how many $TCLK$ cycles are necessary for a control handshake. This value should be incremented by a value $DELTA$ (value see below) and written to $TCR.MDP$.
- The transmitter parity has to be programmed to even parity to be able to generate frames that are not discarded by the receiver.
- Programming the receiver delay for parity error ($RCR.DPE$) to a value bigger than $DELTA$ will lead to a value of $TSTATR.RDC$ bigger than $TCR.MDP$ if the receiver detects a parity error. The value of DPE in the remote receiver is modified by the local transmitter by sending a Command Frame in pipe 1 with the desired value. The difference between $TSTATR.RDC$ and $TCR.MDP$ allows a certain timing tolerance between local transmitter and remote receiver.

Micro Link Interface (MLI)

- The value of DELTA depends on the possible variations of the propagation characteristics of the MLI connection. If the environment does not significantly change, DELTA can be 1. For systems with variations, DELTA could be bigger. The user can check about changing propagation characteristics by reading TSTATR.RDC from time to time and to check if it is constant for correct transfers. If it changes, either a bigger DELTA value can be applied, or the delay adjustment can be repeated, adapting to the new circumstances.

23.3.6 Connection to DMA Mechanism

The MLI module supports the connection to a DMA (direct memory access) mechanism. This mechanism allows the transfer of blocks of data of programmable size via an MLI connection without CPU intervention. Therefore, a DMA mechanism can be used in the Local Controller to write the desired number of data words one after the other to the corresponding MLI Transfer Window. The address ranges of the data blocks and their length has to be handled by the DMA module.

An MLI pipe supporting only one pending Write Frame request at a time, the DMA has to wait until the pipe is capable to handle new data before writing another data word to the Transfer Window. Therefore, the Normal Frame sent events of the pipes can trigger DMA data transfers. Depending on the connection of the MLI module's service request outputs SRx to the DMA trigger inputs, the Normal Frame sent events have to be enabled for service request activation and directed to the desired SRx outputs. It is recommended to use only one type of MLI event per SRx output to trigger a data transfer by DMA. If the DMA mechanism needs a start trigger for the first data word transfer, register GINTR can be written with the appropriate pattern to activate an SRx output.

23.3.7 Connection of MLI to SPI

The handshake signals between a transmitter and a receiver are based on a synchronous transfer protocol. In the SPI protocol, the shift clock and the data signal are equivalent to CLK and DATA. In case of an 4-wire SPI, the slave select signal represents the VALID signal (the leading and the trailing delay have to be set up accordingly).

Contrary to the MLI, in the SPI protocol, a complete control handshake is not defined, so the READY signal does not exist in SPI modules. As a result, the SPI communication does not check by hardware for correct data transfer, but has to handle this on an upper software layer. If using an SPI module for communication with an MLI transmitter or an MLI receiver, the READY signal has to be handled by software or the handshake has to be given up. This can be done by connecting the TVALID signal of an MLI transmitter to one of its own TREADY inputs with polarity inversion. Like this, the TREADY input directly following the inverted TVALID signal, the parity error indication and the Non-Acknowledge error detection are not possible.

Furthermore, in the MLI protocol, the frames may have a different width, depending on their type and selected buffer size. The different numbers of data bits per frame have

Micro Link Interface (MLI)

also to be handled by the SPI module. In order to minimize the number of different frames, it is recommended to restrict the possibility to program different buffer sizes, the use of Read Frames or Command Frames. In order to simplify the data handling by an SPI module, the parity generation could be skipped for frames received by the SPI module and an error detection mechanism on an upper software layer could be implemented. For frames sent by an SPI module, the parity bit has to be calculated and sent correctly. Otherwise, the MLI receiver will discard the received frame.

23.4 MLI Kernel Registers

This section describes the kernel registers of the MLI modules. All MLI kernel register names described in this section will be referenced in other parts of the TC1796 User's Manual by the module name prefix "MLI0_" for the MLI0 interface and "MLI1_" for the MLI1 interface.

MLI Kernel Register Overview

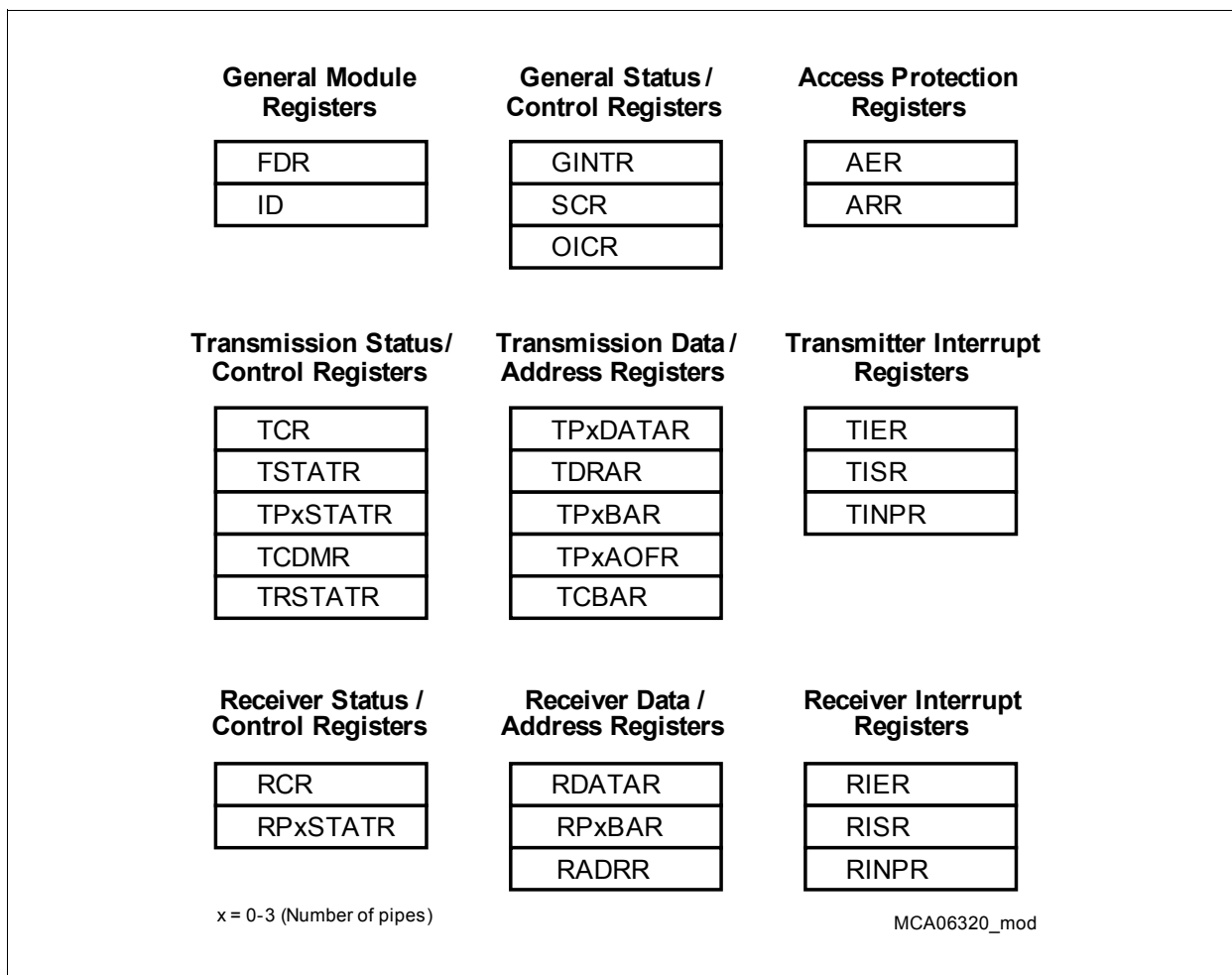


Figure 23-50 MLI Kernel Registers

All registers can be accessed with 8-bit, 16-bit or 32-bit write or read operations. Accesses to address locations inside the MLI address range not targeting the indicated registers are not allowed. The complete and detailed address map of the of the MLI modules is described in [Table 18-30](#) on [Page 18-97](#) of the TC1796 User's Manual System Units part (Volume 1).

Table 23-9 Registers Address Space - MLI Registers

Module	Base Address	End Address	Note
MLI0	F010 C000 _H	F010 C0FF _H	–
MLI1	F010 C100 _H	F010 C1FF _H	–

Table 23-10 Registers Overview - MLI Kernel Registers

Register Short Name	Register Long Name	Offset Address	Description see
ID	Module Identification Register	08 _H	Page 23-78
FDR	Fractional Divider Register	0C _H	Page 23-79
TCR	Transmitter Control Register	10 _H	Page 23-90
TSTATR	Transmitter Status Register	14 _H	Page 23-93
TPxSTATR	Transmitter Pipe x Status Register	18 _H + (x * 4)	Page 23-95
TCMDR	Transmitter Command Register	28 _H	Page 23-97
TRSTATR	Transmitter Receiver Status Register	2C _H	Page 23-99
TPxAOFR	Transmitter Pipe x Address Offset Register	30 _H + (x * 4)	Page 23-103
TPxDATAR	Transmitter Pipe x Data Register	40 _H + (x * 4)	Page 23-101
TDRAR	Transmitter Data Read Answer Register	50 _H	Page 23-101
TPxBAR	Transmitter Pipe x Base Address Register	54 _H + (x * 4)	Page 23-102
TCBAR	Transmitter Copy Base Address Register	64 _H	Page 23-104
RCR	Receiver Control Register	68 _H	Page 23-110
RPxBAR	Receiver Pipe x Base Address Register	6C _H + (x * 4)	Page 23-115
RPxSTATR	Receiver Pipe x Status Register	7C _H + (x * 4)	Page 23-113
RADDR	Receiver Address Register	8C _H	Page 23-116
RDATAR	Receiver Data Register	90 _H	Page 23-114
SCR	Set Clear Register	94 _H	Page 23-81
TIER	Transmitter Interrupt Enable Register	98 _H	Page 23-105

Table 23-10 Registers Overview - MLI Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Address	Description see
TISR	Transmitter Interrupt Status Register	9C _H	Page 23-107
TINPR	Transmitter Interrupt Node Pointer Register	A0 _H	Page 23-108
RIER	Receiver Interrupt Enable Register	A4 _H	Page 23-117
RISR	Receiver Interrupt Status Register	A8 _H	Page 23-120
RINPR	Receiver Interrupt Node Pointer Register	AC _H	Page 23-122
GINTR	Global Interrupt Set Register	B0 _H	Page 23-83
OICR	Output Input Control Register	B4 _H	Page 23-84
AER	Access Enable Register	B8 _H	Page 23-88
ARR	Access Range Register	BC _H	Page 23-89

Micro Link Interface (MLI)

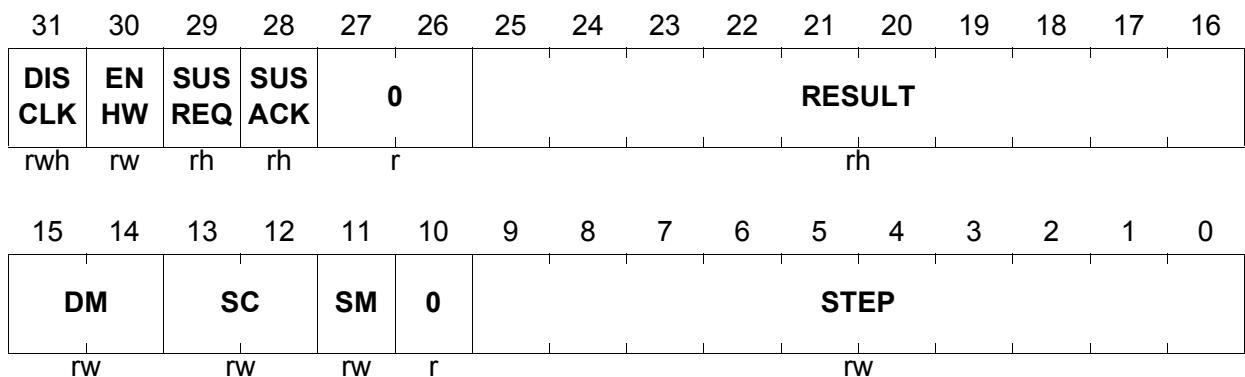
The fractional divider register allows the programmer to control the clock rate and period of the module clocks f_{MLIO} and f_{MLIX} . The period of f_{MLIX} can be either $1/STEP$ or a fraction of $STEP/1024$ (for any value of $STEP$ from 0 to 1023) of clock f_{MLI} . Each MLI has its own fractional divider.

FDR

Fractional Divider Register

(0C_H)

Reset Value: 03FF 43FF_H



Field	Bits	Type	Description
STEP	[9:0]	rw	Step Value Reload or addition value for RESULT.
SM	11	rw	Suspend Mode SM selects between granted or immediate suspend mode.
SC	[13:12]	rw	Suspend Control This bit field determines the behavior of the fractional divider in suspend mode.
DM	[15:14]	rw	Divider Mode This bit field selects normal divider mode or fractional divider mode.
RESULT	[25:16]	rh	Result Value Bit fields for the addition result.
SUSACK	28	rh	Suspend Mode Acknowledge Indicates the state of the SPNDACK signal.
SUSREQ	29	rh	Suspend Mode Request Indicates the state of the SPND signal.
ENHW	30	rw	Enable Hardware Clock Control Controls operation of ECEN input and DISCLK bit.

Micro Link Interface (MLI)

Field	Bits	Type	Description
DISCLK	31	rwh	Disable Clock Hardware controlled disable for f_{OUT} signal.
0	10, [27:26]	rw	Reserved Read as 0; should be written with 0.

*Note: Additional details on the fractional divider register functionality are described in section **“Clock Control Register CLC”** on **Page 3-24** of the TC1796 User’s Manual System Units part (Volume 1).*

23.4.2 General Status/Control Registers

The Set Clear Register SCR is a write only register that makes it possible to set or clear by software several status flags located in registers TSTATR, TRSTATR and RCR. Reading register SCR always returns zeros at all bit locations. Bits that are not written with a 1 have no effect.

SCR

Set Clear Register

(94_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				C	C	C	C	0				C	C		
				NAE	TPE	RPE	AV					BAV	MOD		
W				W	W	W	W	W				W	W		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C	C	C	C	C	C	C	C	0			S	S	S	S	S
CV3	CV2	CV1	CV0	DV3	DV2	DV1	DV0				MOD	CV3	CV2	CV1	CV0
W	W	W	W	W	W	W	W	W			W	W	W	W	W

Field	Bits	Type	Description
SCV_x (x = 0-3)	x	w	Set Command Valid 0 _B No effect 1 _B Bit TRSTATR.CV _x is set.
SMOD	4	w	Set MOD Flag 0 _B No effect 1 _B If CMOD = 0, RCR is set. If CMOD = 1, RCR.MOD is cleared.
CDV_x (x = 0-3)	8+x	w	Clear Data Valid x Flag 0 _B No effect 1 _B Bits TRSTATR.DV _x and TRSTATR.RP _x are cleared.
CCV_x (x = 0-3)	12+x	w	Clear Command Valid x Flag 0 _B No effect. 1 _B If SCV _x = 0, bit TRSTATR.CV _x is cleared. If SCV _x = 1, bit TRSTATR.CV _x is set.
CMOD	16	w	Clear MOD Flag 0 _B No effect. 1 _B Bit RCR.MOD is cleared.

Micro Link Interface (MLI)

Field	Bits	Type	Description
CBAV	17	w	Clear BAV Flag 0 _B No effect. 1 _B Bit TRSTATR.BAV is cleared.
CAV	24	w	Clear AV Flag 0 _B No effect. 1 _B Bit TRSTATR.AV is cleared.
CRPE	25	w	Clear Receiver PE Flag 0 _B No effect. 1 _B Bit RCR.PE is cleared.
CTPE	26	w	Clear Transmitter PE Flag 0 _B No effect. 1 _B Bit TSTATR.PE is cleared.
CNAE	27	w	Clear NAE Flag 0 _B No effect. 1 _B Bit TSTATR.NAE is cleared.
0	[7:5], [23:18], [31:28]	w	Reserved Read as 0; should be written with 0.

Micro Link Interface (MLI)

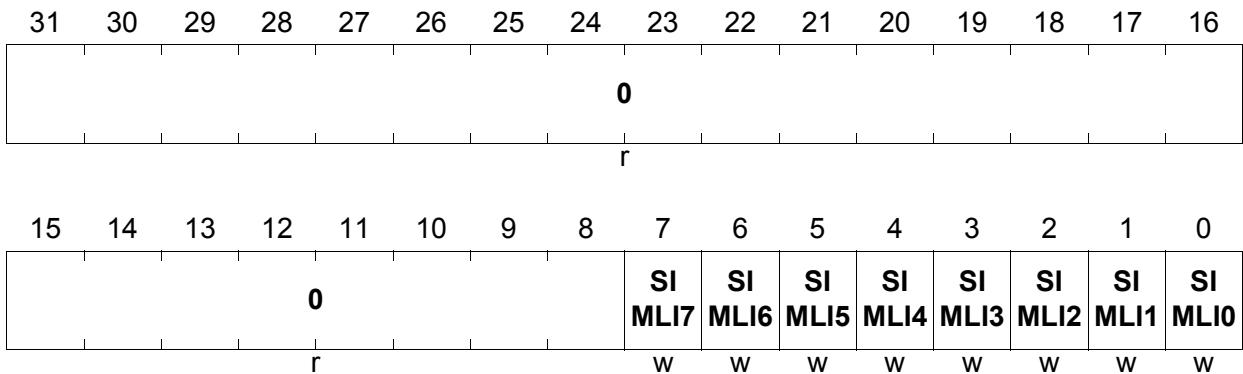
The Global Interrupt Set Register GINTR is a write only register (always reads 0) that allows each of the service request outputs SR_x to be activated under software control (see [Page 23-56](#)).

GINTR

Global Interrupt Set Register

(B0_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SIML_x (x = 0-7)	x	w	Set MLI Service Request Output Line x 0 _B No action 1 _B Service request output SR _x is activated (pulse).
0	[31:8]	r	Reserved Read as 0; should be written with 0.

Micro Link Interface (MLI)

The Output Input Control Register OICR determines the functionality of the MLI transmitter and MLI receiver I/O control logic.

The bits in this register are automatically overwritten after a reset with a value given in the implementation chapter (see [Page 23-127](#)). Furthermore, the connection table of the MLI module signals is given there.

OICR

Output Input Control Register (B4_H) **Reset Value: 1000 8000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDP	RDS	RCE	RCP	RCS	RVP	RVS	RRP D	RRP C	RRP B	RRP A	RRS				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RVE	TDP	TCP	TCE	TRE	TRP	TRS	TVP D	TVP C	TVP B	TVP A	TVE D	TVE C	TVE B	TVE A	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
TVEA, TVEB, TVEC, TVED	0, 1, 2, 3	rw	<p>Transmitter Valid Enable</p> <p>These bits enable the module kernel output signals TVALIDx (x = A, B, C, D) to be driven by MLI transmitter output signal TVALID.</p> <p>0_B TVALIDx is disabled and remains at passive level (as selected by TVPx).</p> <p>1_B Transmitter output signal TVALIDx is enabled and driven by TVALID.</p>
TVPA, TVPB, TVPC, TVPD	4, 5, 6, 7	rw	<p>Transmitter Valid Polarity</p> <p>These bits determine the polarity of the module kernel transmitter output signals TVALIDx (x = A, B, C, D).</p> <p>0_B Non-inverted polarity for TVALIDx selected: TVALIDx is passive when driving a 0. TVALIDx is active when driving a 1.</p> <p>1_B Inverted polarity for TVALIDx selected: TVALIDx is passive when driving a 1. TVALIDx is active when driving a 0.</p>

Field	Bits	Type	Description
TRS	[9:8]	rw	<p>Transmitter Ready Selection This bit field determines the module kernel input signal TREADY_x (x = A, B, C, D) that is used as MLI transmitter input signal TREADY.</p> <p>00_B TREADY_A is selected. 01_B TREADY_B is selected. 10_B TREADY_C is selected. 11_B TREADY_D is selected.</p>
TRP	10	rw	<p>Transmitter Ready Polarity This bit determines the polarity of TREADY_x.</p> <p>0_B Non-inverted polarity for TREADY_x selected: TREADY_x is passive if 0. TREADY_x is active if 1. 1_B Inverted polarity for TREADY_x selected: TREADY_x is passive if 1. TREADY_x if 0.</p>
TRE	11	rw	<p>Transmitter Ready Enable This bit enables the MLI transmitter input signal TREADY.</p> <p>0_B TREADY signal is disabled (always at 0 level). 1_B TREADY signal is enabled and driven by TREADY_x according to the settings of TRS and TRP.</p>
TCE	12	rw	<p>Transmitter Clock Enable This bit enables the module kernel output signal TCLK.</p> <p>0_B TCLK is disabled and remains at passive level (as selected by TCP). 1_B TCLK is enabled and driven according to the setting of TCP.</p>
TCP	13	rw	<p>Transmitter Clock Polarity This bit determines the polarity of the module kernel output clock signal TCLK.</p> <p>0_B Non-inverted polarity for TCLK selected: TCLK is driving a 0 when it is passive. 1_B Inverted polarity for TCLK selected: TCLK is driving a 1 when it is passive.</p>

Micro Link Interface (MLI)

Field	Bits	Type	Description
TDP	14	rw	<p>Transmitter Data Polarity This bit determines the polarity of the module kernel output clock signal TDATA.</p> <p>0_B TDATA is directly driven by MLI transmitter output signal TDATA (non-inverted).</p> <p>1_B TDATA is directly driven by the inverted MLI transmitter output signal TDATA.</p>
RVE	15	rw	<p>Receiver Valid Enable This bit enables the MLI receiver input signal RVALID.</p> <p>0_B RVALID signal is disabled (always at 0 level).</p> <p>1_B RVALID signal is enabled and driven by RVALIDx according to the settings of RVS and RVP (default after reset).</p>
RRS	[17:16]	rw	<p>Receiver Ready Selector This bit field determines the module kernel output signal RREADYx (x = A, B, C, D) that is driven by the MLI receiver output signal RREADY. The RREADYx output signals that are not selected drives a passive level according to the setting of RRPx.</p> <p>00_B RREADYA is selected.</p> <p>01_B RREADYB is selected.</p> <p>10_B RREADYC is selected.</p> <p>11_B RREADYD is selected.</p>
RRPA, RRPB, RRPC, RRPD	18, 19, 20, 21	rw	<p>Receiver Ready Polarity These bits determine the polarity of the module kernel receiver output signals RREADYx (x = A, B, C, D).</p> <p>0_B Non-inverted polarity for RREADYx selected: RREADYx is passive if 0. RREADYx is active if 1.</p> <p>1_B Inverted polarity for RREADYx selected: RREADYx is passive if 1. RREADYx is active if 0.</p>
RVS	[23:22]	rw	<p>Receiver Valid Selector This bit field determines the module kernel input signal RVALIDx (x = A, B, C, D) that is used as MLI receiver input signal RVALID.</p> <p>00_B RVALIDA is selected.</p> <p>01_B RVALIDB is selected.</p> <p>10_B RVALIDC is selected.</p> <p>11_B RVALIDD is selected.</p>

Micro Link Interface (MLI)

Field	Bits	Type	Description
RVP	24	rw	<p>Receiver Valid Polarity</p> <p>This bit determines the polarity of RVALIDx.</p> <p>0_B Non-inverted polarity for RVALIDx selected: RVALIDx is passive if 0. RVALIDx is active if 1.</p> <p>1_B Inverted polarity for RVALIDx selected: RVALIDx is passive if 1. RVALIDx is active if 0.</p>
RCS	[26:25]	rw	<p>Receiver Clock Selector</p> <p>This bit field determines the module kernel input signal RCLKx (x = A, B, C, D) that is used as MLI receiver input clock CLK.</p> <p>00_B RCLKA is selected. 01_B RCLKB is selected. 10_B RCLKC is selected. 11_B RCLKD is selected.</p>
RCP	27	rw	<p>Receiver Clock Polarity</p> <p>This bit determines the polarity of RCLKx.</p> <p>0_B Non-inverted polarity for RCLKx selected: RCLKx is at 0 level in passive state.</p> <p>1_B Inverted polarity for TCLK selected: RCLKx is at 1 level in passive state.</p>
RCE	28	rw	<p>Receiver Clock Enable</p> <p>This bit enables the MLI receiver input clock RCLK.</p> <p>0_B RCLK signal is disabled (always at 0 level). 1_B RCLK signal is enabled and driven by RCLKx according to the settings of RCS and RCP.</p>
RDS	[30:29]	rw	<p>Receiver Data Selector</p> <p>This bit field determines the module kernel input signal RDATAx (x = A, B, C, D) that is used as MLI receiver data input line RDATA.</p> <p>00_B RDATAA is selected. 01_B RDATAB is selected. 10_B RDATAAC is selected. 11_B RDATAAD is selected.</p>
RDP	31	rw	<p>Receiver Data Polarity</p> <p>This bit determines the polarity of RDATAx.</p> <p>0_B Non-inverted polarity for RDATAx selected: RDATAx is passive if 0. RDATAx is active if 1.</p> <p>1_B Inverted polarity for RDATAx selected: RDATAx is passive if 1. RDATAx is active if 0.</p>

23.4.3 Access Protection Registers

The Access Enable Register AER enables write and read operations in the corresponding address ranges ($x = 0$ to 31) in addition to the global move engine enable RCR.MOD. Each address range can be individually enabled or excluded from automatic mode.

Note: See [Table 23-15](#) on [Page 23-141](#) for the TC1796-specific address range definitions.

AER

Access Enable Register

(B8_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN	AEN
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Field	Bits	Type	Description
AENx ($x = 0-31$)	x	rW	<p>Address Range x Enable</p> <p>This bit enables the read and write capability of the MLI move engine for address range x ($x = 0-31$).</p> <p>0_B Automatic MLI read and write moves to address range x are disabled. Read/write moves to address range x are not executed automatically and an MLI service request can be generated. The receiving controller's software has to take care about the move.</p> <p>1_B Automatic MLI read and write moves to address range x are enabled if RCR.MOD = 1.</p>

Micro Link Interface (MLI)

The Access Range Register ARR determines size and number of the address sub-range n (n = 0-3).

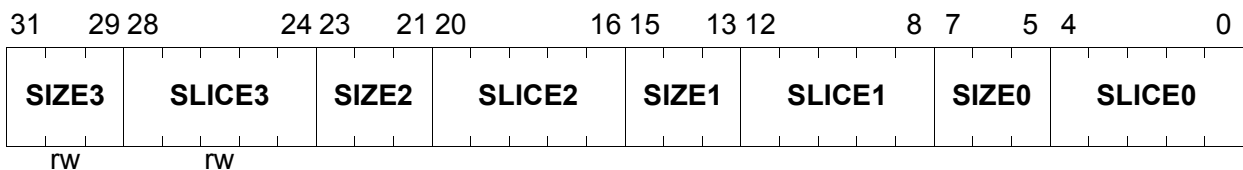
Note: See Page 23-46 for bit field definitions and [Page 23-141](#) for the TC1796-specific address range and address range extension definitions.

ARR

Access Range Register

(BC_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SLICE0	[4:0]	rw	Address Slice 0 SLICE0 selects a specific sub-range within address sub-range 0.
SIZE0	[7:5]	rw	Address Size 0 SIZE0 determines the sub-range size within address sub-range 0.
SLICE1	[12:8]	rw	Address Slice 1 SLICE1 selects a specific sub-range within address sub-range 1.
SIZE1	[15:13]	rw	Address Size 1 SIZE1 determines the sub-range size within address sub-range 1.
SLICE2	[20:16]	rw	Address Slice 2 SLICE2 selects a specific sub-range within address sub-range 2.
SIZE2	[23:21]	rw	Address Size 2 SIZE2 determines the sub-range size within address sub-range 2.
SLICE3	[28:24]	rw	Address Slice 3 SLICE3 selects a specific sub-range within address sub-range 3.
SIZE3	[31:29]	rw	Address Size 3 SIZE3 determines the sub-range size within address sub-range 3.

Field	Bits	Type	Description
MPE	[7:4]	rwh	<p>Maximum Parity Errors</p> <p>This bit field determines the maximum number of transmitter parity error conditions that can be still detected until a transmitter parity error event is generated (see Page 23-42). With each condition detected, MPE is decremented down to 0.</p> <p>0000_B A parity error event is generated if a transmitter parity error condition is detected.</p> <p>0001_B A parity error event is generated if a transmitter parity error condition is detected.</p> <p>0010_B A parity error event is generated if 2 transmitter parity error conditions are detected.</p> <p>0011_B A parity error event is generated if 3 transmitter parity error conditions are detected.</p> <p>..._B ...</p> <p>1110_B A parity error event is generated if 14 transmitter parity error conditions are detected.</p> <p>1111_B A parity error event is generated if 15 transmitter parity error conditions are detected.</p>
MNAE	[9:8]	rwh	<p>Maximum Non Acknowledge Errors</p> <p>This bit field determines the maximum number of consecutive Non-Acknowledge error conditions that can be still detected in the transmitter until a time-out event is generated. MNAE is decremented down to 0 at each Non-Acknowledge error condition. When MNAE = 0 or becoming 0, a time-out event is generated. MNAE is automatically set to 11_B after a successful frame transmission (see Page 23-45).</p> <p>00_B A time-out event is generated if 1 non-ack condition is detected.</p> <p>01_B A time-out event is generated if 1 non-ack condition is detected.</p> <p>10_B A time-out event is generated if 2 consecutive non-ack conditions are detected.</p> <p>11_B A time-out event is generated if 3 consecutive non-ack conditions are detected.</p>

Field	Bits	Type	Description
MDP	[13:10]	rw	<p>Maximum Delay for Parity Error</p> <p>This bit field determines a window for the transmitter in number of TCLK clock periods where a TREADY low-to-high signal transition signal is considered as “correctly received” condition (see Page 23-22).</p> <p>0000_B Zero clock periods selected (not useful)</p> <p>0001_B 1 clock period selected</p> <p>..._B ...</p> <p>1110_B 14 clock periods selected</p> <p>1111_B 15 clock periods selected</p>
NO	14	rw	<p>No Optimized Method</p> <p>This bit field enables/disables the address prediction for Read or Write Frames (see Page 23-45).</p> <p>0_B Optimized method (address prediction) enabled.</p> <p>1_B Optimized method (address prediction) disabled.</p>
TP	15	rw	<p>Type of Parity</p> <p>This bit will determines the type of parity used in frame transmissions. For correct data transfers, TP = 0 has to be programmed. The value TP = 1 can be selected to force parity errors to analyze the propagation delay (see Page 23-24).</p> <p>0_B Even parity selected; parity bit P is set when the modulo-2 sum of frame header bits and data field bits is 1.</p> <p>1_B Odd parity selected; parity bit P is set when the modulo-2 sum of frame header bits and data field bits is 0.</p>
0	3, [31:16]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Micro Link Interface (MLI)

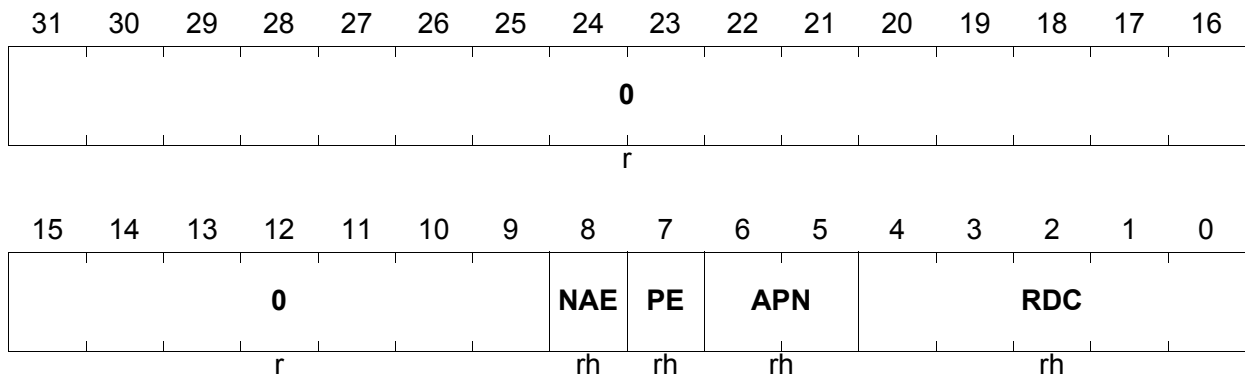
The Transmitter Status Register TSTATR contains transmitter specific status information.

TSTATR

Transmitter Status Register

(14_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
RDC	[4:0]	rh	Ready Delay Counter This bit field counts TCLK periods after the end of a frame transmission. When the TVALID signal goes to low level, RDC is cleared to zero and starts counting up the TCLK clock periods until a TREADY high level is detected (see Page 23-22).
APN	[6:5]	rh	Answer Pipe Number This bit field is written by the MLI receiver with the Pipe Number of a received Read Frame. APN is used by an Answer Frame that is transmitted as response to the Read Frame. 00 _B Pipe 0 is used in Answer Frame. 01 _B Pipe 1 is used in Answer Frame. 10 _B Pipe 2 is used in Answer Frame. 11 _B Pipe 3 is used in Answer Frame.
PE	7	rh	Parity Error Flag This bit is set if a transmitter parity error condition is detected by the transmitter after a frame transmission. PE is cleared by hardware when a frame has been transmitted without a parity error (see Page 23-42). Bit PE can be cleared by software via bit SCR.CTPE.

Micro Link Interface (MLI)

Field	Bits	Type	Description
NAE	8	rh	Non Acknowledge Error Flag This bit is set when a Non-Acknowledge error condition is detected by the MLI transmitter after a frame transmission (see Page 23-45). NAE is cleared by hardware if a transmitted frame has been acknowledged correctly. Bit NAE can be cleared by software via bit SCR.CNAE.
0	[31:9]	r	Reserved Read as 0 if read.

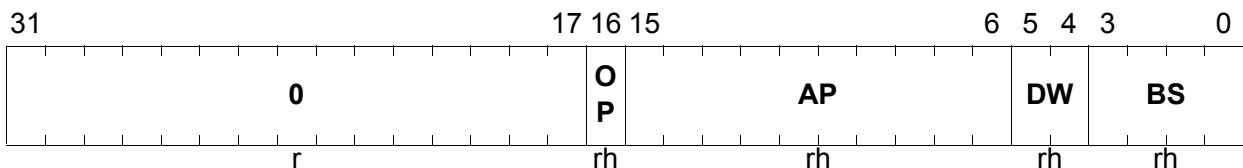
Micro Link Interface (MLI)

The Transmitter Pipe x Status Registers TPxSTATR contain pipe-specific status information related to address optimization and prediction, data width for transmit data, and Remote Window size.

**TPxSTATR (x = 0-3)
Transmitter Pipe x Status Register**

(18_H+4_H*x)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
BS	[3:0]	rh	<p>Buffer Size</p> <p>This bit field indicates the coded buffer size of the pipe x Remote Window in the receiving controller. BS further determines how many address offset bits are transmitted in a Write Offset and Data Frame or in a Discrete Read Frame. When register TPxBAR is written for generation of a Copy Base Address Frame, BS is updated by the Copy Base Address Frame (see Page 23-26).</p> <p>0000_B 1-bit offset address of Remote Window 0001_B 2-bit offset address of Remote Window 0010_B 3-bit offset address of Remote Window ..._B ... 1110_B 15-bit offset address of Remote Window 1111_B 16-bit offset address of Remote Window</p>
DW	[5:4]	rh	<p>Data Width</p> <p>This bit field indicates the data width that has been detected for a read or write access of a bus master to a Transfer Window of pipe x (see Page 23-28 and Page 23-32).</p> <p>00_B 8-bit data width detected 01_B 16-bit data width detected 10_B 32-bit data width detected 11_B Reserved</p>

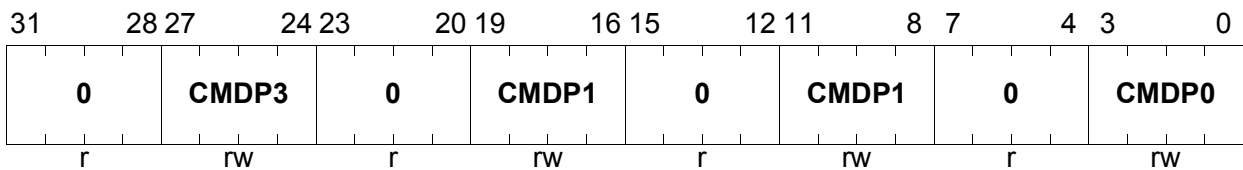
Field	Bits	Type	Description
AP	[15:6]	rh	<p>Address Prediction Factor</p> <p>This bit field indicates the delta value (positive or negative number) of offset address used by the MLI transmitter for the next address prediction. AP is a signed 9-bit number (10th bit is the sign bit) that is written with each transmitter address prediction calculation (see Page 23-24 and Page 23-45).</p>
OP	16	rh	<p>Use Optimized Frame</p> <p>When address optimization is enabled with TCR.NO = 0, this bit indicates if address prediction is possible in the transmitter. OP is written with each transmitter address prediction calculation (see Page 23-24 and Page 23-45).</p> <p>0_B No address prediction is possible. A Write Offset and Data Frame or a Discrete Read Frame are used for transmission.</p> <p>1_B Address prediction is possible. An Optimized Write Frame or an Optimized Read Frame are used for transmission.</p>
0	[31:17]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Micro Link Interface (MLI)

The Transmitter Command Register TCMR contains the command codes that are used during Command Frame transmission. Each time one of the CMDPx bit fields is written, a Command Frame transmission is triggered. Independent of the transferred command code value, a Command Frame transmitted event can be generated in the transmitter for each pipe and a Command Frame received event for each pipe in the receiver, respectively.

TCMDR

Transmitter Command Register (28_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
CMDP0	[3:0]	rw	<p>Command Code for Pipe 0</p> <p>This bit field contains the command code related to pipe 0. The pipe 0 command codes allow an activation (pulse) of one of the service request outputs SR[3:0] in the receiving controller.</p> <p>0001_B Activate service request output SR0 0010_B Activate service request output SR1 0011_B Activate service request output SR2 0100_B Activate service request output SR3 Other bit combinations are reserved for future use; no further action occurs in the receiver.</p>
CMDP1	[11:8]	rw	<p>Command Code for Pipe 1</p> <p>This bit field contains the command code related to pipe 1. The pipe 1 command codes allow to adjust the receiver delay for the parity error condition (see RCR.DPE) in the MLI receiver of the receiving controller.</p> <p>0000_B Set RCR.DPE = 0000_B 0001_B Set RCR.DPE = 0001_B ..._B ... 1110_B Set RCR.DPE = 1110_B 1111_B Set RCR.DPE = 1111_B</p>

Micro Link Interface (MLI)

Field	Bits	Type	Description
CMDP2	[19:16]	rw	<p>Command Code for Pipe 2</p> <p>This bit field contains the command code related to pipe 2. The pipe 2 command codes allow to control the MLI receiver in the receiving controller.</p> <p>0001_B Enable Automatic Data Mode (RCR.MOD = 1) 0010_B Disable Automatic Data Mode (RCR.MOD = 0) 0100_B Clear bit TRSTATR.RP0 0101_B Clear bit TRSTATR.RP1 0110_B Clear bit TRSTATR.RP2 0111_B Clear bit TRSTATR.RP3 1111_B Activate a pulse at break output $\overline{\text{BRKOUT}}$</p> <p>Other bit combinations are reserved for future use; no further action occurs in the receiver.</p>
CMDP3	[27:24]	rw	<p>Command Code for Pipe 3</p> <p>This bit field contains the command code related to pipe 3. The command codes for pipe 3 are free programmable by software.</p>
0	[7:4], [15:12], [23:20], [31:28]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Micro Link Interface (MLI)

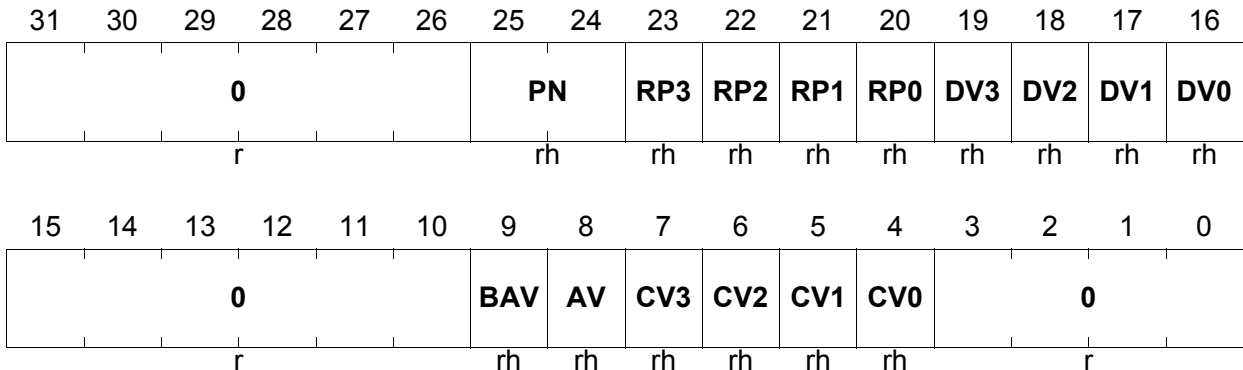
The Transmitter Receiver Status Register TRSTATR contains read-only flags that indicate the status of MLI operations.

TRSTATR

Transmitter Receiver Status Register

(2C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CVx (x = 0-3)	4 + x	rh	Command Valid Bit is set by hardware when a TCMDR.CMDPx bit field is written. It is cleared by hardware when the Command Frame has been correctly transmitted. CVx can be set or cleared by software via bits SCR.SCVx or SCR.CCVx.
AV	8	rh	Answer Valid Bit is set by hardware when the TDRAR register in the the MLI transmitter (in the Remote Controller) is written. AV is cleared by hardware when the Answer Frame has been correctly sent. AV can be cleared by software via bit SCR.CAV.
BAV	9	rh	Base Address Valid Bit is set by hardware when the TCBAR register in the MLI transmitter is written. BAV is cleared by hardware when the Copy Base Address Frame has been correctly sent. BAV can be cleared by software via bit SCR.CBAV.

Micro Link Interface (MLI)

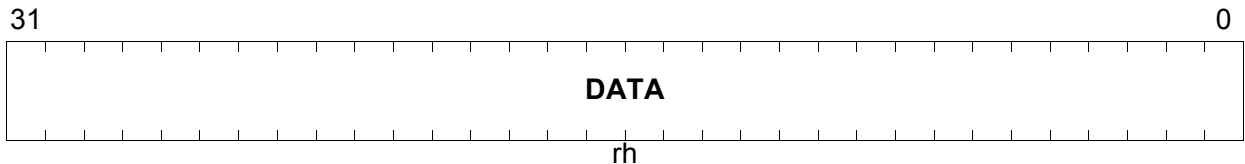
Field	Bits	Type	Description
DV_x (x = 0-3)	16 + x	rh	Data Valid Bit is set by hardware when the TPxDATAR and/or the TPxAOFR registers of the MLI transmitter are updated after a read or write access to a Transfer Window of pipe x. DV _x is cleared again by hardware when the Read or Write Frame has been correctly sent. DV _x can be cleared by software via bit SCR.CDV _x .
RP_x (x = 0-3)	20 + x	rh	Read Pending Bit is set by hardware when the TPxAOFR register of the MLI transmitter is updated after a read access to a Transfer Window of pipe x. RP _x is cleared by hardware when the MLI receiver in the Local Controller receives an Answer Frame for pipe x from the Remote Controller. RP _x can be cleared by software via bit SCR.CDV _x .
PN	[25:24]	rh	Pipe Number This bit field indicates the Pipe Number x of the base address that has been written into register TPxBAR. 00 _B TP0BAR has been last written. 01 _B TP1BAR has been last written. 10 _B TP2BAR has been last written. 11 _B TP3BAR has been last written.
0	[3:0], [15:10], [31:26]	r	Reserved Read as 0.

23.4.5 Transmitter Data/Address Registers

The Transmitter Pipe x Data Register TPxDATAR is a read-only register that stores the data that has been written during the last write access to a Transfer Window of pipe x.

TPxDATAR (x = 0-3)

Transmitter Pipe x Data Register $(40_H + 4_H * x)$ **Reset Value: 0000 0000_H**

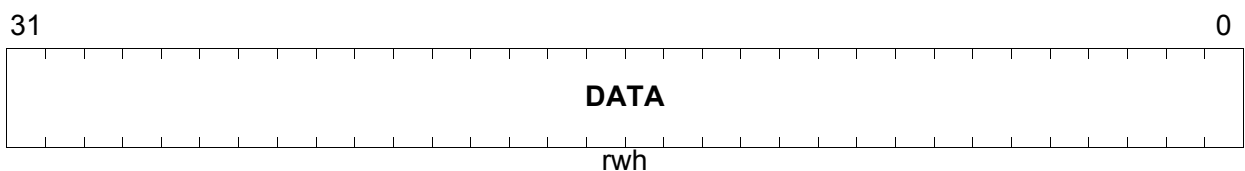


Field	Bits	Type	Description
DATA	[31:0]	rh	Data Whenever a location within a Transfer Window is written, the data is loaded in this bit field.

The Transmitter Data Read Answer Register TDRAR contains the read data for the transmission of an Answer Frame.

TDRAR

Transmitter Data Read Answer Register **(50_H)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
DATA	[31:0]	rwh	Data This bit field is loaded with data that is read from the address requested by a Read Frame. An update of this bit field triggers the start of an Answer Frame with DATA used as content of the Answer Frame. This bit field can be updated either automatically by the move engine (if Automatic Data Mode is enabled) or by the CPU (if Automatic Data Mode is disabled).

Micro Link Interface (MLI)

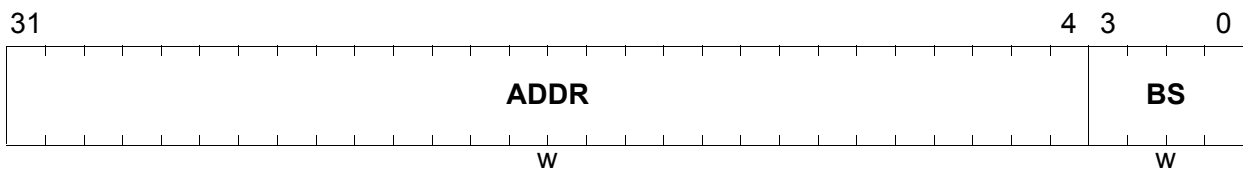
The write-only Transmitter Pipe x Base Address Register TPxBAR represents the 28-bit pipe x Remote Window base address and the Remote Window size that is transmitted to the receiving controller via a Copy Base Address Frame.

TPxBAR (x = 0-3)

Transmitter Pipe x Base Address Register

(54_H+4_H*x)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
BS	[3:0]	w	<p>Buffer Size</p> <p>This bit field determines the coded buffer size of the pipe x Remote Window in the receiving controller. When writing TPxBAR, BS is copied into bit field TPxSTATR.BS.</p> <p>0000_B 1-bit offset address of Remote Window 0001_B 2-bit offset address of Remote Window 0010_B 3-bit offset address of Remote Window ..._B ... 1110_B 15-bit offset address of Remote Window 1111_B 16-bit offset address of Remote Window</p>
ADDR	[31:4]	w	<p>Address</p> <p>This bit field determines the most significant 28 bits of the pipe x Remote Window base address. When writing TPxBAR, ADDR is copied into bit field TCBAR.ADDR[31:4].</p>

Micro Link Interface (MLI)

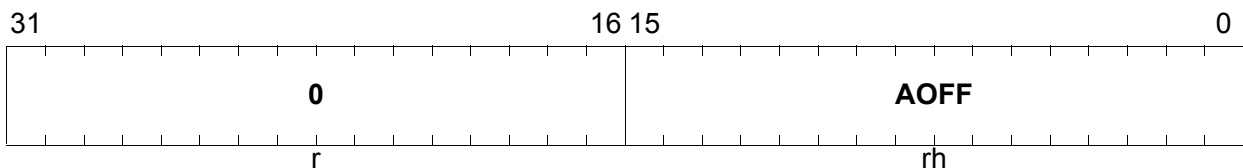
The Transmitter Pipe x Address Offset Register TPxAOFR is a read-only register that stores the offset address that has been used by the last read or write access to a Transfer Window of pipe x.

TPxAOFR (x = 0-3)

Transmitter Pipe x Address Offset Register

(30_H+4_H*x)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
AOFF	[15:0]	rh	Address Offset Whenever a location within a Transfer Window is accessed (read or written) AOFF is loaded with the lowest 16 address bits of the access. Also in the case of a small Transfer Window access, all AOFF bits are loaded, but AOFF[15:13] are not taken into account for further actions.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

Micro Link Interface (MLI)

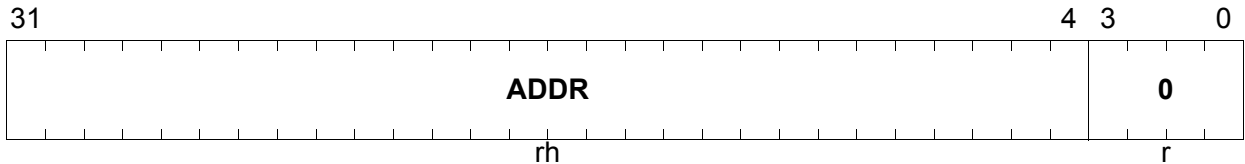
The Transmitter Copy Base Address Register TCBAR contains the 28-bit pipe x Remote Window base address of the latest write access to TPxBAR.ADDR.

TCBAR

Transmitter Copy Base Address Register

(64_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
ADDR	[31:4]	rh	Address This bit field contains the 28 address bits written to TPxBAR.ADDR. This value will be transferred to the receiving controller to define the base address of the Remote Window for pipe x.
0	[3:0]	r	Reserved Read as 0; should be written with 0.

23.4.6 Transmitter Interrupt Registers

The Transmitter Interrupt Enable Register TIER contains the interrupt enable bits and the clear bits for all transmitter events. The bits marked w always read as 0.

TIER

Transmitter Interrupt Enable Register (98_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						TE IR	PE IR	CFS IR3	CFS IR2	CFS IR1	CFS IR0	NFS IR3	NFS IR2	NFS IR1	NFS IR0
r						w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						TE IE	PE IE	CFS IE3	CFS IE2	CFS IE1	CFS IE0	NFS IE3	NFS IE2	NFS IE1	NFS IE0
r						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
NFSIEx (x = 0-3)	x	rw	Normal Frame Sent in Pipe x Interrupt Enable 0 _B Normal frame sent in pipe x event is disabled for activation of an SRx line. 1 _B Normal frame sent in pipe x event is enabled for activation of an SRx line.
CFSIEx (x = 0-3)	4 + x	rw	Command Frame Sent in Pipe x Interrupt Enable 0 _B Command frame sent in pipe x event is disabled for activation of an SRx line. 1 _B Command frame sent in pipe x event is enabled for activation of an SRx line.
PEIE	8	rw	Parity Error Interrupt Enable 0 _B Parity error event is disabled for activation of an SRx line. 1 _B Parity error event is enabled for activation of an SRx line.
TEIE	9	rw	Time-Out Error Interrupt Enable 0 _B Time-out error event is disabled for activation of an SRx line. 1 _B Time-out error event is enabled for activation of an SRx line.

Micro Link Interface (MLI)

Field	Bits	Type	Description
NFSIRx (x = 0-3)	16 + x	w	Normal Frame Sent in Pipe x Flag Clear 0 _B No action. 1 _B Clear TISR.NFSIx.
CFSIRx (x = 0-3)	20 + x	w	Command Frame Sent in Pipe x Flag Clear 0 _B No action. 1 _B Clear TISR.CFSIx.
PEIR	24	w	Parity Error Flag Clear 0 _B No action. 1 _B Clear TISR.PEIx.
TEIR	25	w	Time Out Error Flag Clear 0 _B No action. 1 _B Clear TISR.TEIx.
0	[15:10], [31:26]	r	Reserved Read as 0; should be written with 0.

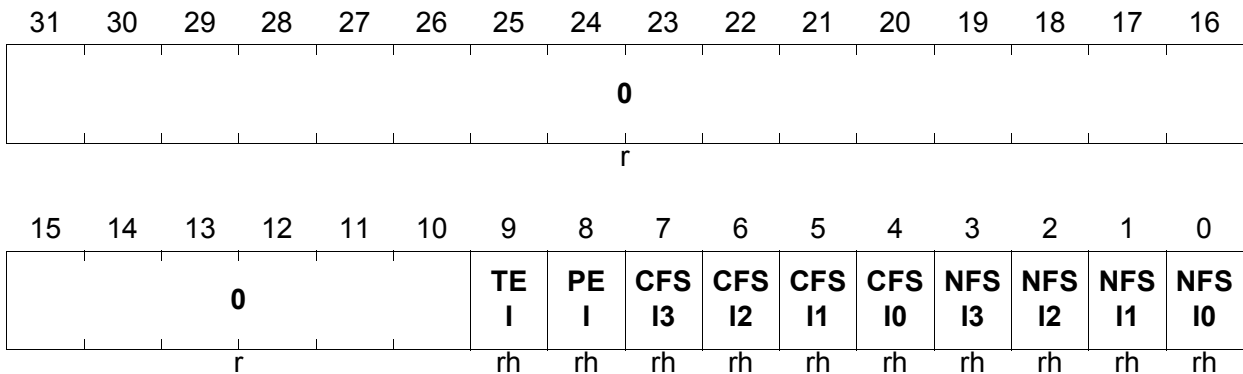
Micro Link Interface (MLI)

The Transmitter Interrupt Status Register TISR contains all MLI event (or interrupt) flags of the MLI transmitter. These flags can be cleared by software when writing the appropriate bits in the TIER register; they are not cleared by hardware.

TISR

Transmitter Interrupt Status Register (9C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
NFS_x (x = 0-3)	x	rh	<p>Normal Frame Sent in Pipe x Flag</p> <p>0_B A Normal Frame has not yet been sent. 1_B A Write or Read Frame has been correctly sent and acknowledged for pipe x.</p> <p>The service request output that can be activated by NFS_x is defined by TINPR.NFSIP_x.</p>
CFS_x (x = 0-3)	4 + x	rh	<p>Command Frame Sent in Pipe x Flag</p> <p>0_B A Command Frame has not yet been sent. 1_B A Command Frame has been correctly sent and acknowledged for pipe x.</p> <p>The service request output that can be activated by CFS_x is defined by TINPR.CFSIP.</p>
PEI	8	rh	<p>Parity Error Flag</p> <p>0_B A parity error event has not yet been detected. 1_B A parity error event has been detected.</p> <p>The service request output that can be activated by PEI is defined by TINPR.PTEIP.</p>
TEI	9	rh	<p>Time-Out Error Flag</p> <p>0_B A time-out error event has not yet been detected. 1_B A time-out error event has been detected.</p> <p>The service request output that can be activated by TEI is defined by TINPR.PTEIP.</p>

Micro Link Interface (MLI)

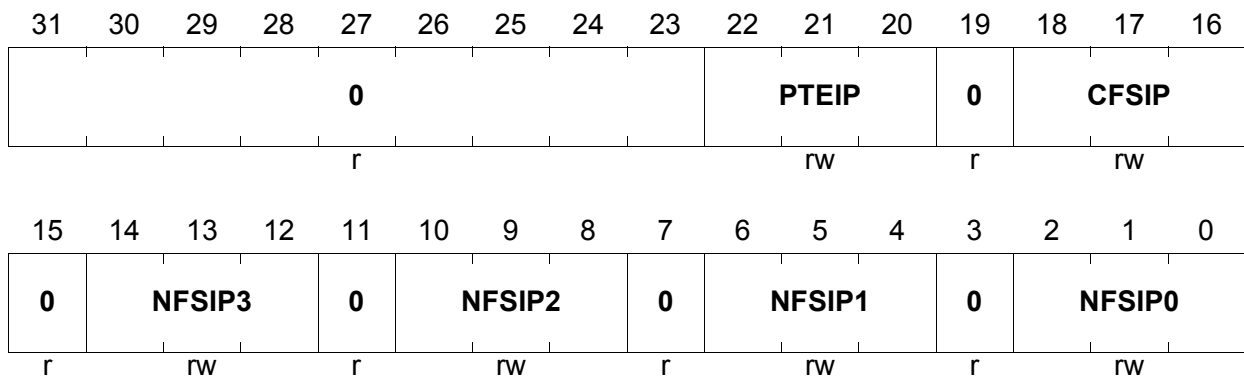
Field	Bits	Type	Description
0	[31:10]	r	Reserved Read as 0.

The Transmitter Interrupt Node Pointer Register TINPR contains the node pointers for the MLI transmitter events.

TINPR

Transmitter Interrupt Node Pointer Register (A0_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
NFSIP0	[2:0]	rw	Normal Frame Sent in Pipe 0 Interrupt Pointer This bit field determines which service request output SR _x becomes active when a Normal Frame sent in pipe 0 event occurs (if enabled). 000 _B The service request output SR0 is selected. 001 _B The service request output SR1 is selected. ... _B ... 110 _B The service request output SR6 is selected. 111 _B The service request output SR7 is selected.
NFSIP1	[6:4]	rw	Normal Frame Sent in Pipe 1 Interrupt Pointer This bit field determines which service request output SR _x becomes active when a Normal Frame sent in pipe 1 event occurs (if enabled). Coding see NFSIP0.
NFSIP2	[10:8]	rw	Normal Frame Sent in Pipe 2 Interrupt Pointer This bit field determines which service request output SR _x becomes active when a Normal Frame sent in pipe 2 event occurs (if enabled). Coding see NFSIP0.

Micro Link Interface (MLI)

Field	Bits	Type	Description
NFSIP3	[14:12]	rw	Normal Frame Sent in Pipe 3 Interrupt Pointer This bit field determines which service request output SRx becomes active when a Normal Frame sent in pipe 3 event occurs (if enabled). Coding see NFSIP0.
CFSIP	[18:16]	rw	Command Frame Sent Interrupt Pointer This bit field determines which service request output SRx becomes active when a Command Frame sent event occurs (if enabled). Coding see NFSIP0.
PTEIP	[22:20]	rw	Parity or Time Out Interrupt Pointer This bit field determines which service request output SRx becomes active when a parity/time-out event occurs (if enabled). Coding see NFSIP0.
0	3, 7, 11, 15, 19, [31:23]	r	Reserved Read as 0; should be written with 0.

23.4.7 Receiver Status/Control Registers

The Receiver Control Register RCR contains control and status bits/bit fields that are related to the MLI receiver operation.

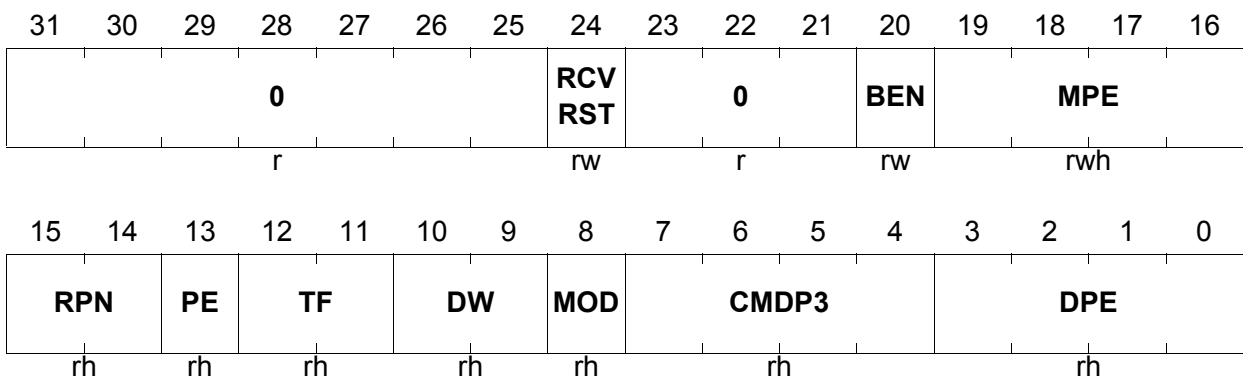
Bit RCVRST is automatically overwritten after a reset (see [Page 23-66](#)) with a value given in the implementation chapter (see [Page 23-127](#)).

RCR

Receiver Control Register

(68_H)

Reset Value: 0100 0000_H



Field	Bits	Type	Description
DPE	[3:0]	rh	<p>Delay for Parity Error</p> <p>DPE determines the number of RCLK clock periods that the MLI receiver waits before the RREADY signal is raised again when it has detected a parity error (see Page 23-22). When a pipe 1 Command Frame is received by the MLI receiver, the command code is stored in this bit field (see Page 23-39).</p> <p>0000_B Zero RCLK clock period delay is selected. 0001_B One RCLK clock period delay is selected. 0010_B Two RCLK clock periods delay is selected. ..._B ... 1110_B Fourteen RCLK clock periods delay is selected. 1111_B Fifteen RCLK clock periods delay is selected.</p>
CMDP3	[7:4]	rh	<p>Command From Pipe 3</p> <p>When a pipe 3 Command Frame is received by the MLI receiver, the command code is stored in this bit field. Pipe 3 commands are free for software use.</p>

Field	Bits	Type	Description
MOD	8	rh	<p>Mode of Operation</p> <p>This bit determines the data transfer operation mode of the MLI receiver. Bit MOD can be set by hardware with the reception of a pipe 2 Command Frame (see Page 23-98). It can be set or cleared by software via bits SCR.SMOD or SCR.CMOD.</p> <p>0_B Automatic Data Mode is disabled. Data read/write operations from/to a Remote Window must be executed by a bus master (e.g. the CPU).</p> <p>1_B Automatic Data Mode is enabled. Data read/write operations from/to a Remote Window are executed by the MLI's move engine.</p>
DW	[10:9]	rh	<p>Data Width</p> <p>This bit field is updated by the MLI receiver whenever new data is received in the RDATAR register. DW indicates the relevant data width.</p> <p>00_B 8-bit relevant data width in RDATAR 01_B 16-bit relevant data width in RDATAR 10_B 32-bit relevant data width in RDATAR 11_B Reserved</p>
TF	[12:11]	rh	<p>Type of Frame</p> <p>This bit field determines the frame type that has most recently been received by the MLI receiver. It is updated whenever the MLI receiver updates RDATAR, RADDR, or RPxBAR.</p> <p>The most recently received frame was a:</p> <p>00_B Copy Base Address Frame 01_B Discrete Read Frame or Optimized Read Frame 10_B Write Offset and Data Frame or Optimized Write Frame 11_B Answer Frame</p> <p>Note that the coding of TF is different from the frame coding as defined in Table 23-1 on Page 23-11.</p>
PE	13	rh	<p>Parity Error</p> <p>PE is set when a parity error is detected in a received frame (see Page 23-42). PE is cleared by hardware when a frame has been received without parity error. PE can be cleared by software via bit SCR.CRPE.</p>

Micro Link Interface (MLI)

Field	Bits	Type	Description
RPN	[15:14]	rh	<p>Received Pipe Number</p> <p>This bit field contains the Pipe Number that was indicated by the Pipe Number bit field of the latest received frame. It is updated by any received frame.</p>
MPE	[19:16]	rwh	<p>Maximum Parity Errors</p> <p>This bit field indicates the number of receive parity error conditions after which a receiver parity error event will be generated. It is set to a desired value by software and it is decremented down to 0 automatically by the MLI each time it detects a receiver parity error condition. If a receiver parity error condition is detected and MPE becomes 0 or is already 0, a receiver parity error event is generated (see Page 23-42).</p> <p>0000_B A receiver parity event is generated if 1 receiver error condition is detected.</p> <p>0001_B A receiver parity event is generated if 1 receiver error condition is detected.</p> <p>0010_B A receiver parity event is generated if 2 receiver error conditions are detected.</p> <p>..._B ...</p> <p>1110_B A receiver parity event is generated if 14 receiver error conditions are detected.</p> <p>1111_B A receiver parity event is generated if 15 receiver error conditions are detected.</p>
BEN	20	rw	<p>Break Out Enable</p> <p>When setting BEN = 1, the MLI receiver generates a pulse on its break output signal $\overline{\text{BRKOUT}}$ when a pipe 2 Command Frame with command code CMD = 1111_B is received.</p> <p>0_B Break output signal generation is disabled.</p> <p>1_B Break output signal is enabled.</p>
RCVRST	24	rw	<p>Receiver Reset</p> <p>This bit forces the receiver to be reset in order to be able to change OICR settings without affecting the receiver registers.</p> <p>0_B The MLI receiver is in operating mode.</p> <p>1_B The MLI receiver is held in reset state and OICR can be modified without unintentional actions in the receiver.</p>

23.4.8 Receiver Data/Address Registers

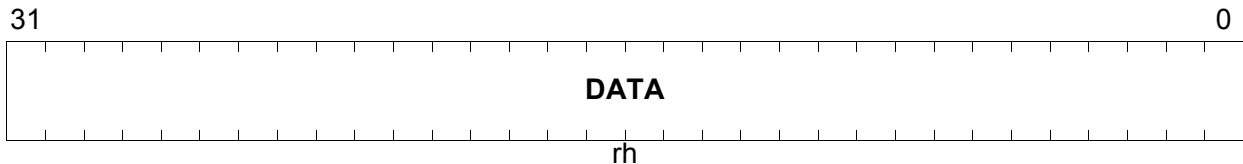
The Receiver Data Register RDATAR is a read-only register that stores data received by a Write Frame or an Answer Frame.

RDATAR

Receiver Data Register

(90_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
DATA	[31:0]	rh	<p>Data</p> <p>In the receiving controller, DATA contains the data received by a Write Frame or an Answer Frame. Bit field RCR.DW determines the width of the relevant data that is stored in RDATAR.</p> <p>RCR.DW = 00_B: RDATAR[7:0] are relevant (8-bit)</p> <p>RCR.DW = 01_B: RDATAR[15:0] are relevant (16-bit)</p> <p>RCR.DW = 10_B: RDATAR[31:0] are relevant (32-bit)</p>

Micro Link Interface (MLI)

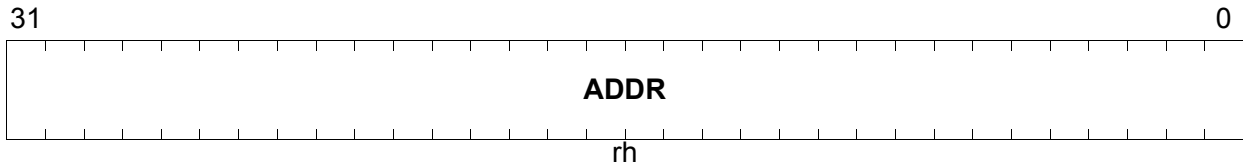
The Receiver Pipe x Base Address Register RPxBAR is a read-only register that contains the complete target address in the Remote Window of pipe x.

RPxBAR (x = 0-3)

Receiver Pipe x Base Address Register

(6C_H+4_H*x)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
ADDR	[31:0]	rh	<p>Address</p> <p>ADDR indicates the complete target address for the pipe x Remote Window.</p> <p>When a pipe x Copy Base Address Frame is received, ADDR[31:4] becomes loaded with the transmitted 28-bit address and bits [3:0] are cleared.</p> <p>When a write or Read Frame with m bits of address offset is received, bits ADDR[31:m] are held constant and bits ADDR[m-1:0] are replaced by the received offset.</p> <p>When an optimized read or data frame is received, the address prediction mechanism adds the predicted address offset RPxSTATR.AP to ADDR and stores the result in ADDR.</p> <p>When an Answer Frame is received, ADDR is not changed.</p>

Micro Link Interface (MLI)

The Receiver Address Register RADRR is a read-only register storing the complete address of the most recently (or currently) targeted Remote Window.

RADRR

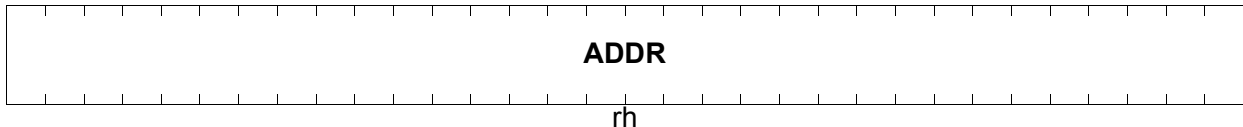
Receiver Address Register

(8C_H)

Reset Value: 0000 0000_H

31

0



Field	Bits	Type	Description
ADDR	[31:0]	rh	<p>Address</p> <p>ADDR indicates the complete target address for the most recently (or currently) targeted Remote Window (pipe x).</p> <p>When a Copy Base Address Frame is received, ADDR is unchanged.</p> <p>When a write or Read Frame with m bits of address offset is received, bits ADDR[31:m] replaced by the bits RPxBAR.ADDR[31:m] and bitsADDR[m-1:0] are replaced by the received offset.</p> <p>When an optimized read or data frame is received, the address prediction mechanism adds the predicted address offset RPxSTATR.AP to RPxBAR.ADDR and stores the result in ADDR.</p> <p>When an Answer Frame is received, ADDR becomes invalid.</p>

23.4.9 Receiver Interrupt Registers

The Receiver Interrupt Enable Register RIER contains the interrupt enable bits and the clear bits for all receiver events. The bits marked w are always read as 0.

RIER

Receiver Interrupt Enable Register (A4_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						DRA	MPE	PE	ICE	CFR	CFR	CFR	CFR	ME	NFR
						IR	IR	IR	R	IR3	IR2	IR1	IR0	IR	IR
r						w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						DRA	M	PEIE	ICE	CFR	CFR	CFR	CFR	NFR	
						IE	PEIE			IE3	IE2	IE1	IE0	IE	
r						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
NFRIE	[1:0]	rw	<p>Normal Frame Received Interrupt Enable This bit field defines if an SRx output is activated if a Normal Frame is correctly received.</p> <p>00_B The SRx activation is disabled.</p> <p>01_B The selected SRx line is activated each time a Normal Frame is correctly received.</p> <p>10_B The selected SRx line is activated each time a Normal Frame is correctly received that is not handled automatically by the MLI move engine (e.g. an Answer Frame).</p> <p>11_B Reserved</p>
CFRIEx (x = 0-3)	2 + x	rw	<p>Command Received in Pipe x Interrupt Enable This bit determines if an SRx output is activated if a Command Frame for pipe x has been received correctly.</p> <p>0_B Command received in pipe x event is disabled for activation of an SRx line.</p> <p>1_B Command received in pipe x event is enabled for activation of an SRx line.</p>

Micro Link Interface (MLI)

Field	Bits	Type	Description
ICE	6	rw	<p>Interrupt Command Enable This bit determines if an SRx output line is activated if a Command Frame is received in pipe 0.</p> <p>0_B Command frame received in pipe 0 event is disabled for activation of an SRx line.</p> <p>1_B Command frame received in pipe 0 event is enabled for activation of an SRx line.</p>
PEIE	7	rw	<p>Parity Error Interrupt Enable This bit determines if an SRx output line is activated if receiver a parity error event is detected.</p> <p>0_B Parity error event is disabled for activation of an SRx line.</p> <p>1_B Parity error event is enabled for activation of an SRx line.</p>
MPEIE	8	rw	<p>Memory Access Protection Interrupt Enable This bit determines if an SRx output line is activated if a memory access protection error is detected.</p> <p>0_B Memory access protection error event is disabled for activation of an SRx line.</p> <p>1_B Memory access protection error event is enabled for activation of an SRx line.</p>
DRAIE	9	rw	<p>Discarded Read Answer Interrupt Enable This bit determines if an SRx output line is activated if a discarded read Answer Frame condition is detected.</p> <p>0_B Discarded read answer event is disabled for activation of an SRx line.</p> <p>1_B Discarded read answer event is enabled for activation of an SRx line.</p>
NFRIR	16	w	<p>Normal Frame Received Interrupt Flag Clear</p> <p>0_B No action.</p> <p>1_B Clear RISR.NFRI.</p>
MEIR	17	w	<p>MLI Move Engine Interrupt Flag Clear</p> <p>0_B No action.</p> <p>1_B Clear RISR.MEI.</p>
CFRIRx (x = 0-3)	18 + x	w	<p>Command Frame Received in Pipe x Interrupt Flag Clear</p> <p>0_B No action.</p> <p>1_B Clear RISR.CFRIx.</p>

Micro Link Interface (MLI)

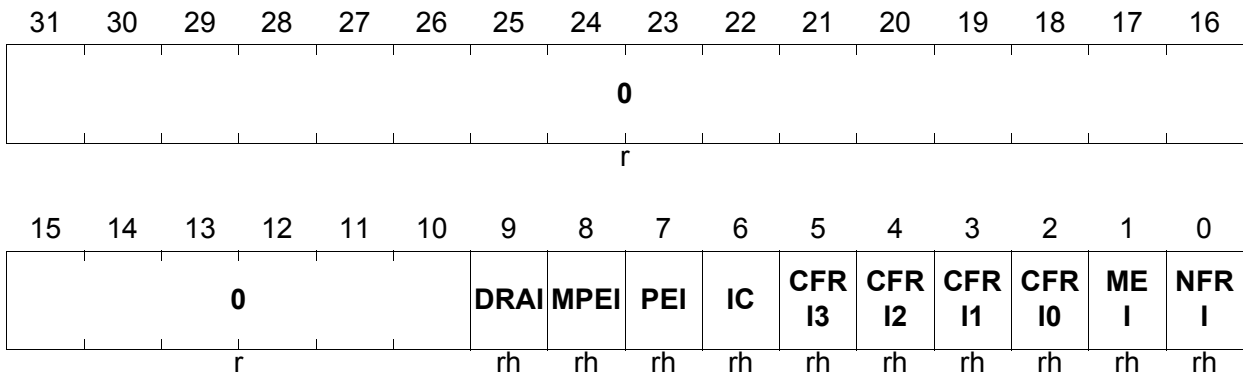
Field	Bits	Type	Description
ICER	22	w	Interrupt Command Flag Clear 0 _B No action. 1 _B Clear RISR.ICE.
PEIR	23	w	Parity Error Interrupt Flag Clear 0 _B No action. 1 _B Clear RISR.PEI.
MPEIR	24	w	Memory Protection Error Interrupt Flag Clear 0 _B No action. 1 _B Clear RISR.MPEI.
DRAIR	25	w	Discarded Read Answer Interrupt Flag Clear 0 _B No action. 1 _B Clear RISR.DRAI.
0	[15:10], [31:26]	r	Reserved Read as 0; should be written with 0.

Micro Link Interface (MLI)

The Receiver Interrupt Status Register RISR contains all event (interrupt) flags of the MLI receiver. These flags can be cleared by software when writing the appropriate bits in the RIER register; they are not cleared by hardware.

RISR

Receiver Interrupt Status Register (A8_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
NFRI	0	rh	Normal Frame Received Interrupt Flag This flag is set when a Write or a Read Frame has been received. The service request output that is activated is defined by RINPR.NFRIP.
MEI	1	rh	MLI Move Engine Interrupt Flag This flag is set when the move engine has finished an operation (read or write, depending on received frame). The service request output that is activated is defined by RINPR.MPPEIP.
CFR_x (x = 0-3)	2 + x	rh	Command Frame Received in Pipe x Interrupt Flag This flag is set when a Command Frame has been received in pipe x. The service request output that is activated is defined by RINPR.CFRIP.
IC	6	rh	Interrupt Command Flag This flag is set when a Command Frame has been received in pipe 0 leading to an activation of one of the service request outputs SR[3:0]. The service request output that is activated is defined by the received command CMD.

Micro Link Interface (MLI)

Field	Bits	Type	Description
PEI	7	rh	Parity Error Interrupt Flag This flag is set when a parity error event has occurred. The service request output that is activated is defined by RINPR.MPPEIP.
MPEI	8	rh	Memory Protection Error Interrupt Flag This flag is set when a memory protection event has occurred. The service request output that is activated is defined by RINPR.MPPEIP.
DRAI	9	rh	Discarded Read Answer Interrupt Flag This flag is set when the discarded read answer event has occurred. This condition occurs if an Answer Frame is received while none of the TRSTATR.RPx bits is set (the Answer Frame was not expected). The service request output that is activated is defined by RINPR.DRAIP.
0	[31:10]	r	Reserved Read as 0.

Micro Link Interface (MLI)

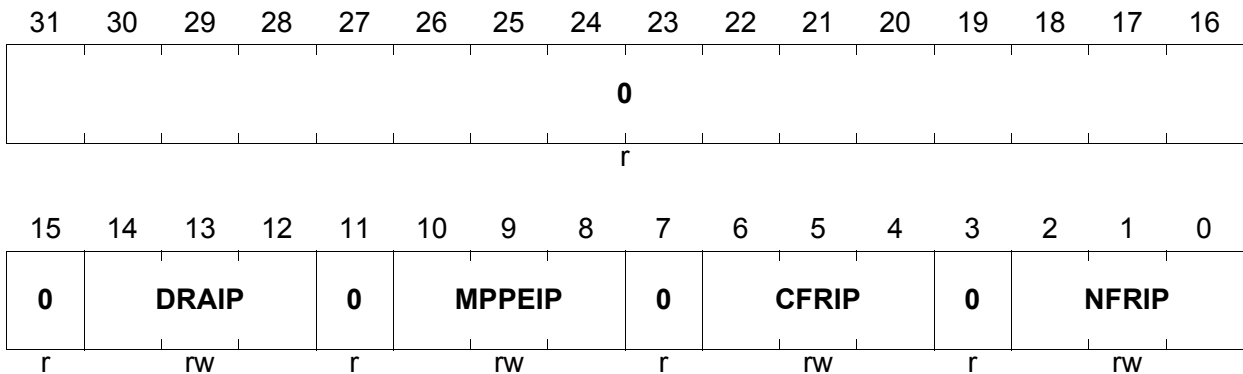
The Receiver Interrupt Node Pointer Register RINPR contains the node pointers for the MLI receiver events.

RINPR

Receiver Interrupt Node Pointer Register

(AC_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
NFRIP	[2:0]	rw	<p>Normal Frame Received Interrupt Pointer This bit field determines which service request output SR_x becomes active when a Normal Frame received event occurs.</p> <p>000_B The service request output SR0 is selected. 001_B The service request output SR1 is selected. ..._B ... 110_B The service request output SR6 is selected. 111_B The service request output SR7 is selected.</p>
CFRIP	[6:4]	rw	<p>Command Frame Received Interrupt Pointer This bit field determines which service request output SR_x becomes active when a Command Frame received event occurs. Coding see NFRIP.</p>
MPPEIP	[10:8]	rw	<p>Memory Protection or Parity Error Interrupt Pointer This bit field determines which service request output SR_x becomes active when a memory protection/parity error event occurs. Coding see NFRIP.</p>
DRAIP	[14:12]	rw	<p>Discarded Read Answer Interrupt Pointer This bit field determines which service request output SR_x becomes active when a discarded read answer event occurs. Coding see NFRIP.</p>

Micro Link Interface (MLI)

Field	Bits	Type	Description
0	3, 7, 11, [31:15]	r	Reserved Read as 0; should be written with 0.

23.5 Implementation of the MLI0/MLI1 in TC1796

This section describes the MLI0/MLI1 module related external functions such as port connections, interrupt and service request control, connections to other on-chip modules, clock control, and the address map.

23.5.1 Interfaces of the MLI Modules

Each MLI module is supplied with separate clock control, address decoding, and interrupt control logic. Four (for MLI0) and two (for MLI1) of the eight module service request outputs are connected to service request nodes. Four service request outputs of each MLI module are connected as DMA request to the DMA controller.

The data, clock, and control lines of each MLI receiver and transmitter are connected to GPIO lines. Alternate functions of Port 1 and Port 5 lines are assigned to the MLI0 module I/O lines while alternate functions of Port 8 lines are assigned to the MLI1 module I/O lines. Additionally, within one MLI module transmitter and receiver signals can be dynamically connected among each other without using pins; this is useful for test purposes in the local loop back mode. In this mode, the connections of the signals in pair are:

- TCLK/RCLKD
- TREADYD/RREADYD
- TVALIDD/RVALIDD
- TDATA/RDATAD

Figure 23-51 and **Figure 23-52** show how the MLI0 and MLI1 modules are interconnected to port lines and other on-chip functional blocks.

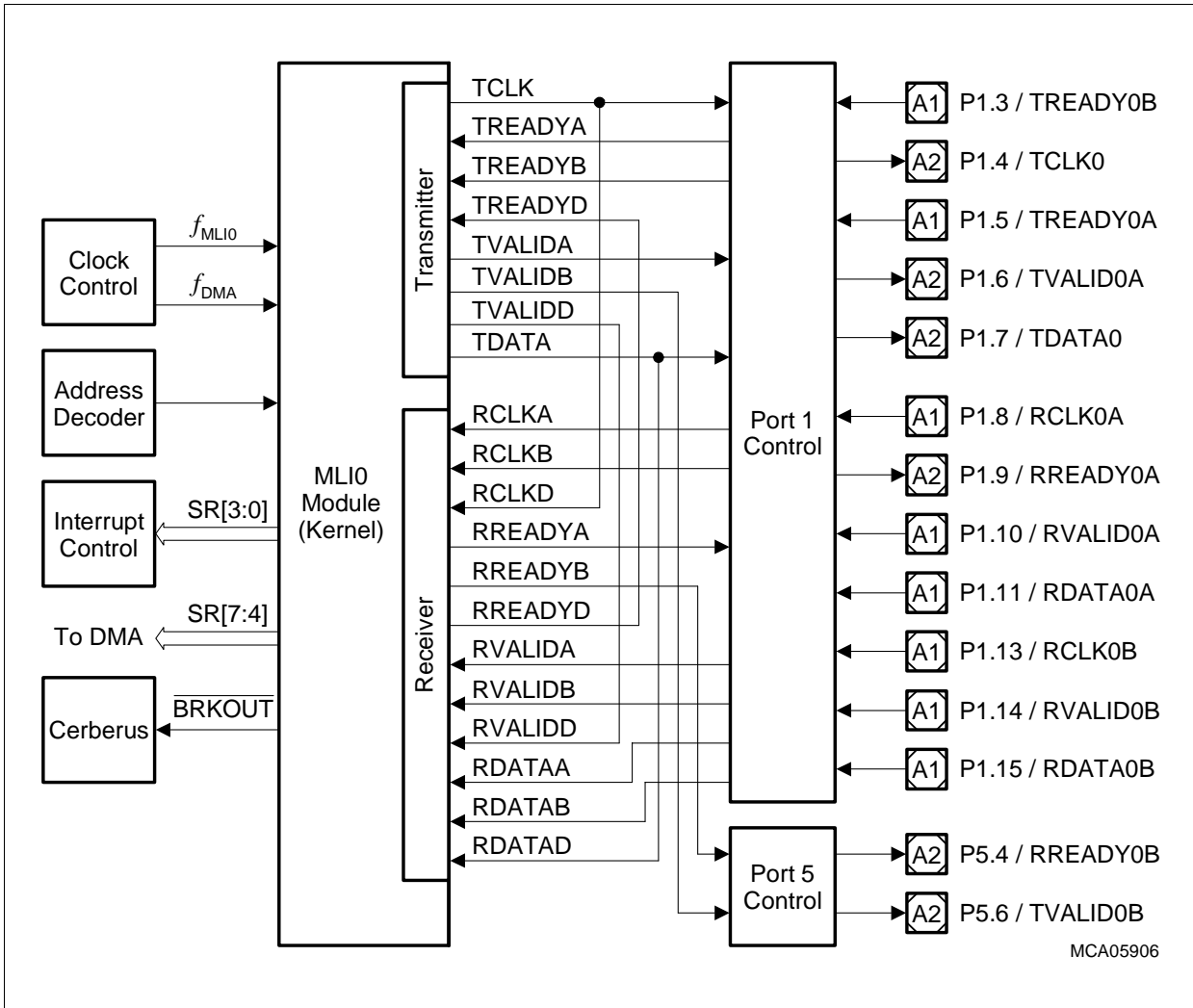


Figure 23-51 MLI0 Module Implementation and Interconnections

When programming the MLI0_OICR register, the following additional items must be considered:

- Unused transmitter/receiver output lines with index “C” (TVALIDC and RREADYC, not shown in the figure above) are not connected.
- Unused transmitter/receiver input lines with index “C” (TREADYC, RCLKC, RVALIDC, and RDATAAC, not shown in the figure above) are connected to low level.

See also [Page 23-130](#) for additional details on I/O line control and function selection.

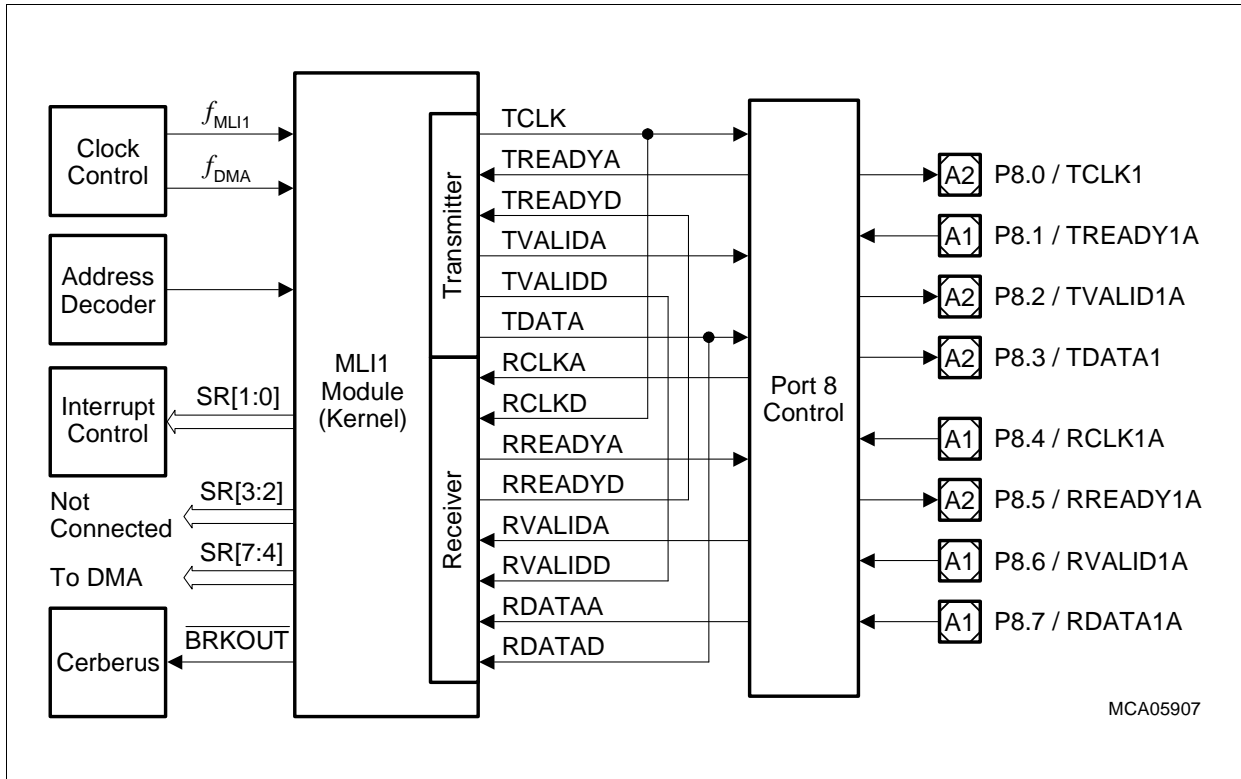


Figure 23-52 MLI1 Module Implementation and Interconnections

When programming the MLI1_OICR register, the following additional items must be considered:

- Lines with index “B” (not shown in the figure above)
 - Unused transmitter/receiver output lines TVALIDB and RREADYB are not connected.
 - Unused transmitter/receiver input lines TREADYB, RCLKB, RVALIDB, and RDATA B are connected to low level.
- Lines with index “C” (not shown in the figure above)
 - Unused transmitter/receiver output lines TVALIDC and RREADYC are reserved for emulation purposes.
 - Unused transmitter/receiver input lines TREADYC, RCLKC, RVALIDC, and RDATA C are reserved for emulation purposes and should not be selected during normal operation of the TC1796.

See also [Page 23-130](#) for additional details on I/O line control and function.

23.5.2 MLI Module External Registers

Figure 23-53 summarizes the module related external registers that are required for MLI0/MLI1 programming. Details on MLI0/MLI1 related register settings are shown in the following sections.

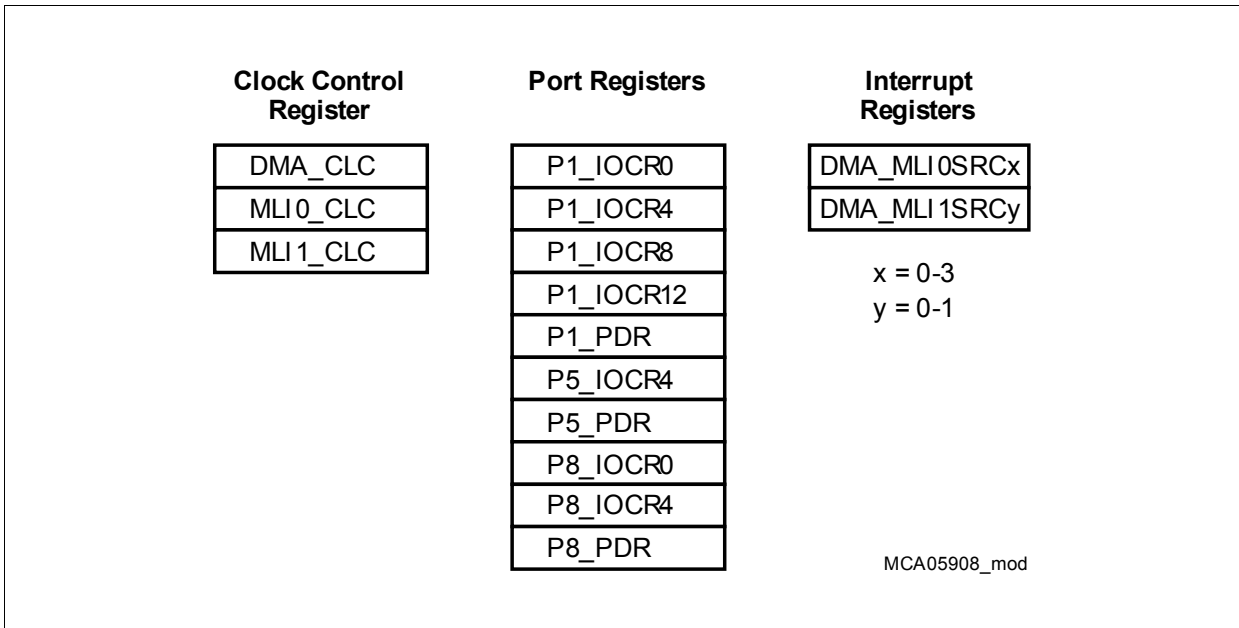


Figure 23-53 MLI0/MLI1 Implementation-Specific Special Function Registers

23.5.2.1 Automatic Register Overwrite

The following values are applied after reset (see [Page 23-66](#)).

- OICR = 1000 8000_H; Setting “A” is selected
- RCR.RCVRST = 0: the receiver is enabled for reception.

23.5.3 Module Clock Generation

The module clock generation configuration for the two MLI modules is shown in [Figure 23-54](#).

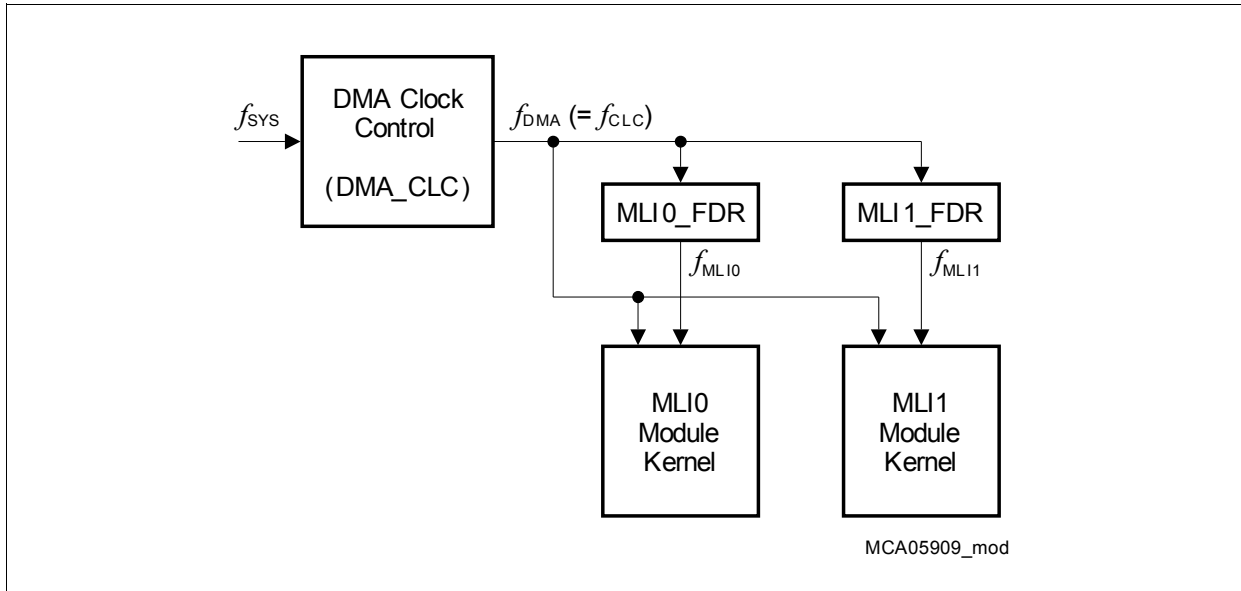


Figure 23-54 Clock Configuration of the MLI Modules

The two MLI modules (MLI0 and MLI1) are supplied from a common module clock f_{DMA} , that has the frequency of the system clock $f_{SYS} (= f_{FPI})$ and is controlled via the DMA_CLC clock control register. The MLI modules do not have its own clock control registers. Its module clocks f_{MLI0} and f_{MLI1} are derived from f_{DMA} by two separate fractional divider registers, MLI0_FDR and MLI1_FDR (FDR description see [Page 23-79](#) and baud rate generation see [Section 23.2.7](#) on [Page 23-65](#)).

Note: Additional details on the fractional divider register functionality are described in section “[Fractional Divider Operation](#)” on [Page 3-29](#) of the TC1796 User’s Manual System Units part (Volume 1).

- f_{DMA}
This is the module clock used inside the MLI kernels for control purposes such as for clocking of control logic and register operations. The clock control register DMA_CLC makes it possible to enable/disable f_{DMA} under certain conditions. DMA_CLC is described in the DMA chapter of this document.
- f_{MLI0} and f_{MLI1}
This clock is the module clock used in the MLI kernels as base for the shift clock and therefore determines the baud rate of the synchronous serial data transmission. The fractional divider registers MLI0_FDR and MLI1_FDR control the frequencies of f_{MLI0} and f_{MLI1} . This configuration makes it possible to enable/disable the module clocks f_{MLI0} and f_{MLI1} independently of f_{DMA} .

Micro Link Interface (MLI)

Combined with the baud rate as derived in the MLI module (see [Equation \(23.1\)](#) on [Page 23-65](#)) and the MLix_FDR fractional divider setup, the resulting MLI baud rate is defined by:

$$\text{Baud rate}_{\text{MLIX}} = \frac{f_{\text{DMA}}}{2} \times \frac{1}{n} \quad \text{with } n = 1024 - \text{FDR.STEP} \quad (23.4)$$

$$\text{Baud rate}_{\text{MLIX}} = \frac{f_{\text{DMA}}}{2} \times \frac{n}{1024} \quad \text{with } n = 0-1023 \quad (23.5)$$

[Equation \(23.4\)](#) applies to normal divider mode of the fractional divider (FDR.DM = 01_B). [Equation \(23.5\)](#) applies to fractional divider mode (FDR.DM = 10_B).

After a reset operation, both MLI modules are enabled in normal divider mode. According the MLix_FDR register's reset value of 03FF 43FF_H, the selected baud rate is $f_{\text{DMA}}/2$. Note that the DMA controller is also enabled after a reset operation with clock $f_{\text{DMA}} = f_{\text{SYS}}$. The receiver baud rate is defined by the following formula.

$$\text{Baud rate}_{\text{RLCKmax}} = f_{\text{MLI}} \quad (23.6)$$

23.5.4 Port Control and Connections

MLI0 and MLI1 clock and data output lines are connected to GPIO ports and are, therefore, controlled in the port logics (see also [Page 23-125](#) and [Page 23-126](#)). The following port control operations selections must be executed for these I/O lines:

- Input/output function selection (IOCR registers)
- Pad driver characteristics selection for the outputs (PDR registers)

23.5.4.1 Input/Output Function Selection

The port input/output control registers contain the bit fields that select the digital output and input driver characteristics such as port direction (input/output) with alternate output selection, pull-up/down devices, and open-drain selections. The I/O lines for the MLI modules are controlled by the Port 1, Port 5 and Port 8 input/output control registers. When the MLI modules are connected to the GPIO port lines, the correct settings of the enable/polarity control bits and bit fields in the output input control registers MLI0_IOCR and MLI1_IOCR must also be regarded (transmitter I/O line control see [Page 23-51](#), receiver I/O line control see [Page 23-51](#)). Note that after a reset operation the MLI0 and MLI1 modules (although enabled) have no direct connections to the GPIO lines.

[Table 23-11](#) shows how IOCR register bits and bit fields must be programmed for the required GPIO functionality of the MLI I/O lines.

Table 23-11 MLI0 and MLI1 I/O Line Selection and Setup

Module	Port Lines	Input/Output Control Register Bits	I/O
MLI0	P1.3 / TREADY0B	P1_IOCR0.PC3 = 0XXX _B ¹⁾ MLI0_IOCR.TRE = 1 MLI0_IOCR.TRP = X ²⁾ MLI0_IOCR.TRS = 01 _B	Input
	P1.4 / TCLK0	P1_IOCR4.PC4 = 1X01 _B ¹⁾ MLI0_IOCR.TCE = 1 MLI0_IOCR.TCP = X ²⁾	Output
	P1.5 / TREADY0A	P1_IOCR4.PC5 = 0XXX _B ¹⁾ MLI0_IOCR.TRE = 1 MLI0_IOCR.TRP = X ²⁾ MLI0_IOCR.TRS = 00 _B	Input
	P1.6 / TVALID0A	P1_IOCR4.PC6 = 1X01 _B ¹⁾ MLI0_IOCR.TVEA = 1 MLI0_IOCR.TVPA = X ²⁾	Output
	P1.7 / TDATA0	P1_IOCR4.PC7 = 1X01 _B ¹⁾ MLI0_IOCR.TDP = X ²⁾	Output

Micro Link Interface (MLI)

Table 23-11 MLI0 and MLI1 I/O Line Selection and Setup (cont'd)

Module	Port Lines	Input/Output Control Register Bits	I/O
MLI0	P1.8 / RCLK0A	P1_IOC8.PC8 = 0XXX _B ¹⁾ MLI0_OICR.RCE = 1 MLI0_OICR.RCP = X ²⁾ MLI0_OICR.RCS = 00 _B	Input
	P1.9 / RREADY0A	P1_IOC8.PC9 = 1X01 _B ¹⁾ MLI0_OICR.RRS = 00 _B MLI0_OICR.RRPA = X ²⁾	Output
	P1.10 / RVALID0A	P1_IOC8.PC10 = 0XXX _B ¹⁾ MLI0_OICR.RVE = 1 MLI0_OICR.RVP = X ²⁾ MLI0_OICR.RVS = 00 _B	Input
	P1.11 / RDATA0A	P1_IOC8.PC11 = 0XXX _B ¹⁾ MLI1_OICR.RDP = X ²⁾ MLI1_OICR.RDS = 00 _B	Input
	P1.13 / RCLK0B	P1_IOC12.PC13 = 0XXX _B ¹⁾ MLI0_OICR.RCE = 1 MLI0_OICR.RCP = X ²⁾ MLI0_OICR.RCS = 01 _B	Input
	P1.14 / RVALID0B	P1_IOC12.PC14 = 0XXX _B ¹⁾ MLI0_OICR.RVE = 1 MLI0_OICR.RVP = X ²⁾ MLI0_OICR.RVS = 01 _B	Input
	P1.15 / RDATA0B	P1_IOC12.PC15 = 0XXX _B ¹⁾ MLI1_OICR.RDP = X ²⁾ MLI1_OICR.RDS = 01 _B	Input
	P5.4 / RREADY0B	P5_IOC4.PC4 = 1X10 _B ¹⁾ MLI0_OICR.RRS = 01 _B MLI0_OICR.RRPB = X ²⁾	Output
	P5.6 / TVALID0B	P5_IOC4.PC6 = 1X10 _B ¹⁾ MLI0_OICR.TVEB = 1 MLI0_OICR.TVPB = X ²⁾	Output

Micro Link Interface (MLI)

Table 23-11 MLI0 and MLI1 I/O Line Selection and Setup (cont'd)

Module	Port Lines	Input/Output Control Register Bits	I/O
MLI1	P8.0 / TCLK1	P8_IOCR0.PC0 = 1X11 _B ¹⁾ MLI1_OICR.TCE = 1 MLI1_OICR.TCP = X ²⁾	Output
	P8.1 / TREADY1A	P8_IOCR0.PC1 = 0XXX _B ¹⁾ MLI1_OICR.TRE = 1 MLI1_OICR.TRP = X ²⁾ MLI1_OICR.TRS = 00 _B	Input
	P8.2 / TVALID1A	P8_IOCR0.PC2 = 1X11 _B ¹⁾ MLI1_OICR.TVEA = 1 MLI1_OICR.TVPA = X ²⁾	Output
	P8.3 / TDATA1	P8_IOCR0.PC3 = 1X11 _B ¹⁾ MLI1_OICR.TDP = X ²⁾	Output
	P8.4 / RCLK1A	P8_IOCR4.PC4 = 0XXX _B ¹⁾ MLI1_OICR.RCE = 1 MLI1_OICR.RCP = X ²⁾ MLI1_OICR.RCS = 00 _B	Input
	P8.5 / RREADY1A	P8_IOCR4.PC5 = 1X11 _B ¹⁾ MLI1_OICR.RRS = 00 _B MLI1_OICR.RRPA = X ²⁾	Output
	P8.6 / RVALID1A	P8_IOCR4.PC6 = 0XXX _B ¹⁾ MLI1_OICR.RVE = 1 MLI1_OICR.RVP = X ²⁾ MLI1_OICR.RVS = 00 _B	Input
	P8.7 / RDATA1A	P8_IOCR4.PC7 = 0XXX _B ¹⁾ MLI1_OICR.RDP = X ²⁾ MLI1_OICR.RDS = 00 _B	Input

1) Possible PCx bit field combinations see [Table 23-12](#).

2) With polarity control bit = 0, normal polarity is selected. With polarity control bit = 1, inverted polarity is selected.

Table 23-12 PCx Coding

PCx[3:0]	I/O	Output Characteristics	Selected Pull-up/Pull-down/ Selected Output Function
0X00 _B	Input	–	No pull device connected
0X01 _B			Pull-down device connected
0X10 _B ¹⁾			Pull-up device connected
0X11 _B			No pull device connected
1001 _B	Output	Push-pull	Output function ALT1
1101 _B		Open-drain	Output function ALT1
1010 _B		Push-pull	Output function ALT2
1110 _B		Open-drain	Output function ALT2
1011 _B		Push-pull	Output function ALT3
1111 _B		Open-drain	Output function ALT3

1) This bit field value is default after reset.

23.5.4.2 Input/Output Control Register

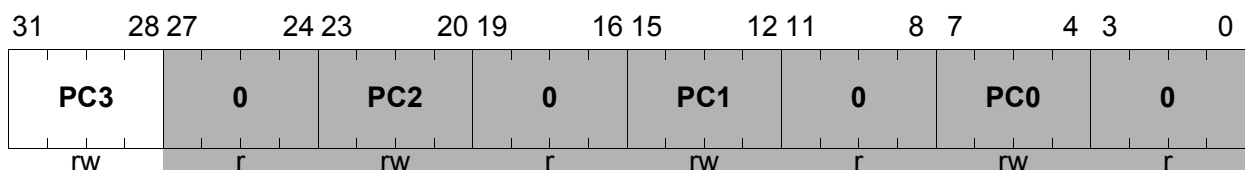
The following pages describe all IOCR registers that contain MLI0/MLI1 related input/output control bit fields. Shaded bits and bit fields are “don’t care” for MLI0/MLI1 I/O port control purposes.

P1_IOCR0

Port 1 Input/Output Control Register 0

(10_H)

Reset Value: 2020 2020_H



Field	Bits	Type	Description
PC3	[31:28]	rw	Port Input/Output Control for Port 1 Pins 0-3¹⁾ Port input/output control for P1.3/TREADY0B

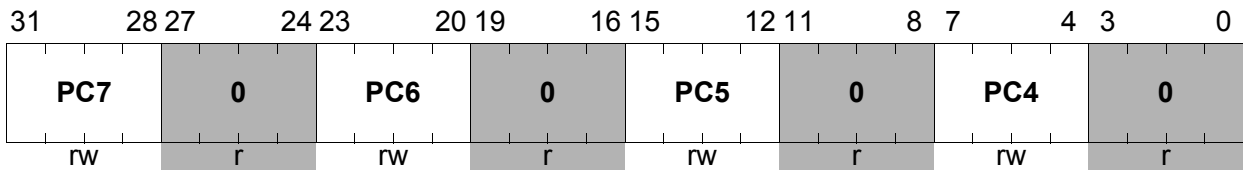
1) Coding of bit field see [Table 23-12](#). Shaded bits and bit fields are “don’t care” for MLI I/O port control.

Micro Link Interface (MLI)

P1_IOCRA4

Port 1 Input/Output Control Register 4(14_H)

Reset Value: 2020 2020_H



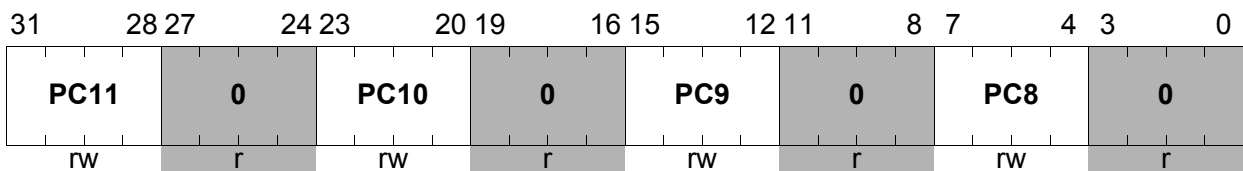
Field	Bits	Type	Description
PC4, PC5, PC6, PC7	[7:4], [15:12], [23:20], [31:28]	rw	Port Input/Output Control for Port 1 Pins 4-7¹⁾ Port input/output control for P1.4/TCLK0 Port input/output control for P1.5/TREADY0A Port input/output control for P1.6/TVALID0A Port input/output control for P1.7/TDATA0

1) Coding of bit field see [Table 23-12](#). Shaded bits and bit fields are “don’t care” for MLI I/O port control.

P1_IOCRA8

Port 1 Input/Output Control Register 8(18_H)

Reset Value: 2020 2020_H



Field	Bits	Type	Description
PC8, PC9, PC10, PC11	[7:4], [15:12], [23:20], [31:28]	rw	Port Input/Output Control for Port 1 Pins 8-11¹⁾ Port input/output control for P1.8/RCLK0A Port input/output control for P1.9/RREADY0A Port input/output control for P1.10/RVALID0A Port input/output control for P1.11/RDATA0A

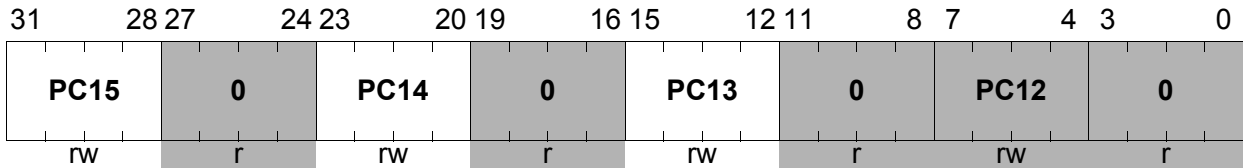
1) Coding of bit field see [Table 23-12](#). Shaded bits and bit fields are “don’t care” for MLI I/O port control.

Micro Link Interface (MLI)

P1_IOC12

Port 1 Input/Output Control Register 12(1C_H)

Reset Value: 2020 2020_H



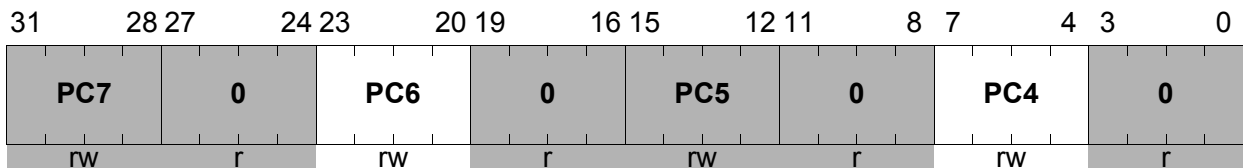
Field	Bits	Type	Description
PC13, PC14, PC15	[15:12], [23:20], [31:28]	rw	Port Input/Output Control for Port 1 Pins 12-15¹⁾ Port input/output control for P1.13/RCLK0B Port input/output control for P1.14/RVALID0B Port input/output control for P1.15/RDATA0B

1) Coding of bit field see [Table 23-12](#). Shaded bits and bit fields are “don’t care” for MLI I/O port control.

P5_IOC4

Port 5 Input/Output Control Register 4(14_H)

Reset Value: 2020 2020_H



Field	Bits	Type	Description
PC4, PC6	[7:4], [23:20]	rw	Port Input/Output Control for Port 5 Pins 4-7¹⁾ Port input/output control for P5.4/RREADY0B Port input/output control for P5.6/TVALID0B

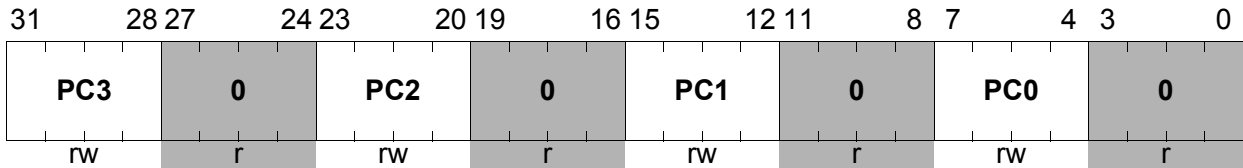
1) Coding of bit field see [Table 23-12](#). Shaded bits and bit fields are “don’t care” for MLI I/O port control.

Micro Link Interface (MLI)

P8_IOCRO

Port 8 Input/Output Control Register 0(10_H)

Reset Value: 2020 2020_H



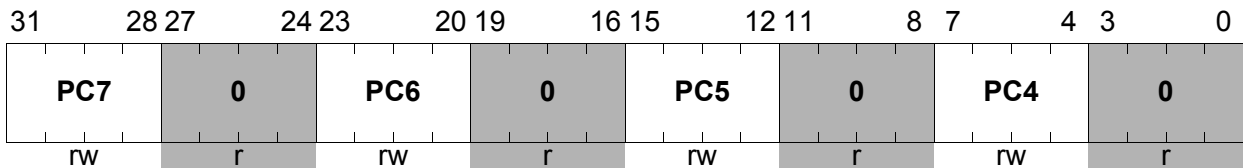
Field	Bits	Type	Description
PC0, PC1, PC2, PC3	[7:4], [15:12], [23:20], [31:28]	rw	Port Input/Output Control for Port 8 Pins 0-4¹⁾ Port input/output control for P8.0/TCLK1 Port input/output control for P8.1/TREADY1A Port input/output control for P8.2/TVALID1A Port input/output control for P8.3/TDATA1

1) Coding of bit field see [Table 23-12](#). Shaded bits and bit fields are “don’t care” for MLI I/O port control.

P8_IOCRA

Port 8 Input/Output Control Register 4(14_H)

Reset Value: 2020 2020_H



Field	Bits	Type	Description
PC4, PC5, PC6, PC7	[7:4], [15:12], [23:20], [31:28]	rw	Port Input/Output Control for Port 8 Pins 4-7¹⁾ Port input/output control for P8.4/RCLK1A Port input/output control for P8.5/RREADY1A Port input/output control for P8.6/RVALID1A Port input/output control for P8.7/RDATA1A

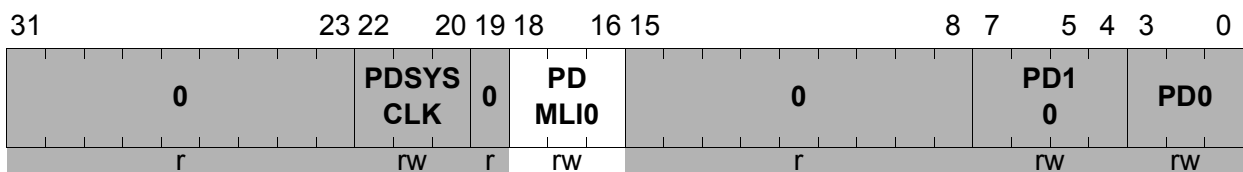
1) Coding of bit field see [Table 23-12](#). Shaded bits and bit fields are “don’t care” for MLI I/O port control.

23.5.4.3 Pad Driver Characteristics Selection

The Port 1, Port 5 and Port 8 pad driver mode registers contain bit fields that determine the output driver strength and the slew rate of the MLI output lines. A detailed description of all available PDx bit field combinations is given in Chapter “GPIO Ports” in the TC1796 System Units User’s Manual. Shaded bits and bit fields are “don’t care” for MLI0/MLI1 I/O port control purposes.

P1_PDR

Port 1 Pad Driver Mode Register (40_H) Reset Value: 0000 0000_H

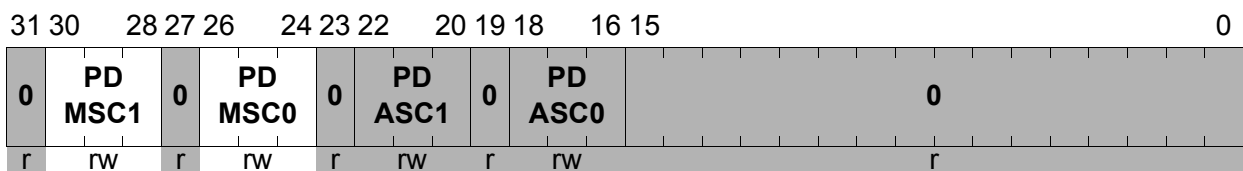


Field	Bits	Type	Description
PDMLIO	[18:16]	rw	Pad Driver Mode for P1.4/TCLK0, P1.6/TVALID0A, P1.7/TDATA0, and P1.9/RREADY0A ¹⁾

1) Coding of bit field see [Table 23-13](#). Shaded bits and bit fields are “don’t care” for MLI I/O port control.

P5_PDR

Port 5 Pad Driver Mode Register (40_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
PDMSC0	[26:24]	rw	Pad Driver Mode for P5.4/RREADY0B ¹⁾
PDMSC1	[30:28]	rw	Pad Driver Mode for P5.6/TVALID0B ¹⁾

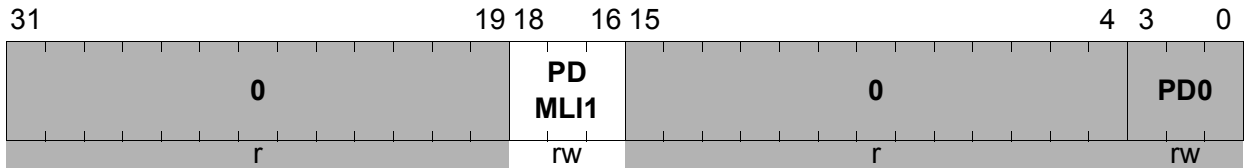
1) Coding of bit field see [Table 23-13](#). Shaded bits and bit fields are “don’t care” for MLI I/O port control.

Micro Link Interface (MLI)

P8_PDR

Port 8 Pad Driver Mode Register (40_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PDMLI1	[18:16]	rw	Pad Driver Mode for P8.0/TCLK1, P8.2/TVALID1A, P8.3/TDATA1, and P8.5/RREADY1A ¹⁾

1) Coding of bit field see [Table 23-13](#). Shaded bits and bit fields are “don’t care” for MLI I/O port control.

PDx Selection Table

Table 23-13 Pad Driver Mode Mode Selection (Class A2 Pads)

PDx Bit Field	Driver Strength	Signal Transitions
000 _B	Strong driver	Sharp edge ¹⁾
001 _B		Medium edge ¹⁾
010 _B		Soft edge ¹⁾
011 _B	Weak driver	–
100 _B	Medium driver	–
101 _B		–
110 _B		–
111 _B	Weak driver	–

1) In strong driver mode, the output driver characteristics of class A2 pads can be additionally controlled by the temperature compensation logic.

23.5.5 On-Chip Connections

23.5.5.1 Service Request Output Connections

Each MLI module provides eight service request outputs SR[7:0] that can be used to generate interrupts or DMA requests. In the TC1796, four service request outputs SR[3:0] of the MLI0 module and two service request outputs SR[1:0] of the MLI1 module are connected to an interrupt node. Service request outputs SR[3:2] of the MLI1 module are not connected. Four service request outputs (SR[7:4]) of each MLI module are connected to DMA request inputs of the TC1796 DMA controller.

Each of the service request outputs used as interrupt requests are controlled by a service request control register. The service request control registers of the MLI modules are located inside the DMA address area. Therefore, all MLI0/MLI1 service request control registers are named as DMA_MLIxSRCy and described in the DMA chapter implementation part (see [Page 12-107](#)) of the TC1796 System Units User's Manual.

All MLI service request output connections are listed in [Table 23-14](#).

Table 23-14 Service Request Lines and Interconnections of MLI0/MLI1

Module	Service Req. Output Line	Connected to Node or DMA Request Input	Description
MLI0	SR0	DMA_MLI0SRC0	MLI0 Service Request Node 0 (in DMA)
	SR1	DMA_MLI0SRC1	MLI0 Service Request Node 1 (in DMA)
	SR2	DMA_MLI0SRC2	MLI0 Service Request Node 2 (in DMA)
	SR3	DMA_MLI0SRC3	MLI0 Service Request Node 3 (in DMA)
	SR4	CH00_REQI7	DMA Channel 00 Request Input 7
		CH04_REQI7	DMA Channel 04 Request Input 7
	SR5	CH01_REQI7	DMA Channel 01 Request Input 7
		CH05_REQI7	DMA Channel 05 Request Input 7
	SR6	CH02_REQI7	DMA Channel 02 Request Input 7
		CH06_REQI7	DMA Channel 06 Request Input 7
	SR7	CH03_REQI7	DMA Channel 03 Request Input 7
		CH07_REQI7	DMA Channel 07 Request Input 7

Table 23-14 Service Request Lines and Interconnections of MLI0/MLI1 (cont'd)

Module	Service Req. Output Line	Connected to Node or DMA Request Input	Description	
MLI1	SR0	DMA_MLI1SRC0	MLI1 Service Request Node 0 (in DMA)	
	SR1	DMA_MLI1SRC1	MLI1 Service Request Node 1 (in DMA)	
	SR2	–	Not connected	
	SR3	–	Not connected	
	SR4		CH10_REQI14	DMA Channel 10 Request Input 15
			CH14_REQI14	DMA Channel 14 Request Input 15
	SR5		CH11_REQI6	DMA Channel 11 Request Input 15
			CH15_REQI5	DMA Channel 15 Request Input 15
	SR6		CH12_REQI5	DMA Channel 12 Request Input 15
			CH16_REQI6	DMA Channel 16 Request Input 15
	SR7		CH13_REQI5	DMA Channel 13 Request Input 15
			CH17_REQI6	DMA Channel 17 Request Input 15

23.5.5.2 Break Signals

The $\overline{\text{BRKOUT}}$ output signals of MLI0 and MLI1 are connected as break input signals to the Multi Core Break Switch (MCBS) that is a part of the Cerberus on-chip debug control module. These connections allow MLI0/MLI1 initiated break conditions to be generated in the Cerberus.

23.5.6 Access Protection

23.5.6.1 Fixed Address Range Definition

Table 23-15 below shows the fixed address ranges covered by the access protection (for read and write) of the MLI modules. The access enable bits AER.AENx are related to address range x. If bit AENx is set, read/write accesses to the corresponding address range x are allowed. Some of the access enable bits AENx control more than one address range.

Table 23-15 MLI Access Protection Address Ranges

Access Protection Range			Related Module(s)
No. x	Enable Bit in AER	Selected Address Range	
0	AEN0	F000 0000 _H to F000 00FF _H F010 C200 _H to F010 C2FF _H	SCU, incl. WDT MEMCHK
1	AEN1	F000 0100 _H to F000 01FF _H F010 0000 _H to F010 00FF _H	SBCU RBCU
2	AEN2	F000 0200 _H to F000 02FF _H	STM
3	AEN3	F000 0400 _H to F000 04FF _H	OCDS
4	AEN4	F000 0800 _H to F000 08FF _H F000 0900 _H to F000 09FF _H	MSC0 MSC1
5	AEN5	F000 0A00 _H to F000 0AFF _H	ASC0
6	AEN6	F000 0B00 _H to F000 0BFF _H	ASC1
7	AEN7	F000 0C00 _H to F000 0FFF _H	P0, P1, P2, P3
8	AEN8	F000 1000 _H to F000 17FF _H	P4, P5, P6, P7, P8, P9, P10
9	AEN9	F000 1800 _H to F000 2FFF _H	GPTA0, GPTA1, LTCA2
10	AEN10	F000 3C00 _H to F000 3EFF _H	DMA
11	AEN11	F000 4000 _H to F000 5FFF _H	MultiCAN
12	AEN12	F004 0000 _H to F005 FFFF _H	PCP Registers, PCP Data Memory
13	AEN13	F006 0000 _H to F006 7FFF _H	PCP Code Memory
14	AEN14	F010 0100 _H to F010 01FF _H	SSC0
15	AEN15	F010 0200 _H to F010 02FF _H	SSC1
16	AEN16	F010 0300 _H to F010 03FF _H	FADC

Table 23-15 MLI Access Protection Address Ranges (cont'd)

Access Protection Range			Related Module(s)
No. x	Enable Bit in AER	Selected Address Range	
17	AEN17	F010 0400 _H to F010 07FF _H	ADC0, ADC1
18	AEN18	F010 A000 _H to F010 BFFF _H	DPRAM
19	AEN19	F010 C000 _H to F010 C0FF _H F01E 0000 _H to F01E 7FFF _H F020 0000 _H to F023 FFFF _H	MLI0 Module, MLI0 Small TWs, MLI0 Large TWs
20	AEN20	F010 C100 _H to F010 C1FF _H F01E 8000 _H to F01E FFFF _H F024 0000 _H to F027 FFFF _H	MLI1, MLI1 Small TWs, MLI1 Large TWs
21	AEN21	F7E0 FF00 _H to F7E0 FFFF _H F7E1 0000 _H to F7E1 FFFF _H F800 0500 _H to F87F FFFF _H	CPS CPU SFRs & GPRs PMU, Flash Regs, DMU, DBCUC, DMI, PMI, PBCUC, LFI
22	AEN22	F800 0000 _H to F800 03FF _H	EBU
23	AEN23	8000 0000 _H to 807F FFFF _H A000 0000 _H to A07F FFFF _H	Program Flash Space
24	AEN24	8080 0000 _H to 8FDF FFFF _H A080 0000 _H to AFDF FFFF _H	External EBU Space
25	AEN25	8FE0 0000 _H to 8FEF FFFF _H AFE0 0000 _H to AFEF FFFF _H	Data Flash Space
26	AEN26	8FF0 0000 _H to 8FFF BFFF _H AFF0 0000 _H to AFFF BFFF _H	Emulation Device Memory Space
27	AEN27	8FFF C000 _H to 8FFF FFFF _H AFFF C000 _H to AFFF FFFF _H	Boot ROM
28	AEN28	D800 0000 _H to DEFF FFFF _H E000 0000 _H to E7FF FFFF _H	External Peripherals & External Emulator
29	AEN29	E800 0000 _H to E83F FFFF _H	LMU Image (E80x translated to C00x)
30	AEN30	E840 0000 _H to E84F FFFF _H	DMI Image (E84x translated to D00x)
31	AEN31	E850 0000 _H to E85F FFFF _H	PMI Image (E85x translated to D40x)

23.5.6.2 Programmable Address Sub-Range Definitions

In the TC1796, two of four possible programmable address range extensions are implemented for internal memory areas:

- 8-Kbyte Dual-Ported RAM (DPRAM) assigned to **address range 18**; sub-ranges are controlled by bit fields ARR.SIZE0 and ARR.SLICE0 (see [Table 23-16](#)).
- 64-Kbyte SRAM assigned as **address range 29**; sub-ranges are controlled by bit fields ARR.SIZE1 and ARR.SLICE1 (see [Table 23-17](#)).

Attention: *Bit fields ARR.SLICE2/ARR.SIZE2 and ARR.SLICE3/ARR.SIZE3 are not used in TC1796.*

The coding of bit fields SIZE0 and SLICE0 for the DPRAM sub-range access protection are defined as shown in [Table 23-16](#).

Table 23-16 DPRAM Address Protection Sub-Range Definitions

SIZE0	Sub-Ranges	SLICE0	Selected Address Range
000 _B	32 sub-ranges of 64 bytes	00000 _B 00001 _B ... 11111 _B	F010 A000 _H to F010 A03F _H F010 A040 _H to F010 A07F _H ... F010 A7C0 _H to F010 A7FF _H F010 A800 _H to F010 BFFF _H is not selectable
001 _B	32 sub-ranges of 128 bytes	00000 _B 00001 _B ... 11111 _B	F010 A000 _H to F010 A07F _H F010 A080 _H to F010 A0FF _H ... F010 AF80 _H to F010 AFFF _H F010 B000 _H to F010 BFFF _H is not selectable
010 _B	32 sub-ranges of 256 bytes	00000 _B 00001 _B ... 11111 _B	F010 A000 _H to F010 A0FF _H F010 A100 _H to F010 A1FF _H ... F010 BF00 _H to F010 BFFF _H
011 _B	16 sub-ranges of 512 bytes	X0000 _B X0001 _B ... X1111 _B	F010 A000 _H to F010 A1FF _H F010 A200 _H to F010 A3FF _H ... F010 BE00 _H to F010 BFFF _H
100 _B	8 sub-ranges of 1 Kbyte	XX000 _B XX001 _B ... XX111 _B	F010 A000 _H to F010 A3FF _H F010 A400 _H to F010 A7FF _H ... F010 BC00 _H to F010 BFFF _H

Table 23-16 DPRAM Address Protection Sub-Range Definitions (cont'd)

SIZE0	Sub-Ranges	SLICE0	Selected Address Range
101 _B	4 sub-ranges of 2 Kbytes	XXX00 _B XXX01 _B XXX10 _B XXX11 _B	F010 A000 _H to F010 A7FF _H F010 A800 _H to F010 AFFF _H F010 B000 _H to F010 B7FF _H F010 B800 _H to F010 BFFF _H
110 _B	2 sub-ranges of 4 Kbytes	XXXX0 _B XXXX1 _B	F010 A000 _H to F010 AFFF _H F010 B000 _H to F010 BFFF _H
111 _B	8 Kbytes	XXXXX _B	F010 A000 _H to F010 BFFF _H

The coding of bit fields SIZE1 and SLICE1 for the SRAM sub-range access protection (with address translation from E800 to C000) are coded as shown in [Table 23-17](#).

Table 23-17 SRAM Address Protection Sub-Range Definitions

SIZE1	Sub-Ranges	SLICE1	Selected Address Range
000 _B	32 sub-ranges of 512 bytes	00000 _B 00001 _B ... 11111 _B	E800 0000 _H to E800 01FF _H E800 0200 _H to E800 03FF _H ... E800 3E00 _H to E800 3FFF _H E800 4000 _H to E800 FFFF _H is not selectable
001 _B	32 sub-ranges of 1 Kbyte	00000 _B 00001 _B ... 11111 _B	E800 0000 _H to E800 03FF _H E800 0400 _H to E800 07FF _H ... E800 7C00 _H to E800 7FFF _H E800 8000 _H to E800 FFFF _H is not selectable
010 _B	32 sub-ranges of 2 Kbytes	00000 _B 00001 _B ... 11111 _B	E800 0000 _H to E800 07FF _H E800 0800 _H to E800 0FFF _H ... E800 F800 _H to E800 FFFF _H
011 _B	16 sub-ranges of 4 Kbytes	X0000 _B X0001 _B ... X1111 _B	E800 0000 _H to E800 0FFF _H E800 1000 _H to E800 1FFF _H ... E800 F000 _H to E800 FFFF _H
100 _B	8 sub-ranges of 8 Kbytes	XX000 _B XX001 _B ... XX111 _B	E800 0000 _H to E800 1FFF _H E800 2000 _H to E800 3FFF _H ... E800 E000 _H to E800 FFFF _H

Table 23-17 SRAM Address Protection Sub-Range Definitions (cont'd)

SIZE1	Sub-Ranges	SLICE1	Selected Address Range
101 _B	4 sub-ranges of 16 Kbytes	XXX00 _B XXX01 _B XXX10 _B XXX11 _B	E800 0000 _H to E800 3FFF _H E800 4000 _H to E800 7FFF _H E800 8000 _H to E800 BFFF _H E800 C000 _H to E800 FFFF _H
110 _B	2 sub-ranges of 32 Kbytes	XXXX0 _B XXXX1 _B	E800 0000 _H to E800 7FFF _H E800 8000 _H to E800 FFFF _H
111 _B	64 Kbytes	XXXXX _B	E800 0000 _H to E800 FFFF _H

23.5.7 MLI0/MLI1 Transfer Window Address Maps

In the TC1796, the transfer windows for the MLI0 and MLI1 modules are located in the address ranges as identified in [Table 23-18](#).

Table 23-18 MLI0/MLI1 Transfer Windows

Module	Window Type	Pipe	Address Range
MLI0	Small Transfer Window	Pipe 0	F01E 0000 _H to F01E 1FFF _H
		Pipe 1	F01E 2000 _H to F01E 3FFF _H
		Pipe 2	F01E 4000 _H to F01E 5FFF _H
		Pipe 3	F01E 6000 _H to F01E 7FFF _H
	Large Transfer Window	Pipe 0	F020 0000 _H to F020 FFFF _H
		Pipe 1	F021 0000 _H to F021 FFFF _H
		Pipe 2	F022 0000 _H to F022 FFFF _H
		Pipe 3	F023 0000 _H to F023 FFFF _H
MLI1	Small Transfer Window	Pipe 0	F01E 8000 _H to F01E 9FFF _H
		Pipe 1	F01E A000 _H to F01E BFFF _H
		Pipe 2	F01E C000 _H to F01E DFFF _H
		Pipe 3	F01E E000 _H to F01E FFFF _H
	Large Transfer Window	Pipe 0	F024 0000 _H to F024 FFFF _H
		Pipe 1	F025 0000 _H to F025 FFFF _H
		Pipe 2	F026 0000 _H to F026 FFFF _H
		Pipe 3	F027 0000 _H to F027 FFFF _H

24 General Purpose Timer Array (GPTA)

This chapter describes the General Purpose Timer Array of the TC1796. The GPTA consists of three sub-modules: GPTA0 and GPTA1 with identical functionality; LTCA2 has reduced GPTA0/1 functionality.

This chapter contains the following sections:

- A summary on the structure and basic functionalities (see [Page 24-2](#))
- Functional description of the GPTA kernel, applicable for GPTA0 and GPTA1 (see [Page 24-6](#))
- Register descriptions of all GPTA kernel specific registers, applicable for GPTA0 and GPTA1 (see [Page 24-150](#))
- Functional description of the LTCA2 kernel (see [Page 24-217](#))
- Register descriptions of all LTCA2 kernel specific registers (see [Page 24-233](#))
- TC1796 implementation-specific details and registers of the GPTA module, including port connections and control, interrupt control, address decoding, and clock control (see [Page 24-247](#)).

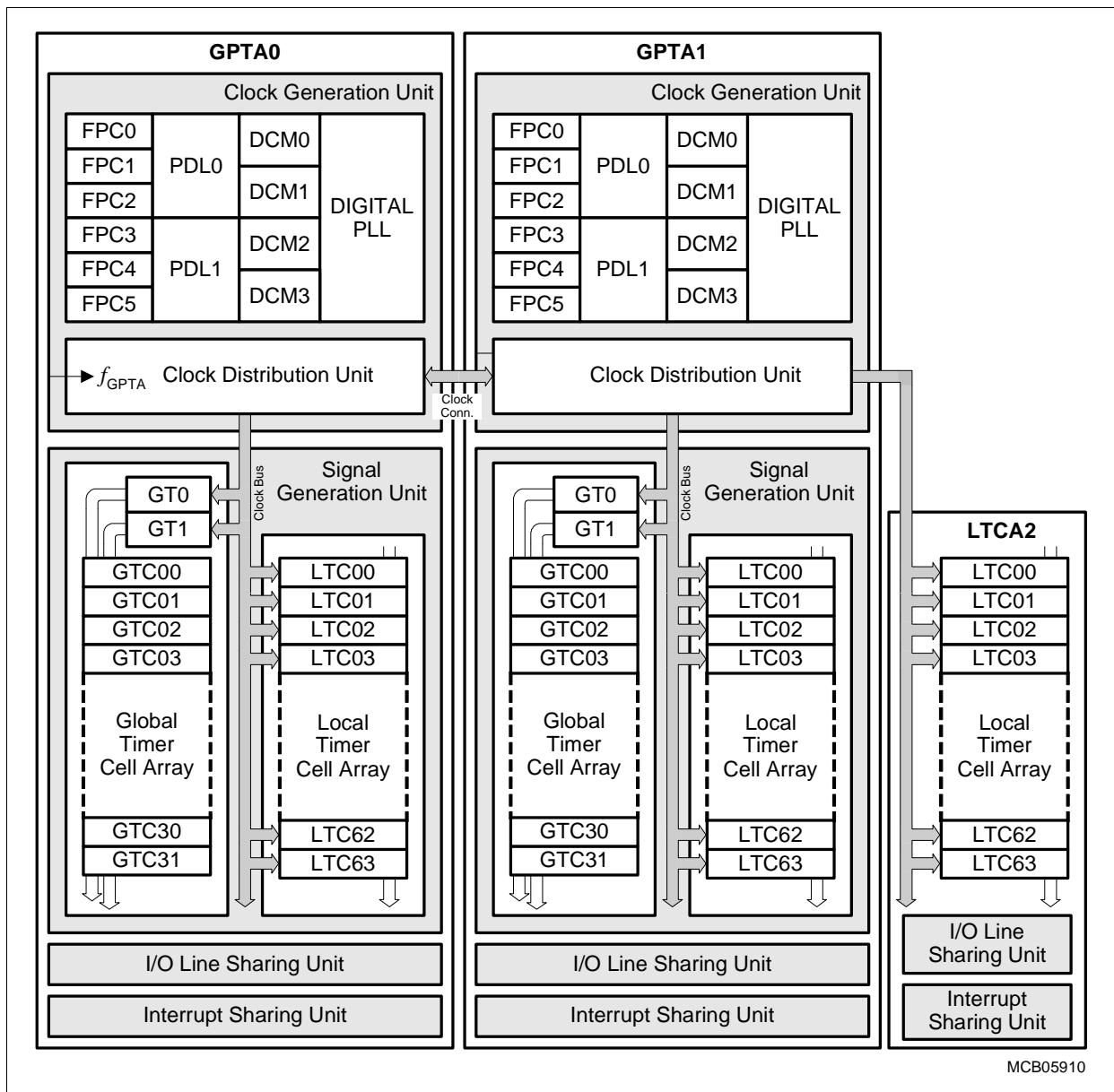
Note: The GPTA kernel register names described in [Section 24.3](#) and [Section 24.5](#) will be referenced in the TC1796 User's Manual by the module name prefix "GPTA0_" for the GPTA0 module, by "GPTA1_" for the GPTA1 module, and by "LTCA2_" for the LTCA2 module.

General Purpose Timer Array (GPTA)

24.1 GPTA Overview

The TC1796 contains two General Purpose Timer Arrays (GPTA0 and GPTA1) with identical functionality, plus an additional Local Timer Cell Array (LTCA2). **Figure 24-1** shows a global view of the GPTA modules.

The GPTA provides a set of timer, compare, and capture functionalities that can be flexibly combined to form signal measurement and signal generation units. They are optimized for tasks typical of engine, gearbox, and electrical motor control applications, but can also be used to generate simple and complex signal waveforms needed in other industrial applications.



MCB05910

Figure 24-1 General Block Diagram of the GPTA Modules in the TC1796

General Purpose Timer Array (GPTA)

24.1.1 Functionality of GPTA0 and GPTA1

Each of the General Purpose Timer Arrays (GPTA0 and GPTA1) provides a set of hardware modules required for high-speed digital signal processing:

- Filter and Prescaler Cells (FPC) support input noise filtering and prescaler operation.
- Phase Discrimination Logic units (PDL) decode the direction information output by a rotation tracking system.
- Duty Cycle Measurement Cells (DCM) provide pulse-width measurement capabilities.
- A Digital Phase Locked Loop unit (PLL) generates a programmable number of GPTA module clock ticks during an input signal's period.
- Global Timer units (GT) driven by various clock sources are implemented to operate as a time base for the associated Global Timer Cells.
- Global Timer Cells (GTC) can be programmed to capture the contents of a Global Timer on an external or internal event. A GTC may also be used to control an external port pin depending on the result of an internal compare operation. GTCs can be logically concatenated to provide a common external port pin with a complex signal waveform.
- Local Timer Cells (LTC) operating in Timer, Capture, or Compare Mode may also be logically tied together to drive a common external port pin with a complex signal waveform. LTCs – enabled in Timer Mode or Capture Mode – can be clocked or triggered by various external or internal events.

Input lines can be shared by an LTC and a GTC to trigger their programmed operation simultaneously.

The following list summarizes the specific features of the GPTA units.

Clock Generation Unit

- Filter and Prescaler Cell (FPC)
 - Six independent units
 - Three basic operating modes:
Prescaler, Delayed Debounce Filter, Immediate Debounce Filter
 - Selectable input sources:
Port lines, GPTA module clock, FPC output of preceding FPC cell
 - Selectable input clocks:
GPTA module clock, prescaled GPTA module clock, DCM clock, compensated or uncompensated PLL clock.
 - $f_{\text{GPTA}}/2$ maximum input signal frequency in Filter Modes
- Phase Discriminator Logic (PDL)
 - Two independent units
 - Two operating modes (2- and 3- sensor signals)
 - $f_{\text{GPTA}}/4$ maximum input signal frequency in 2-sensor Mode, $f_{\text{GPTA}}/6$ maximum input signal frequency in 3-sensor Mode

General Purpose Timer Array (GPTA)

- Duty Cycle Measurement (DCM)
 - Four independent units
 - 0 - 100% margin and time-out handling
 - f_{GPTA} maximum resolution
 - $f_{\text{GPTA}}/2$ maximum input signal frequency
- Digital Phase Locked Loop (PLL)
 - One unit
 - Arbitrary multiplication factor between 1 and 65535
 - f_{GPTA} maximum resolution
 - $f_{\text{GPTA}}/2$ maximum input signal frequency
- Clock Distribution Unit (CDU)
 - One unit
 - Provides nine clock output signals:
 - f_{GPTA} , divided f_{GPTA} clocks, FPC1/FPC4 outputs, DCM clock, LTC prescaler clock

Signal Generation Unit

- Global Timers (GT)
 - Two independent units
 - Two operating modes (Free-Running Timer and Reload Timer)
 - 24-bit data width
 - f_{GPTA} maximum resolution
 - $f_{\text{GPTA}}/2$ maximum input signal frequency
- Global Timer Cell (GTC)
 - 32 units related to the Global Timers
 - Two operating modes (Capture, Compare and Capture after Compare)
 - 24-bit data width
 - f_{GPTA} maximum resolution
 - $f_{\text{GPTA}}/2$ maximum input signal frequency
- Local Timer Cell (LTC)
 - 64 independent units
 - Three basic operating modes (Timer, Capture and Compare) for 63 units
 - Special compare modes for one unit
 - 16-bit data width
 - f_{GPTA} maximum resolution
 - $f_{\text{GPTA}}/2$ maximum input signal frequency

Interrupt Sharing Unit

- 286 interrupt sources, generating up to 92 service requests

General Purpose Timer Array (GPTA)

I/O Sharing Unit

- Interconnecting inputs and outputs from internal clocks, FPC, GTC, LTC, ports, and MSC interface

24.1.2 Functionality of LTCA2

- 64 Local Timer Cells (LTCs)
 - Three basic operating modes (Timer, Capture and Compare) for 63 units
 - Special compare modes for one unit
 - 16-bit data width
 - f_{GPTA} maximum resolution
 - $f_{\text{GPTA}}/2$ maximum input signal frequency

General Purpose Timer Array (GPTA)

24.2 GPTA0/GPTA1 Kernel Description

The functionality of the General Purpose Timer Arrays GPTA0/GPTA1 kernel is described in this section. Clock control, address decoding, and service (interrupt) request control are managed outside the GPTA0/GPTA1 module kernel.

Figure 24-2 shows a global block diagram of the GPTA module kernel.

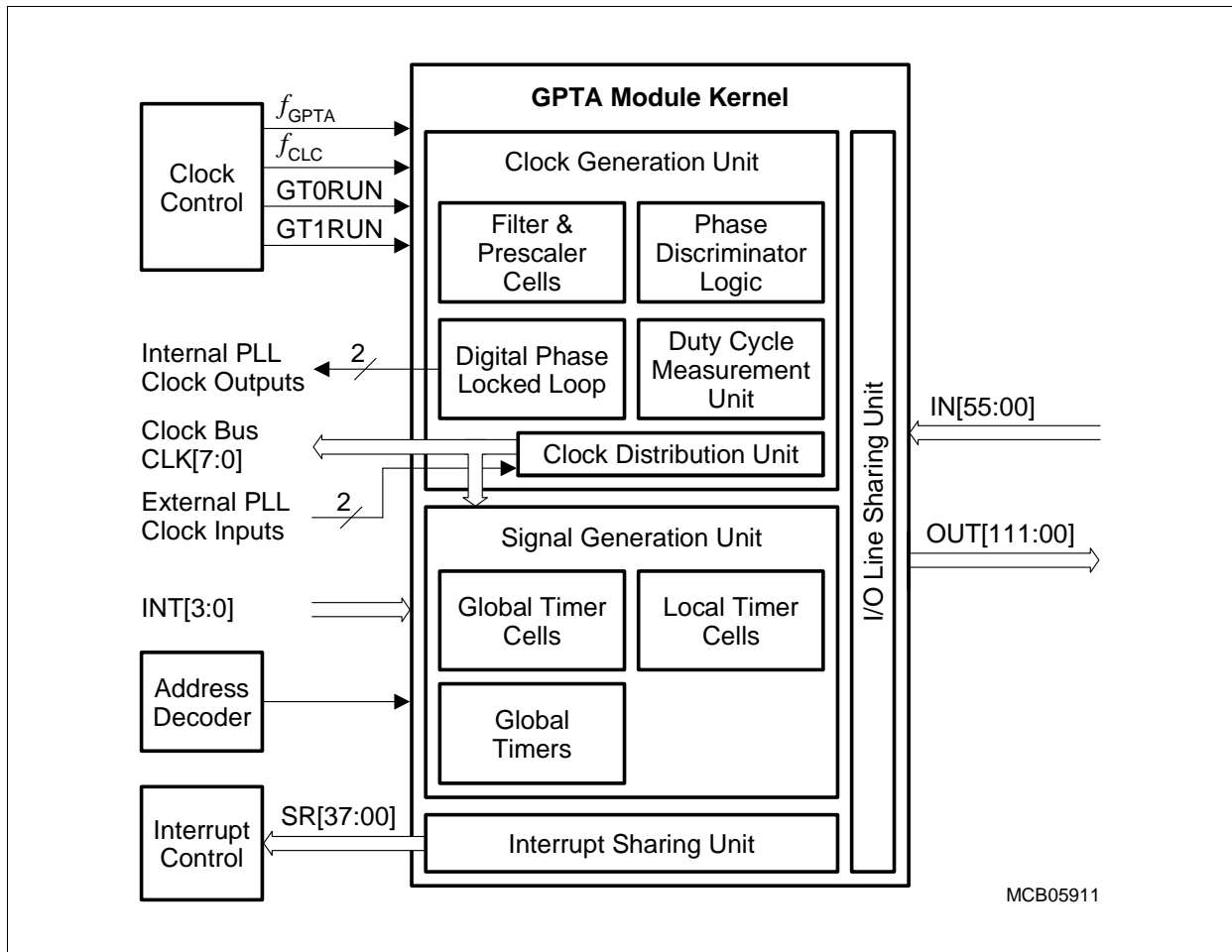


Figure 24-2 Block Diagram of GPTA Kernel

Each GPTA0/1 kernel has 56 input signals, 112 output signals, and four input signals, that can be connected to port pins or other on-chip logic units (see “GPTA Module Implementation” on Page 24-247 for the TC1796 specific interconnections). Further, several clock input and output signals are provided.

General Purpose Timer Array (GPTA)

24.2.1 GTPA Units

Each of the General Purpose Timer Arrays GPTA0 and GPTA1 ([Figure 24-2](#)) is split into a Clock Generation Unit (CGU) and a Signal Generation Unit (SGU):

- The **Clock Generation Unit** (see [Page 24-8](#)) allows a preprocessing of the input signals using filter, timer, capture, compare and enhanced digital PLL Modules:
 - The **Filter and Prescaler Cells** (FPC) provide input noise filtering (Immediate Debounce and Delayed Debounce) and may also work as prescalers for the GPTA module clock and external signals.
 - The **Phase Discrimination Logic** (PDL) may take the outputs of the FPCs to decode phase encoded signals from a position and rotation direction sensor system.
 - The **Duty Cycle Measurement Cells** (DCM) provide signal measurement capabilities (timer plus capture register, single and double capture on rising and falling edges or both) as well as missing pulse detection/reconstruction functions.
 - The **Digital Phase Locked Loop** (Digital PLL) is intended to generate a higher resolution clock from the values measured by DCM cells. Any arbitrary multiplication factor between 1 and 65535 is supported and may be changed from input clock period to input clock period.
 - The **Clock Distribution Unit** (CDU) provides all LTCs and GTs with a variety of different clock signals. It is equipped with GPTA module clock prescalers and multiplexers supporting alternate clock sources.

The original signals and all outputs of the preprocessing units are distributed to the Global Timers and LTCs via the clock bus.

- The **Signal Generation Unit** (see [Page 24-37](#)) provides a set of timers, capture and compare units:
 - The two 24-bit **Global Timers** (GT) can be individually configured as free-running counters or as reload counters starting at a programmable value from 0_H to $FFFFFF_H$. Each GT is equipped with a scalable greater-or-equal comparator; the number of bits to be compared is selectable.
 - The **Global Timer Cell** registers (GTC) are 24-bit wide. GTCs may be used as comparators (modifying the logical state of a related output port pin), or as capture units, storing the current GT0 or GT1 value on rising, falling or both signal edges detected on a related input port pin. Several adjacent GTCs may be connected to logical units operating on the same pin, allowing complex functions to be implemented.
 - The **Local Timer Cell** registers (LTC) are 16-bit wide. 63 LTCs can be configured to operate in one of four different modes: free-running or resettable counter, capture or compare unit. Adjacent cells can be combined to operate on the same pin, thus generating complex waveforms. One LTC (LTC63) can be used for special compare modes.

General Purpose Timer Array (GPTA)

24.2.2 Clock Generation Unit

As described in detail in the following sections, the Clock Generation Unit (CGU) provides five signal pre-processing modules:

- Filter and Prescaler Cell (FPC)
- Phase Discrimination Logic (PDL)
- Duty Cycle Measurement Unit (DCM)
- Digital Phase Locked Loop Cell (PLL)
- Clock Distribution Module (CDU)

The **Filter and Prescaler Cells** (FPC) provide input noise filtering using a debounce filter. FPCs are also able to operate as a prescaler for the GPTA module clock and external signals. Each FPC can select among different data and clock input signals.

The **Phase Discrimination Logic** (PDL) is able to decode FPC debounce filtered and phase encoded signals coming from a position and rotation direction sensor system. In the PDL, phase encoding can be bypassed.

The **Duty Cycle Measurement Units** (DCM) provide signal measurement capabilities (timer plus capture register, single and double capture on rising and falling edges or both) as well as missing pulse detection/reconstruction functions.

The **Digital Phase Locked Loop** (PLL) is intended to generate a higher resolution clock out of the values measured by DCM cells. Any arbitrary multiplication factor between 1 and 65535 is supported and may be changed from input clock period to input clock period.

The **Clock Distribution Unit** (CDU) provides all Local and Global Timer Cells with a variety of different clock signals. It is equipped with GPTA module clock prescalers and multiplexers supporting alternate clock sources.

Figure 24-3 shows how the five modules of the CGU are interconnected. The external interface signals of the CGU are:

- GPTA module clock f_{GPTA}
- GPTA module input signals (connected to the FPCs)
- Clock bus outputs (generated by the CDU)
- PDL bus outputs
- External PLL clock inputs (fed into CDU)

General Purpose Timer Array (GPTA)

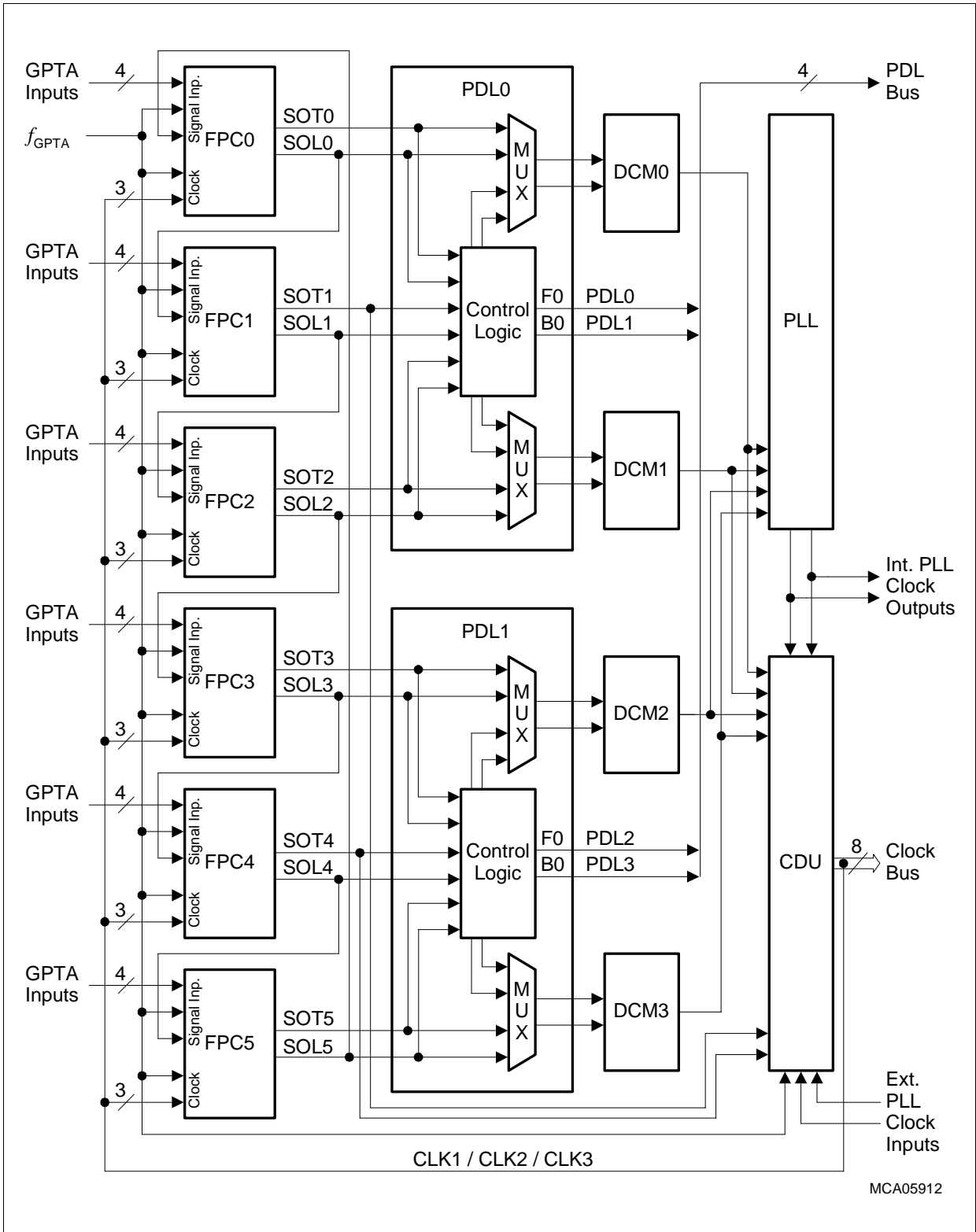


Figure 24-3 Interconnections in the CGU Module

General Purpose Timer Array (GPTA)

24.2.2.1 Filter and Prescaler Cell (FPC)

Each GPTA contains six filter and prescaler cells, FPC0 to FPC5. As shown in **Figure 24-4**, each FPC is equipped with an signal input multiplexer, a clock multiplexer, an edge detection unit, a 16-bit timer, a 16-bit compare register, a 16-bit comparator, and an FPC control unit (see also **Page 24-115** for the FPC functional algorithm description).

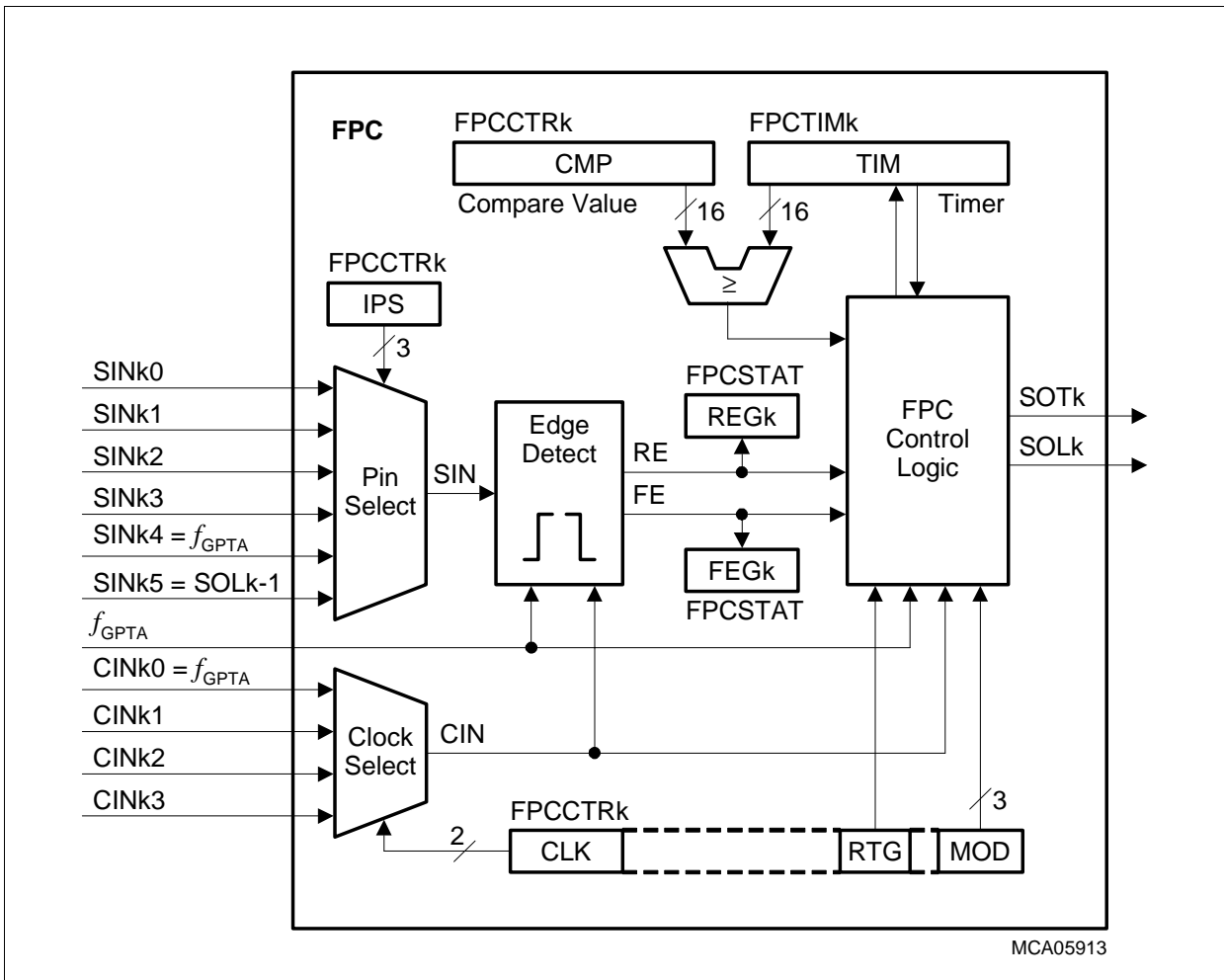


Figure 24-4 Filter and Prescaler Cell Architecture

FPC Registers

The following registers are assigned to the filter and prescaler cells FPCk (k = 0-5):

- FPCSTAT = Filter and Prescaler Cell Status Register (see **Page 24-155**)
- FPCCTRk = Filter and Prescaler Cell Control Register k (see **Page 24-156**)
- FPCTIMk = Filter and Prescaler Cell Timer Register k (see **Page 24-158**)

General Purpose Timer Array (GPTA)

FPC Operating Modes

Each filter and prescaler cell can be individually configured to operate in one of the following operating modes:

- Delayed Debounce Filter Mode on both edges
- Immediate Debounce Filter Mode on both edges
- Rising edge: Immediate Debounce Filter Mode, falling edge: No filtering
- Rising edge: No filtering, falling edge: Immediate Debounce Filter Mode
- Rising edge: Delayed Debounce Filter Mode, falling edge: Immediate Debounce Filter Mode
- Rising edge: Immediate Debounce Filter Mode, falling edge: Delayed Debounce Filter Mode
- Prescaler Mode (triggered on rising edge)
- Prescaler Mode (triggered on falling edge)

The operation mode is selected by bit field FPCCTRk.MOD ([Page 24-156](#)).

FPC Input Signals

Bit field FPCCTRk.IPS (see [Page 24-156](#)) selects one of the following inputs for FPCk:

- Signal input 0 (SINK0)
- Signal input 1 (SINK1)
- Signal input 2 (SINK2)
- Signal input 3 (SINK3)
- GPTA module clock f_{GPTA} (SINK4)
- Preceding FPC level output signal SOLk-1 (SIN05 is connected to SOL5)

When the preceding FPC level output signal is selected as input, two or more FPCs may be concatenated; for example, to combine a delayed debounce filter and an immediate debounce filter.

The maximum FPC input signal frequency must be less than or equal to the sampling rate ($f_{GPTA}/2$). The assignment of GPTA I/O line and FPC signal inputs SINK is defined in [“FPC Input Line Selection” on Page 24-94](#).

FPC Filter Clocks

Bit field FPCCTRk.CLK (see [Page 24-157](#)) selects one of four filter clocks for FPCk:

- Clock input line 0 (CINK0) = GPTA module clock f_{GPTA}
- Clock input line 1 (CINK1) = local PLL clock,
- Clock input line 2 (CINK2) = (prescaled) GPTA module clock f_{GPTA} or PLL clock from other unit or DCM 3 clock
- Clock input line 3 (CINK3) = DCM 2 clock or PLL clock of other unit or uncompensated PLL clock or uncompensated PLL clock of other unit

When using a PLL clock for the FPC, no software is needed to adapt the FPC filter to changing speed for angle-based input signals. The standard PLL clock can be either the

General Purpose Timer Array (GPTA)

compensated or uncompensated PLL clock or can be the PLL clock of the other GPTA module. The uncompensated PLL clock is useful in applications in which bursts (due to acceleration) might disturb the filter function.

With a prescaled GPTA module clock, very long time filter time periods can be achieved.

Output Signal Splitting

Two output lines are provided by each FPC module as follows:

- An trigger output signal SOTk, reporting a falling or rising signal edge on the FPC input by a single f_{GPTA} clock pulse,
- A level output signal SOLk, indicating the direction of the detected signal transition.

This signal-splitting scheme (pair of trigger and level output) provides subsequent PDL and DCM cells with the information about an input signal transition in the same f_{GPTA} clock cycle. This feature avoids cascading a one clock delay per edge detection unit implemented at the input of each subsequent cell. **Figure 24-5** shows the FPC output signal splitting scheme.

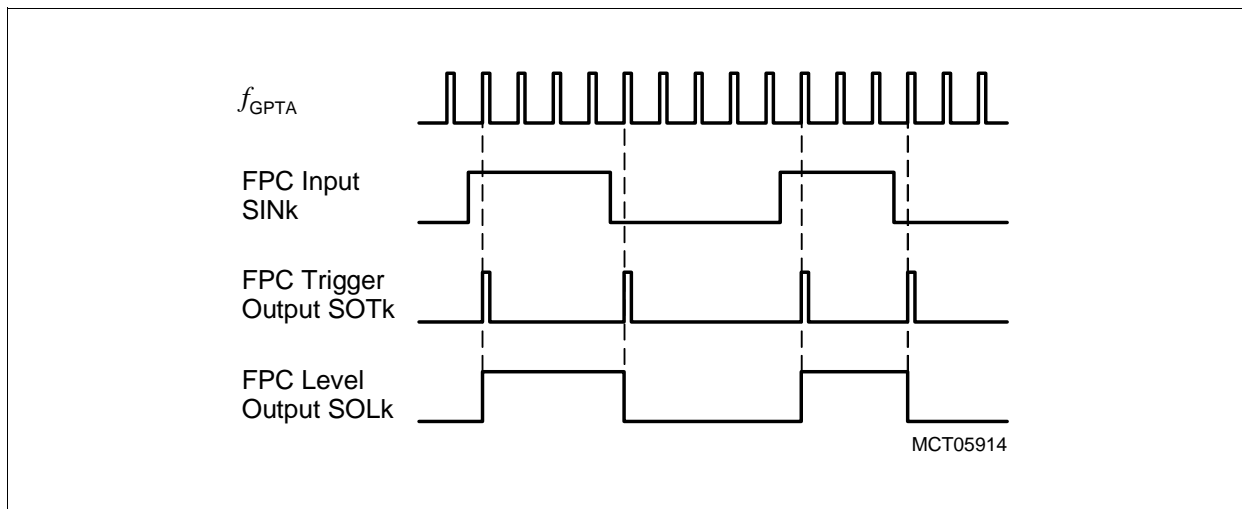


Figure 24-5 FPC Output Splitting into Trigger and Level Information

General Purpose Timer Array (GPTA)

Delayed Debounce Filter Mode

In Delayed Debounce Filter Mode, the signal input SIN is filtered from all signal transitions and glitches with a width smaller than the selected clock period length multiplied by the compare register value.

The input signal SIN (sampled with f_{GPTA}) is analyzed at the selected filter clock rate of CIN. If the state of the input sample differs from the current output signal value, the 16-bit timer is incremented by one. When the timer register FPCTIMk is not in its idle state (0000_H) **and** the state of the input sample matches the current output signal value, the 16-bit timer is decremented by one (see [Figure 24-6](#)); if bit FPCCTRk.RTG is set, the timer will be set to idle state again (see [Figure 24-7](#)). A rising or falling edge, occurring on the signal input line SIN when the timer is greater than zero but less than the compare value, sets the corresponding glitch flag FPCSTAT.REG (on rising edge glitch) or FPCSTAT.FEGk (on falling edge glitch). When the timer matches the 16-bit compare value stored in FPCCTRk.CMP (timer threshold), the level output signal line SOLk is inverted, a GPTA module clock pulse is generated at the trigger output signal SOTk, and the timer is reset to 0000_H. The rising/falling edge glitch flags must be cleared by software.

The filter is by-passed if the compare value FPCCTRk.CMP is programmed to zero (0000_H). In this case, the input signal is directly copied to the output signal.

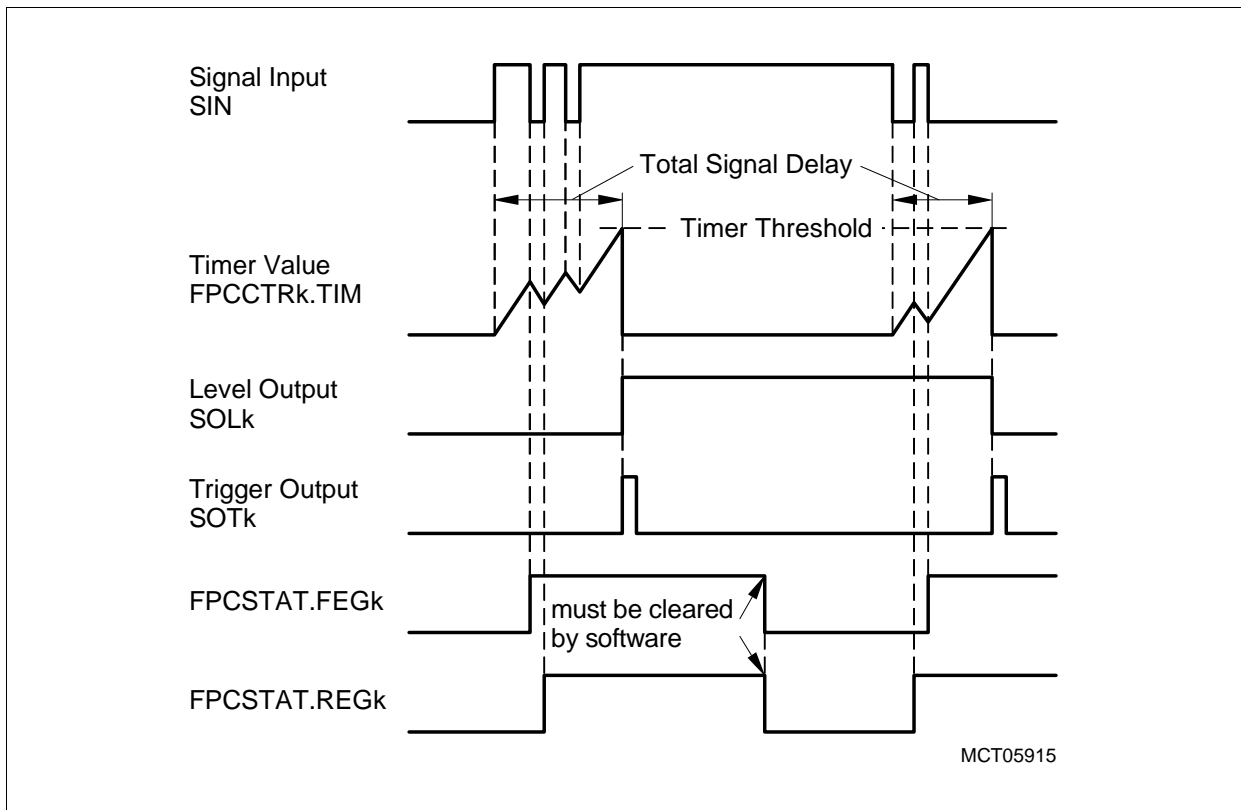


Figure 24-6 FPC Delayed Debounce Filter Algorithm with Timer Decrement

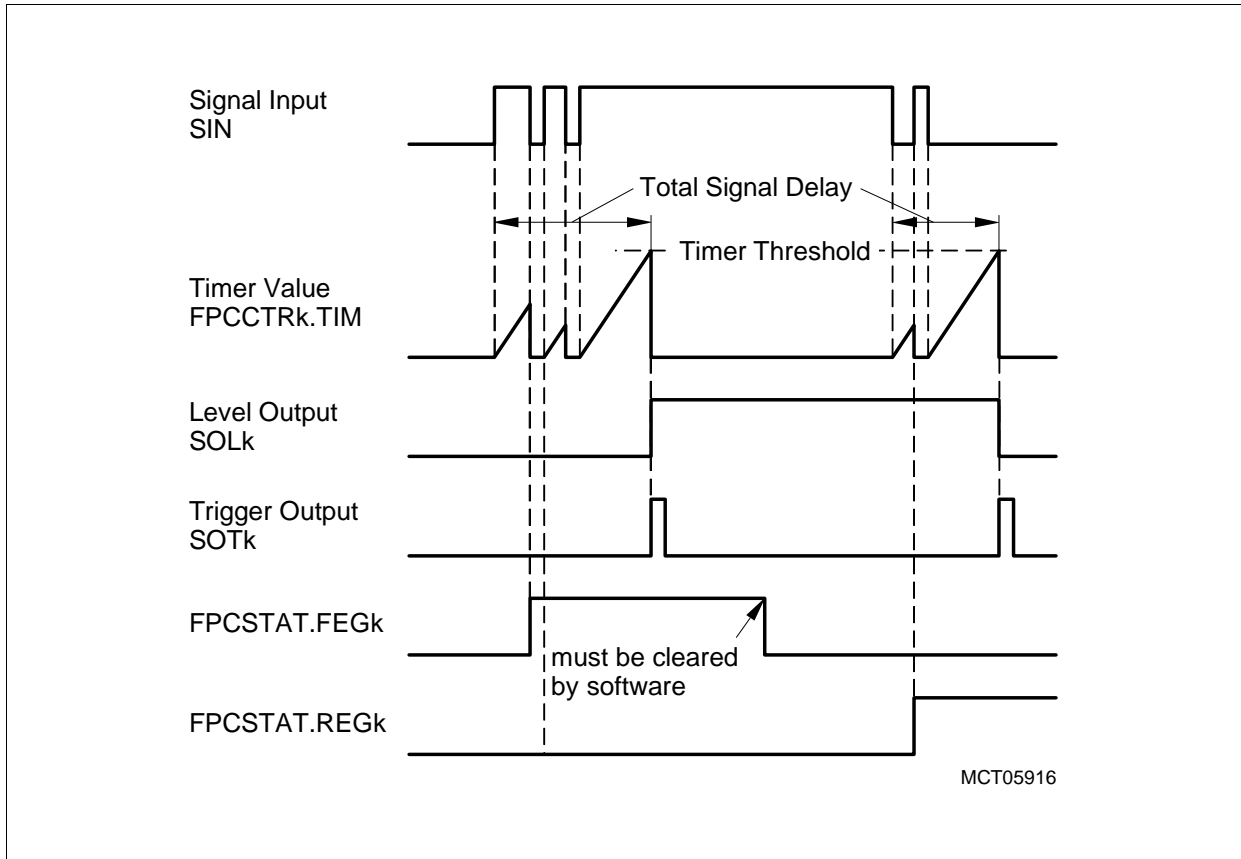


Figure 24-7 FPC Delayed Debounce Filter Algorithm with Timer Reset

The total signal delay from input to output depends on the programmed compare register value, the number of high-frequency pulses (glitches) during the filter operating time, and the timer behavior in case of a glitch (decrement or reset).

The FPC Delayed Debounce Filter Mode is selected by:

- FPCCTRk.MOD = 000_B

General Purpose Timer Array (GPTA)

Immediate Debounce Filter Mode

In Immediate Debounce Filter Mode, the input signal is filtered from signal transitions and glitches arriving a programmable time after an input signal edge detection (see [Figure 24-8](#)).

The input signal SIN is sampled with f_{GPTA} and the glitch edge detection is also performed with f_{GPTA} . The further analysis (e.g. filter timer increment) is done at the selected filter clock rate of CIN. As long as the timer is reset, the FPC control unit copies the sampled input value directly to the level output signal line SOLk. When a rising or falling edge occurs on the signal input line SIN and the 16-bit compare value FPCCTRk.CMP is not zero, the timer is enabled to be incremented by the selected clock and the copy mechanism is disabled. When the timer value FPCTIMk.TIM matches the compare value FPCCTRk.CMP, the timer is reset and the copy mechanism is enabled again. A rising or falling edge, occurring on SIN while the timer is greater than zero but less than the compare value, sets the corresponding glitch flag FPCSTAT.REG (on rising edge glitch) or FPCSTAT.FEGk (on falling edge glitch). The rising/falling edge glitch flags must be cleared by software.

The filter is bypassed if the compare value FPCCTRk.CMP is programmed to zero (0000_H). In this case, the input signal is directly copied to the output signal without any disable periods.

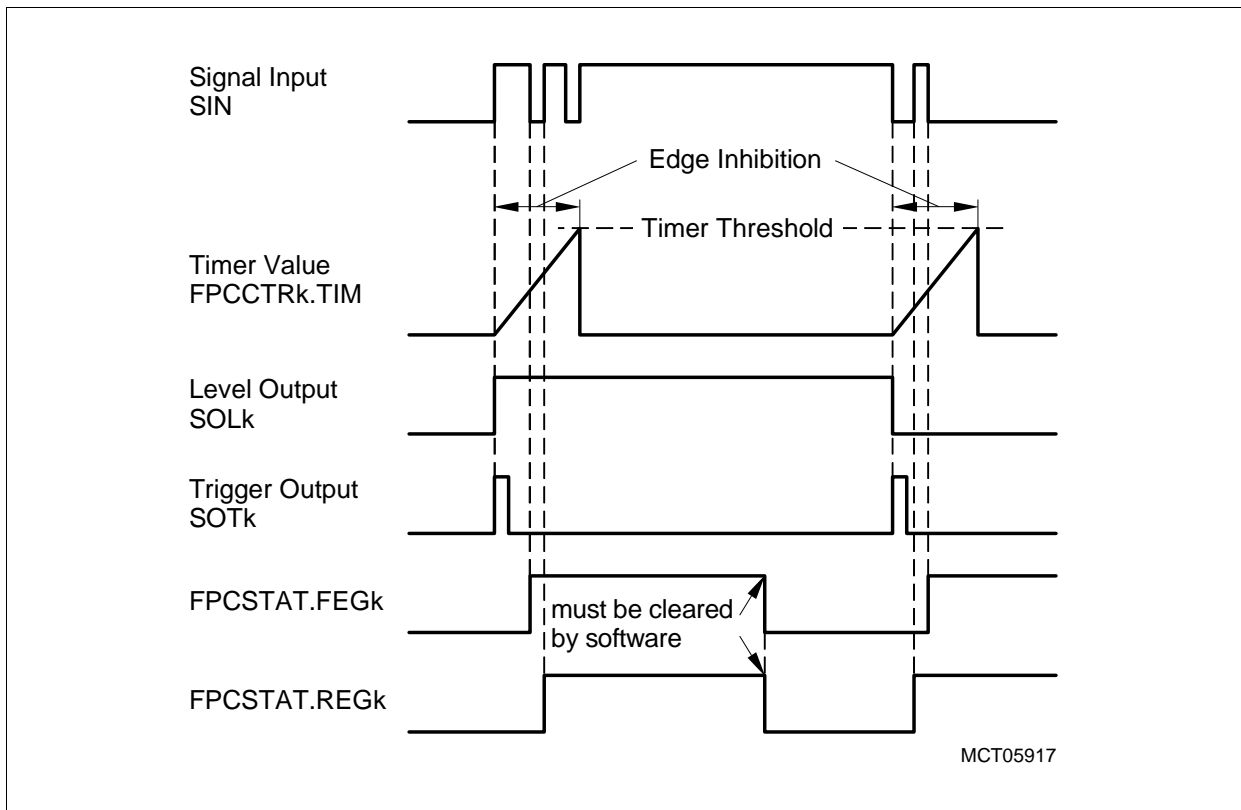


Figure 24-8 FPC Immediate Debounce Filter Algorithm on Both Edges

General Purpose Timer Array (GPTA)

Note: During the last clock cycle of edge inhibition time (where timer value is equal to the compare value) an input signal glitch will be filtered but the corresponding glitch status flag in register FPCSTAT is not set.

The Immediate Debounce Filter can be enabled only for one edge, either rising or falling. In this case, the signal output follows the signal input value immediately after the timer threshold of the filtered edge is reached, without re-starting the timer (see [Figure 24-9](#)).

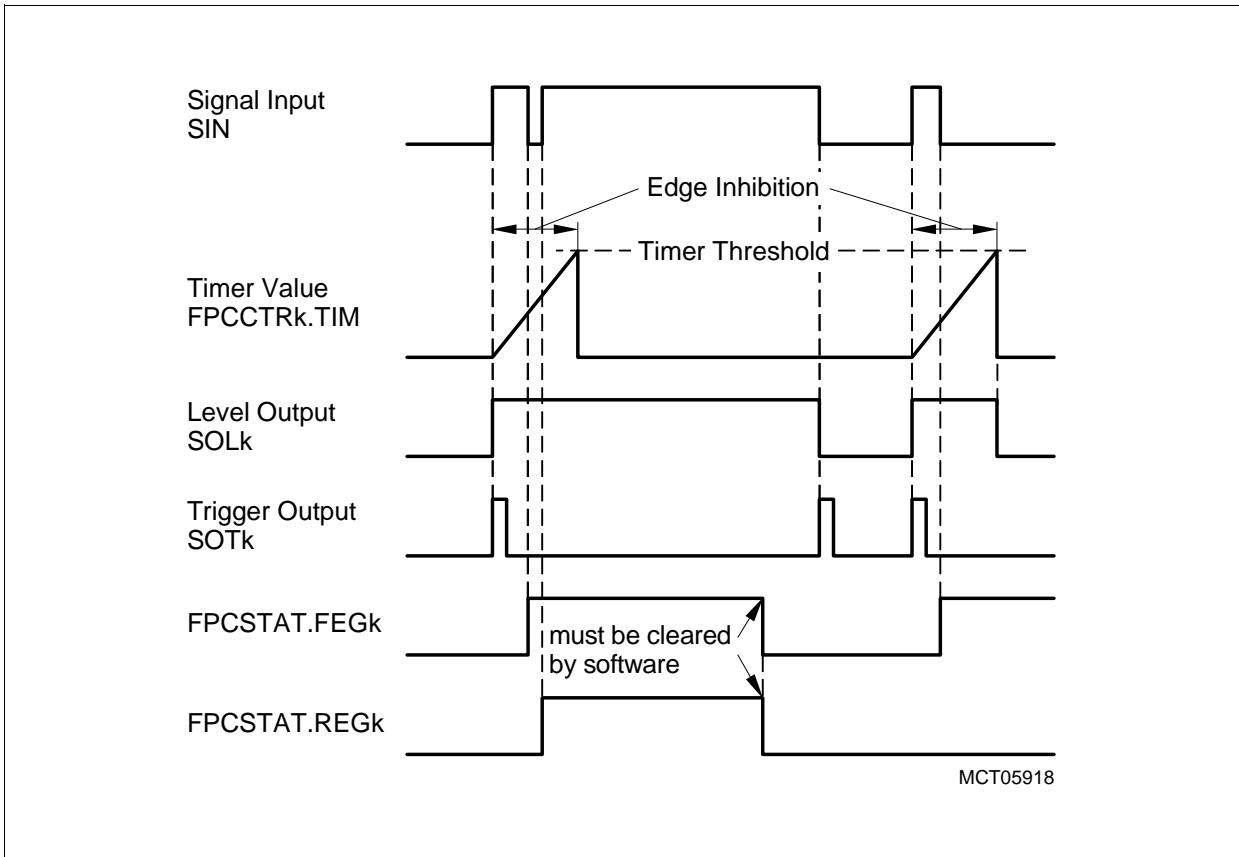


Figure 24-9 FPC Immediate Debounce Filter Algorithm on Rising Edge only

The FPC Immediate Debounce Filter Modes are selected by:

- FPCCTRk.MOD = 001_B: Immediate Debounce Filter Mode on both edges
- FPCCTRk.MOD = 010_B: Immediate Debounce Filter Mode on rising edge only, no filtering on falling edge.
- FPCCTRk.MOD = 011_B: Immediate Debounce Filter Mode on falling edge only, no filtering on rising edge.

General Purpose Timer Array (GPTA)

Mixed Filter Modes

In the Mixed Filter Modes, one edge of a signal is filtered in the Delayed Debounce Mode, and the other edge is filtered in the Immediate Debounce Mode. The Debounce Mode is switched when the timer threshold is reached. Note that both filter modes use the same timer threshold in this case (see [Figure 24-10](#), demonstrating Delayed Debounce Mode with Timer Decrement on Rising Edge and Immediate Debounce of on Falling Edge).

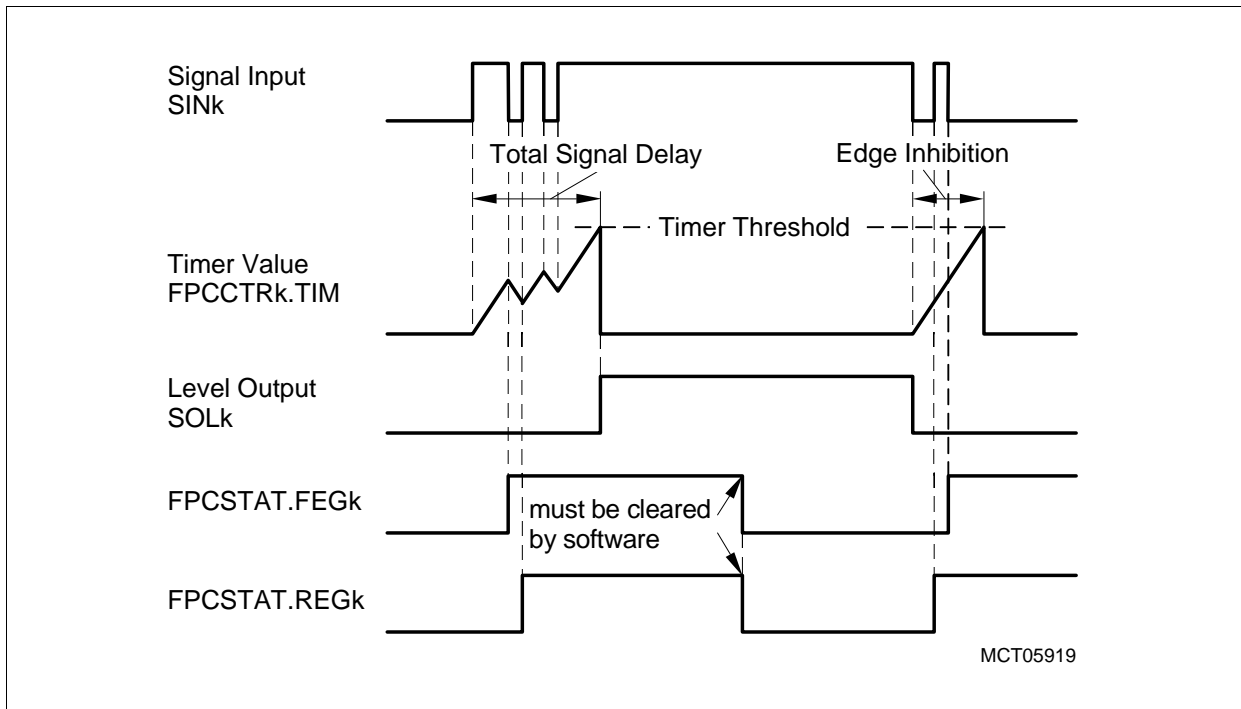


Figure 24-10 FPC Mixed Filter Algorithm

The FPC Mixed Filter Modes are selected by:

- FPCCTRk.MOD = 100_B: Delayed Debounce Filter Mode on rising edge
Immediate Debounce Filter Mode on falling edge
- FPCCTRk.MOD = 101_B: Immediate Debounce Filter Mode on rising edge
Delayed Debounce Filter Mode on falling edge

General Purpose Timer Array (GPTA)

Prescaler Mode

In Prescaler Mode, the input signal is sampled and analyzed with f_{GPTA} . The FPC control unit counts each rising (or falling) edge of the input signal. When the timer value matches the compare value:

- one GPTA module clock pulse is generated at the trigger output signal SOTk and level output signal SOLk
- the timer FPCTIMk.TIM is reset to 0000_H

Figure 24-11 shows a divide-by-6 operation using the FPC in Prescaler Mode with trigger on rising edge selected.

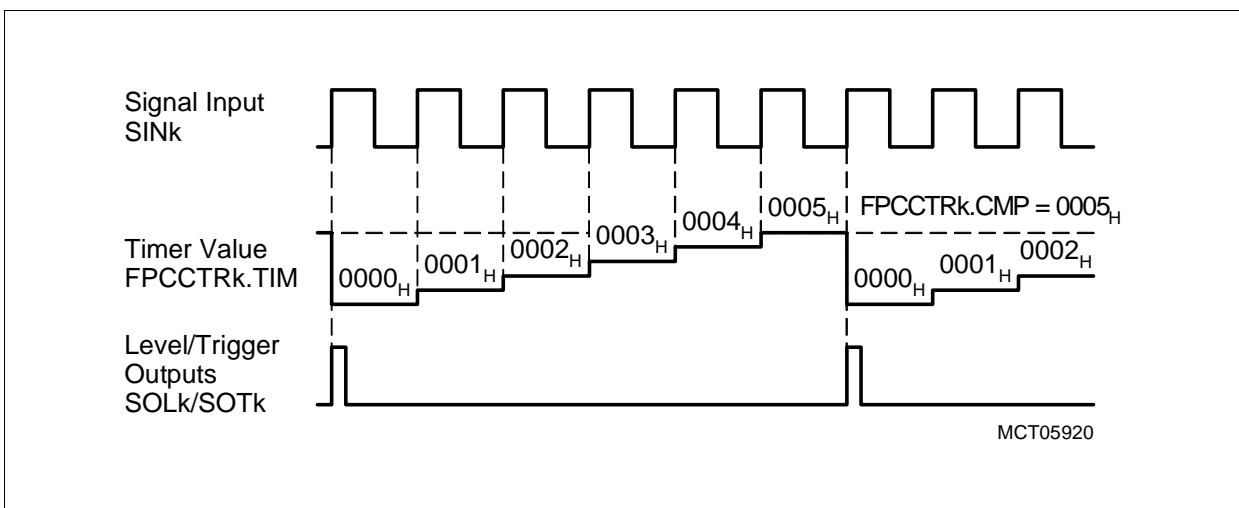


Figure 24-11 FPC Prescaler Mode

For a divide-by-n operation, the compare value FPCCTRk.CMP must be set to n - 1.

The FPC Prescaler Modes are selected by:

- FPCCTRk.MOD = 110_B: Prescaler Mode triggered on rising edge
- FPCCTRk.MOD = 111_B: Prescaler Mode triggered on falling edge

General Purpose Timer Array (GPTA)

24.2.2.2 Phase Discrimination Logic (PDL)

The GPTA provides two Phase Discrimination Logic modules (PDL0, PDL1) driven by two signal lines coming from an FPC cell (for description, see [Page 24-12](#)):

- An event input signal
- A level input signal

Both Phase Discrimination Logic modules are controlled by the Phase Discrimination Logic Control Register PDLCTR (see [Page 24-159](#)).

Each PDL is equipped with an edge detection unit, a phase detection unit, a PDL control unit, and an output multiplexer. Six output lines are provided by each PDL Module:

- A forward output signal (F0, F1) is driven by one f_{GPTA} clock pulse if an input signal edge is recognized as forward rotation. These signals can be connected to any Local Timer Cell via the PDL bus.
- A backward output signal (B0, B1) is driven by one f_{GPTA} clock pulse if an input signal edge is recognized as backward rotation. This signal can be connected to any Local Timer Cell via the PDL bus.
- Two pairs of output signals, carrying the bypassed input level and event information from the driving FPC cells or the angular velocity and error information provided by the PDL function. These output lines are directly connected to the adjacent Duty Cycle Measurement Cells, DCM0/DCM1(for PDL0) and DCM2/DCM3 (for PDL1).

The PDL processes the output signal of a 2-sensor or 3-sensor positioning system. With bit PDLCTR.TSEx = 1, a 3-sensor system execution is selected providing the DCM1 and/or DCM3 cell with information concerning erroneous states in the signal input. When PDLCTR.TSEx = 0, a 2-sensor system is selected and DCM1 and/or DCM3 are supplied with the input event and level information from the driving FPC2 and/or FPC5.

The rotation direction, monitored by the connected sensors, is automatically derived from the sequence in which the input signals change. Each edge detected on an input signal line generates a pulse on the F0, F1 forward output lines or on the B0, B1 backward output lines. Input jitter, which might occur if a sensor rests near to one of its switching points, is compensated.

If bit PDLCTR.MUXx = 1, the trigger output signal to DCM0/DCM2 (angular velocity information) is driven by a boolean 'OR' operation of the corresponding forward trigger and backward trigger signal while the level output signal at DCM0/DCM2 is at fixed high level. In this case, every pulse at F0/B0 and F1/B1 generates a rising edge at the DCM0/DCM trigger signal.

If bit PDLCTR.MUXx = 0, the associated DCM0/DCM2 signals are directly connected with the input event and level signals from the driving FPC0/FPC3.

To calculate the sensor's current position, the associated LTCs should be clocked with the PDL forward and backward output pulses. A software operation, subtracting the backward counter contents from the forward counter contents, provides the absolute

General Purpose Timer Array (GPTA)

position. Dynamic information (speed, acceleration, deceleration) may be obtained by analyzing the angular velocity signal periods with the associated DCM cell.

The maximum input frequency is $f_{GPTA}/4$ for a 2-sensor positioning system and $f_{GPTA}/6$ for a 3-sensor positioning system. To ensure that a transition of any input signal is correctly recognized, its level should be held high or low for at least two f_{GPTA} cycles before it changes (three f_{GPTA} cycles for a 3-sensor positioning system).

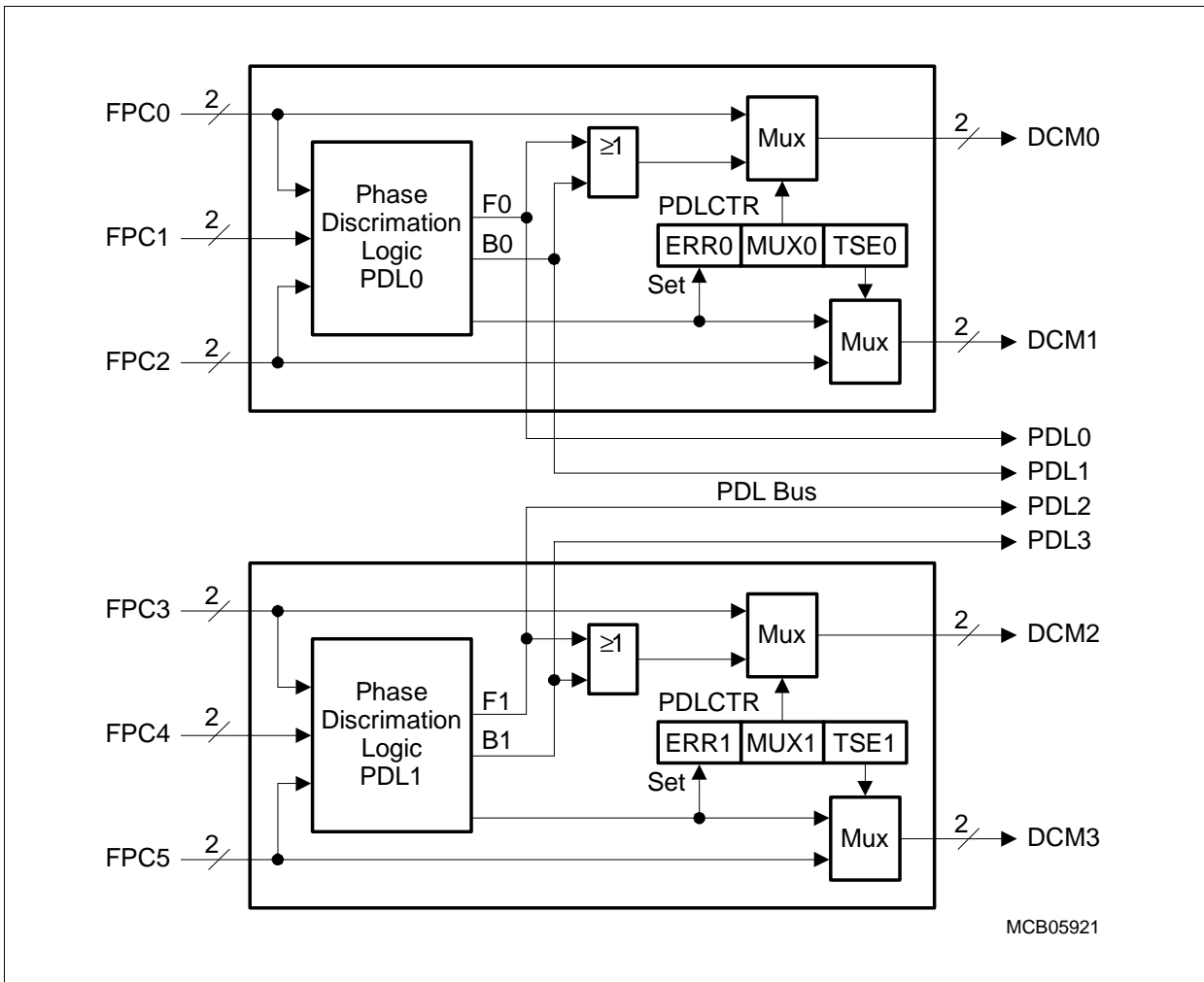


Figure 24-12 Block Diagram of Phase Discrimination Logic Unit

General Purpose Timer Array (GPTA)

Positioning System With Two Sensors

The 2-sensor Mode is enabled when bit PDLCTR.TSEx is cleared. The sensors are mounted at a 90° angle to each other (see [Figure 24-13](#)). The third sensor input of the PDL module is internally disabled and DCM1/DCM3 cell inputs are driven by fed-through FRC2/FPC5 output lines.

This configuration can measure an absolute position with a resolution of 90°. No error conditions can be detected.

!	Means not
Re	Means rising edge
Fe	Means falling edge
Forward	$ReS1*!S2 + S1*ReS2 + FeS1*S2 + !S1*FeS2$
Backward	$ReS1*S2 + !S1*ReS2 + FeS1*!S2 + S1*FeS2$
Position	Forward_Counter - Backward_Counter

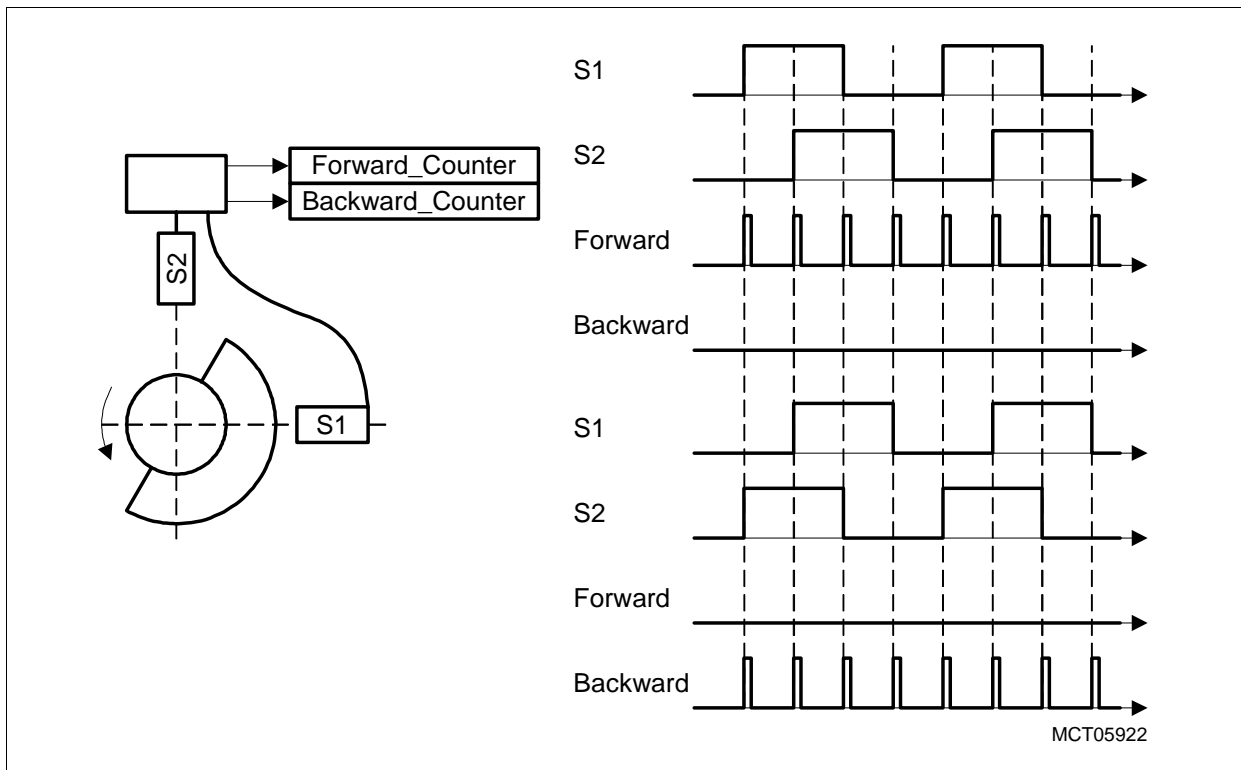


Figure 24-13 Interface Signals of a PDL in a 2-Sensor Positioning System

[Figure 24-14](#) illustrates how the output signals of a 2-sensor system superimposed with noise are processed by the PDL unit. Jitter pulses are completely compensated if they do not occur on both signal lines simultaneously.

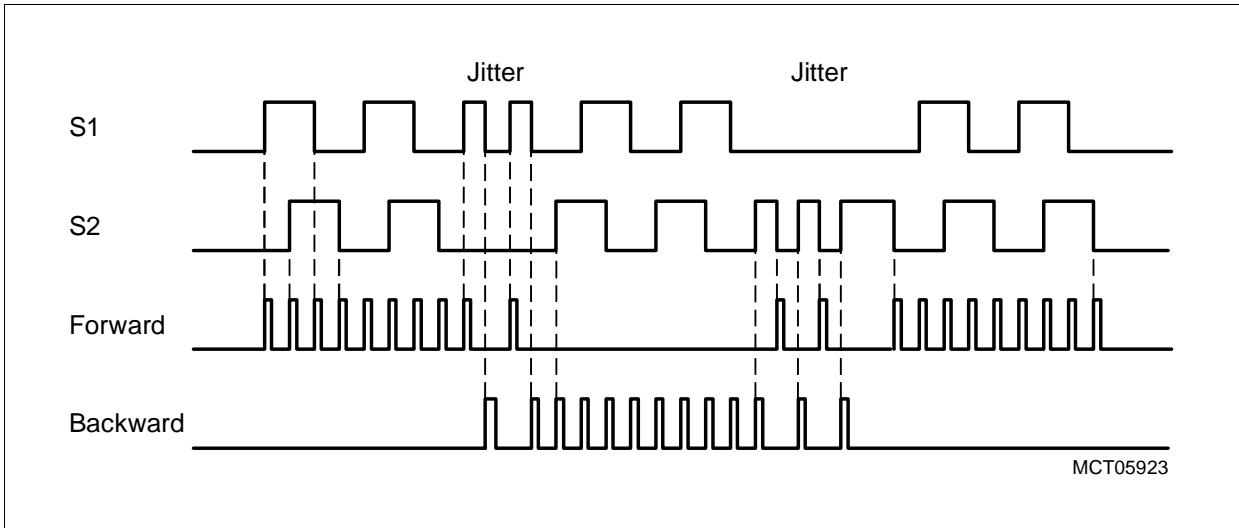


Figure 24-14 Compensation of Input Jitter

Positioning System with Three Sensors

The 3-sensor Mode is enabled when bit PDLCTR.TSEx is set to 1. The sensors are mounted at an 120° angle to each other (see [Figure 24-15](#)). This configuration can measure an absolute position with a resolution of 60°.

Input signal combinations that are not allowed in a properly-working positioning system (all inputs low or all inputs high) cause the following to occur:

- An error signal is generated, driving the Duty Cycle Measurement cells DCM1 and/or DCM3,
- The error flag PDLCTR.ERRx is set,
- No forward or backward pulses are generated.

When the error disappears, the error signal will be cleared. The error flag PDLCTR.ERRx must be cleared by software.

!	Means not
Re	Means rising edge
Fe	Means falling edge
Forward	$ReS1*!S2*S3 + FeS3*S1*!S2 + ReS2*S1*!S3 + FeS1*S2*!S3 + ReS3*!S1*S2 + FeS2*!S1*S3$
Backward	$ReS1*S2*!S3 + FeS3*!S1*S2 + ReS2*!S1*S3 + FeS1*!S2*S3 + ReS3*S1*!S2 + FeS2*S1*!S3$
Error	The input signal states $S1*S2*S3$ and $!S1*!S2*!S3$ are not allowed
Position	Forward_Counter - Backward_Counter

General Purpose Timer Array (GPTA)

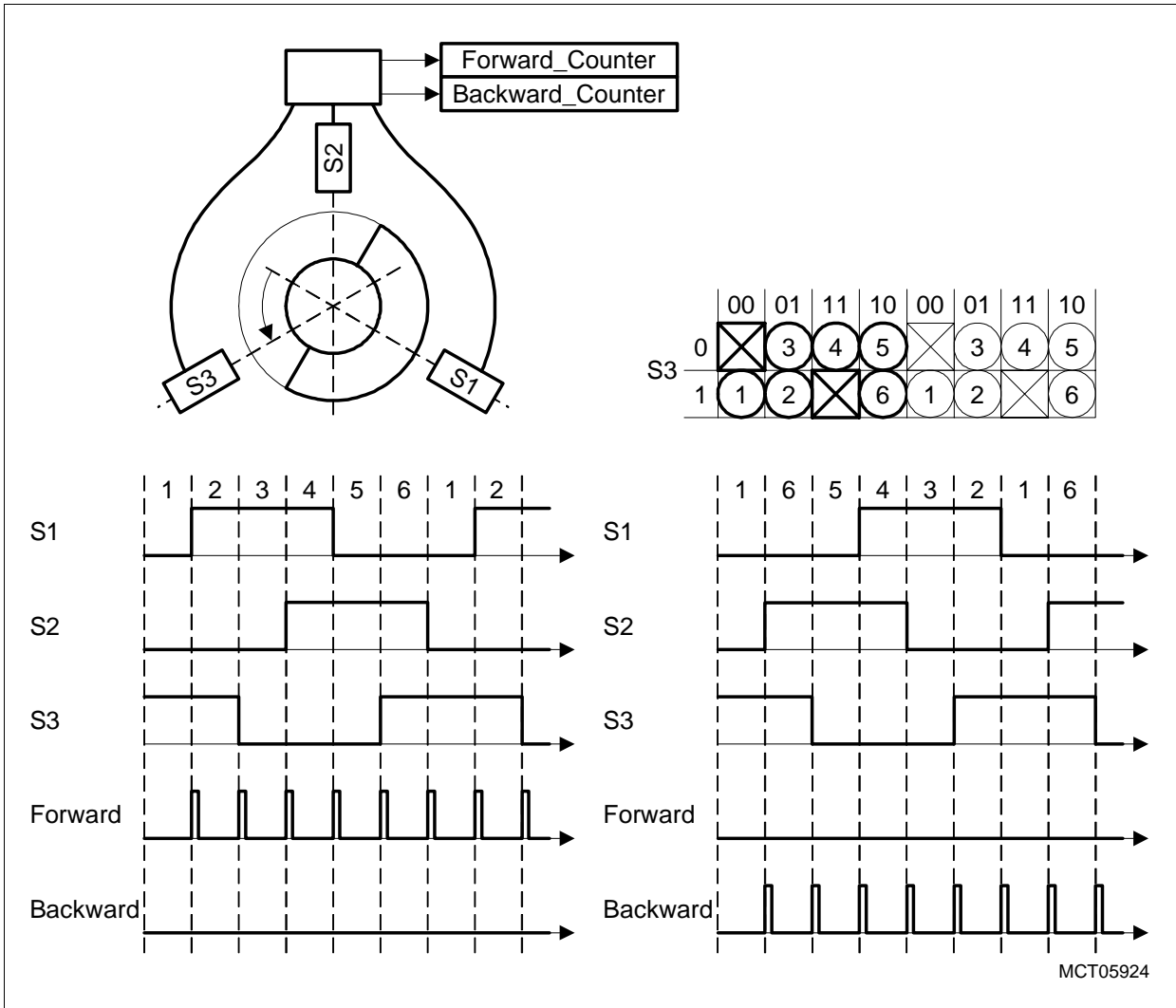


Figure 24-15 Interface Signals of a PDL in a 3-Sensor Positioning System

Jitter pulses are completely compensated as illustrated in [Figure 24-14](#).

General Purpose Timer Array (GPTA)

24.2.2.3 Duty Cycle Measurement Unit (DCM)

The GPTA contains four DCM units (DCM0 to DCM3). The input signal to be analyzed is delivered as a 2-line signal input (see [Figure 24-5](#) for the event/level input signal splitting scheme). It is built by:

- An event input, and
- A signal level input.

Each DCM unit has four outputs:

- An event output line,
- An interrupt output that can become active at a signal input rising edge,
- An interrupt output that can become active at a signal input falling edge,
- An interrupt output that can become active at a compare event.

Each DCM unit is equipped with a 24-bit timer, a 24-bit capture register, a 24-bit capture/compare register, a 24-bit comparator and a DCM control unit ([Figure 24-16](#)).

The following registers are assigned to the DCM units:

- DCMCTR_k = Duty Cycle Measurement Control Register k (see [Page 24-161](#))
- DCMTIM_k = Duty Cycle Measurement Timer Register k (see [Page 24-163](#))
- DCMCAV_k = Duty Cycle Measurement Capture Register k (see [Page 24-163](#))
- DCMCOV_k = Duty Cycle Measurement Capture/Compare Register k (also referred as “capcom”, see [Page 24-164](#))
- SRSC0 = Service Request State Clear Register 0 (see [Page 24-206](#))
- SRSS0 = Service Request State Set Register 0 (see [Page 24-208](#))

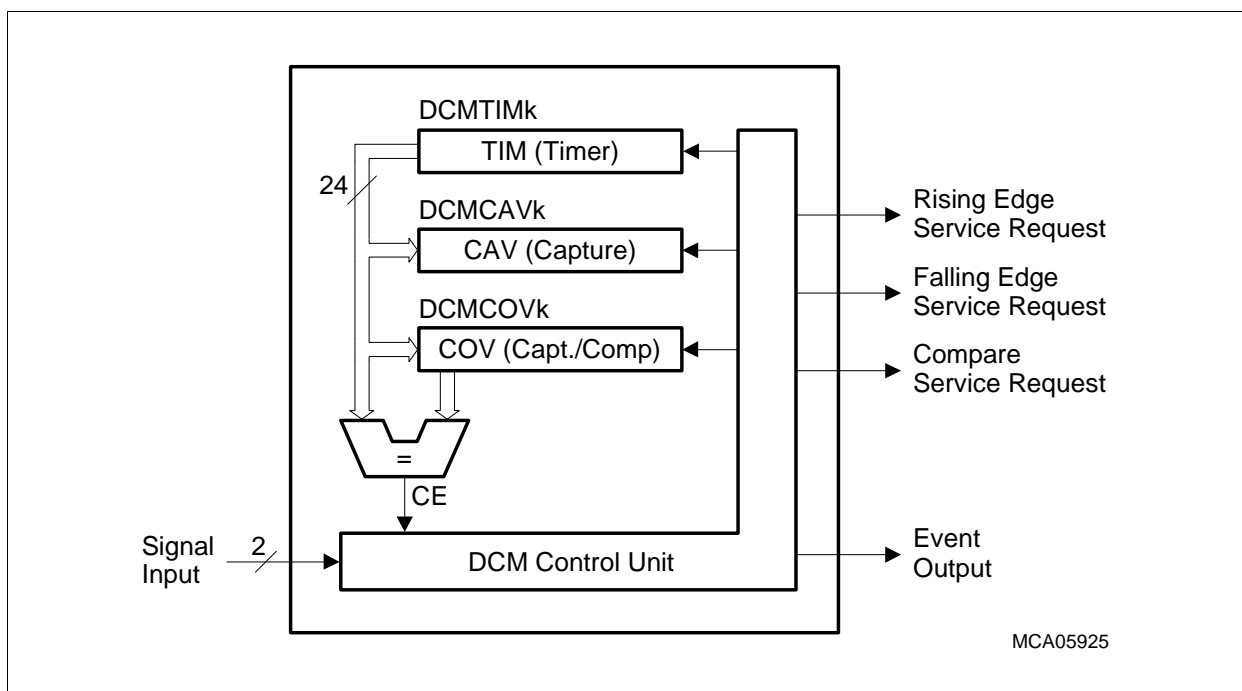


Figure 24-16 Block Diagram of a Duty Cycle Measurement Unit

General Purpose Timer Array (GPTA)

The DCM unit inputs are connected to the PDL outputs. Depending on the configuration of the associated PDL cell, the DCM units can also be driven by an FPC directly (as shown in [Figure 24-12](#)):

- DCM0 is driven by FPC0 or PDL0 angular velocity signal,
- DCM1 is driven by FPC2 or PDL0 error signal,
- DCM2 is driven by FPC3 or PDL1 angular velocity signal,
- DCM3 is driven by FPC5 or PDL1 error signal.

When the driving FPCs and PDL cells are programmed in feed-through mode, an external port pin signal as selected by the FPC input multiplexer can be directly processed by a DCM cell.

The duty cycle of the DCM unit signal input can be determined by measuring its period length and the width of its low or high state. For this purpose, several operations can be started on an signal input edge:

- **Reset Timer**

The local timer can be reset on rising, falling, or both edges of the signal input line as selected via control bits DCMCTRk.RZE (for rising edge) and DCMCTRk.FZE (for falling edge). After a reset timer event, the timer is continuously incremented by the GPTA module clock f_{GPTA} until the next reset condition occurs. If no reset timer event is enabled, the timer operates in Free-Running Timer Mode, repeatedly counting from its lower limit (000000_H) to its upper limit (FFFFFF_H).

- **Capture**

The current timer value is stored in the capture register DCMCAV on the rising edge (DCMCTR.RCA = 1) or falling edge (DCMCTRk.RCA = 0) of the signal input line. The current timer value is stored in the capture/compare register DCMCOV on the opposite signal edge as selected by DCMCTRk.RCA and if enabled by bit DCMCTRk.OCA = 1. With DCMCTRk.OCA = 0 the capture/compare register DCMCOV is not affected.

- **Edge Service Request and Interrupt Request**

On a rising input signal edge of the DCMk unit (k = 0-3) the service request flag SRS0.DCM0kR is set. Additionally, a service request signal is triggered if bit DCMCTRk.RRE = 1. A falling input signal edge sets the service request flag SRS0.DCM0kF. An interrupt request generation on this edge is triggered if bit DCMCTRk.FRE = 1. Both edges of the signal input line initiate an interrupt request when both bits, DCMCTRk.FRE and DCMCTRk.RRE, are set. The interrupt on signal input edges is disabled if both bits are cleared.

- **Hardware Generated Output Pulse**

A single f_{GPTA} clock pulse is generated on the DCM output line if enabled by control register bit DCMCTRk.RCK (rising edge at signal line) and/or DCMCTRk.FCK (falling edge at signal line) and an appropriate edge is detected at the input.

The 0% or 100% duty cycle exception (no edge or only one edge detected) can be handled by a **limit checking** option. The expected input signal's maximum period length (measured in f_{GPTA} clock ticks) can be loaded into the capture/compare

General Purpose Timer Array (GPTA)

register DCMCOV that is continuously compared with the timer value. When the timer is incremented up to the limit stored in capture/compare register, the service request flag SRS0.DCM0xC is set. If the compare service request is enabled (control register bit DCMCTRk.CRE = 1), an interrupt request is generated.

- **Software Generated Output Pulse**

If the **software** intends to compensate an input pulse backlog, bit DCMCTRk.QCK should be set to 1. This immediately triggers a single **clock pulse generation** on the DCM output signal line.

DCM Interrupt Control

Each DCM unit is able to generate three service request output signals. The service request outputs of a DCMk unit are controlled as shown in [Figure 24-17](#). When a service request condition occurs, the corresponding service request flag is always set. The service request output is activated only if it is enabled by the corresponding enable bit. Further details on service request and interrupt handling are provided in section [“Interrupt Sharing Unit \(IS\)” on Page 24-112](#).

General Purpose Timer Array (GPTA)

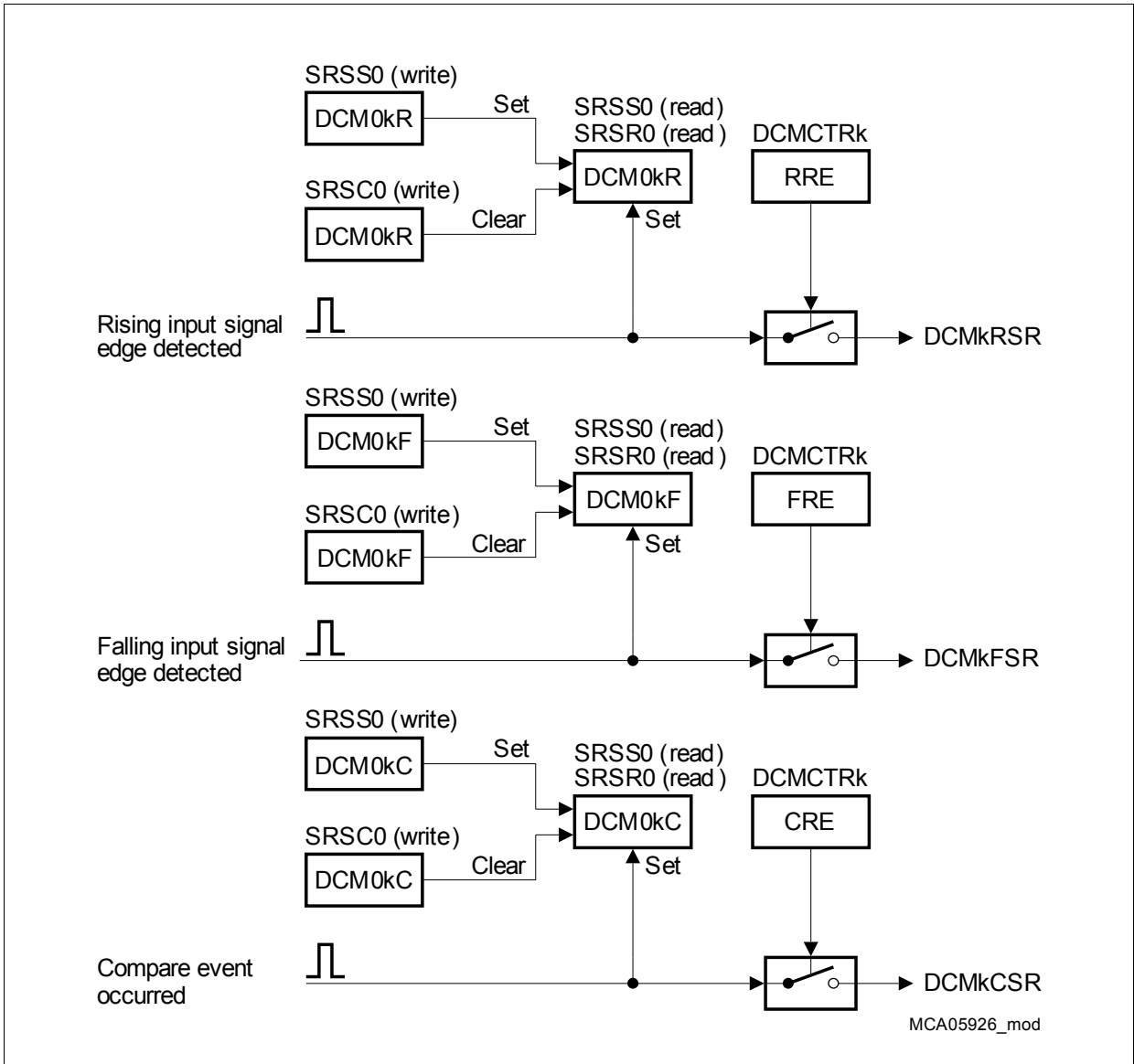


Figure 24-17 DCMk Service Request Generation

24.2.2.4 Digital Phase Locked Loop Cell (PLL)

The GPTA provides a digital Phase Locked Loop cell (PLL) with a frequency multiplier function. An input signal edge is used as a trigger to generate a programmable number of GPTA module clocks f_{GPTA} on the output signal line. The four signal output lines of the DCM units can be used as PLL trigger input. The PLL control unit distributes the desired number of GPTA clocks in regular time intervals over the input signal period length. The PLL can automatically follow an acceleration or deceleration of the input signal. Alternatively, an external software routine may handle the input signal's period length variation.

The PLL includes a 4-channel input multiplexer, a 16-bit timer, a 16-bit step register, a 24-bit reload register, a 24-bit adder, a 24-bit multiplexer, a 25-bit delta register extended by one sign bit and a PLL control unit (see [Figure 24-18](#)).

The following registers are assigned to the Phase Locked Loop cell:

- PLLCTR = Phase Locked Loop Control Register (see [Page 24-165](#))
- PLLMTI = Phase Locked Loop Microtick Register (see [Page 24-166](#))
- PLLCNT = Phase Locked Loop Counter Register (see [Page 24-167](#))
- PLLSTP = Phase Locked Loop Step Register (see [Page 24-167](#))
- PLLREV = Phase Locked Loop Reload Register (see [Page 24-168](#))
- PLLDTR = Phase Locked Loop Delta Register (see [Page 24-168](#))
- SRSC0 = Service Request State Clear Register 0 (see [Page 24-206](#))
- SRSS0 = Service Request State Set Register 0 (see [Page 24-208](#))

Three output signals are available on the PLL cell:

- PLL signal output line
- Uncompensated PLL signal output line
- Service request line

General Purpose Timer Array (GPTA)

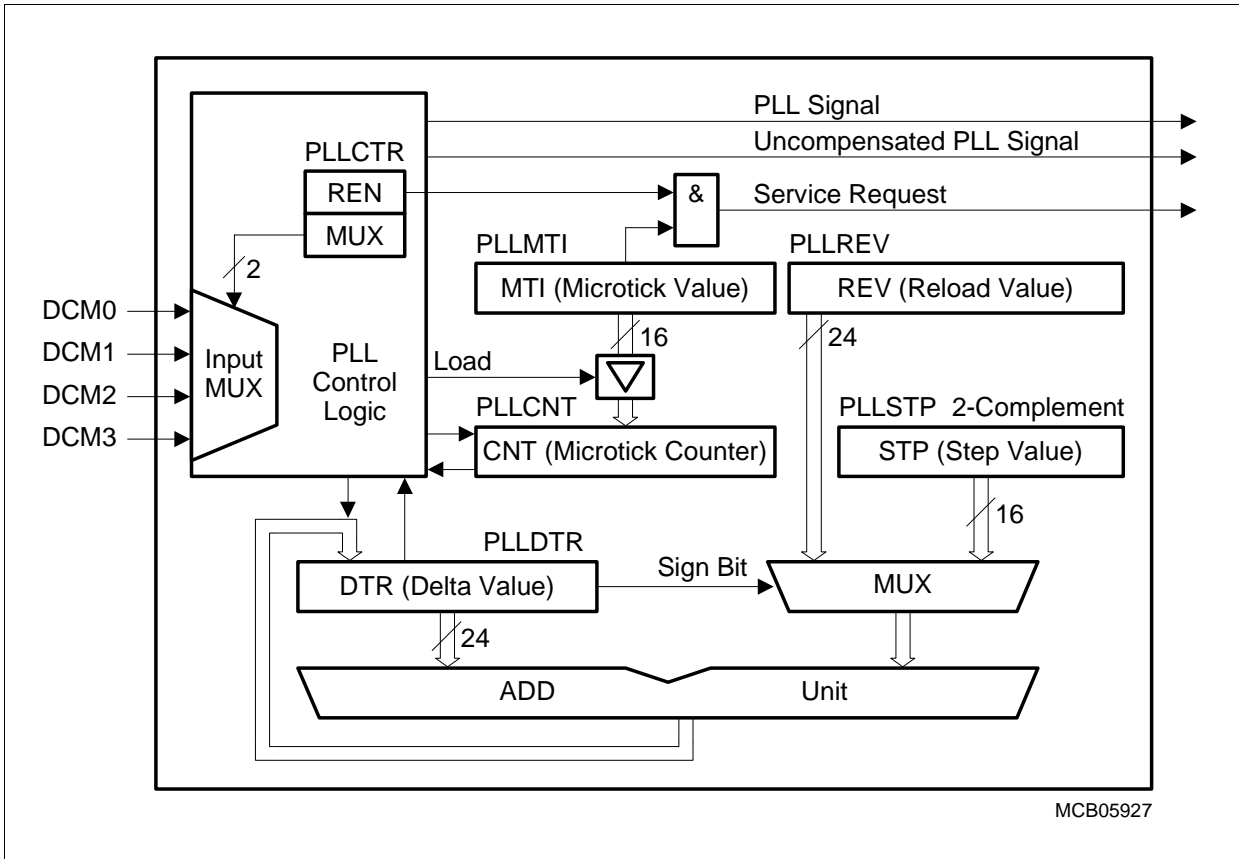


Figure 24-18 Block Diagram of Digital PLL Cell

The desired input signal is selected by programming bit field PLLCTR.MUX. The number of output pulses to be generated within one input signal period must be stored in the microtick register PLLMTI and (coded in 2-complement data format) in the step register PLLSTP. The PLLREV reload register must be programmed with a reload value. This reload value is calculated by subtracting the number of output pulses to be generated within one input signal period from the input signal's period length (measured in number of f_{GPTA} clocks). An automatic compensation of an input signal acceleration or deceleration is enabled by setting bit PLLCTR.AEN to 1 (Automatic End Mode). After disabling the Automatic End Mode, the PLL continuously generates output pulses without synchronization to an input signal edge.

When the counter for the number of remaining output signal pulses PLLCNT decrements to zero, the PLL service request flag is set. Additionally, a service request signal PLLSR will be generated if the control register bit PLLCTR.REN is set.

General Purpose Timer Array (GPTA)

PLL Interrupt Control

The PLL unit is able to generate a service request output signal PLLSR. This signal is controlled as shown in **Figure 24-19**. When the service request condition $PLL\text{CNT} = 0$ occurs, the service request flag is always set. The service request output PLLSR is activated only if it is enabled by the enable bit $PLL\text{CTR.REN}$. Additional information about service request and interrupt handling are given in section **“Interrupt Sharing Unit (IS)”** on **Page 24-112**.

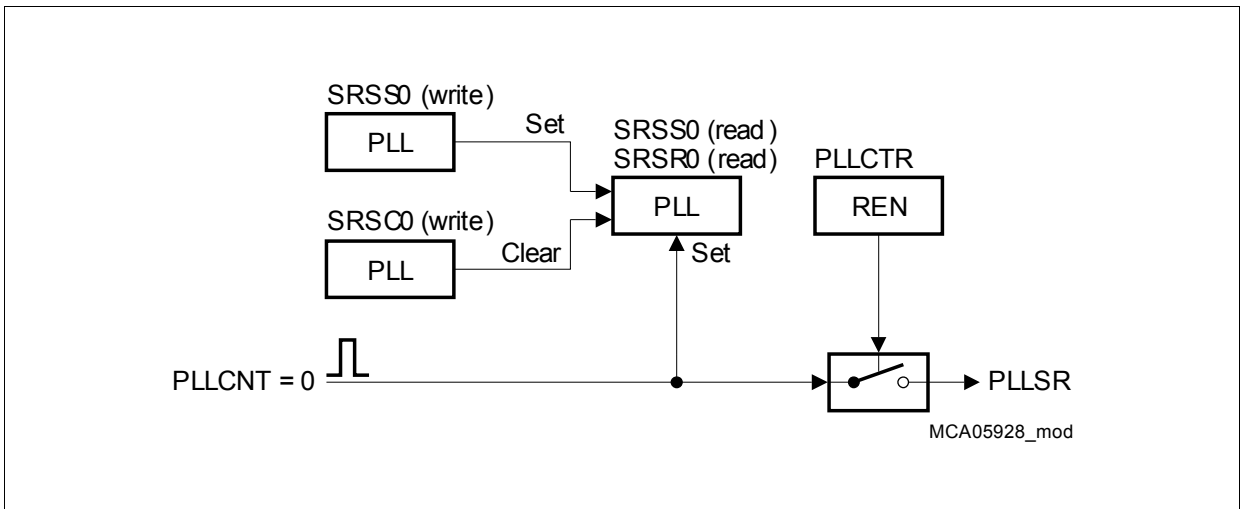


Figure 24-19 PLL Service Request Generation

General Purpose Timer Array (GPTA)

Steady Input Signal Example

In the following example, the input signal's period length is $13f_{GPTA}$ clock periods, which should be subdivided into three equally spaced sections. The reload value to be stored in PLLREV.REV register is calculated to $0A_H$ ($10 = 13 - 3$). PLLMTI.MTI is loaded with 03_H (number of output pulses) and its 2-complement representation (FFD_H) is written into PLLSTP.STP.

After a reset, a state machine driven by the GPTA module clock, updates the delta register PLLDTR with the reload value. Afterwards, the PLLSTP register's contents are continuously added to the delta register value (Figure 24-20). In fact, the difference between both values is computed and stored in the PLLDTR register again, because the PLLSTP register has been loaded with a negative value (2-complement data format). When the PLLDTR register has been decremented to a negative value, the reload register contents are added to Delta register's current contents.

A rising edge detected on the selected input signal triggers the counter register PLLCNT to load the number of requested output pulses from PLLMTI. When a negative content of the PLLDTR register is detected, the microtick counter is decremented by one. In Automatic Mode ($AEN = 1$), the output pulse generation is stopped when the microtick counter reaches zero.

The period length of a single output pulse varies between four and five f_{GPTA} clocks; the maximum period length variation of output pulses is restricted to one f_{GPTA} clock. The total period length of all three output pulses, generated by one PLL loop corresponds to the input signal period width ($5 + 4 + 4 = 13f_{GPTA}$ clocks).

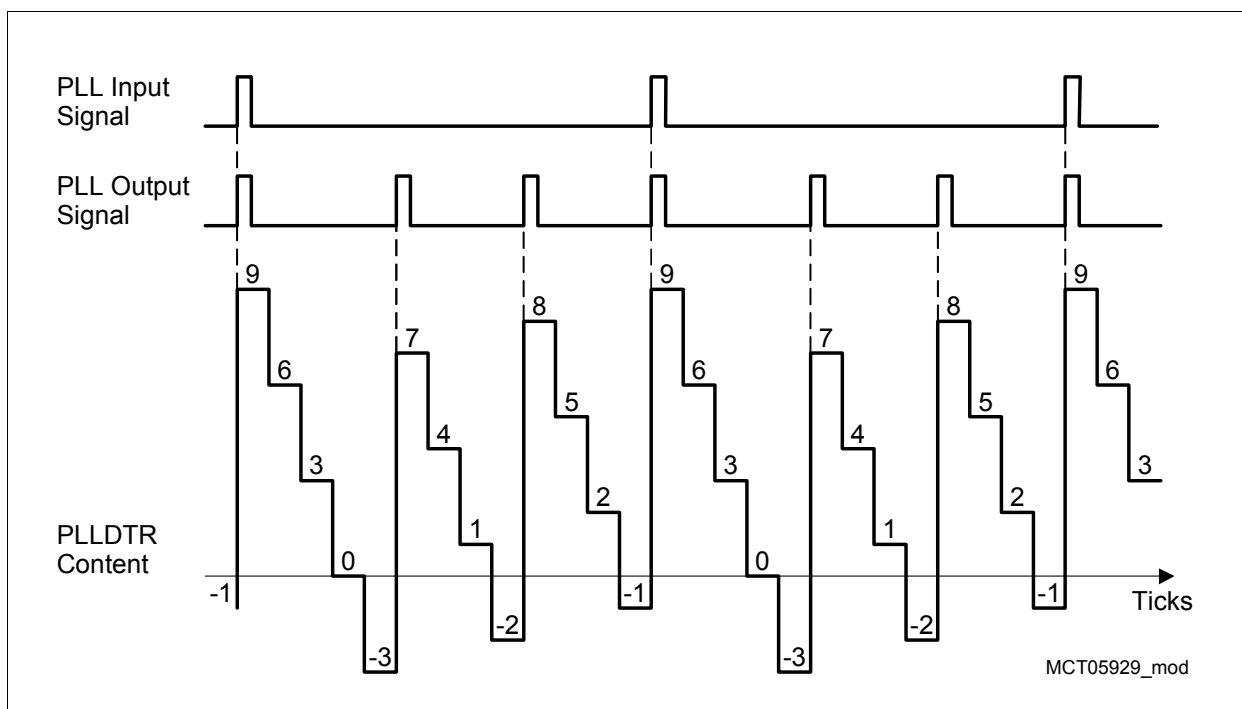


Figure 24-20 Digital PLL Steady State Simulation

General Purpose Timer Array (GPTA)

This type of PLL implementation presents a valuable advantage compared to classic PLL implementation. Indeed, the generated microticks are equally distributed. The division remainder is distributed to several clocks instead of adding this remainder to the last pulse clock of the period.

Figure 24-21 illustrates this advantage. Considering a period of 15 clock pulses to be divided by a factor of 4, it gives a result of 3 with a remainder equal to 3. The reload value is calculated to $0B_H$ ($11 = 15 - 4$). The number of output pulses is equal to 4 and its 2-complement representation ($FFFC_H$) is written into the step register.

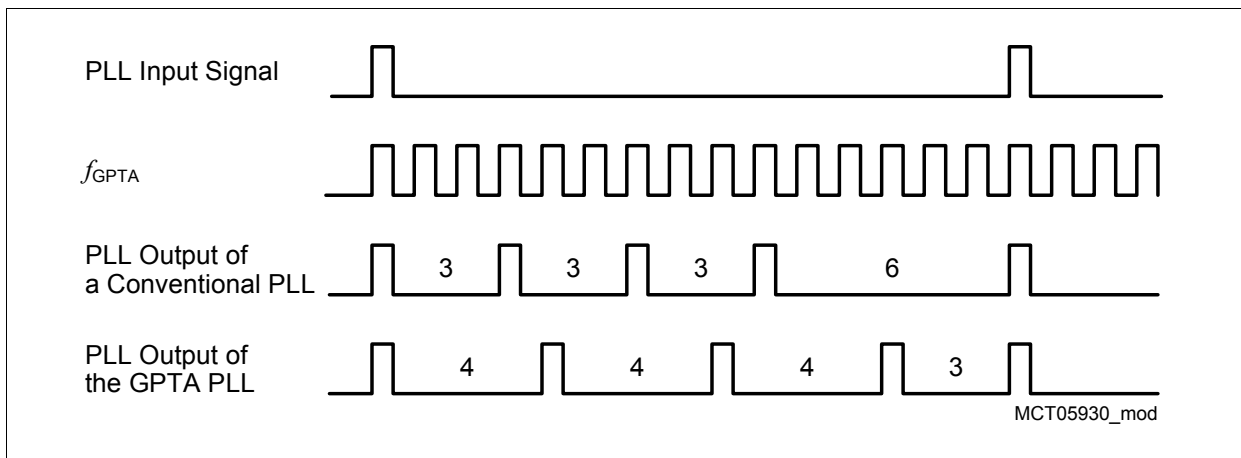


Figure 24-21 Advantage of the GPTA PLL

Input Signal Acceleration and Deceleration

The consequence of an input signal acceleration or deceleration can be compensated either automatically or by an external software routine. It detects an input signal's period length variation by comparing the current period length (measured in the associated DCM cell) with the expected period length used as calculation base for the PLLREV register contents.

- Compensation of input signal deceleration
 - Compensation by PLL Automatic End Mode

If Automatic End Mode is enabled ($PLLCTR.AEN = 1$), the PLL stops at the calculated end of the current input signal period. Due to the deceleration, the rising edge of the following input signal period is delayed, starting the next PLL operation later than expected. A gap occurs between the last output pulse of the current input signal period and the first pulse of the following one (see **Figure 24-22**).
 - Compensation by Software

After disabling the Automatic End Mode ($PLLCTR.AEN = 0$), the PLL generates output pulses without synchronization to an input signal edge. In case of a deceleration, more output pulses than calculated are generated during one input signal period. Several algorithms can be implemented to compensate the surplus of generated output pulses:

General Purpose Timer Array (GPTA)

The length of the current input signal period has been underestimated by a certain number of f_{GPTA} clock periods. This deficit could be added to the calculated length of the next input signal period.

The PLL can continue to operate with the old input signal period length estimation, but the number of output pulses to be generated during the next input clock period may be decreased by the surplus of output pulses initiated during the last signal period.

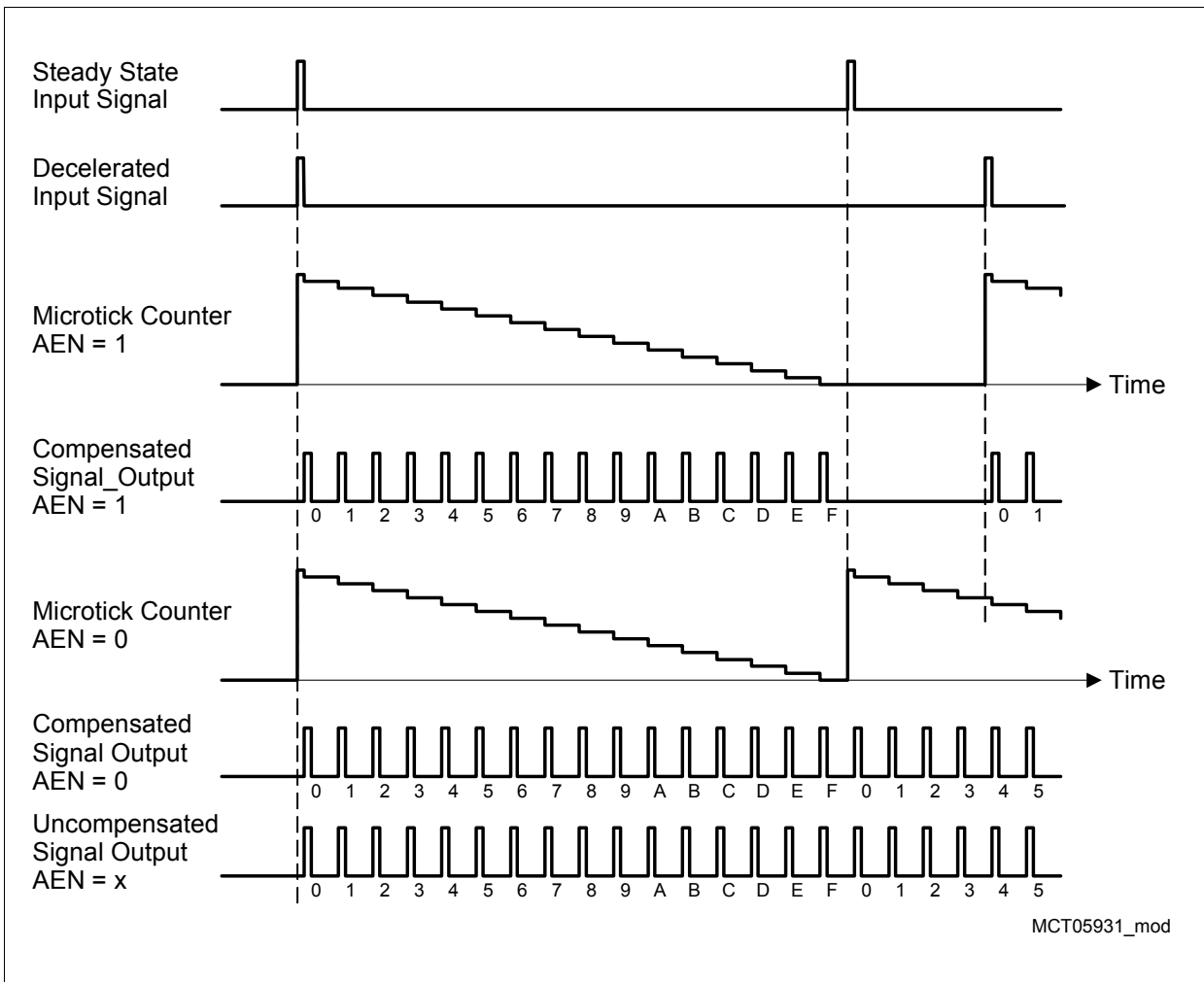


Figure 24-22 Compensation of Input Signal Deceleration

- Compensation of input signal acceleration
 - Compensation by PLL Automatic End Mode

The next rising edge of the input signal arrives while the counter has not been decremented to zero. The PLL performs all remaining output signal pulses at full speed (f_{GPTA}), when control register bit AEN is set to 1. Subsequently, counter and Delta register are reloaded with their calculated values and the PLL operates at normal speed (see [Figure 24-23](#)).

General Purpose Timer Array (GPTA)

– Compensation by Software

After disabling the Automatic End Mode, the PLL generates fewer output pulses than calculated during one input signal period. Several algorithm can be implemented to compensate for the lack of generated output pulses:

The length of the current input signal period has been overestimated by a certain number of f_{GPTA} clock periods. This deficit should be subtracted from the calculated length of the next input signal period.

The PLL can continue to operate with the old input signal period length estimation, but the number of output pulses to be generated during the next input clock period may be increased by the lack of output pulses initiated during the last signal period.

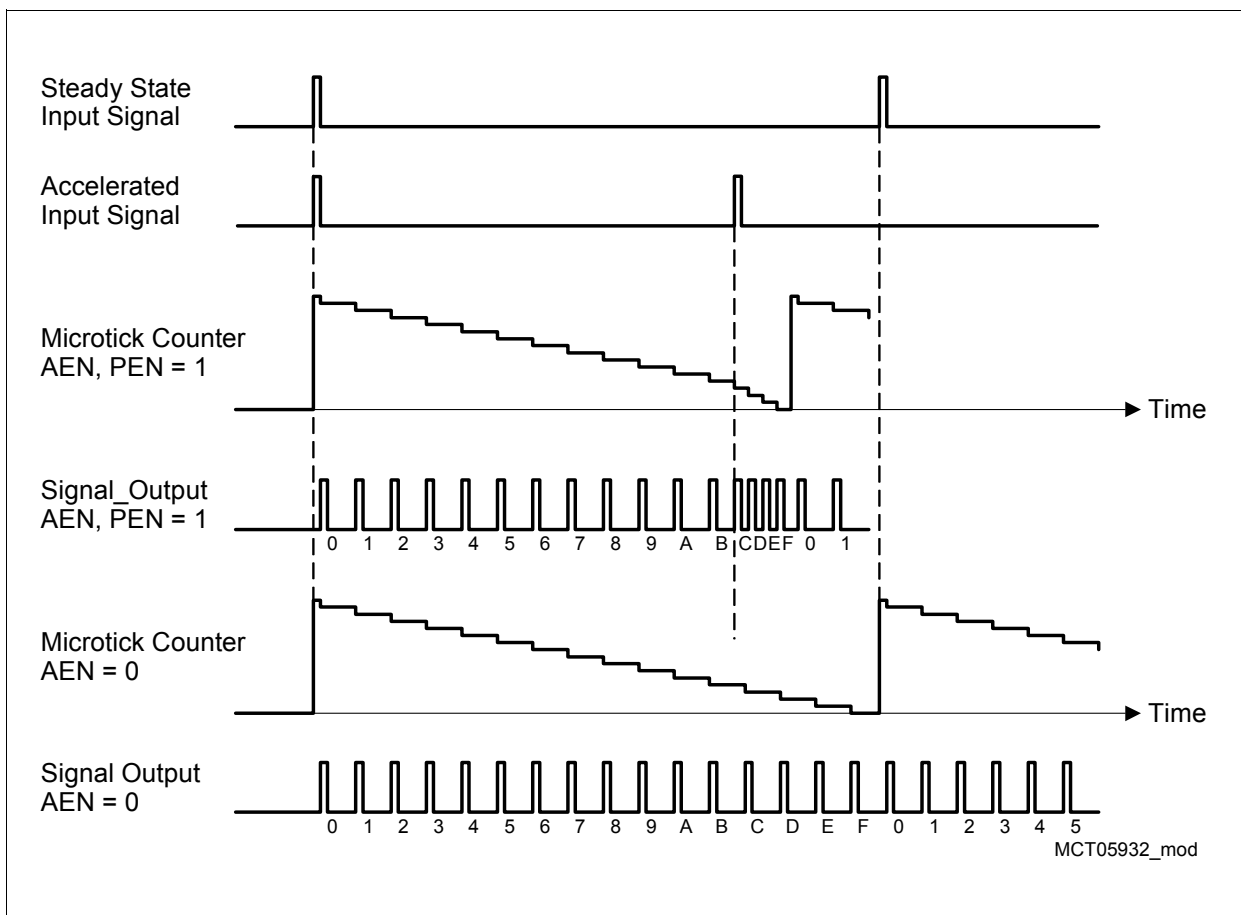


Figure 24-23 Compensation of Input Signal Acceleration

Additionally to the normal output signal, the PLL provides an uncompensated output signal. This signal has no gaps or acceleration bursts. However, the number of microticks during one signal period may be incorrect.

General Purpose Timer Array (GPTA)**24.2.2.5 Clock Distribution Unit (CDU)**

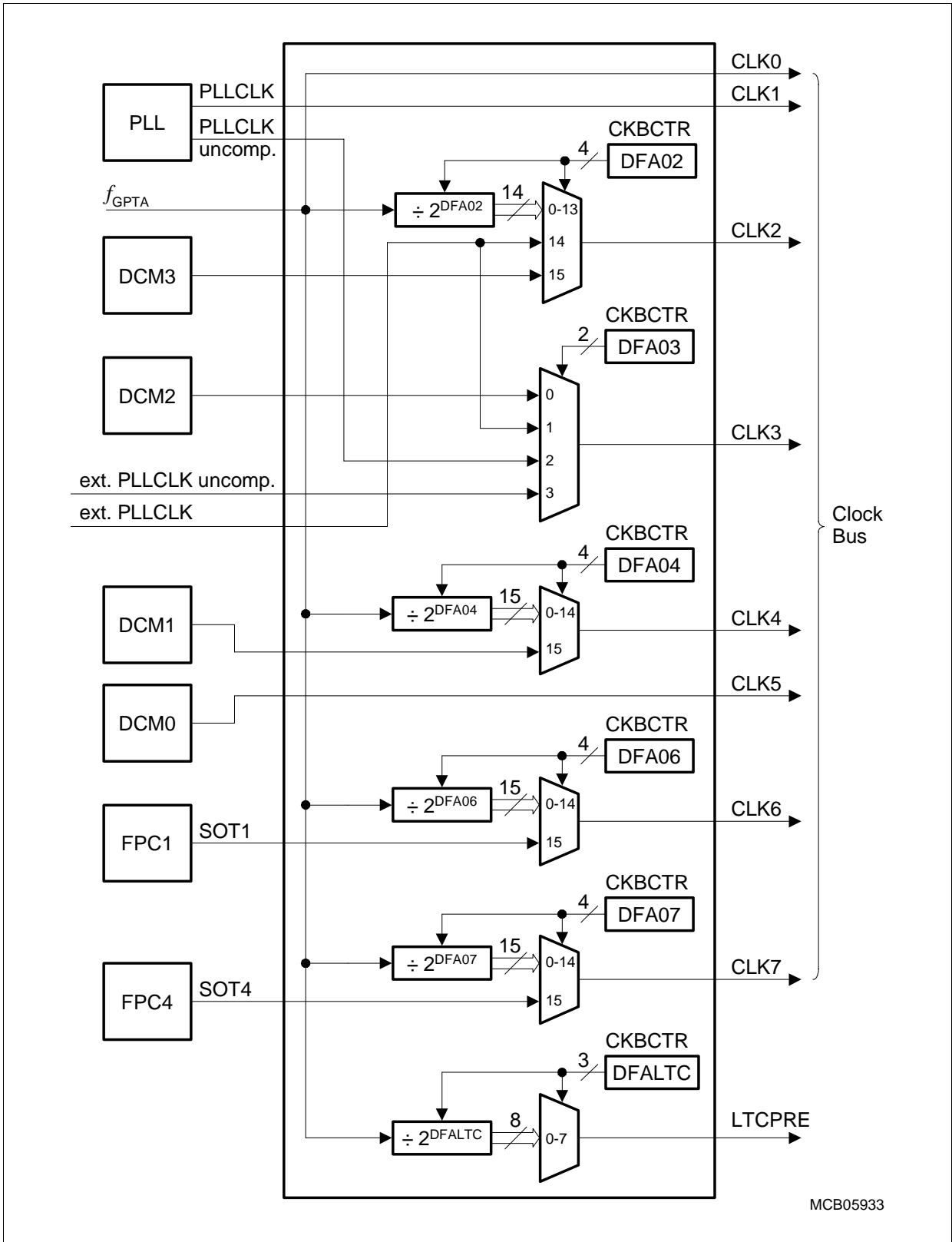
The Clock Distribution Unit (CDU) provides all Local and Global Timer Cells with a clock bus containing eight different clock output signals CLK[7:0] and a special LTC prescaler clock LTCPRE. These nine clock signals are generated out of eleven clock input signals coming from different clock sources (see [Figure 24-24](#)).

The prescalers divide the GPTA module clock f_{GPTA} by a programmable 2^n factor. Factor n is defined by bit fields DFA02, DFA04, DFA06 and DFA07 of control register CKBCTR. A bit field value of 15 disables the related prescaler and selects alternate sources for clock bus lines 2, 4, 6 and 7. For clock bus line CLK2, a bit field value of 14 selects an alternate source.

For clock bus line CLK3, the 2-bit wide bit field DFA03 of control register CKBCTR selects one of the four available clocks.

The LTC prescaler clock LTCPRE is generated by dividing the f_{GPTA} module clock by a factor defined by the 3-bit wide bit field DFALTC of control register CKBCTR. Note that the LTCPRE clock is not a part of the clock bus but a clock signal that is distributed directly from the CDU to each LTC except LTC63.

General Purpose Timer Array (GPTA)



MCB05933

Figure 24-24 Clock Distribution Unit Block Diagram

General Purpose Timer Array (GPTA)

24.2.3 Signal Generation Unit

As described in detail in the following sections, the Signal Generation Unit contains three types of modules:

- Global Timer (GT)
- Global Timer Cell (GTC)
- Local Timer Cell (LTC)

24.2.3.1 Global Timers (GT)

The GPTA provides two global 24-bit timers (GT) that are connected to the clock bus with its eight clock lines. Each GT is locally equipped with a clock source multiplexer, a 24-bit up-counter, a 24-bit reload register, and a 24-bit greater/equal comparator (see [Figure 24-25](#)).

Note: Index variable k ($= 0, 1$) determines the number of the Global Timer.

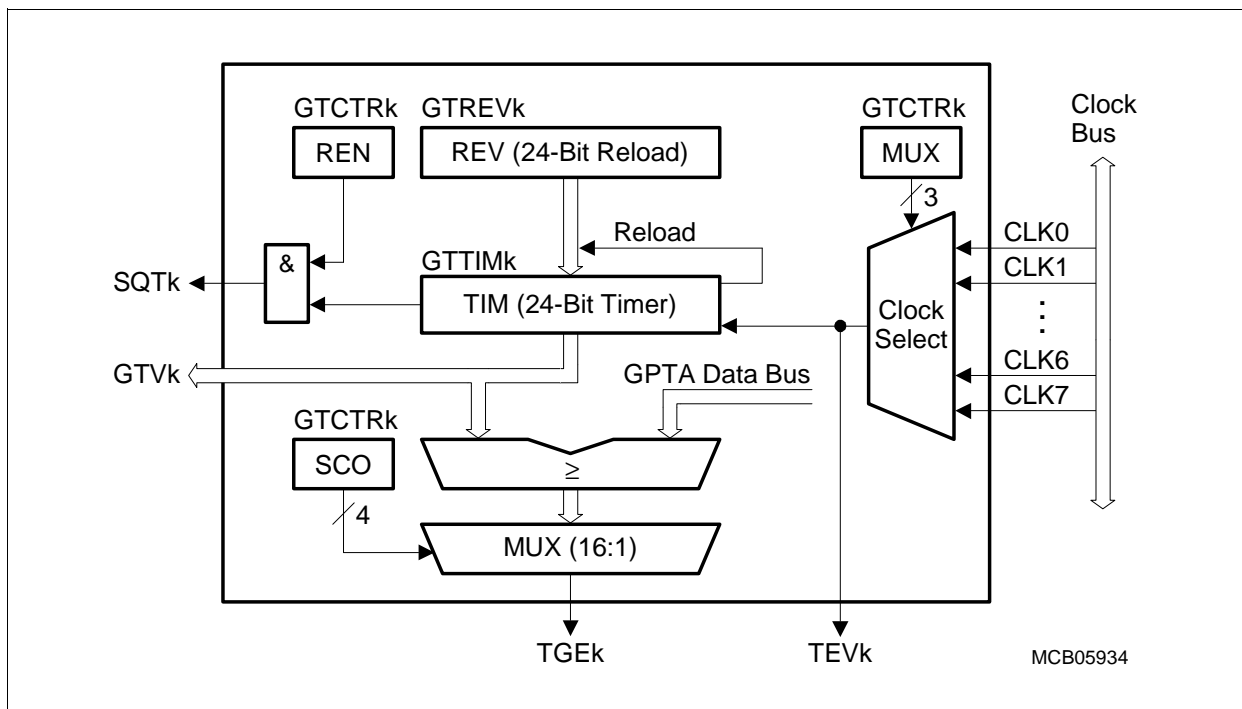


Figure 24-25 Block Diagram of Global Timer (GT)

The following registers are assigned to the Global Timers GT $_k$ ($k = 0, 1$):

- GTCTR $_k$ = Global Timer Control Register k (see [Page 24-170](#))
- GTREV $_k$ = Global Timer Reload Value Register k (see [Page 24-172](#))
- GTTIM $_k$ = Global Timer Register k (see [Page 24-171](#))
- SRSC0 = Service Request State Clear Register 0 (see [Page 24-206](#))
- SRSS0 = Service Request State Set Register 0 (see [Page 24-208](#))

General Purpose Timer Array (GPTA)

Each of the two GT modules provides the following input/output signals:

- Eight clock inputs, connected to the clock bus from the clock distribution unit (CDU)
- Global timer value bus GTVk (outputs), carrying the 24-bit GTk counter value
- TEVk output, indicating a GT counter update
- TGEk output, indicating the result of a compare operation
- SQTk service request output, triggered at a timer overflow.

The Global Timer output signals GTVk, TEVk, and TGEk are available as input signals at each GTC (see also [Page 24-56](#)).

Global timer k can be initialized with a start value, that is written by software into the GTTIMk register. The 24-bit Global Timer value GTTIMk.TIM is incremented by each rising edge of clock input signal TEVk that is selected from the 8-bit clock bus via bit field GTCTRk.MUX. On a Global Timer overflow (transition of $FFFFFF_H$ to 000000_H), the following events occur:

- The 24-bit reload value GTREVk.REV is copied into GTTIMk.TIM
- Bit SRSC0.GT0k is set
- The service request output SQTk is activated (if enabled by bit GTCTRk.REN)

A free-running timer is configured by programming GTREVk.REV with 000000_H .

The “Timer Event” (TEVk) output is activated if the GTk value changes because of a clock edge, a timer reload operation, or a software write access to GTCTRk. The TEVk output is connected to all GTCs. TEVk is used in the GTCs to trigger a compare operation, re-checking the equality of their compare register contents and the updated Global Timer value.

General Purpose Timer Array (GPTA)

GT Interrupt Control

Each of the GTs is able to generate a service request output signal SQTk. This signal is controlled as shown in **Figure 24-26**. On a GTk timer overflow, the service request flag GT0k is always set. The service request output SQTk is activated only if it is enabled by the enable bit GTCTRk.REN. Additional information about service request and interrupt handling is given in section **“Interrupt Sharing Unit (IS)”** on **Page 24-112**.

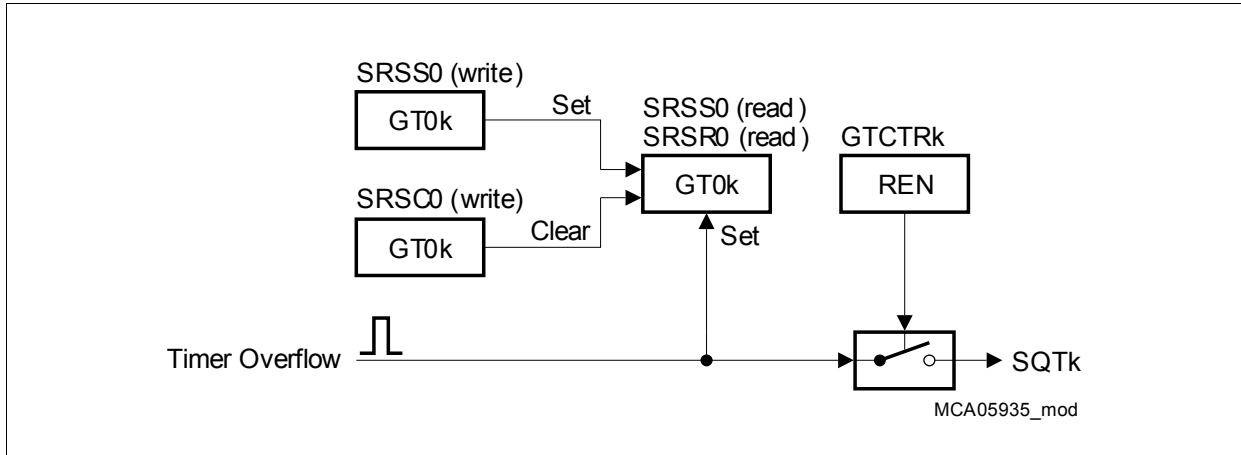


Figure 24-26 GTk Service Request Generation

Synchronization of Global Timers

Both Global Timers, GT0 and GT1, can be enabled and disabled individually. Each GT has its own run signal GTkRUN that is generated outside the GPTA kernel (see also **Page 24-6**). Signal GTkRUN is generated in a GPTA clock control unit. This external control capability makes it possible to control the run signals GTkRUN in a way that all Global Timers of one or more GPTA modules can be enabled/disabled synchronously.

The two Global Timers will run synchronously only if all of the following conditions are true:

- Timers use the same input signal
- Timers are started (and stopped, if required) synchronously
- Timers use identical start and reload values
- Timers are not written while they are running

General Purpose Timer Array (GPTA)

Scalable Signed Greater or Equal Compare

This section (up to [Page 24-54](#)) explains the **classical timer update problem**, and the solutions supported by the GPTA.

The two Global Timers embedded into the GPTA include a 24-bit greater/equal comparator. This comparator unit performs compare operations between the GT timer contents and the data value found on the GPTA-internal data bus (coming from a GTC compare register update). The goal of this comparator is to be able to perform an action immediately if the compare cell is updated with a new threshold but the timer has already passed this value. [Figure 24-27](#) gives an example on this greater/equal concept.

Assumption: a timer is running and a new threshold (value T) is set.

The different points Px represent different cases of present time. When at P1 or P2, the moment represented by T lies in the future and no action is yet required. When at P3 or P4, the moment represented by T lies in the past, and an action is required immediately.

So, the problem is to determine if the threshold T has been passed or not.

Considering an **infinite counter**, the situation is simple. The evaluation consists in determining if point P is before or after T.

Considering a **reloaded counter**, as the timer rolls over at its maximum value, the situation is more complex.

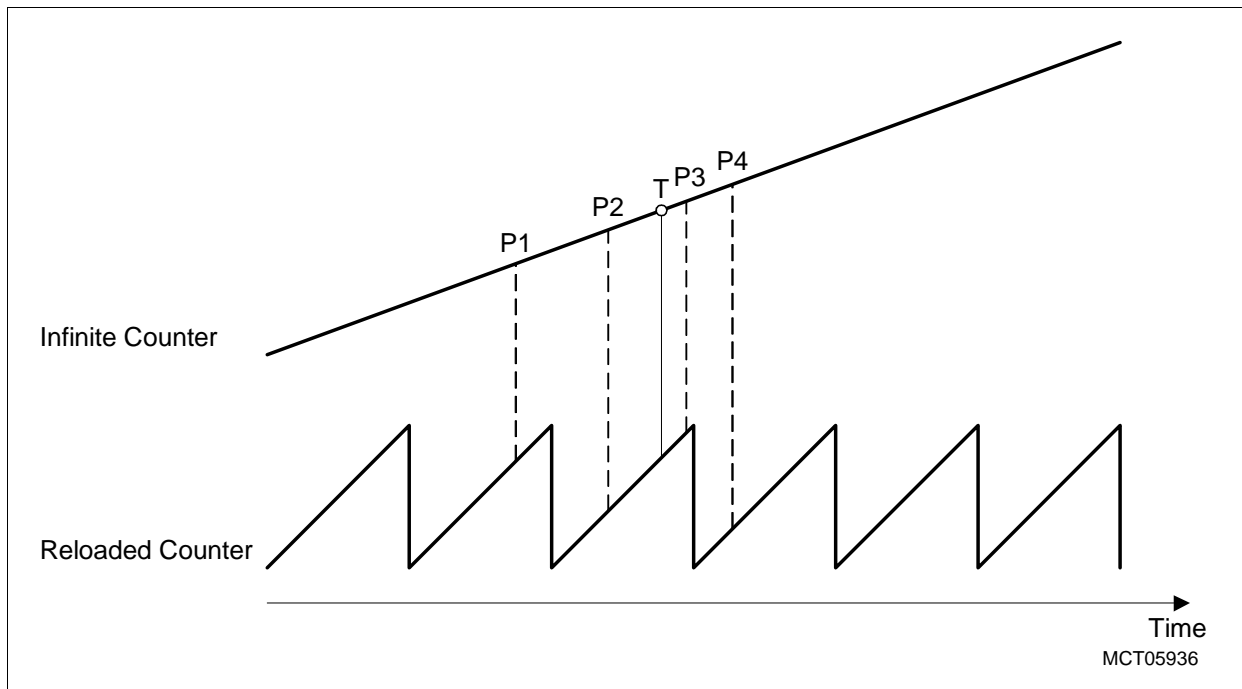


Figure 24-27 Greater/Equal Concept

The **observation window** determines the space in time where writing the value T to the comparator will lead to correct observation (meaning, there is an event if “After”; there is no event if “Before”). Considering an observation window, an event (threshold T) is

General Purpose Timer Array (GPTA)

programmed and then the window is split into two windows, the “After” window and the “Before” window (Figure 24-28). If the timer lies in the “After” window at the time of programming the threshold, the event is performed immediately. If it lies in the “Before” window, the event will happen later when the timer reaches the threshold T. The “Before” window refers to a “prediction range”, and the “After” window refers to the “history buffer”.

From a practical point of view, once the value T is determined, it is necessary to calculate the observation window (position and width). Before updating the value T, the application must ensure that the observation window was entered but has not yet been left.

The width of the **observation window** cannot exceed the timer period. To support reloaded counters where the overflow can occur within the observation window, a **signed** comparison is performed.

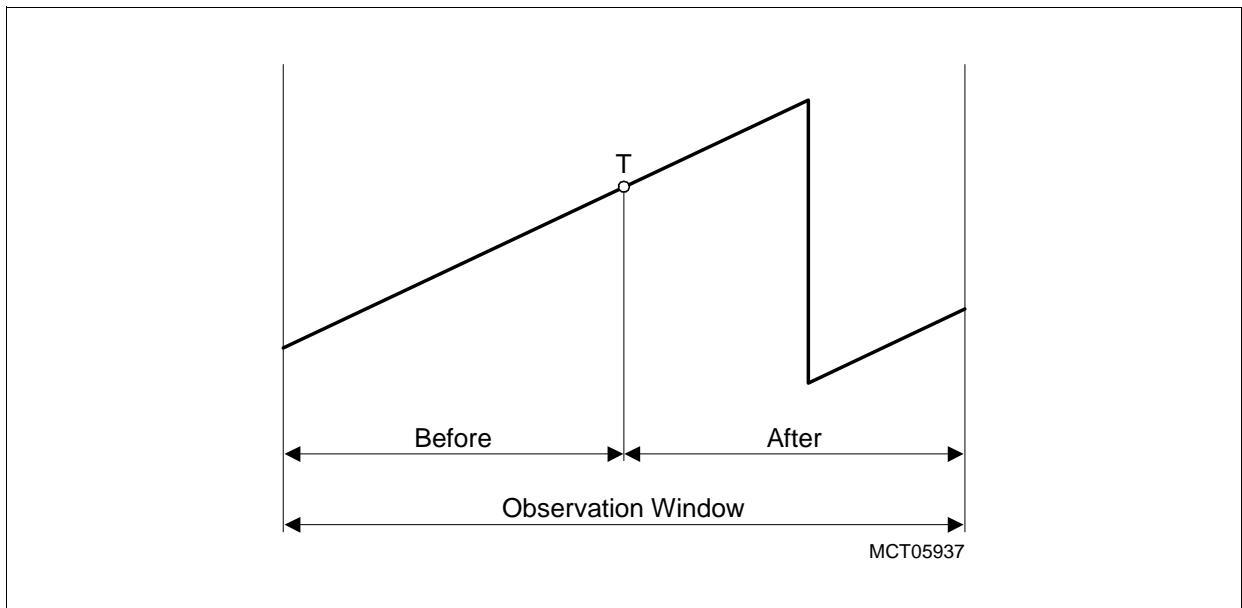


Figure 24-28 Before and After Windows

General Purpose Timer Array (GPTA)

Comparison Between Unsigned and Signed Compare

To be able to support different timer periods and to support correct observation even beyond timer overflow, the GPTA embeds the **scalable** and **signed** greater/equal comparator. Using a signed comparison allows one overflow of the timer to occur within the observation window. This is illustrated in **Figure 24-29**.

Using a signed compare in order to take into account the timer overflow, the comparator window is introduced. The comparator window is centered to the point T and its width can be selected by the user.

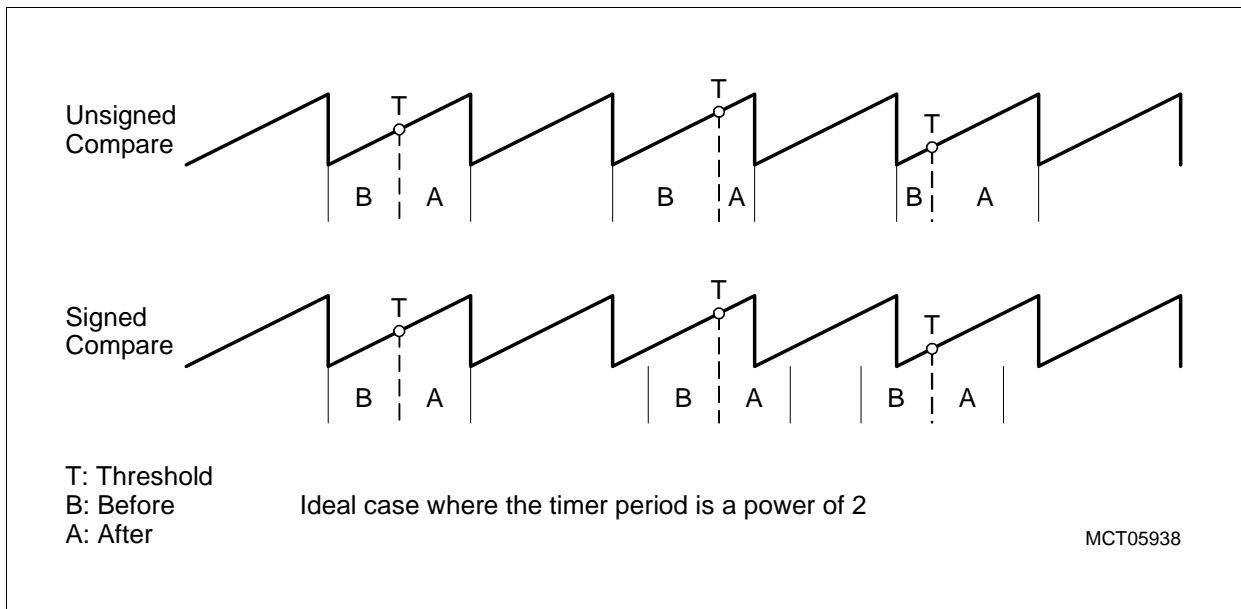


Figure 24-29 Unsigned Versus Signed Compare

When the timer range is a multiple of 2 and because the comparator is scalable, the observation window and the comparator window are identical. See **Figure 24-30**.

General Purpose Timer Array (GPTA)

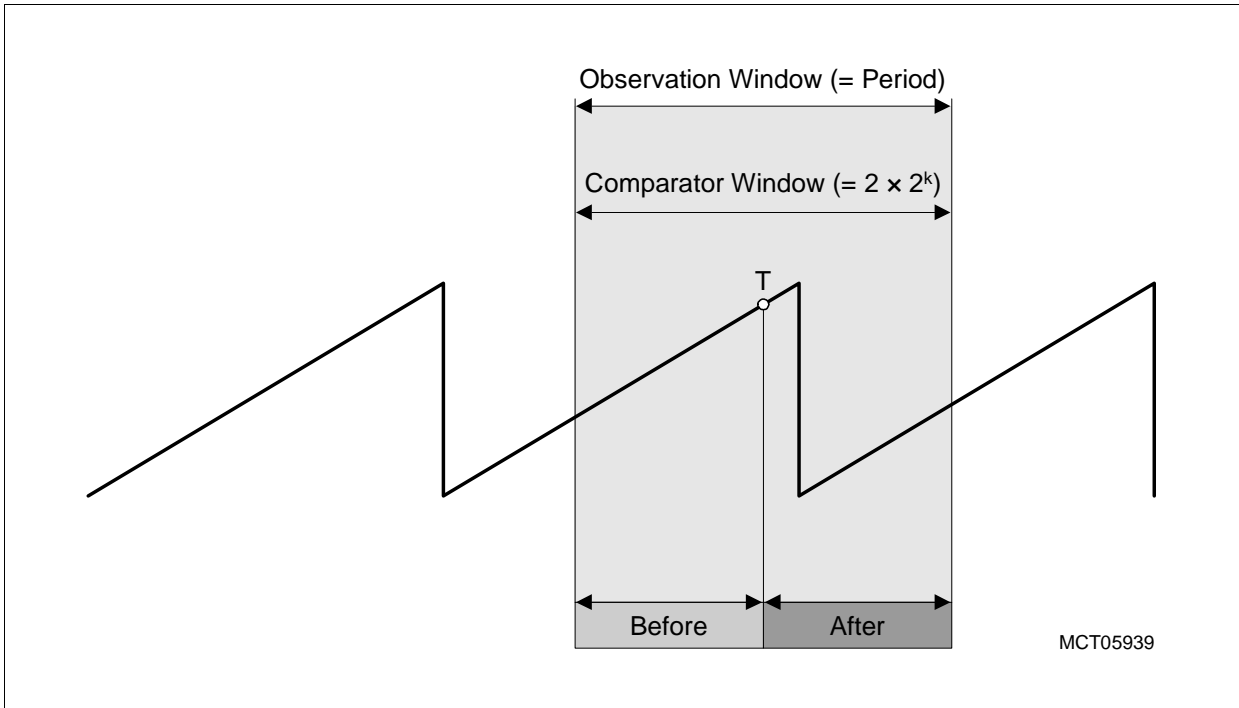


Figure 24-30 Observation and Comparator Windows (timer is a power of 2)

The scalable and signed greater/equal comparator scheme leads to a limitation that must be considered when programming the GPTA Module. If the timer range is not a power of 2, the comparator window (always a power of 2) will no longer match the timer period. This will impact the observation window as described in the following paragraph.

Observation window for reloaded timers (period is not a power of 2)

In that case, the comparator window must exceed the timer period. The user must find the comparator window (by selecting the scale factor k) which fits best the timer period.

The following equation must apply:

$$2^k < \text{Period} \leq 2 \times 2^k \tag{24.1}$$

Figure 24-31 and **Figure 24-32** show that one part of the comparator window must be discarded in order to avoid inconsistency, resulting in the observation window.

General Purpose Timer Array (GPTA)

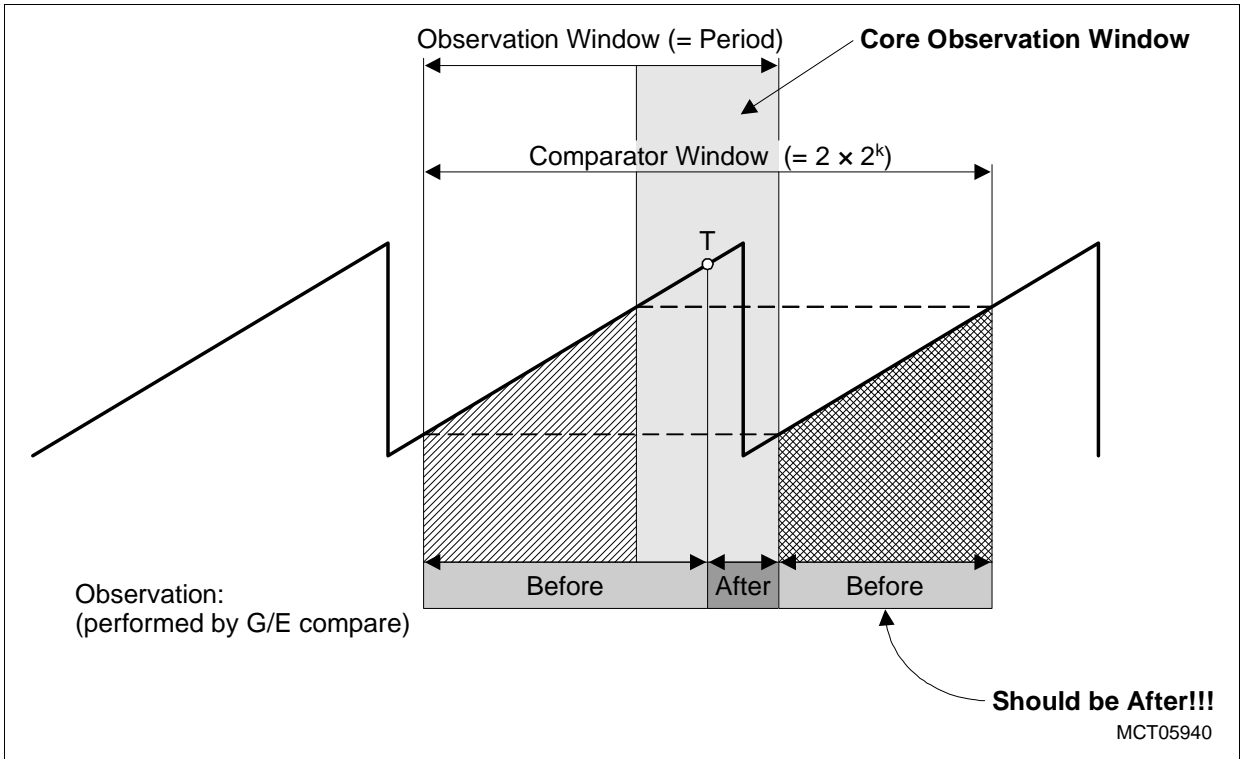


Figure 24-31 Observation Window when Threshold T is High

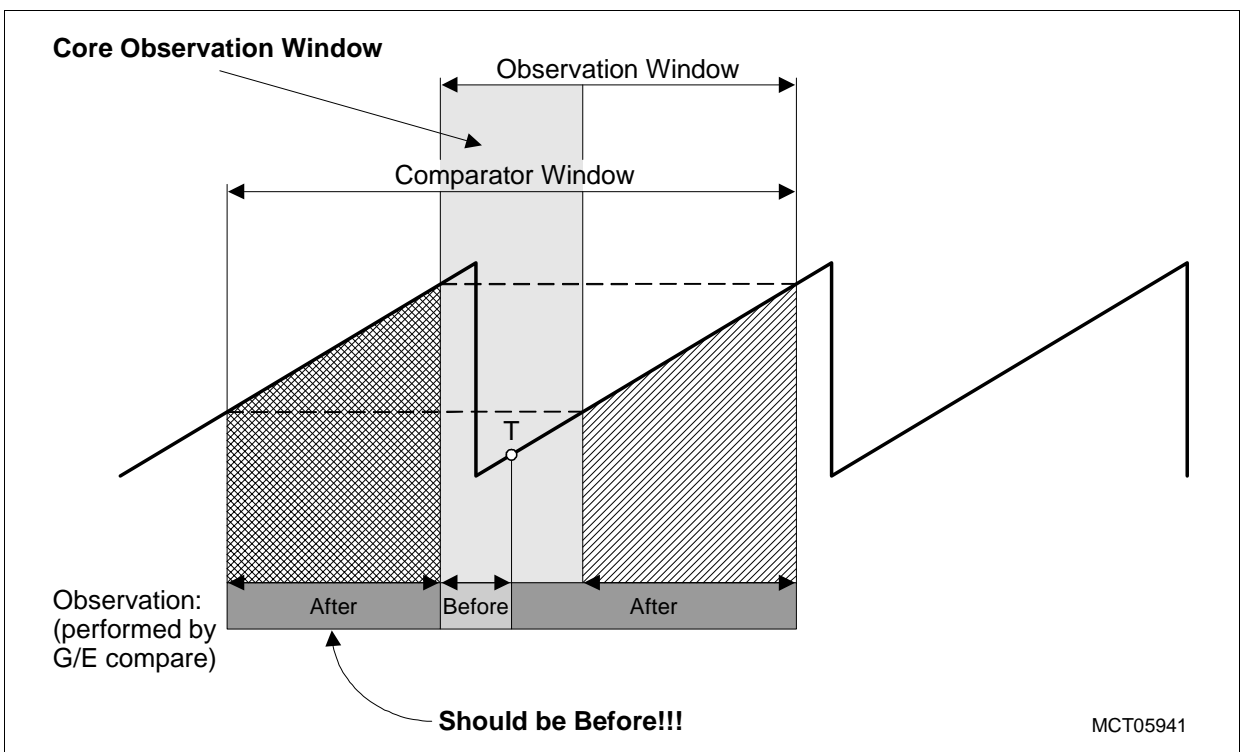


Figure 24-32 Observation Window when Threshold T is Low

General Purpose Timer Array (GPTA)

A comparison of the previous figures shows that the position of the observation window with respect to T is dependent on the value of T itself. That means the user, before updating the comparator with T, needs to calculate the observation window as a function of T. To avoid this calculation, a **core observation window** can be defined that is independent of T. It will always be centered on T, whatever its value is. However, one particularity exists when using the core observation window: the size of the core observation window varies depending on two static values: the timer period and the comparator window's sizes. In particular, the core observation window reduces as the value of the timer period is just after a power of 2. This is shown in **Figure 24-34**.

For any timer period (whatever the range) and any threshold position, a symmetrical core observation window of a statically defined size can be determined.

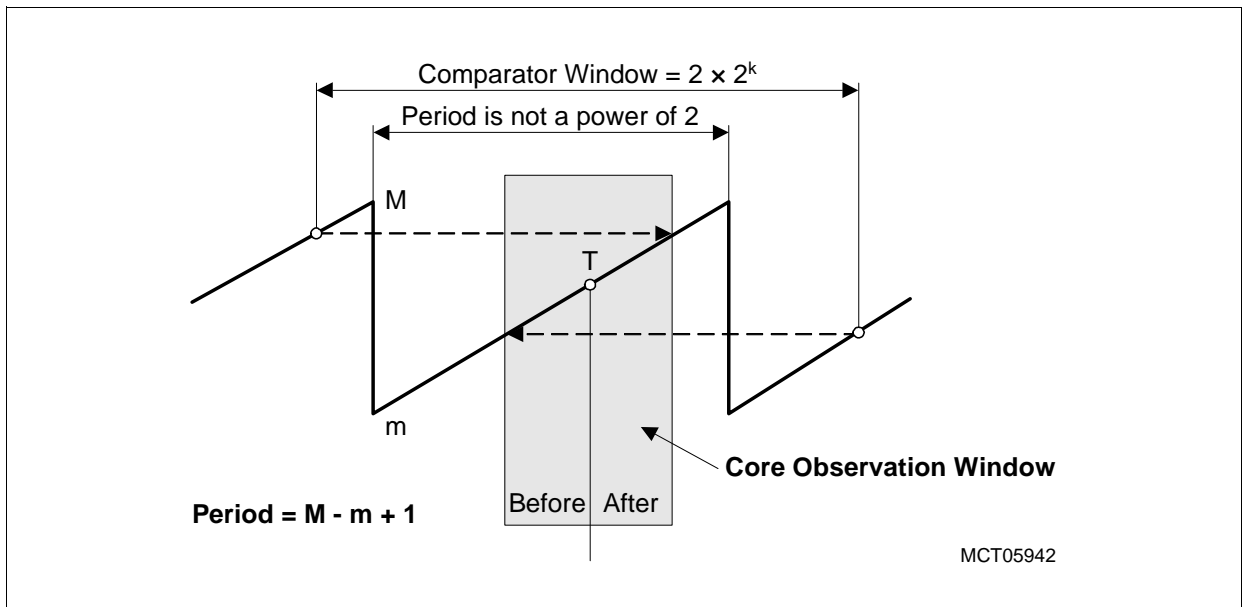


Figure 24-33 The Core Observation Window

General Purpose Timer Array (GPTA)

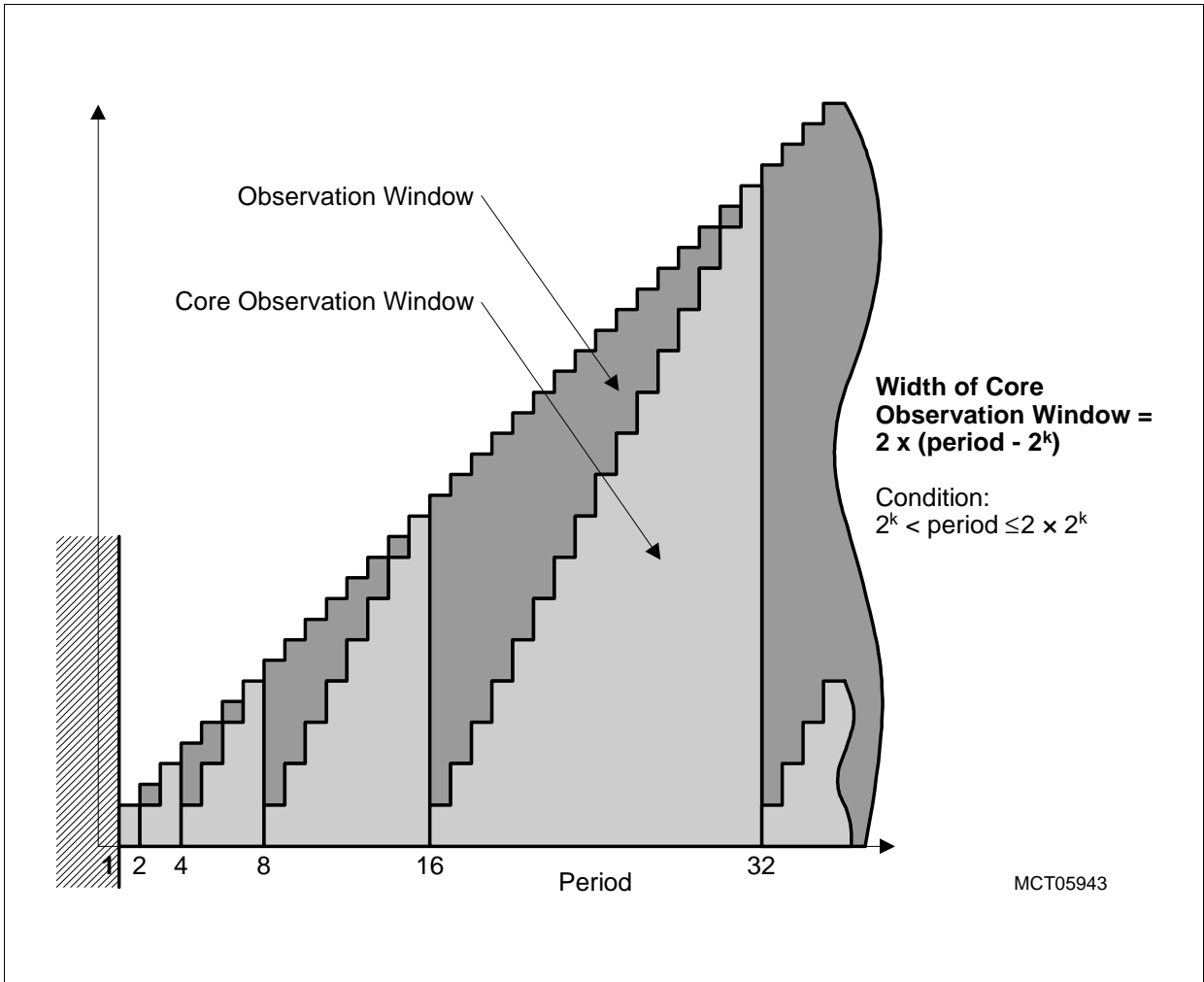


Figure 24-34 Core Observation Window Sizes Versus Period Sizes

General Purpose Timer Array (GPTA)

Implementation

The hardware implementation of the **scalable and Signed/Unsigned** Greater/Equal compare is illustrated in **Figure 24-35**. The function consists of subtracting the threshold T from the GT timer value. The result is in 2s complement format. The result's sign bit and the 15 most significant bits are available for observation. One of those bits is selected according to the mode of operation (Unsigned or Signed) and the period length (bit field GTCTRk.SCO). This bit drives the TGE (Timer Greater Equal) flag.

Unsigned compare: Select Sign bit (SCO = 0F_H)

Signed compare: Select one of the 15 most significant result bits (SCO = 00_H to 0E_H)

Note: On how to choose one of the 15 bits is explained later.

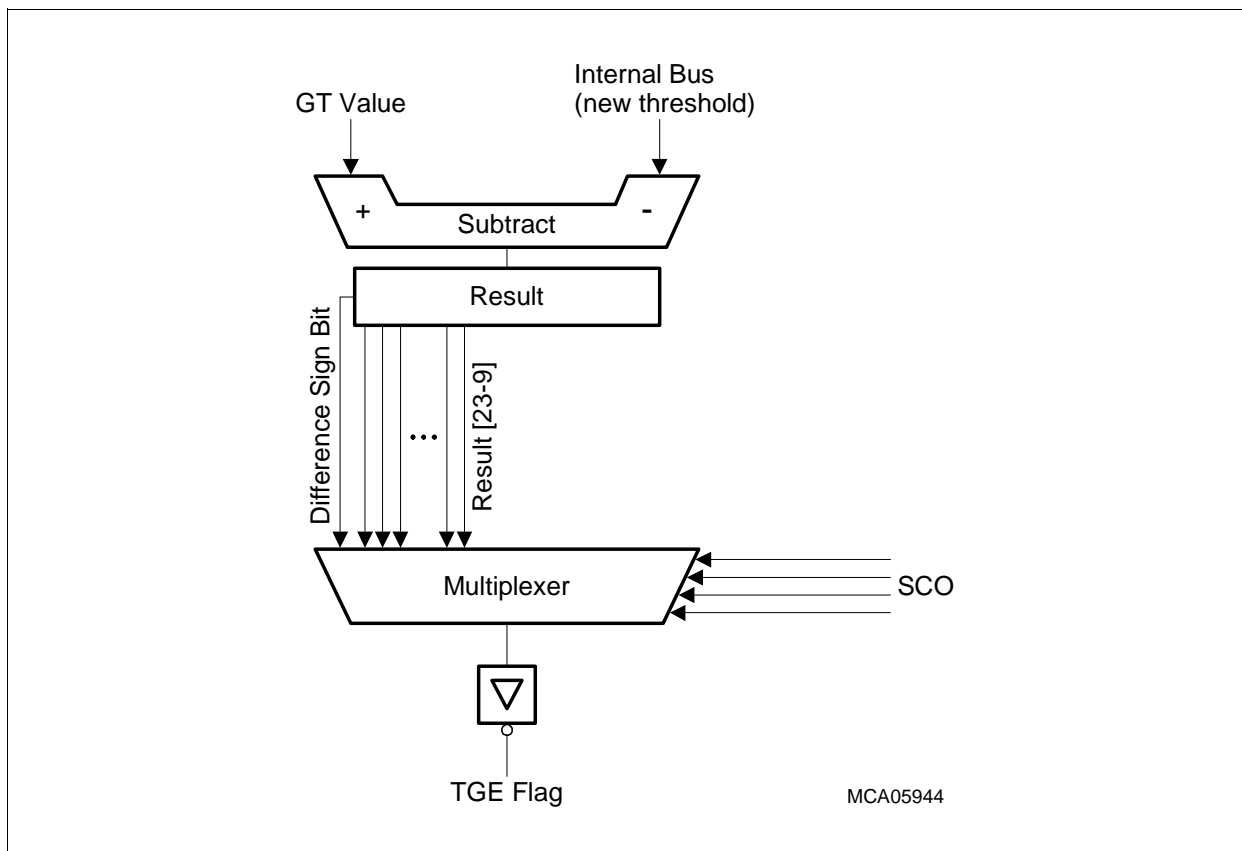


Figure 24-35 Comparator Implemented by a Subtraction Unit

The interpretation of the selected result bit is provided in the following simple example: For a 4-bit timer, the subtraction of the threshold T from the timer value, leads to a 4-bit signed result, as illustrated in **Figure 24-36**. This example is selected for simplicity although 4-bit periods are not covered by the implementation.

When using Unsigned compare, the sign bit S is selected. If it equals 0, the result is positive, indicating that the timer is greater or equal the threshold, and hence **After**. If it equals 1, the result is negative, and the observation indicates **Before**.

General Purpose Timer Array (GPTA)

When using Signed compare, the result bit R_3 can be selected and interpreted, provided that the timer period is at least 9. Here, the range of the result can be split into four sub-ranges. Because the result is in 2s complement format, a value of 0 for R_3 is interpreted as **After**, and a value of 1 is interpreted as **Before**. A comparison of [Figure 24-36](#) and [Figure 24-37](#) shows why this proceeding leads to correct interpretation within the observation window. [Figure 24-37](#) shows the case of a period equal to a multiple of 2.

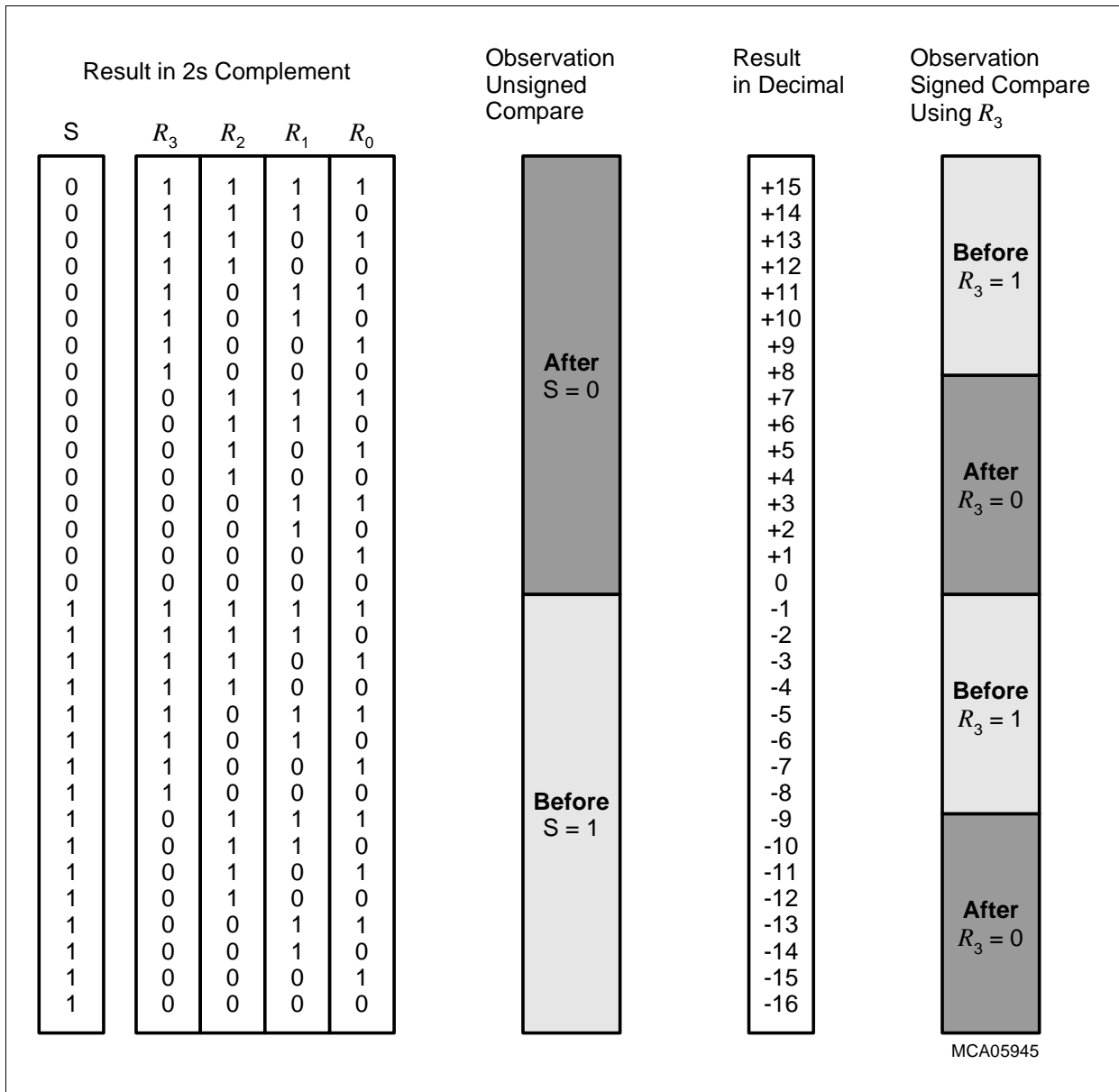


Figure 24-36 Result and Observation for a 4-bit Timer

General Purpose Timer Array (GPTA)

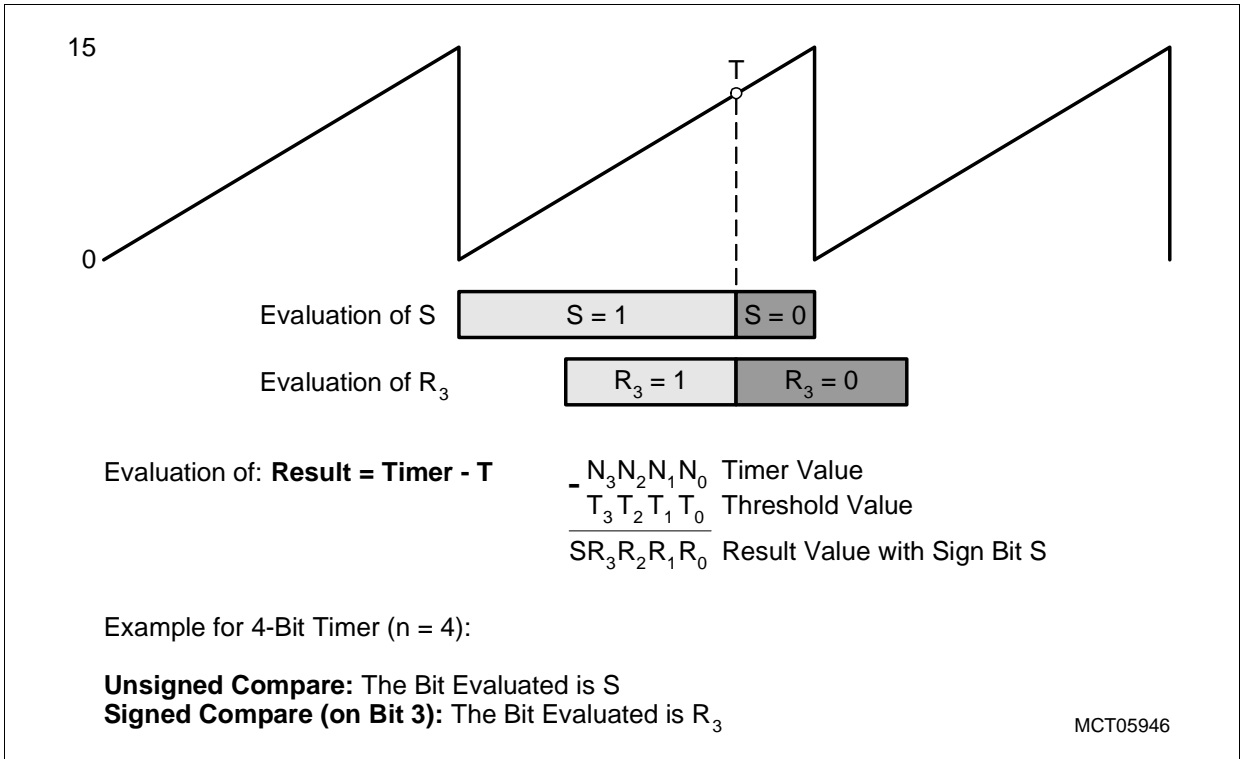


Figure 24-37 Result and Observation (Period = 16)

Figure 24-38 shows the case of a period of 12 which is not a power of 2. Here again, the Table in Figure 24-36 applies.

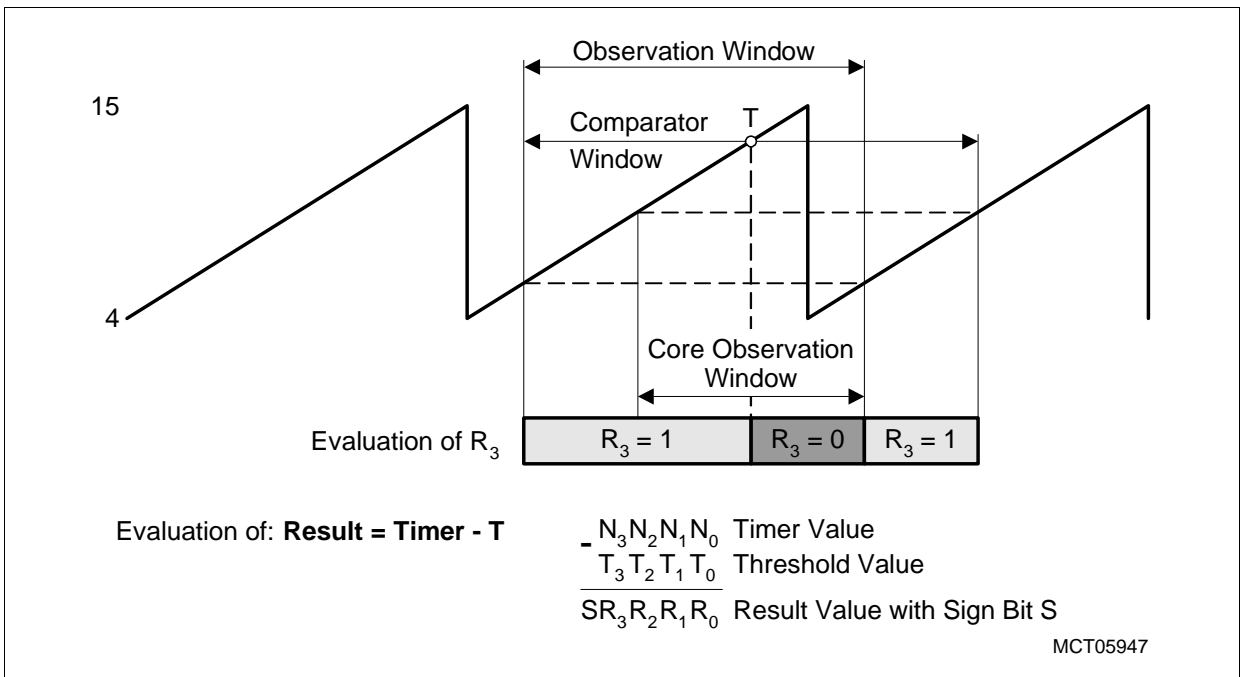


Figure 24-38 Result and Observation (Period = 12)

General Purpose Timer Array (GPTA)

The previous examples show that the result bit to select for observation (R_3) corresponds to the comparator window's size ($k = 3$).

Considering the case in which the period is not a multiple of 2, choose a comparator window whose width is between 1 and 2 times the timer period:

$$2^k < \text{Period} \leq 2 \times 2^k \quad (24.2)$$

In no case may the comparator window be equal to or greater than twice the period.

The letter "k" represents the Result bit to select.

General Purpose Timer Array (GPTA)

How to Proceed

- Unsigned greater/equal compare:

SCO bit field = $0F_H$ (15_d)

Thereby, the sign bit of the result is selected to drive TGE flag.

This setting is valid for all possible periods. The observation window always matches the period.

- Signed greater/equal compare:

Depending on the period, the appropriate k is selected, so that:

$$\text{Period} = M - m + 1 \quad (= \text{Max} - \text{Min} + 1) \quad (24.3)$$

$$2^k < \text{Period} \leq 2 \times 2^k \quad (24.4)$$

SCO bit field = 0 to $0E_H$ (0 to 14_d)

Thereby, the result bit R_k is selected to drive TGE flag.

This setting is possible for periods greater than 512.

Table 24-1 Period Range Depending on Selected k

$2^k < \text{Period} \leq 2 \times 2^k$	k	SCO Bit Field (decimal)
$0 < \text{period} \leq 512$	Not covered by implementation	
$512 < \text{period} \leq 1024$	9	0
$1024 < \text{period} \leq 2048$	10	1
$2048 < \text{period} \leq 4096$	11	2
$4096 < \text{period} \leq 8192$	12	3
$8192 < \text{period} \leq 16384$	13	4
$16384 < \text{period} \leq 32768$	14	5
$32768 < \text{period} \leq 65536$	15	6
$65536 < \text{period} \leq 131072$	16	7
$131072 < \text{period} \leq 262144$	17	8
$262144 < \text{period} \leq 524288$	18	9
$524288 < \text{period} \leq 1048576$	19	10
$1048576 < \text{period} \leq 2097152$	20	11
$2097152 < \text{period} \leq 4194304$	21	12
$4194304 < \text{period} \leq 8388608$	22	13
$8388608 < \text{period} \leq 16777216$	23	14

General Purpose Timer Array (GPTA)

The width of the core observation window is defined by:

$$2 \times (\text{period} - 2^k) \tag{24.5}$$

As a consequence, the width of the “Before” window within the core observation window is $(\text{period} - 2^k)$ and the width of the “After” window within the core observation window is $(\text{period} - 2^k)$, including the value T.

Additional Information: Illustration on the General Case

The previous section illustrated the greater/equal compare for the particular case of a 4-bit timer. The purpose of this section is to describe the implementation from a general point of view, that is, for a timer period equal to $M - m + 1$.

In the following figures, the X axis indicates the timer value (elapsing time) and the Y axis indicates the threshold value T. The 45° line starting at (m, m) represents the position in time of T. The graphic shows the observation performed by the hardware for all cases of T ($m \leq T \leq M$).

Figure 24-39 illustrates the Unsigned compare. A particular case is shown in which, for a higher value of T, the observation indicates “Before” at the beginning of the period, and until the timer reaches the value T. Thereafter, the observation switches to “After” and remains there until the timer exits the period.

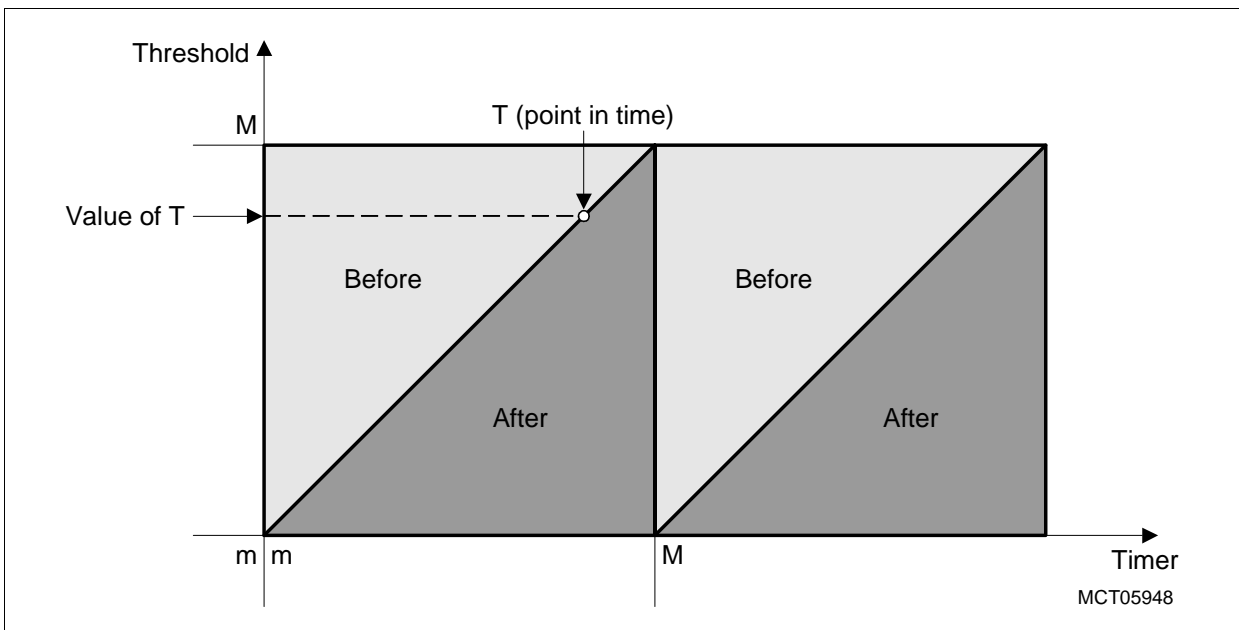


Figure 24-39 Graphical Representation of Unsigned Compare

Figure 24-40 illustrates the Signed compare where the period equals a multiple of 2 (that means $M - m + 1 = 2 \times 2^k$). In this case, for a higher value of T, the observation indicates “After” at the beginning of the period (not yet inside the observation window). When entering the observation window, “Before” is indicated until the timer reaches the

General Purpose Timer Array (GPTA)

value T. Thereafter, the observation switches to “After” and remains there until the timer exits the observation window. This graphic can be related to [Table 24-37](#) where the comparator window equals the period, and the observation window is always centered on the threshold T.

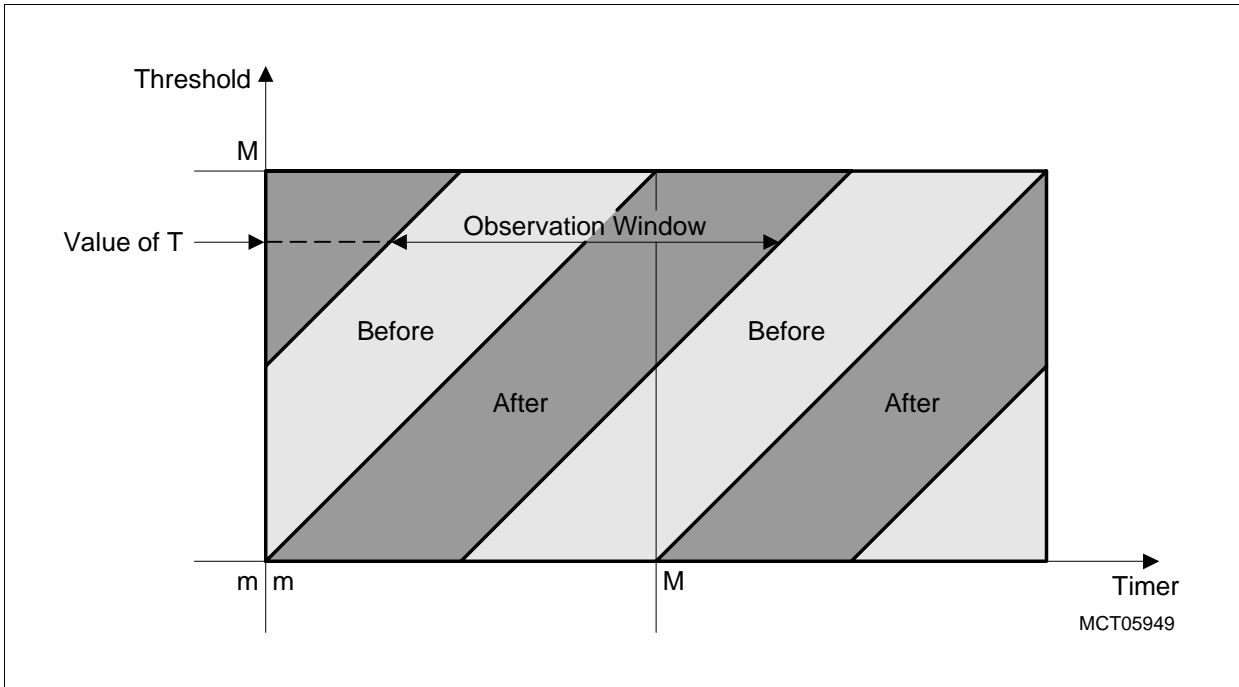


Figure 24-40 Graphical Representation of Signed Compare (Period = 2×2^k)

The [Figure 24-41](#) illustrates the Signed compare where the period may also be unequal a multiple of 2. The graphical representation of this general case is analogous to the one described in [Figure 24-31](#).

If the period is not a multiple of 2, the graphical representation of the Signed compare shows a discontinuity in the “Before” and “After” ranges. Indeed, the widths of the “Before” and “After” windows are not constant, as they depend on the value T. As a consequence, the observation window is not centered on T. The result is that the position of the observation window would have to be re-evaluated for each value T (i.e. determining the widths of the “After” and the “Before” window). For this calculation, the principal characteristic is shown in [Table 24-41](#) (2×2^k - period = comparator window - period).

General Purpose Timer Array (GPTA)

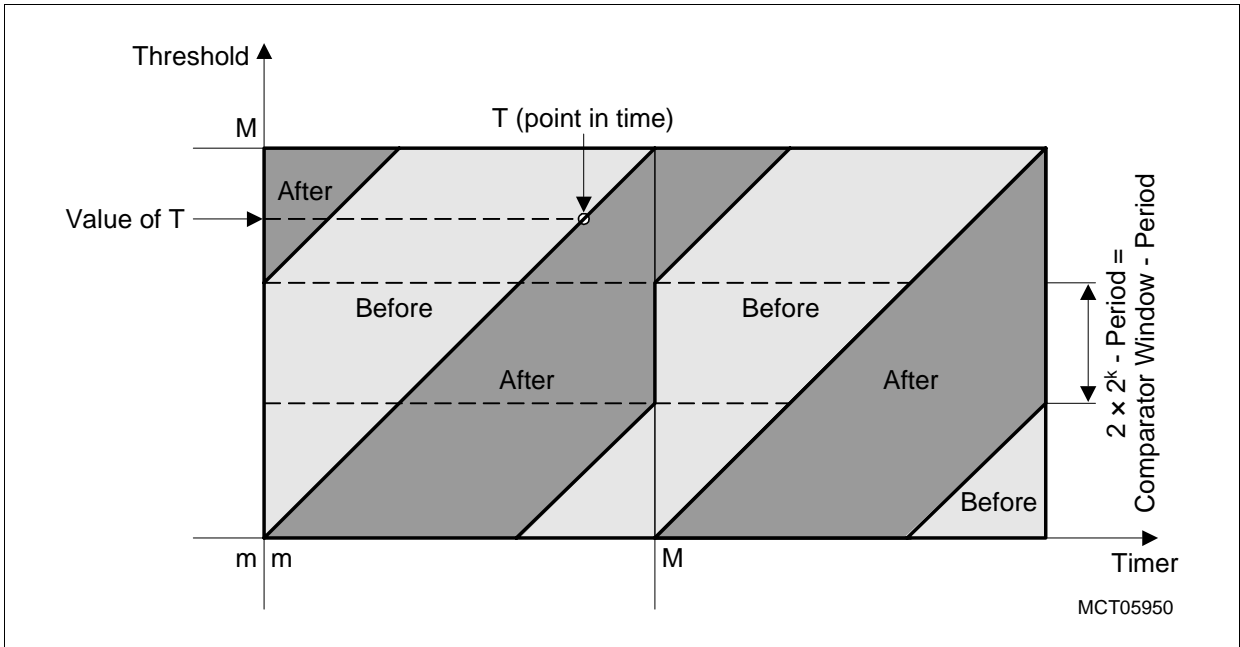


Figure 24-41 Graphical Representation of Signed Compare ($2^k < \text{Period} \leq 2 \times 2^k$)

Figure 24-42 shows how the observation window is positioned with respect to T. It also shows the **core observation window** that is always centered on T and which has a constant width.

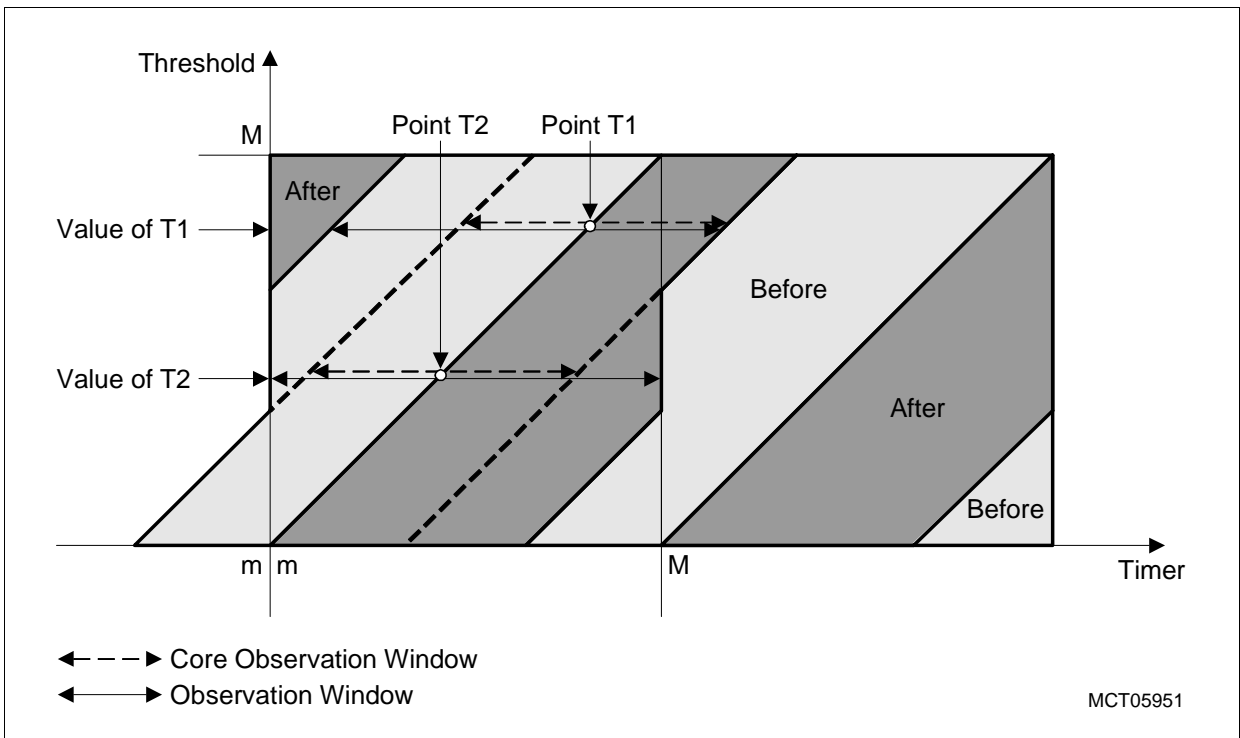


Figure 24-42 Core Observation Window in the Graphic

General Purpose Timer Array (GPTA)

24.2.3.2 Global Timer Cell (GTC)

The GPTA provides 32 Global Timer Cells (GTC00 to GTC31) used for capture/compare operations.

Registers

The following registers are assigned to a GTC_k (k = 00-31):

- GTCCTR_k = Global Timer Cell Control Register k (see [Page 24-175](#))
- GTCXR_k = Global Timer Cell X Register k (see [Page 24-179](#))
- SRSC1 = Service Request State Clear Register 1 (see [Page 24-209](#))
- SRSS1 = Service Request State Set Register 1 (see [Page 24-210](#))

Features

- **24-bit based timer cells** related to two Global Timers GT0 and GT1.
- **Capture Mode** on rising, falling or both edges with following actions:
 - Service request generation
 - Output signal transition generation (set, reset, toggle the output signal)
- **Compare Mode** on equal compare, or greater than, or equal to compare with following actions:
 - Service request generation
 - Output signal transition generation (set, reset, toggle the output signal)
 - Capture (after compare match) the value of the selected Global Timer or the opposite Global Timer
- **One Shot Mode** allows the selected (capture or compare) mode to be stopped after the first event.
- **Flexible mechanism** to link pin actions and allow complex combination of cells. (A cell has the ability to propagate actions over adjacent cells with higher number, in order to perform complex waveforms such as PWMs).

Architecture

The architecture of a GTC is shown in [Figure 24-43](#). Each GTC has a multiplexer that allows selection of the GT0 or GT1 Global Timer value bus as data source, a 24-bit capture/compare register GTCX_k, and a 24-bit equal comparator.

The 32 Global Timer Cells (GTC00 to GTC31) have the following inputs:

- Two Global Timer value buses, GTV0 and GTV1, coming from the two Global Timers and carrying the GT0 and GT1 timer values
- Two inputs, TEV0 and TEV1, reporting GT0 and GT1 timer value updates
- Two inputs, TGE0 and TGE1, reporting the result of the GT0 and GT1 compare operations
- A trigger input (GTC_kIN) that is connected via the GTC input multiplexer to one of the following signal sources:

General Purpose Timer Array (GPTA)

- External port lines
- Local Timer Cell outputs
- Filter and Prescaler Cell outputs
- Internal input signals INTx
- Two action mode inputs (M0I, M1I) coming from the adjacent GTC with lower order number (M1I and M0I of GTC00 are 0)

Each GTC provides the following outputs:

- One data output (GTCKOUT) that can be connected to:
 - External port lines
 - Inputs of an MSC module
 - Outputs and/or inputs of Local Timer Cell inputs
- Two action mode outputs (M0O, M1O) going to the adjacent GTC with higher order number
- One service request line (SQSk) triggered by a capture/compare event.

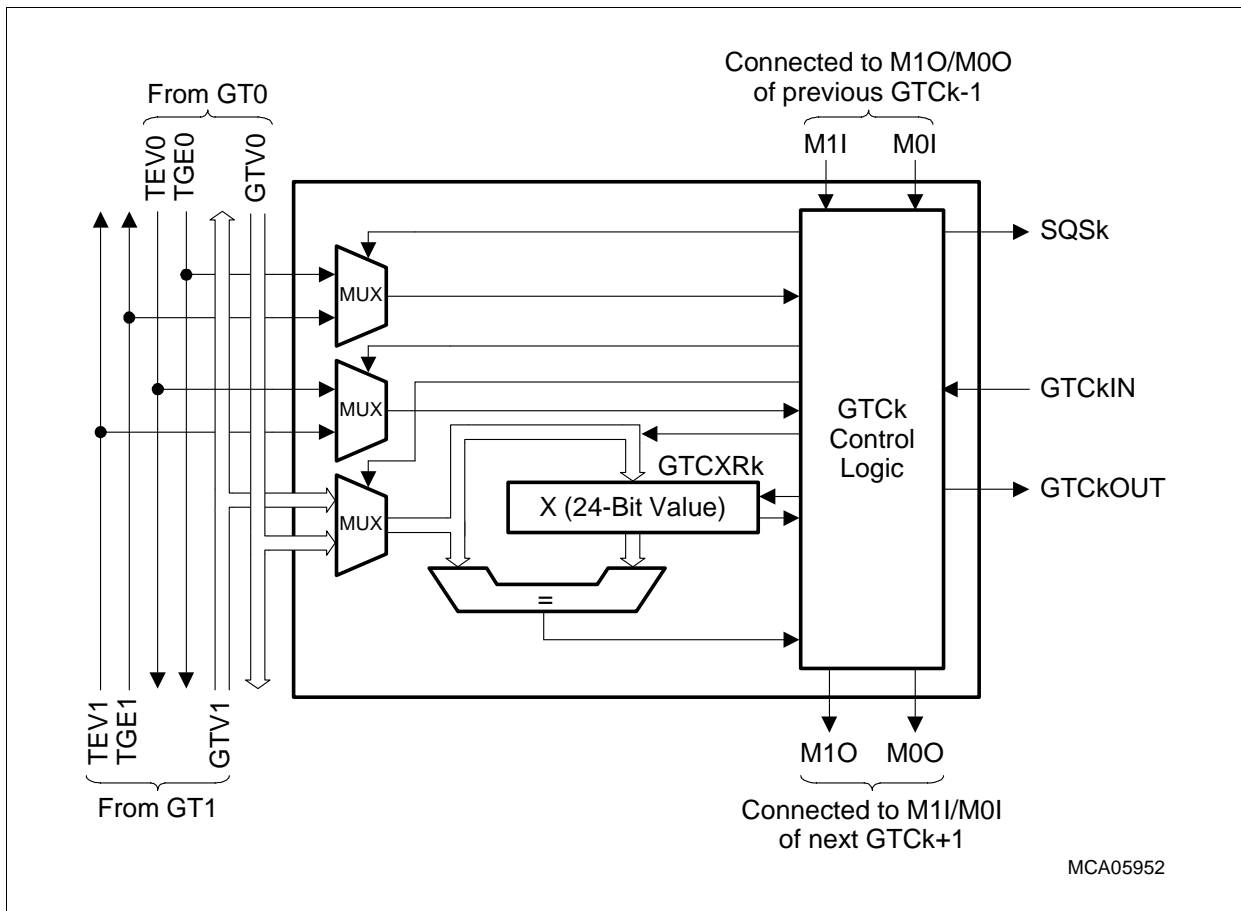


Figure 24-43 Architecture of Global Timer Cells

General Purpose Timer Array (GPTA)

Figure 24-44 shows how the GTCs are arranged and connected to the adjacent GTCs and with the Global Timers GT0 and GT1.

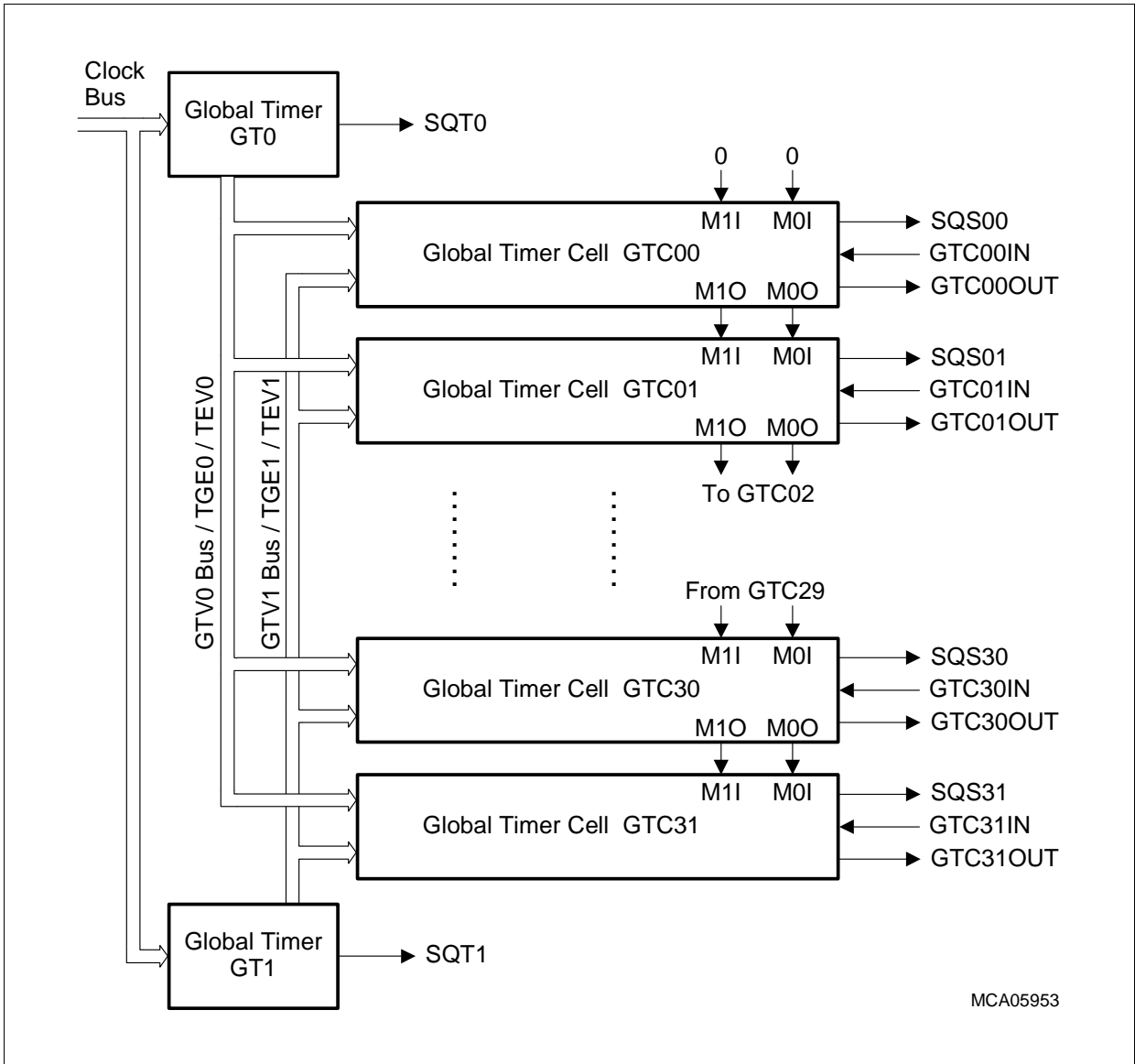


Figure 24-44 GTC Interconnections

Note: Cascading of GTCs is limited. TC1796 specific details are given on [Page 24-272](#).

General Purpose Timer Array (GPTA)**Capture Mode**

The capture function of a GTCK cell is performed on a rising edge (GTCCTRk.REN = 1), a falling edge (GTCCTRk.FED = 1) or both edges of the selected GTCKIN input signal. On the requested event, the GTC:

- Copies the 24-bit value of the selected Global Timer into the 24-bit capture/compare register GTCXkR.X,
- Sets the GTCK service request flag in register SRSS1/SRSC1,
- Activates the service request output SQSk if control register bit GTCCTRk.REN = 1,
- Performs a GTCKOUT output signal line manipulation (set, reset, toggle, unchanged) as defined by bit field GTCCTRk.OCM,
- Transfers an action request, generated by an internal event or received on the M11, M0I input lines, to the M1O, M0O output lines.

Compare Mode

In the Compare Mode of a GTCK cell, several functions can be performed when the value of the selected Global Timer matches and/or exceeds the value stored in register GTCXR. With GTCCTRk.GES = 0 an “Equal Compare” match is selected while GTCCTRk.GES = 1 selects a “Greater Equal Compare” match. On the requested event, the GTC:

- Sets the GTCK service request flag in register SRSS1/SRSC1,
- Activates service request output SQSk if control register bit GTCCTRk.REN = 1,
- Performs a GTCKOUT output signal line manipulation (set, reset, toggle, unchanged) as defined by bit field GTCCTRk.OCM,
- Transfers an action request, generated by an internal event or received on the M11, M0I input lines, to the M1O, M0O output lines.

If a greater or equal compare is selected, the condition is evaluated only when the compare value is written to the GTCXRk register. The user should then ensure that the GTC is already enabled so that the evaluation can take place.

Capture after Compare Mode

When bit GTCCTRk.CAC = 1 and a compare event has occurred, register GTCXR is loaded with:

- The Global Timer value as selected by bit field GTCCTRk.MOD (GTCCTRk.CAT = 0),
- The alternate Global Timer value (GTCCTRk.CAT = 1). If a greater or equal compare match has been detected, the GTCK should be set into One Shot Mode in order to prevent double capturing.

General Purpose Timer Array (GPTA)

One Shot Mode

In One Shot Mode ($GTCCTRk.OSM = 1$), a self-disable of $GTck$ is executed after each GTC event ($GTCCTRk.CEN = 0$). The current state of a $GTck$ can be evaluated by reading the control register flag bit $GTCCTRk.CEN$.

Note: The contents of the $GTck$ capture/compare register $GTCXRk$ are write-protected for Capture_After_Compare in Single Shot Mode. Write protection is activated when the compare value is reached and released after a read access of register $GTCXRk$ occurred.

Data Output Line Control

The data output $GTckOUT$ can be controlled by the $GTck$ itself and by adjacent GTCs with a lower order number. For this purpose, two communication signals between GTCs are available connecting all GTCs via their $M1I/M0I$ inputs and their $M1O/M0O$ outputs, respectively (see [Figure 24-45](#)).

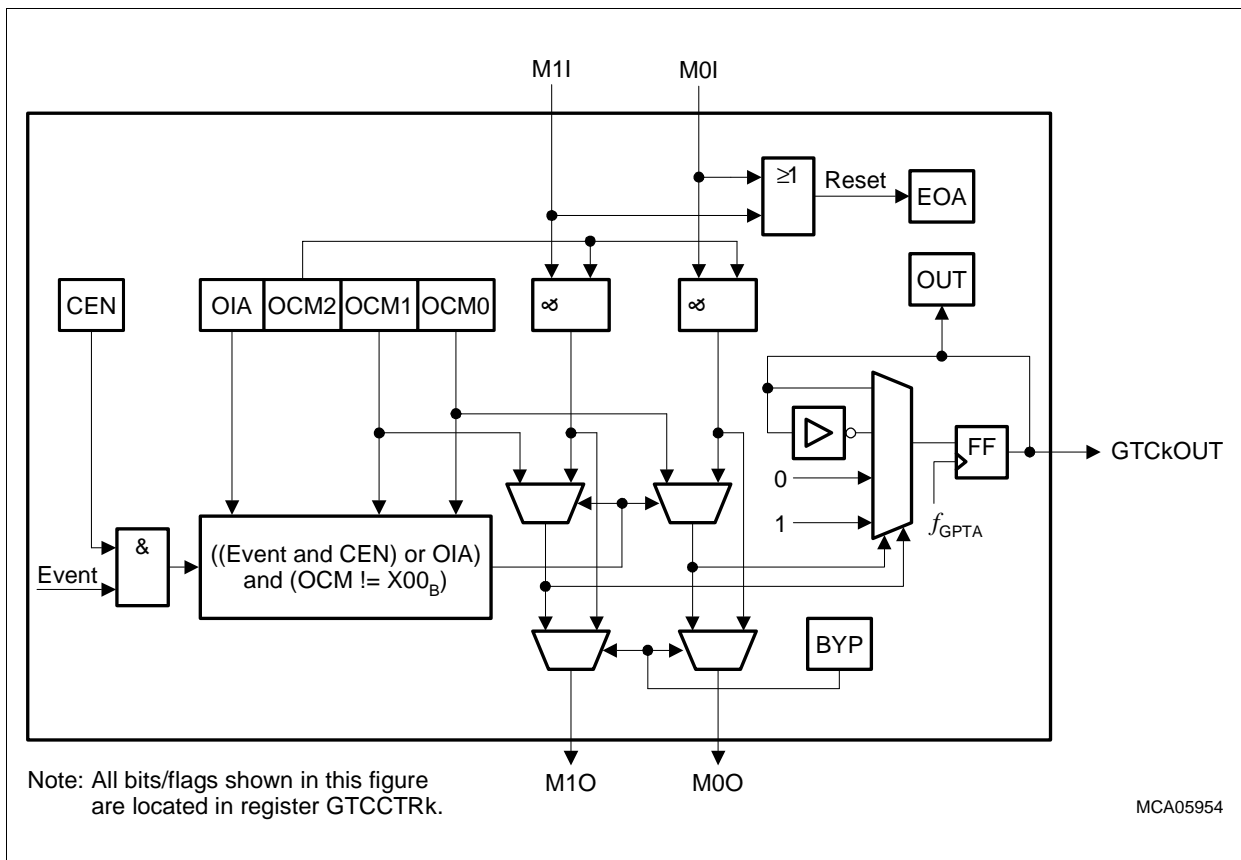


Figure 24-45 GTC Output Operation and Action Transfer

General Purpose Timer Array (GPTA)

When bit GTCCTRk.OCM2 is cleared, the data output GTCKOUT is only controlled by the local GTCK. A set, reset, toggle, or hold operation can be performed as selected by bits GTCCTRk.OCM1 and GTCCTRk.OCM0 ([Table 24-2](#)).

When bit GTCCTRk.OCM2 is set, the data output GTCKOUT is affected either by the local GTCCTRk.OCM1 and GTCCTRk.OCM0 bits or by the M1I/M0I input lines, which are connected to the adjacent GTCK-1 Global Timer output lines M1O/M0O. An enabled GTCK event superimposes an action request generated simultaneously by the M1I/M0I inputs.

When the bypass bit GTCCTRk.BYP is cleared, the M1O/M0O output lines logically OR together the local GTCK events and, if enabled by bit GTCCTRk.OCM2, the action requests received via the M1I/M0I input lines.

When bit GTCCTRk.BYP is set to 1, a local GTCK event will not modify the M1O/M0O output lines.

Table 24-2 Selection of GTC Output Operations and Action Transfer Modes

Bit Field OCM[2:0]	Local Capture or Compare Event	M1O/M0O BYP = 0	M1O/M0O BYP = 1	State of Local Data Output Line
0 0 0	not occurred	0 0	0 0	not modified
	occurred	0 0	0 0	not modified
0 0 1	not occurred	0 0	0 0	not modified
	occurred	0 1	0 0	inverted
0 1 0	not occurred	0 0	0 0	not modified
	occurred	1 0	0 0	0
0 1 1	not occurred	0 0	0 0	not modified
	occurred	1 1	0 0	1
1 0 0	not occurred	M1I M0I	M1I M0I	modified according M1I/M0I
	occurred	M1I M0I	M1I M0I	modified according M1I/M0I
1 0 1	not occurred	M1I M0I	M1I M0I	modified according M1I/M0I
	occurred	0 1	M1I M0I	inverted
1 1 0	not occurred	M1I M0I	M1I M0I	modified according M1I/M0I
	occurred	1 0	M1I M0I	0
1 1 1	not occurred	M1I M0I	M1I M0I	modified according M1I/M0I
	occurred	1 1	M1I M0I	1

The GTCKOUT output line can be connected to output ports, on-chip peripheral inputs, and/or LTC inputs via the I/O Line Sharing Unit (see [Page 24-90](#)). GTCKOUT can be updated directly by software (setting bit GTCCTRk.OIA = 1) or upon a timer, capture or compare event within the local GTCK or a preceding GTC. The current state of the data output line can be evaluated by reading status flag GTCCTRk.OUT.

General Purpose Timer Array (GPTA)**Cell Deactivation**

Normally, the GTCs are always enabled. However, by programming a GTC to Capture Mode with no edge selected ($GTCCTRk.FED = GTCCTRk.RED = 0$), the cell becomes inactive and performs no action, but still passes action commands via the communication link from M1I/M0I to M1O/M0O.

Cell Enabling on Event

A GTC is enabled by writing (ST byte, word, half-word operation) $GTCCTRk.EOA$ (Enable-Of-Action) with 0. Because bit EOA is hardware protected, read-modify-write operations (LDMST, ST.X, SWAP) only enable the GTC if bit EOA is modified from 1 to 0. Alternatively, a GTC can be enabled by an event in a GTC with lower index number. For this purpose, the local event function of a GTC must be disabled by initially setting $GTCCTRk.EOA$. This will clear $GTCCTRk.CEN$ and now a local event cannot affect the GTC. When a preceding GTC generates and communicates an event (or OIA) via its communication link M1O/M0O, at least one of the M1I/ M0I input lines changes its state to 1. This condition clears bit $GTCCTRk.EOA$ of the disabled GTC via the OR gate as shown in [Figure 24-45](#). Now $GTCCTRk.CEN$ is set and the cell is enabled for local events.

It is also possible to enable the following GTC via the communication link for local events. For this purpose, the $GTCCTRk.EOA$ bit of the following GTC must be set, too. If bit $GTCCTRk.OCM2$ of the preceding GTC is 1, the enable action will take place at the same time as in the preceding GTC. Otherwise the GTC will be enabled later on a capture/compare event in the preceding GTC, provided $OCM0$ or $OCM1$ of this GTC is different from 0.

In this way, several GTCs can be enabled at the same time or one after the other. Normally, the cells will be used in One Shot Mode, and an interrupt will be generated after the last event to evaluate the data and to prepare the next enable sequence.

A disabled GTC ($GTCCTRk.CEN = 0$) behaves as an inactive cell.

Logical Operating Units

The inter-cell communication architecture allows implementation of a complex waveform generation to be distributed over several GTCs, controlling a common port pin.

For example, one GTC may be configured in Capture Mode triggered by a rising edge detected on the associated input pin line. The related interrupt service routine can increment the captured timer value by a delay offset and store the result in the $GTCXR$ register of the adjacent GTC configured in Compare Mode. Upon a compare event in the second GTC, the output port line of a third GTC can be set via M1O, M0O interface lines. When the $GTCXR$ register of the third cell is loaded with another compare value by the interrupt service routine related to the second GTC, the output port line may be reset by the next compare event within GTC3.

General Purpose Timer Array (GPTA)

This logical operating unit provides an output signal with programmable pulse width and configurable delay with minimal software overhead.

GTC Service Request

The service request output SQSk of a Global Timer Cell GTck is controlled as shown in **Figure 24-46**. When the GTck service request condition becomes active, the service request flag is always set. The service request output SQSk is only activated if it is enabled by the enable bit GTCCTRk.REN. Additional information about service request and interrupt handling is given on **Page 24-112**.

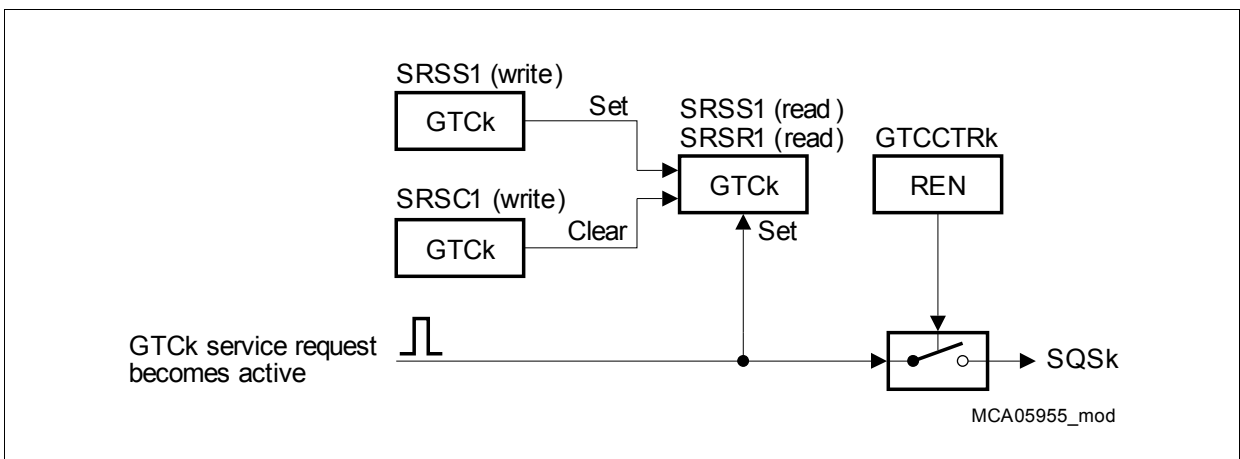


Figure 24-46 GTck Service Request Generation

General Purpose Timer Array (GPTA)

GTC Application Examples

The Global Timers together with GTCs can typically be used for input signal timing analysis of very complex input signals as well as for generation of complex output signals. Figure 24-47 shows a configuration with Global Timer 0 and four GTCs, which is used in the following two examples:

- Example 1: Complex input signal capturing and analyzing
- Example 2: Complex periodical output signal generation

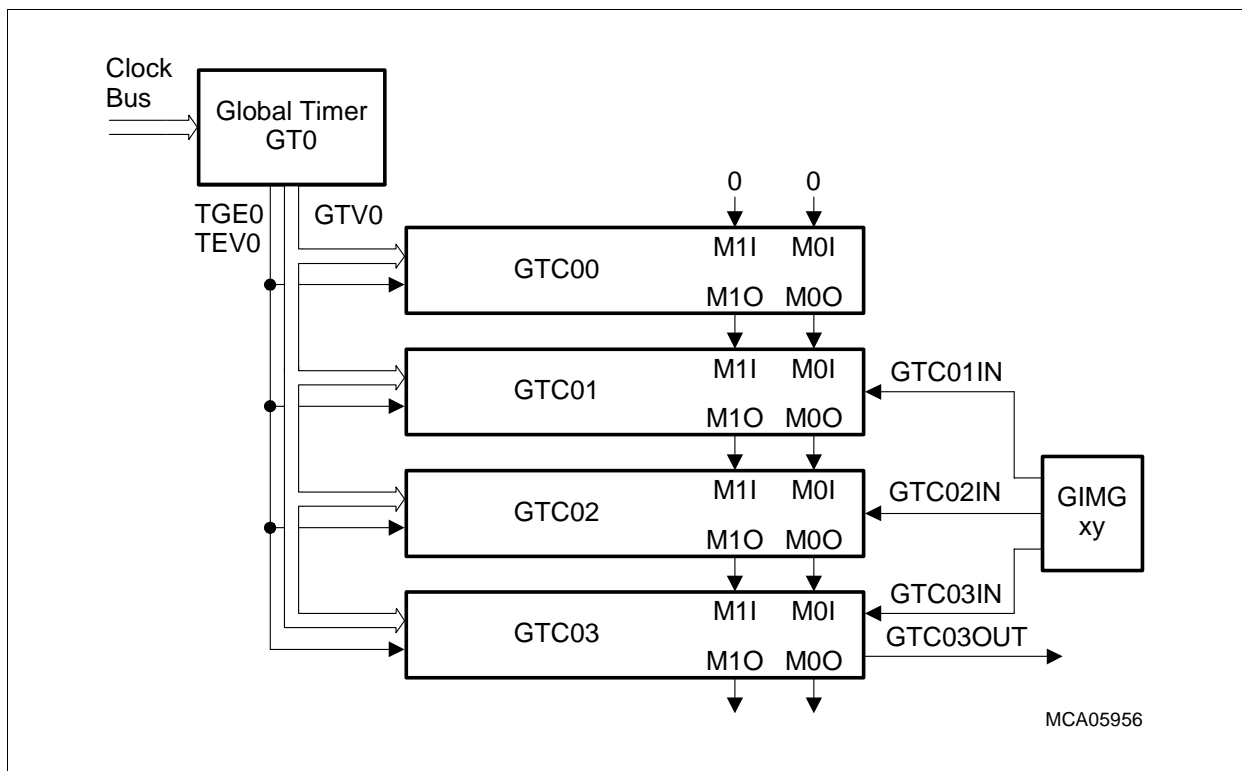


Figure 24-47 Complex Input/Output Signal Capturing/Generation with GTs and GTCs

Complex Input Signal Capturing and Analyzing

In this application example, one input signal from a GTC input multiplexer group becomes analyzed from a timing reference point for three consecutive signal transitions. This common input signal (e.g. a port line) is selected by a GTC input multiplexer group (GIMG) common for GTC01, GTC02, and GTC03 (see also Page 24-100). The GTCs are configured in the following way:

- GT0 operates as free-running up-counting 24-bit timer with reload to GTREV0.REV on overflow. It is clocked by one clock signal from the clock bus.
- GTC00 operates in Compare Mode with timer GT0. The compare match event is reported on the M0O/M1O output lines to the GTC01.

General Purpose Timer Array (GPTA)

- GTC01 operates in Capture Mode at **rising** edge with enable-on-action set (EOA set) and One Shot Mode enabled (OSM set).
- GTC02 operates in Capture Mode at **falling** edge with enable-on-action set (EOA set) and One Shot Mode enabled (OSM set).
- GTC03 operates in Capture Mode at **rising** edge with enable-on-action set (EOA set) and One Shot Mode enabled (OSM set).

With the compare event of GTC00 (time stamp), GTC01 becomes active and waits for the next rising edge at its data input GTC01IN. While GTC01 is active, GTC02 and GTC03 are inactive.

When GTC01 detects a rising edge at its data input, it captures the current GT0 value into its GTCXR01 register, enables GTC02, and becomes disabled afterwards because it was operating in One Shot Mode. When GTC02 detects a falling edge at its data input, it captures the current GT0 value into its GTCXR02 register, enables GTC03, and becomes disabled afterwards because it was operating in One Shot Mode. When GTC03 detects a rising edge at its data input, it captures the current GT0 value into its GTCXR03 register and becomes disabled afterwards because it was operating in One Shot Mode. Optionally, the capture event at GTC03 may generate a service request to indicate that the three capture events have occurred and the captured values can be checked by software. Note that all of these capture events are executed by the GPTA hardware without any software interactions and with a resolution of the GT0 clock rate.

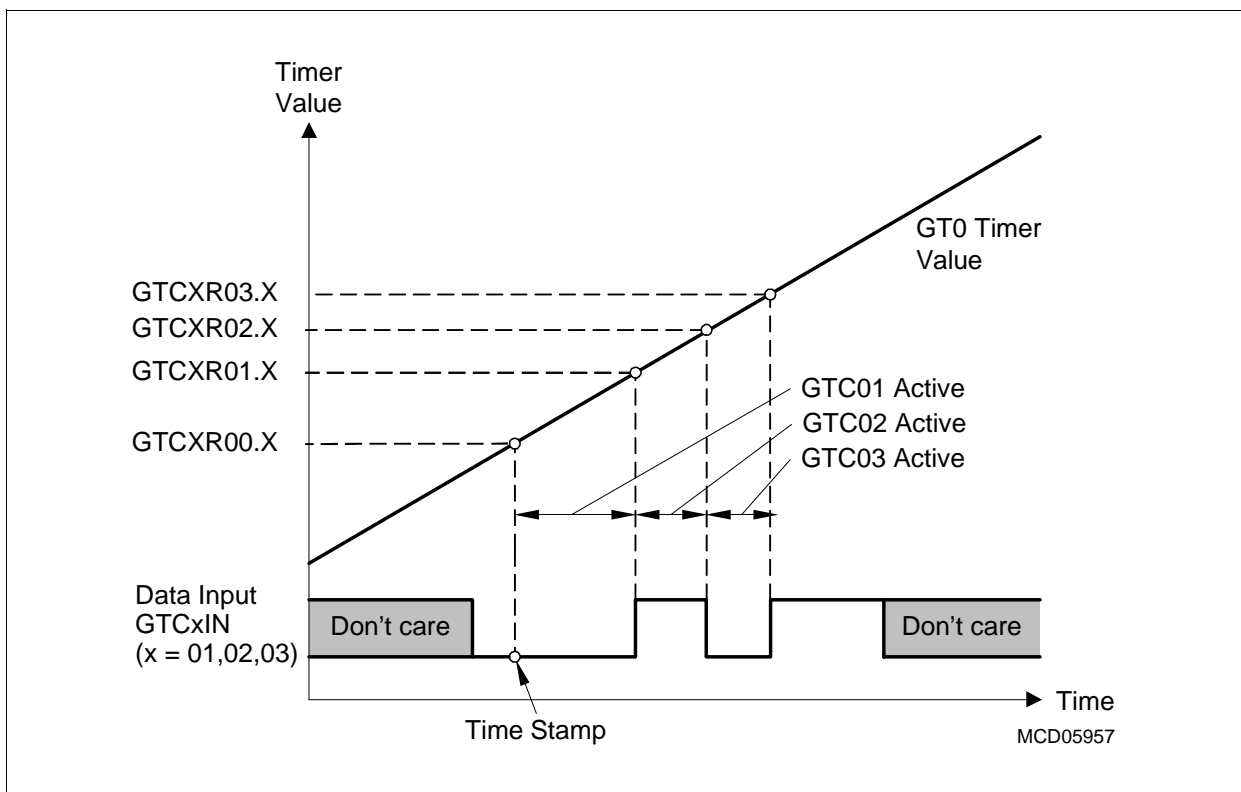


Figure 24-48 Complex Input Signal Analysis/Capturing with GTCs

General Purpose Timer Array (GPTA)

Complex Periodic Output Signal Generation

This application example uses the GT/GTC configuration as shown in [Figure 24-47](#). The generated output signal is available at GTC03OUT. The GTC input signals of the GT/GTC configuration are not used in this example.

- GT0 operates as free-running up-counting 24-bit timer with reload to GTREV0.REV on overflow. It is clocked by a clock signal from the clock bus. Its reload period determines the period of the generated PWM output signal.
- GTC00 operates in Compare Mode with timer GT0 with enable-on-action set (EOA set) and One Shot Mode enabled (OSM set). Furthermore, the output GTC03OUT becomes **set** on a local event (OCM = X11_B).
- GTC01 operates in Compare Mode with timer GT0 with enable-on-action set (EOA set) and One Shot Mode enabled (OSM set). Furthermore, the output GTC03OUT becomes **reset** on a local event (OCM = 110_B).
- GTC02 operates in Compare Mode with timer GT0 with enable-on-action set (EOA set) and One Shot Mode enabled (OSM set). Furthermore, the output GTC03OUT becomes **set** on a local event (OCM = 111_B).
- GTC03 operates in Compare Mode with timer GT0 with enable-on-action set (EOA set) and One Shot Mode enabled (OSM set). Furthermore, the output GTC03OUT becomes **reset** on a local event (OCM = 110_B).

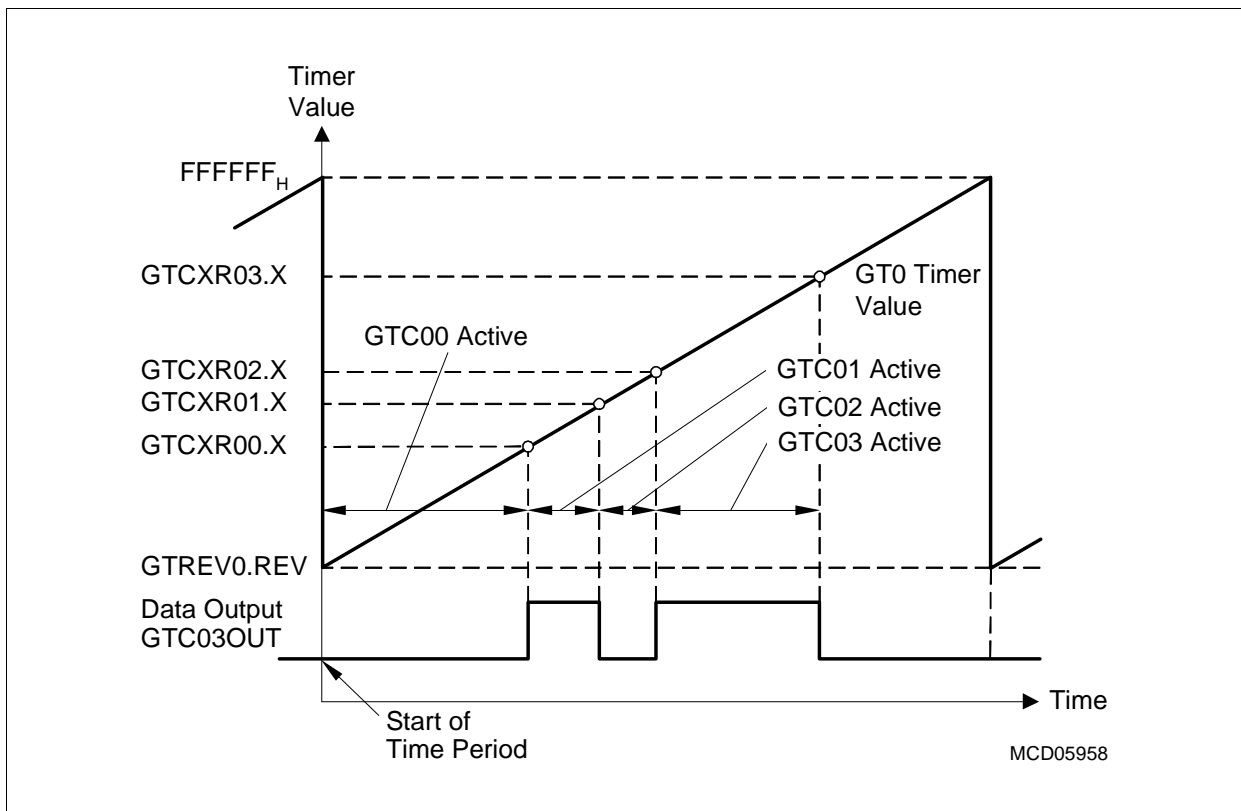


Figure 24-49 Complex Output Signal Generation with GTCs

General Purpose Timer Array (GPTA)

At the start of the time period (reload of GT0), GTC00 becomes active and waits for the compare event. At this event, it sets the output signal GTC03OUT, enables GTC01 for compare operation, and becomes disabled afterwards because it was operating in One Shot Mode. When the GTC01 compare event occurs, the output signal GTC03OUT is reset, GTC02 becomes enabled, and GTC01 becomes disabled because it was operating in One Shot Mode. When the GTC02 compare event occurs, the output signal GTC03OUT is set, GTC03 becomes enabled, and GTC02 becomes disabled because it was operating in One Shot Mode. When the GTC03 compare event occurs, the output signal GTC03OUT is reset, and GTC02 becomes disabled because it was operating in One Shot Mode.

The capture event at GTC03 should generate a service request to indicate that the three compare events have occurred and that GTC01 can be enabled again (setting EOA and OSM). Note that all of the compare events are executed by the GPTA hardware without any software interactions and with a resolution of the GT0 clock rate.

24.2.3.3 Local Timer Cell (LTC00 to LTC62)

LTC00 to LTC62 are functionally identical. The functionality of LTC63 is different to LTC00 to LTC62 and therefore described separately at [Page 24-79](#).

Registers

The following registers are assigned to a Local Timer Cell LTC_k (k = 00-62):

- LTCCTR_k = Local Timer Cell Control Register k (see [Page 24-180](#))
- LTCXR_k = Local Timer Cell X Register k (see [Page 24-190](#))
- SRSC2 = Service Request State Clear Register 2 (see [Page 24-211](#))
- SRSC3 = Service Request State Clear Register 3 (see [Page 24-213](#))
- SRSS2 = Service Request State Set Register 2 (see [Page 24-212](#))
- SRSS3 = Service Request State Set Register 3 (see [Page 24-214](#))

Features

- **16-bit based timer cells** providing capture, compare, and timer functions.
- **Capture Mode** on rising, falling or both edges with following actions:
 - Service request generation
 - Output signal transition generation (set, reset, toggle the output signal).
- **Compare Mode** on equal compare of the corresponding (Reset-)Timer LTC with following actions:
 - Service request generation
 - Output signal transition generation (set, reset, toggle the output signal).
- **Timer Mode** incremented on hardware signal with following actions:
 - Event generation at overflow
 - Service request generation
 - Output signal transition generation (set, reset, toggle the output signal).
- **Reset Timer Mode** allows the selected LTC to be reset by an adjacent cell. Coherent update capability of adjacent LTCs for PWM management is provided.
- **One Shot Mode** allows the selected (capture, compare, timer or reset timer) mode to be stopped after the first event.
- **Flexible mechanism** to link pin actions and allow complex combination of cells. (A cell has the ability to propagate actions over adjacent cells with higher number, in order to perform complex waveforms such as multi channel PWMs).

Architecture

The architecture of an LTC is shown in [Figure 24-50](#). Each LTC has a 16-bit capture/compare register and a 16-bit equal to comparator.

General Purpose Timer Array (GPTA)

The first 63 Local Timer Cells (LTC00 to LTC62) have the following inputs:

- A local input data bus (YI) carrying the local timer value of the adjacent LTC with lower order number (YI of LTC00 is always 0000_H)
- A TI input reporting the occurrence of a local timer value update of the adjacent LTC with lower order number (TI of LTC00 is 0)
- A SI input used by the LTC in Compare Mode as enable line (SI of LTC00 is 0)
- Two action mode inputs (M1I, M0I) coming from the adjacent LTC cell with lower order number (M1I and M0I of LTC00 are 0)
- An EI input reporting an event coming from the adjacent LTC with higher order number
- A trigger/clock/enable input LTCKIN hooked to one of the following signals sources:
 - External port lines
 - GTC00 to GTC31 outputs
 - Clock bus signals
 - PDL0 or PDL1 outputs
 - Internal GPTA kernel input signals INT_x (x = 0-3)

Each LTC provides the following output signals:

- One data output line (LTCKOUT) that can be connected to:
 - External port lines
 - Inputs of an MSC module
 - Outputs and/or inputs of Global Timer Cell inputs
- One LTC prescaler clock input (LTCPRE) for Timer Mode
- One service request line (SQT) triggered by a capture/compare event
- A local output data bus (YO) carrying the local timer value to the adjacent LTC with higher order number or the value on YI
- A TO output reporting the occurrence of a local timer value update to the adjacent LTC with higher order number
- An SO output used by the adjacent LTC with higher order number as enable signal for a compare function
- An EO output reporting the occurrence of a local event to the adjacent LTC with lower order number
- Two mode lines (M1O, M0O) going to the adjacent LTC with higher order number.

General Purpose Timer Array (GPTA)

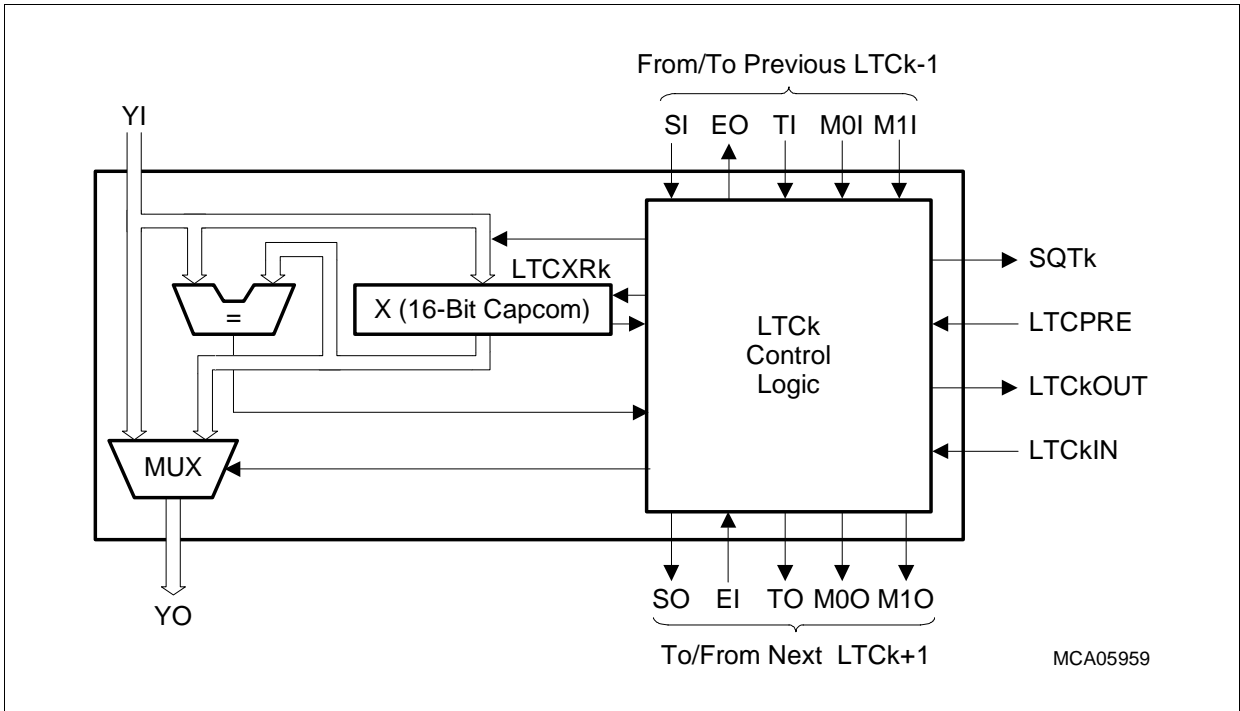


Figure 24-50 Architecture of Local Timer Cells

General Purpose Timer Array (GPTA)

Figure 24-51 shows the arrangement of the LTCs and the connections with adjacent LTCs. LTC63 is a Local Timer Cell that differs from all other LTCs (LTC00 to LTC62). LTC63 is described in detail on Page 24-79.

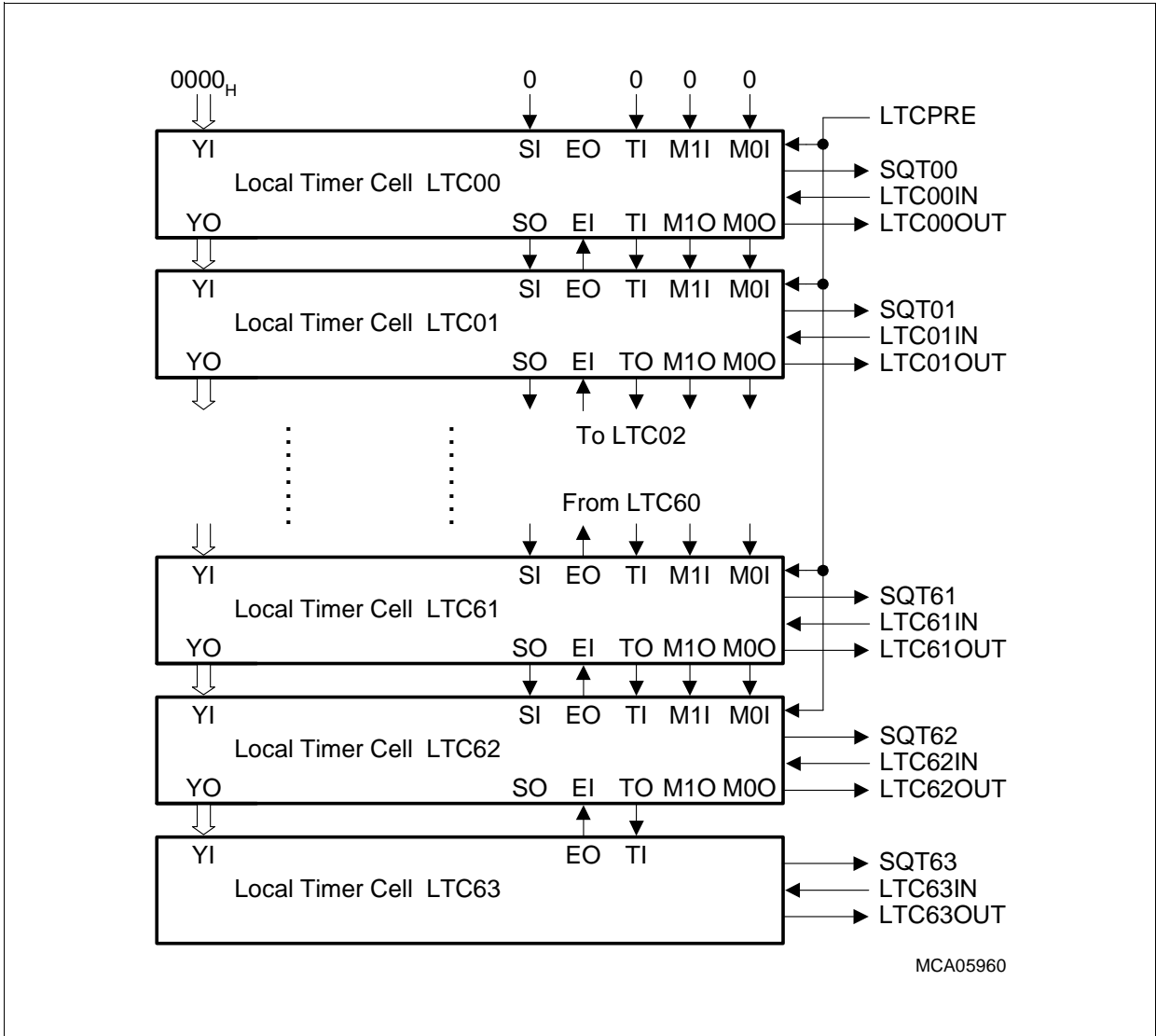


Figure 24-51 Interconnections between the LTCs

Note: Cascading of LTCs is limited. TC1796 specific details are given on Page 24-272.

General Purpose Timer Array (GPTA)**Operating Mode Selection**

The operating mode of an LTC – Free-Running Timer, Reset Timer, Capture, or Compare Mode – is defined by bit field LTCCTRk.MOD.

Free-Running Timer Mode

The content of the Local Timer Cell Register LTCXRk is initialized by a software write operation. LTCXRk is incremented by the selected LTCKIN input signal. Level or Edge Sensitive Mode can be selected for LTCKIN (see [Page 24-73](#)). In Level Sensitive Mode, prescaler clock from the CDU (LTCPRE) can be used to reduce the timer frequency. Every change of the Local Timer Cell Register LTCXRk (increment, reset, or write access) is indicated by output signal TO = 1. When the timer reaches its overflow value (FFFF_H),

- the LTCK service request flag is set,
- the service request output SQTk is activated if control register bit LTCCTRk.REN = 1,
- the LTCK output line LTCKOUT can be altered (set, reset, toggle, unchanged),
- the LTCKOUT output line can be altered (set, reset, toggle, unchanged), depending on bit field LTCCTRk.OCM,
- an action request, generated by an LTCK internal event or received on the M1I/M0I input lines, is transferred via the M1O/M0O output lines to the LTC with higher order number (LTCK+1).

The event output line EO is also activated (set to high) by a software reset when writing FFFF_H to register LTCXRk.

Reset Timer Mode

An LTC that is configured in Reset Timer Mode provides the same functionality as in Free-Running Timer Mode, but is extended by two additional features:

- The Local Timer Cell Register LTCXRk can be reset to FFFF_H via the EI line, which can be activated by an event that occurred in the adjacent LTC with higher order number.
- If bit LTCCTRk.CUD is set to 1, the EI line reset event also toggles the logic state of the SO output line before it clears register bit LTCCTRk.CUD automatically. By accessing register bit LTCCTRk.SLO, the state of the timer's output line SO can be read or explicitly written.

Capture Mode

In Capture Mode, the LTCKIN input signal is used for capture function. Level or edge Sensitive Modes can be selected (see [Page 24-73](#)). On a capture event, the LTCK:

- copies the state of the local input data bus (YI) to the LTCXRk register (LTC00 always copies 0000_H),
- sets the LTCK service request flag,

General Purpose Timer Array (GPTA)

- activates the service request line SQTk, if LTCCTRk.REN is set to 1,
- changes the LTCKOUT output line state (set, reset, toggle, unchanged), depending on bit field LTCCTRk.OCM and the M1I/ M0I input line state,
- generates and/or passes an action request via the M1O/M0O output lines to the LTC with higher order number (LTCK+1),
- sets the event output EO to high level for one f_{GPTA} clock cycle.

Compare Mode

The Compare Mode can be enabled on a low, high, or both levels of the select input line SI (LTCCTRk.SOL = 1, LTCCTRk.SOH = 1). The current state of SI is indicated by bit field LTCCTRk.SLL and can be read. When the value of the local input data bus (YI) matches the LTCXRk contents,

- The LTCK service request flag is set,
- The service request line SQTk is activated if LTCCTRk.REN is set to 1,
- The LTCKOUT output line state is changed (set, reset, toggle, unchanged), depending on bit field LTCCTRk.OCM,
- An action request is generated and/or passed via the M1O/M0O output lines to the LTC with higher order number (LTCK+1),
- The event output EO is set to high level for one f_{GPTA} clock cycle.

Note: To enable the compare function in all cases (on every timer or compare register update caused by a software write access, a reset event or a compare match), bits LTCCTRk.SOL and LTCCTRk.SOH must be set to 1.

An inactive cell (LTCCTRk.SOL = LTCCTRk.SOH = 0, or SI does not match the programmed value) will transfer the state of the event input line EI to the event output line EO.

One Shot Operation

When bit LTCCTRk.OSM is set to 1, a self-disable is executed after each LTC event. The disable state is cleared with the next write access to control register LTCCTRk. The current state of LTCK can be checked by reading the control register flag bit LTCCTRk.CEN.

Note: The contents of register LTCXRk is write-protected for Capture_After_Compare in Single Shot Mode. Write protection is activated when the compare value is reached and is released after an access to register LTCXRk.

General Purpose Timer Array (GPTA)

Data Input Line Control

The data input line LTCkIN can operate in two modes (selected by bit LTCCTRk.ILM):

- Level Sensitive or
- Edge Sensitive.

In Edge Sensitive Mode, the active edges are selected by bits LTCCTRk.FED and LTCCTRk.RED. For the Level Sensitive Mode, the active level of the input signal can be selected by bit FED/AIL of register LTCCTRk.

Depending on which source is selected for the input line by the input multiplexer, different clocking modes of the LTC cell are possible ([Table 24-3](#)).

Table 24-3 LTC Data Input Line Operation (in Timer Mode)

Input Source	Level Sensitive Input Line LTCCTRk.ILM = 1	Edge Sensitive Input Line LTCCTRk.ILM = 0
External Signal (Port line)	The external signal operates as gating signal for the cell. The active input level can be selected with control register bit AIL. Additionally, the LTC prescaler mode can be enabled with LTCCTRk.PEN to reduce the timer frequency. The programmed function of the LTC is performed with the GPTA module clock frequency, or with the programmed prescaler clock LTCPRE (see Page 24-36).	The programmed function of the LTC cell is performed on selected edge(s).
Internal Clock Bus Line or PDL output or INT input	The programmed function is performed with the internal clock or PDL/INT signal. Note that all internal clock bus lines and PDL signals are active high pulses. The LTC Prescaler Mode and the input signal inversion must not be used.	The programmed function of the LTC cell is performed on selected edge(s). In case of full speed GPTA module clock selection as input clock, the Level Sensitive Mode must be selected. The Edge Sensitive Mode will not produce an event in this special case.

General Purpose Timer Array (GPTA)

Table 24-3 LTC Data Input Line Operation (in Timer Mode) (cont'd)

Input Source	Level Sensitive Input Line LTCCTRk.ILM = 1	Edge Sensitive Input Line LTCCTRk.ILM = 0
GTC output	The GTC output signal operates as gating signal for the cell. The active input level can be selected with bit LTCCTRk.AIL. Additionally, the LTC prescaler clock LTCPRE can be enabled with bit LTCCTRk.PEN to reduce the timer frequency.	The programmed function of the LTC cell is performed on selected edge(s).

Note: If Capture Mode and level sensitive input is selected for an LTCk (bit LTCCTRk.ILM = 1), a capture event occurs on every LTC timer clock event if the corresponding LTC input signal is a high level.

General Purpose Timer Array (GPTA)

Data Output Line Control

The data output LTckOUT can be controlled by the LTck itself and by adjacent LTCs with a lower order number. For this purpose, two communication signals between LTCs are available that makes it possible to connect all LTCs via their M11/M0I inputs and their M10/ M0O outputs respectively (Figure 24-52).

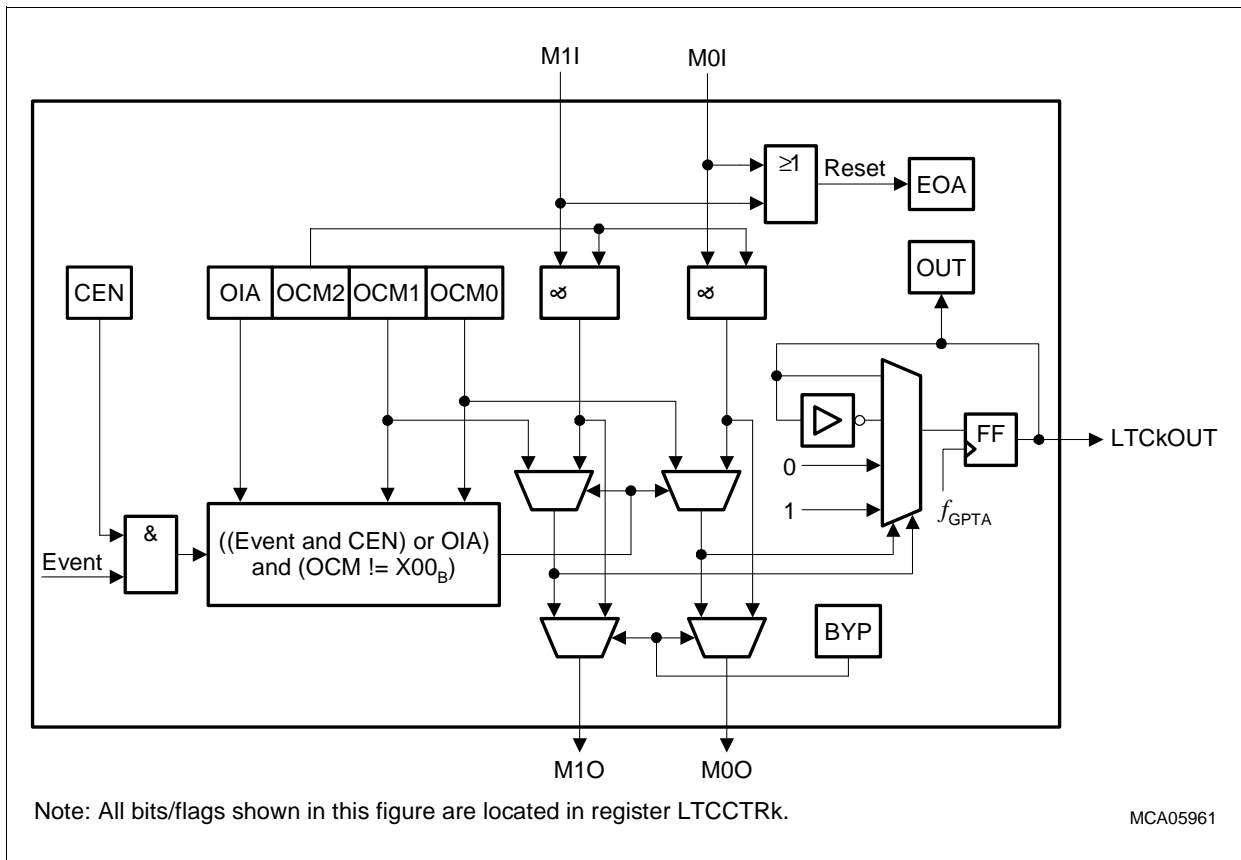


Figure 24-52 LTC Output Operation and Action Transfer

When bit LTCCTRk.OCM2 is cleared, the data output LTckOUT is controlled only by the local LTck. A set, reset, toggle, or hold operation can be performed as selected by bits LTCCTRk.OCM1 and LTCCTRk.OCM0 (see Table 24-4).

When bit LTCCTRk.OCM2 is set, the data output LTckOUT is affected either by the local LTCCTRk.OCM1 and LTCCTRk.OCM0 bits or by the M11/M0I input lines, which are connected to the adjacent LTck-1 Global Timer output lines M10/M0O. An enabled LTck event superimposes an action request generated simultaneously by the M11/M0I inputs.

When the bypass bit LTCCTRk.BYP is cleared, the M10/M0O output lines logically OR together the local LTck events and, if enabled by bit LTCCTRk.OCM2, the action requests received via the M11/M0I input lines. When bit LTCCTRk.BYP is set to 1, a local LTck event will not modify the M10/M0O output lines.

General Purpose Timer Array (GPTA)

Table 24-4 Selection of LTC Output Operations and Action Transfer Modes

Bit Field OCM[2:0]	Local Capture, Compare, or Timer Overflow Event	M1O/M0O BYP = 0	M1O/M0O BYP = 1	State of Local Data Output Line
0 0 0	not occurred	0 0	0 0	not modified
	occurred	0 0	0 0	not modified
0 0 1	not occurred	0 0	0 0	not modified
	occurred	0 1	0 0	inverted
0 1 0	not occurred	0 0	0 0	not modified
	occurred	1 0	0 0	0
0 1 1	not occurred	0 0	0 0	not modified
	occurred	1 1	0 0	1
1 0 0	not occurred	M1I M0I	M1I M0I	modified according M1I/M0I
	occurred	M1I M0I	M1I M0I	modified according M1I/M0I
1 0 1	not occurred	M1I M0I	M1I M0I	modified according M1I/M0I
	occurred	0 1	M1I M0I	inverted
1 1 0	not occurred	M1I M0I	M1I M0I	modified according M1I/M0I
	occurred	1 0	M1I M0I	0
1 1 1	not occurred	M1I M0I	M1I M0I	modified according M1I/M0I
	occurred	1 1	M1I M0I	1

The LTCKOUT output line can be connected to output ports, on-chip peripheral inputs and/or LTC inputs via the I/O Line Sharing Unit (see [Page 24-90](#)). LTCKOUT can be updated directly by software (setting bit LTCCTRk.OIA = 1) or upon a timer, capture, or compare event within the local LTCK or a preceding LTC. The current state of the data output line can be evaluated by reading status flag LTCCTRk.OUT.

Cell Deactivation

Normally, the LTCs are always enabled. However, by programming an LTC to Capture Mode with no edge selected (LTCCTRk.ILM = LTCCTRk.FED = LTCCTRk.RED = 0), the cell becomes inactive and performs no action, but still passes action commands via the communication link from M1I/M0I to M1O/M0O. Output EO is inactive.

Alternatively, the LTC can be deactivated by setting it into Compare Mode with no active select line level (LTCCTRk.SOL = LTCCTRk.SOH = 0) but the communication link remains active. In this mode configuration, EI will be passed to EO.

General Purpose Timer Array (GPTA)**Cell Enabling on Event**

An LTC is enabled by writing (ST byte, word, half-word operation) LTCCTRk.EOA (Enable-Of-Action) with 0 in Capture Mode or Compare Mode. Because bit EOA is hardware protected, read-modify-write operations (LDMST, ST.X, SWAP) only enable the LTC if bit EOA is modified from 1 to 0 in Capture Mode or Compare Mode. If switching to Timer Mode, the LTC cell is enabled. If in Timer Mode every write operation into bit 0-7 will enable the LTC. Alternatively, an LTC can be enabled by an event in an LTC with lower index number. For this purpose, the local event function of an LTC must be disabled by setting LTCCTRk.EOA initially. This will clear LTCCTRk.CEN and now a local event cannot affect the LTC. When a preceding LTC generates and communicates an event (or OIA) via the communication link M1O/M0O, at least one of the M1I/M0I input lines changes its state to 1. This condition clears bit LTCCTRk.EOA of the disabled LTC via the OR gate as shown in [Figure 24-52](#). Now LTCCTRk.CEN is set and the LTC is enabled for local events.

It is also possible to enable the following LTC via the communication link for local events. For this purpose, the bit LTCCTRk.EOA of this cell must be set, too. If bit LTCCTRk.OCM2 of the preceding cell is 1, the enable action will take place at the same time as in the preceding cell. Otherwise, the LTC will be enabled later on a capture/compare event in the preceding LTC, provided LTCCTRk.OCM0 or LTCCTRk.OCM1 of this cell is different from 0.

In this way, several LTCs can be enabled at the same time or one after the other. Normally, the LTCs will be used in One Shot Mode, and a service request will be generated after the last event to evaluate the data and to prepare the next enable sequence.

A disabled LTC (LTCCTRk.CEN = 0) behaves as an inactive capture LTC.

Logical Operating Units

The inter-cell communication architecture allows concatenation of several LTCs to a logical unit. A logical unit contains any number of LTCs communicating via M1 and M0 lines and ends at an LTC disabled for action input or transfer (such as an LTC configured as timer, reset timer or LTC initiated with LTCCTRk.OCM2 = 0).

Therefore, the LTC with the lowest order number should be configured in Reset Timer Mode, thus providing all other LTCs of the logical unit with a time base (YO) and a compare enable signal (SO). Another LTC of the same logical unit can be initiated in Compare Mode to reset the LTC via its event output line EO, when a programmed threshold value is reached (register LTCXR) and the current state of its select line input SI matches the condition selected by the LTCCTRk bits SOH/SOL. Additional LTCs of the same logical unit can operate in Capture Mode triggered by a rising edge, falling edge, or both edges of a GPTA input line or a clock line of the clock bus. On the generated event, these LTCs capture the current contents of the timer cell, can generate

General Purpose Timer Array (GPTA)

a service request, can perform a manipulation of a GPTA output line (set, reset or toggle), and can also reset the LTC via the event output line EO.

LTC Service Request

The service request output SQTk of a Local Timer Cell LTck is controlled as shown in **Figure 24-53**. When the LTck service request condition becomes active, the service request flag becomes always set. The service request output SQTk is only activated if it is enabled by the enable bit LTCCTRk.REN. Additional information about service request and interrupt handling is given on **Page 24-112**.

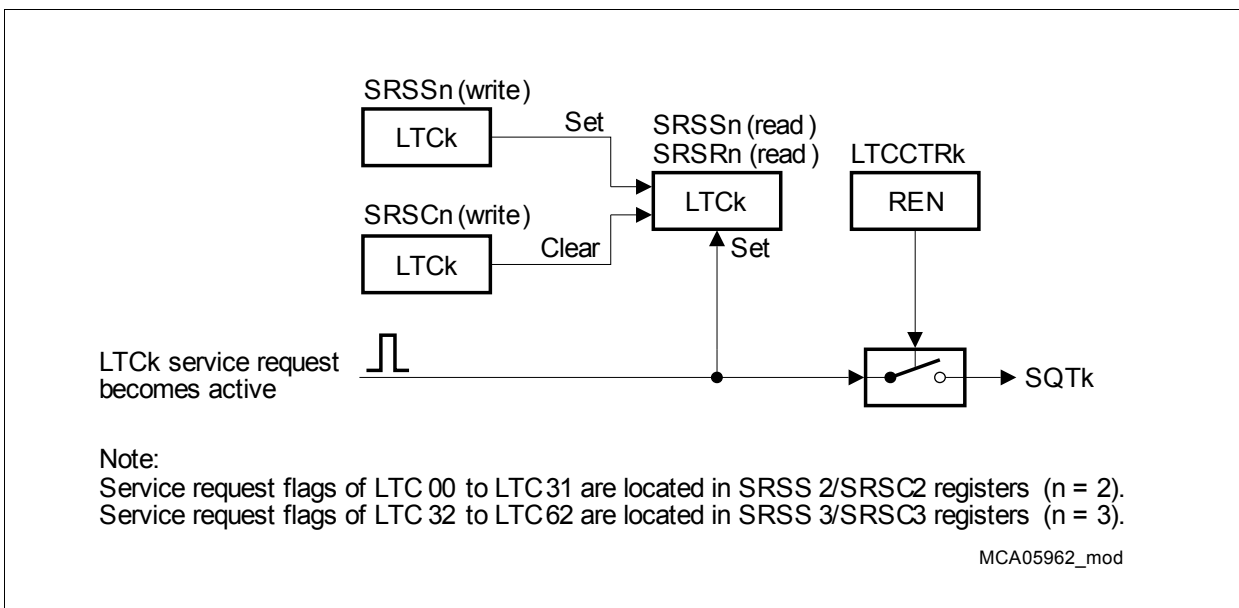


Figure 24-53 LTck Service Request Generation

24.2.3.4 Local Timer Cell LTC63

Registers

The following registers are assigned to Local Timer Cell LTC63:

- LTCCTR63 = Local Timer Cell Control Register 63 (see [Page 24-188](#))
- LTCXR63 = Local Timer Cell X Register 63 (see [Page 24-190](#))
- SRSC3 = Service Request State Clear Register 3 (see [Page 24-213](#))
- SRSS3 = Service Request State Set Register 3 (see [Page 24-214](#))

Features

The GPTA Local Timer Cell array has one special cell, LTC63, which provides the following special features:

- **Compare Mode** on greater equal compare of the last timer, 16-bit based with following actions:
 - Service request generation
 - Output signal transition generation (set, reset, toggle the output signal).
- **Bit Reversal Mode:**
 - Timer can be selected to enable a special PWM Mode, called pulse count modulation (PCM)
- **Compare Value Switching** can be triggered by a hardware signal. This function can generate a service request. One Shot Mode makes it possible to stop the function after the first event.

Architecture

LTC63 is locally equipped with a 16-bit compare register, a 16-bit shadow register and a 16-bit greater comparator ([Figure 24-54](#)).

The LTC63 has the following inputs:

- A local input data bus (YI) carrying the local timer value of the adjacent LTC with lower order number
- A TI input reporting the occurrence of a local timer value update of the adjacent LTC with lower order number
- A trigger/enable input LTCKIN for compare value switching hooked to one of the following signals sources:
 - External port lines
 - GTC00 to GTC31 outputs
 - Clock bus signals
 - PDL0 or PDL1 outputs
 - Internal GPTA kernel input signals INTx (x = 0-3)

General Purpose Timer Array (GPTA)

The LTC63 provides the following output signals:

- One data output line (LTCkOUT) that can be connected to:
 - External port lines
 - Inputs of an MSC module
 - Outputs and/or inputs of Global Timer Cell inputs
- One service request line (SQT) triggered by a compare or copy event
- An EO output reporting the occurrence of a local event to the adjacent LTC with lower order number

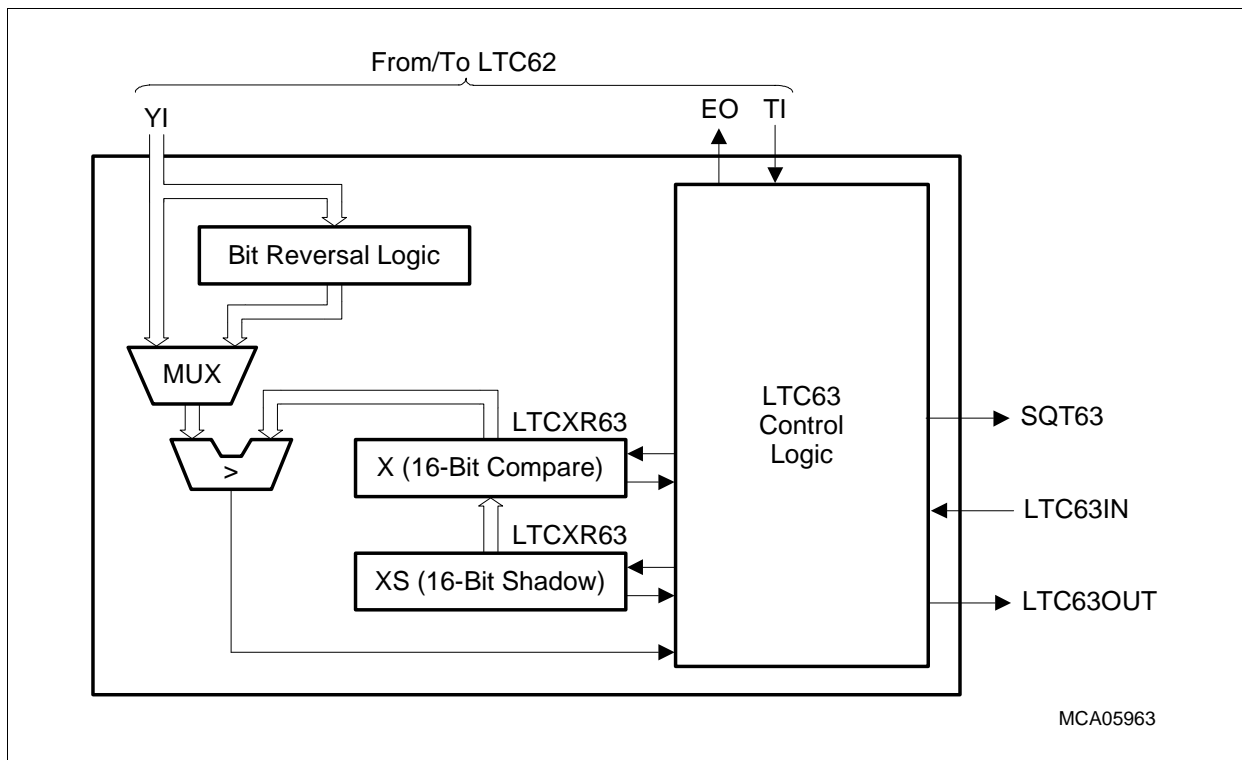


Figure 24-54 Architecture of Local Timer Cell 63

Compare

The compare function is always enabled. As long as the 16-bit compare value LTCXR63.X is greater than the timer value provided at YI, the comparator output signal is 1. The timer value at YI comes from the LTC62 either in original or in reversed order (Bit0 <-> Bit15, Bit1 <-> Bit14, etc.). The greater comparator output is connected directly to the output line LTC63OUT.

The 16-bit compare value LTCXR63.X is never greater than the LTC timer value (FFFF_H) coming from YI and which is used on LTC timer reset. Without special measures, a duty cycle of 100% cannot be achieved. There is always one LTC timer clock missing. Therefore, additional logic generates a permanent high signal whenever the 16-bit compare value LTCXR63.X is FFFF_H.

General Purpose Timer Array (GPTA)

When the comparator output signal changes from 1 to 0,

- the service request flag LTC63 is set,
- an interrupt request will be activated if enabled by bit field LTCCTR63.REN,
- the event output line EO is set to high level for one f_{GPTA} clock cycle.

As well as the 16-bit compare register LTCXR63.X, the LTC63 also contains a 16-bit shadow register LTCXR63.XS. Both 16-bit registers are combined in the 32-bit register LTCXR. On an LTC input signal selected via the LTC input multiplexer, the contents of the shadow register are copied to the compare register.

Standard PWM Mode

The LTC63 can be used for standard PWM duty cycle generation with enhanced update features. For this purpose, a pair of LTCs with lower index is configured as reset timer/period compare register. The user must set the period compare register to the desired period - 2 and LTC63 to the desired duty cycle. With LTCCTR63.BRM = 0 (Bit Reversal Mode), timer bit reversal is disabled. LTC63 is used for standard PWM Mode but with enhanced update features due to the “greater” comparator. The compare register LTCXR63.X can be written on-the-fly. If the duty cycle is changed at an arbitrary time, the actual duty cycle for the current period will reflect the old duty cycle, the new one, or a mixture of both. A duty cycle of 100% will be generated if the compare register is set to $FFFF_H$.

Pulse Count Modulation Mode (PCM)

With a period of 100 clocks and a duty cycle of 64%, standard PWM will produce an output signal that is ON for 64 clock cycles and OFF for the remaining 36 clock cycles. In contrast, pulse count modulation will generate 64 ON pulses and 36 OFF pulses distributed over the whole period as evenly as possible. PCM offers higher output frequency than standard PWM. This allows faster settling time e.g. when building a D/A converter in conjunction with an external low-pass filter. Only for very short or very long duty cycles does the method show no advantage or just little advantage compared to standard PWM.

As with standard PWM, a pair of LTCs with lower index is configured as reset timer/period compare register and LTC63 is used as duty cycle compare register. But now, Bit BRM (Bit Reversal Mode) in the LTCCTR63 register is set to 1 which enables the timer bit reversal to activate PCM.

The algorithm will also work if fewer than 16 timer bits are effectively used, even if the period is not a power of two. In any case, the user must write the duty cycle in unsigned 16-bit fractional format to the compare register.

General Purpose Timer Array (GPTA)

Figure 24-55 shows an PCM example for an effective period of 6 clocks.

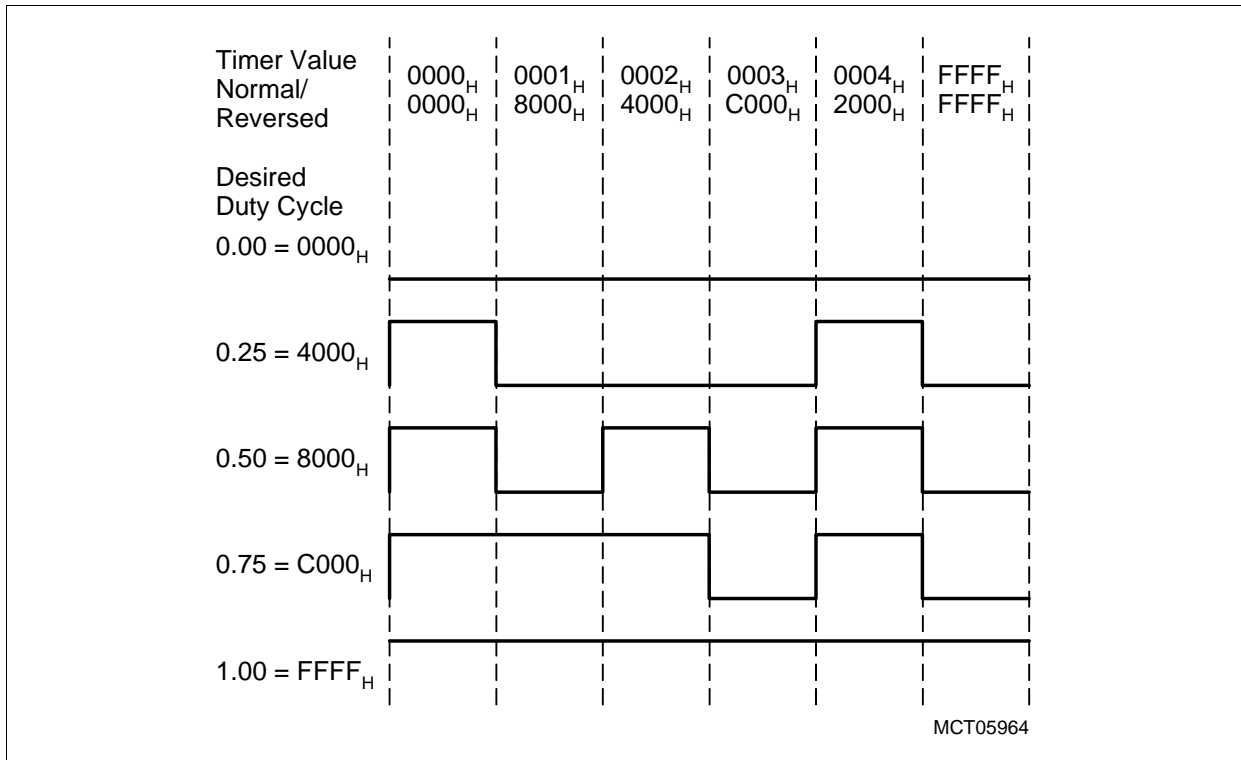


Figure 24-55 Pulse Count Modulation Example 1

Table 24-5 shows the rounding behavior for a period of 100 clocks.

Table 24-5 Implicit PCM Rounding

Desired Duty Cycle	Expected ON Pulses	Actual ON Pulses
0.000 = 0000	0	0
0.100 = 199A _H	10	11
0.500 = 8000 _H	50	50
0.800 = CCCD _H	80	82
0.900 = E666 _H	90	90
0.999 = FFBE _H	100	99
1.000 = FFFF _H	100	100

The output is OFF for the remaining cycles of the period. The worst case error is approximately +2/-1 ON pulses. A subtraction performed via software may be used to reduce the worst case error.

If the duty cycle is changed at an arbitrary time, the actual duty cycle for the entire current period will reflect the old duty cycle, the new one, or a mixture of both.

General Purpose Timer Array (GPTA)

Figure 24-56 shows another PCM example that demonstrates the difference between a standard PWM signal and the derived PCM signal. During one PWM period (128 clock cycles), the standard PWM signal is ON for 8 clock cycles and OFF for the remaining 120 clock cycles (duty cycle of 6.25%). The PCM signal operates with a PCM duty cycle of $1/8 = 0.125$ resulting in 8 ON pulses of 1 clock cycle width within 1 PWM period.

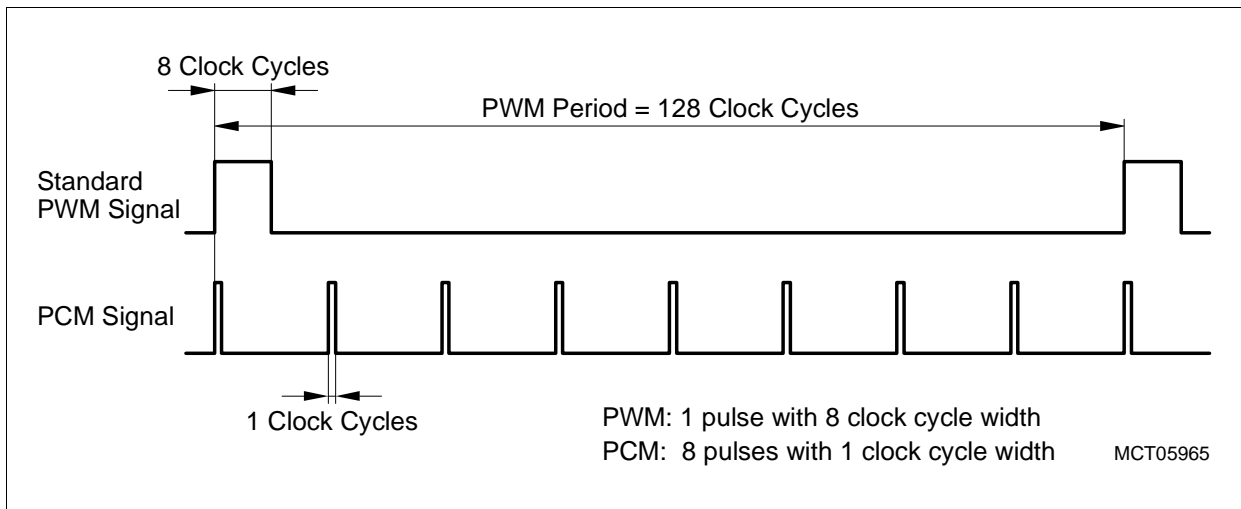


Figure 24-56 Pulse Count Modulation Example 2

Compare Value Switching

In both pulse modulation modes, it is possible to change the duty cycle either by software or on an LTC input signal. LTC63 contains two registers, the compare register and a shadow register. For software access, the compare register LTCXR63.X (= 16-bit low part of LTCXR63) is written directly.

For compare value switching triggered by hardware, the shadow register LTCXR63.XS (= 16-bit high part of LTCXR63) is pre-loaded with the desired duty cycle. On an LTC input signal selected via the LTC input multiplexer,

- The shadow register content LTCXR63.XS is copied to the compare register LTCXR63.X,
- The LTC63 service request flag is set,
- An interrupt request will be activated if enabled by bit field LTCCTR63.REN.

The data input line LTC63IN can operate in two modes (selected by bit LTCCTR63.ILM):

- Level Sensitive Mode or
- Edge Sensitive Mode

In Edge Sensitive Mode, the active edges are selected by bits LTCCTR63.FED and LTCCTR63.REN. In Level Sensitive Mode, the data input line LTC63IN is sensitive on a high level.

General Purpose Timer Array (GPTA)

Various clocking modes of the LTC63 copy function are possible, depending on the source selected for the input line by the input multiplexer (see [Table 24-6](#)).

Table 24-6 LTC63 Data Input Line Operation

Input Source	Level Sensitive Input Line	Edge Sensitive Input Line
External Signal (Port line)	The external signal operates as gating signal for the cell. If the input is high the copy function of the LTC cell is performed with each rising edge of the GPTA module clock f_{GPTA} .	The copy function of the LTC cell is performed on selected edge(s).
Internal Clock Bus Line or PDL output or INT input	The copy function is performed with the internal clock or PDL/INT signal.	The copy function of the LTC cell is performed on selected edge(s). In case of full speed GPTA module clock selection, the Level Sensitive Mode must be selected. The Edge Sensitive Mode will not produce an event in this special case.
GTC output	The GTC output signal operates as gating signal for the cell. If the input is high the copy function of the LTC cell is performed with each rising edge of the GPTA module clock f_{GPTA} .	The copy function of the LTC cell is performed on selected edge(s).

When bit LTCCTR63.OSM is set to 1, a self-disable is executed after each copy event (PWM is not affected). The current state of the LTC copy enable may be evaluated by reading the control register flag bit LTCCTR63.CEN.

The output can be switched immediately to 0 or 1 in any pulse modulation mode by writing 0000_H or FFFF_H to the duty cycle compare register LTCXR63.X.

LTC63 Service Request

The service request SQT63 can be generated by one of the following events:

- Comparator output changes from 1 to 0 (this makes sense mainly for standard PWM),
- Copy event.

Bit combinations 01_B and 10_B of bit field LTCCTR63.REN selects one of the two service request sources and enables it. Output SQT63 becomes active in these two cases. With

General Purpose Timer Array (GPTA)

the other two bit combinations of bit field LTCCTR63.REN (00_B, 11_B), the SQT63 output will not be activated. The LTC63 service request flag SRSS3.LTC63 will be set on a service request independently of LTCCTR63.REN. Additional information on service request and interrupt handling is given on [Page 24-112](#).

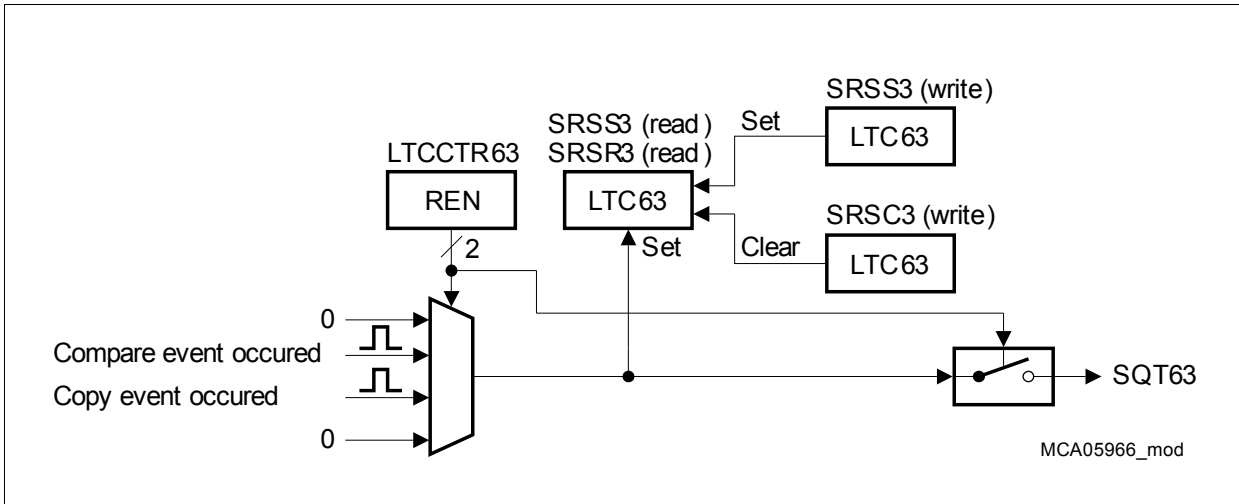


Figure 24-57 LTC63 Service Request Generation

24.2.3.5 LTC Application Examples

This section describes a PWM generation example using Local Timer Cells.

Fully Programmable PWM Signal Generation with 5 LTCs

As shown in [Figure 24-58](#), a logical unit of five LTCs can be used to generate a PWM signal with a programmable duty cycle, period length, and fully coherent update of the period and duty cycle. In this example, LTC00 up to LTC04 are used to produce a PWM signal at the output of LTC04.

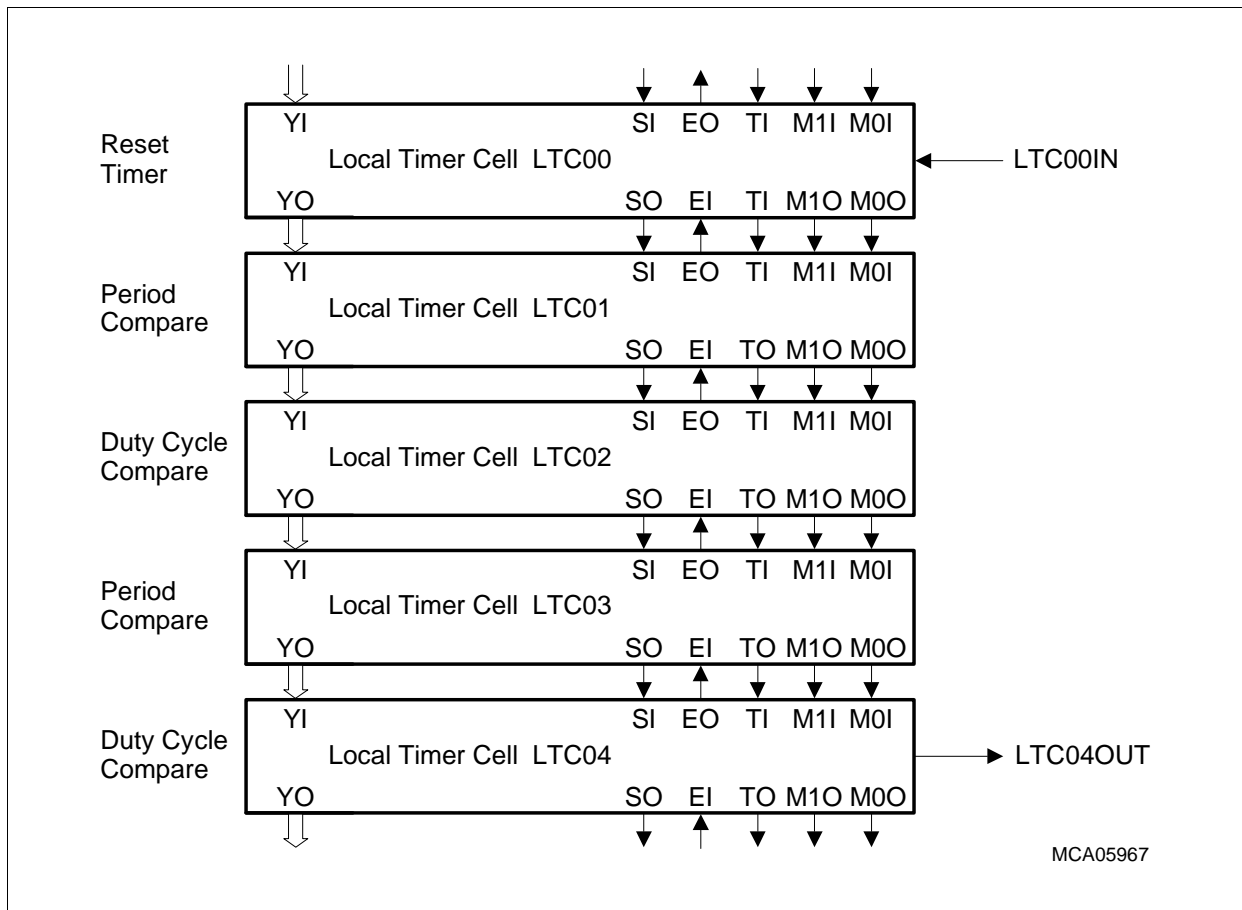


Figure 24-58 PWM Signal Generation with LTCs (Full Coherent Update)

LTC00 is configured in Reset Timer Mode thus providing all subsequent cells with a time base. LTC00 is clocked by a clock signal at the LTC00IN which has been selected by the LTC input multiplexer.

LTC01 and LTC02 are configured in Compare Mode. They are enabled if its SI inputs are at low level and responsible for the LTC04OUT signal generation in Phase 1. With the programmed values from [Table 24-7](#), the LTC04OUT signal of Phase 1 has a period of $1000_D (= 3E8_H)$ clocks of the LTC00IN clock signal and a duty cycle of 20% ($= 200_D$ or $C8_H$).

General Purpose Timer Array (GPTA)

LTC01 is configured in such a way ($LTCCTR01.OCM = 011_B$) that its output LTC01OUT is set to 1 whenever the LTC00 timer value LTCXR00.X is equal to the LTC01 compare value LTCXR01.X.

LTC02 is configured in such a way ($LTCCTR02.OCM = 110_B$) that its output LTC02OUT is reset whenever the LTC00 timer value LTCXR00.X is equal to the LTC02 compare value LTCXR02.X or it copies the action from the previous LTC01.

LTC03 and LTC04 are configured in Compare Mode. They are enabled if its SI inputs are at high level and are responsible for the LTC04OUT signal generation in Phase 2. With the programmed values from [Table 24-7](#), the LTC04OUT signal of Phase 2 has a period of $2000_D (= 7D0_H)$ clocks of the LTC00IN clock signal and a duty cycle of 75% ($= 1500_D$ or $5DC_H$).

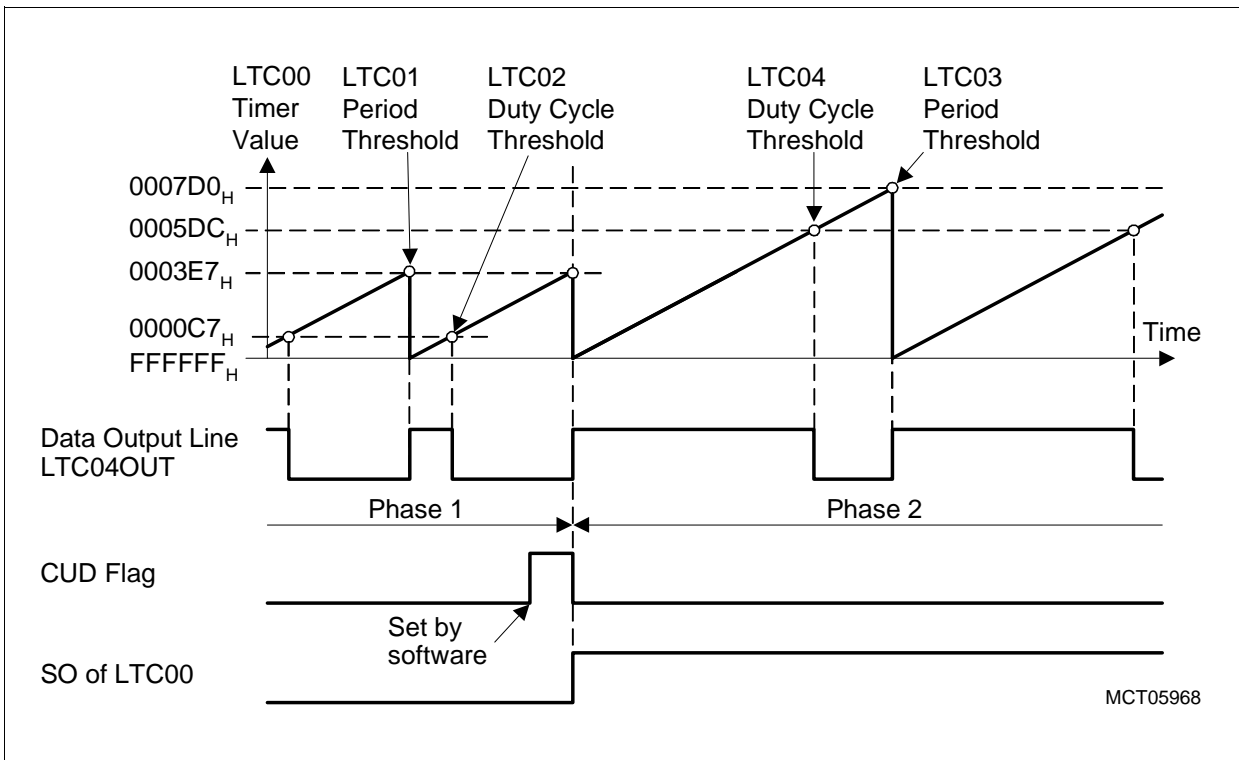


Figure 24-59 Internal Signal States of the PWM Signal Generation with 5 LTCs

LTC00 to LTC05 for the PWM example must be configured as defined in [Table 24-7](#).

General Purpose Timer Array (GPTA)

Table 24-7 Programming Values for PWM Signal Generation with 5 LTCs

Register	Value	Function
LTC00 Configuration Setup		
GPTA0_LTCXR00	0000 0000 _H	LTC00 data register value = 0
GPTA0_LTCCTR00	0000 0413 _H	MOD = 11 _B : Reset Timer Mode selected OSM = 0: LTC00 continuously enabled ILM = 0, RED = 1, FED = 0: Input LTC00IN operates in Edge Sensitive Mode with rising edge; one clock bus signal is selected via the LTC input multiplexer SLO = 0: state of select line output SO is 0 CEN = 0: enable LTC00 for local events OCM = 000 _B : hold LTC00OUT state
LTC01 Configuration Setup		
GPTA0_LTCXR01	0000 03E7 _H	Load compare value = 3E7 _H = 999 _D
GPTA0_LTCCTR01	0000 5C11 _H	MOD = 01 _B : Compare Mode with LTC00 selected OSM = 0: LTC01 continuously enabled SOH = 0, SOL = 1: compare enabled by low level at SI BYP = 0: bypass in LTC02 is disabled EOA = 0: LTC02 enabled for local events OCM = 011 _B : set LTC01OUT by a local event only OIA = 1: output action defined by OCM must be performed immediately
LTC02 Configuration Setup		
GPTA0_LTCXR02	0000 00C7 _H	Load compare value = C7 _H = 199 _D
GPTA0_LTCCTR02	0000 3411 _H	MOD = 01 _B : Compare Mode with LTC00 selected OSM = 0: LTC02 continuously enabled SOH = 0, SOL = 1: compare enabled by low level at SI BYP = 0: bypass in LTC02 is disabled EOA = 0: LTC02 enabled for local events OCM = 110 _B : reset LTC02OUT by a local event or copy the previous cell action OIA = 0: no immediate output action required

General Purpose Timer Array (GPTA)

Table 24-7 Programming Values for PWM Signal Generation with 5 LTCs (cont'd)

Register	Value	Function
LTC03 Configuration Setup		
GPTA0_LTCXR03	0000 07CF _H	Load compare value = 7CF _H = 1999 _D
GPTA0_LTCCTR03	0000 7C21 _H	MOD = 01 _B : Compare Mode with LTC00 selected OSM = 0: LTC03 continuously enabled SOH = 1, SOL = 0: compare enabled by high level at SI BYP = 0: bypass in LTC03 is disabled EOA = 0: LTC03 enabled for local events OCM = 111 _B : set LTC03OUT by a local event or copy the previous cell action OIA = 1: output action defined by OCM must be performed immediately
LTC04 Configuration Setup		
GPTA0_LTCXR04	0000 05DB _H	Load compare value = 5DB _H = 1499 _D
GPTA0_LTCCTR04	0000 3421 _H	MOD = 01 _B : Compare Mode with LTC00 selected OSM = 0: LTC04 continuously enabled SOH = 1, SOL = 0: compare enabled by high level at SI BYP = 0: bypass in LTC04 is disabled EOA = 0: LTC04 enabled for local events OCM = 110 _B : reset LTC04OUT by a local event or copy the previous cell action OIA = 0: no immediate output action required

General Purpose Timer Array (GPTA)

24.2.4 Input/Output Line Sharing Unit (IOLS)

The I/O Line Sharing Unit allows the 56 inputs and 112 outputs of the GPTA unit to be routed with high flexibility between I/O lines, output lines, clock inputs, other on-chip peripherals and other GPTA cells. The GPTA module provides a total of 56 input lines and 112 output lines, assigned to seven I/O groups IOG[6:0] and seven output groups OG[6:0].

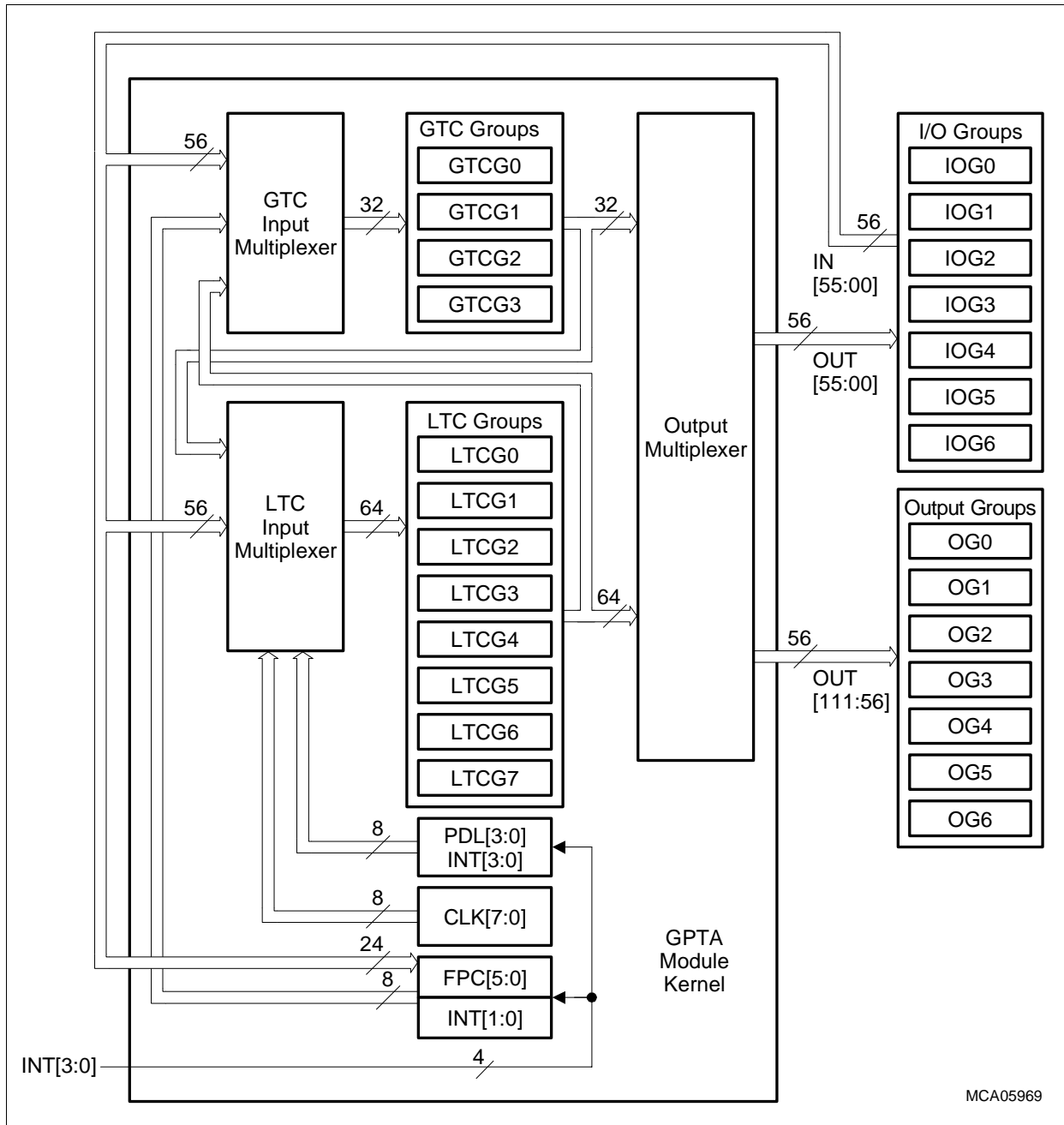


Figure 24-60 Input/Output Line Sharing Unit Overview

General Purpose Timer Array (GPTA)

The I/O Line Sharing Unit does the following selections:

- FPC input line selection
- GTC and LTC output multiplexer selection
- GTC input multiplexer selection
- LTC input multiplexer selection

For choosing these selection, the input and output lines of the related cells are integrated into groups with eight parts each. Seven I/O groups, seven output groups, four GTC groups, eight LTC groups, one clock group, one FPC/INT group, and one PDL/INT group are defined.

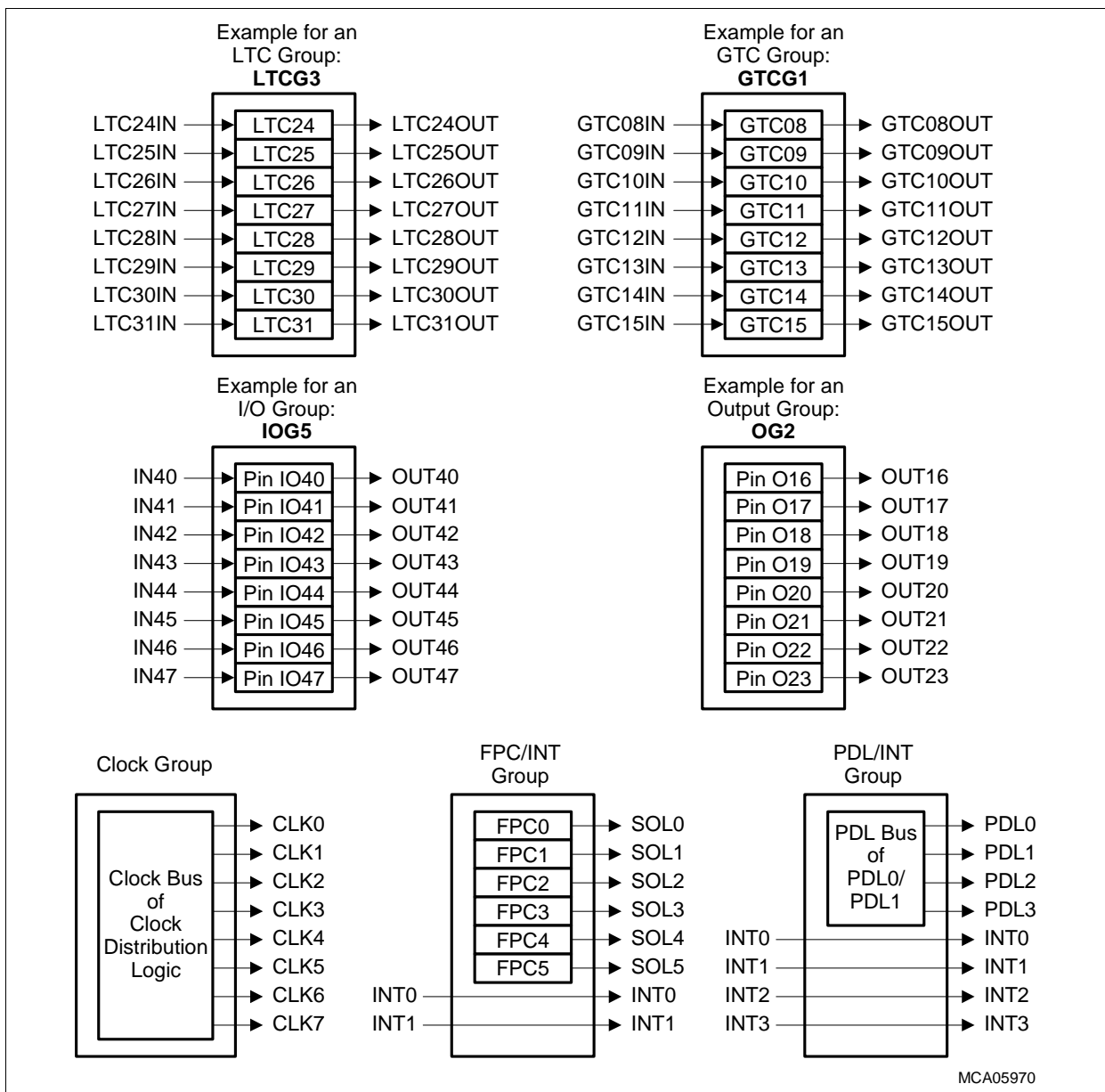


Figure 24-61 Groups Definitions for I/O Line Sharing Unit

General Purpose Timer Array (GPTA)

An **LTC group** combines eight LTC cells with its input and output lines. This results in eight LTC groups, LTCG0 to LTCG7.

A **GTC group** combines eight GTC cells with its input and output lines. This results in four GTC groups, GTCG0 to GTCG3.

An **I/O group** combines eight GPTA I/O lines connected to bi-directional device pins with its input and output lines. This results in seven I/O groups, IOG0 to IOG6, supporting 56 I/O lines.

An **Output group** combines eight GPTA output lines connected to device pins as an output. This results in seven output groups, OG0 to OG6, supporting 56 output lines.

The **Clock group** is a group that combines the eight clock bus output signals CLK[7:0] generated by the clock distribution unit.

The **FPC/INT group** is a group that combines the six level output signals SOL[5:0] of the FPCs with two external input lines INT[1:0] of the GPTA module.

The **PDL/INT group** is a group that combines the four PDL output lines of the PDL bus with four external input lines INT[3:0] of the GPTA module.

Table 24-8 Group to I/O Line/Cell Assignment

Group/Module	Cell/Line	Input	Output
LTC Groups			
LTCG0	LTC[07:00]	LTC[07:00]IN	LTC[07:00]OUT
LTCG1	LTC[15:08]	LTC[15:08]IN	LTC[15:08]OUT
LTCG2	LTC[23:16]	LTC[23:16]IN	LTC[23:16]OUT
LTCG3	LTC[31:24]	LTC[31:24]IN	LTC[31:24]OUT
LTCG4	LTC[39:32]	LTC[39:32]IN	LTC[39:32]OUT
LTCG5	LTC[47:40]	LTC[47:40]IN	LTC[47:40]OUT
LTCG6	LTC[55:48]	LTC[55:48]IN	LTC[55:48]OUT
LTCG7	LTC[63:56]	LTC[63:56]IN	LTC[63:56]OUT
GTC Groups			
GTCG0	GTC[07:00]	GTC[07:00]IN	GTC[07:00]OUT
GTCG1	GTC[15:08]	GTC[15:08]IN	GTC[15:08]OUT
GTCG2	GTC[23:16]	GTC[23:16]IN	GTC[23:16]OUT
GTCG3	GTC[31:24]	GTC[31:24]IN	GTC[31:24]OUT
I/O Groups			
IOG0	–	IN[07:00]	OUT[07:00]
IOG1	–	IN[15:08]	OUT[15:08]

General Purpose Timer Array (GPTA)

Table 24-8 Group to I/O Line/Cell Assignment (cont'd)

Group/Module	Cell/Line	Input	Output
IOG2	–	IN[23:16]	OUT[23:16]
IOG3	–	IN[31:24]	OUT[31:24]
IOG4	–	IN[39:32]	OUT[39:32]
IOG5	–	IN[47:40]	OUT[47:40]
IOG6	–	IN[55:48]	OUT[55:48]
Output Groups			
OG0	–	–	OUT[07:00]
OG1	–	–	OUT[15:08]
OG2	–	–	OUT[23:16]
OG3	–	–	OUT[31:24]
OG4	–	–	OUT[39:32]
OG5	–	–	OUT[47:40]
OG6	–	–	OUT[55:48]
Clock Group			
–	–	–	CLK[7:0]
FPC/INT Groups			
FPC[5:0]	–	–	SOT[5:0]
External Input [1:0]	–	–	INT[1:0]
PDL/INT Groups			
PDL[1:0] PDL Bus	–	–	PDL[3:0]
External Input [3:0]	–	–	INT[3:0]

General Purpose Timer Array (GPTA)

24.2.4.1 FPC Input Line Selection

As shown on [Page 24-10](#), each FPC cell can be connected to one out of four input lines SINK[3:0], to the GPTA module clock f_{GPTA} , or to the output of the preceding FPC. In total, 24 input lines out of the 56 input lines IN[55:00] from the I/O groups are connected (not programmable) with the FPCK inputs. The FPCK input line selection is controlled by the FPCCTRk.IPS bit fields. [Table 24-23](#) shows the FPC port pin connections.

Table 24-9 FPC Input Line Assignments

FPC Control Register	Bit Field IPS	Selected Input Pin
FPCCTR0	000 _B	IN0
	IN12	001 _B
	010 _B	IN24
	011 _B	IN36
FPCCTR1	000 _B	IN2
	001 _B	IN14
	010 _B	IN26
	011 _B	IN38
FPCCTR2	000 _B	IN4
	001 _B	IN16
	010 _B	IN28
	011 _B	IN40
FPCCTR3	000 _B	IN6
	001 _B	IN18
	010 _B	IN30
	011 _B	IN42
FPCCTR4	000 _B	IN8
	001 _B	IN20
	010 _B	IN32
	011 _B	IN44
FPCCTR5	000 _B	IN10
	001 _B	IN22
	010 _B	IN34
	011 _B	IN46

General Purpose Timer Array (GPTA)

24.2.4.2 GTC and LTC Output Multiplexer Selection

The output multiplexer shown in [Figure 24-60](#) and [Figure 24-62](#) below connects the 32 GTC output lines and the 64 LTC output lines with the I/O groups (7 × 8 = 56 output lines) and the output groups (7 × 8 = 56 output lines).

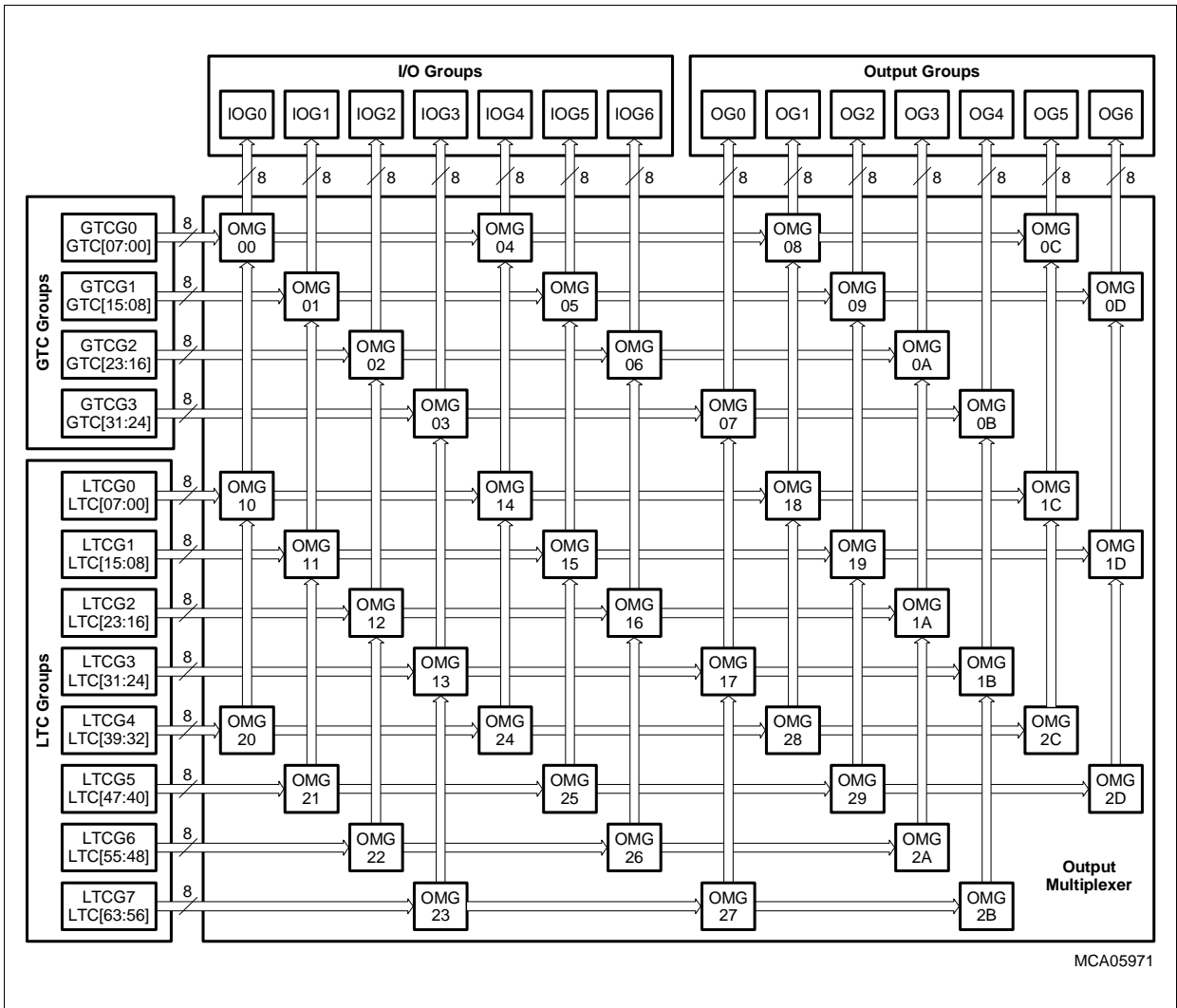


Figure 24-62 Output Multiplexer

The output multiplexer contains Output Multiplexer Groups (OMGs) that connect the Global Timer Cells or Local Timer Cells with the input lines of the I/O groups and output groups. GTCs and LTCs are grouped into four GTC groups (GTTCG[3:0]) and eight LTC groups (LTTCG[7:0]) with 8 cells each. In the same way, I/O groups and output groups are grouped into 14 groups (seven I/O groups and seven output groups) with 8 lines each.

General Purpose Timer Array (GPTA)

Figure 24-63 shows the logical structure of an OMG.

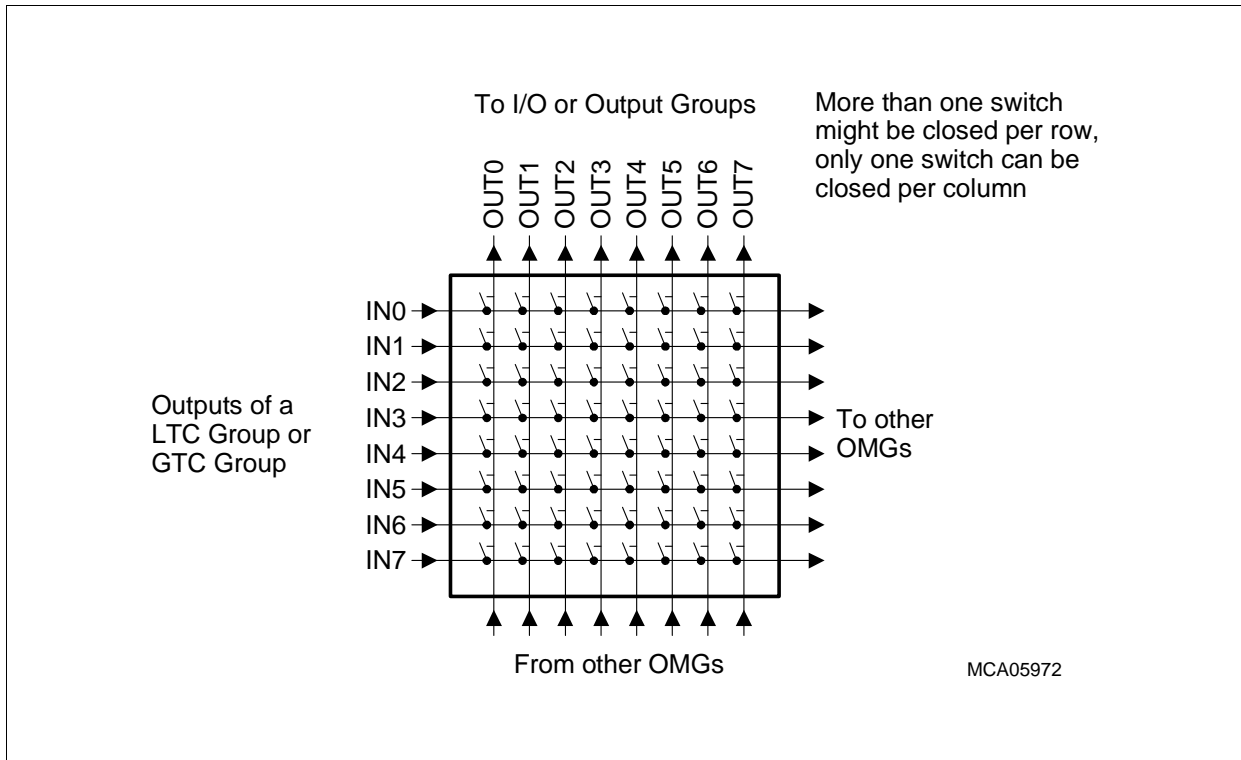


Figure 24-63 Output Multiplexer Group (OMG) Structure

Rules for connections to Output Multiplexer Group OMG:

- Within a GTC or LTC group, the output of the cell with the lowest index number is connected to OMG input line IN0. The remaining cells of a cell group are connected to OMG input lines IN1 to IN7 with ascending cell index numbers.
Example: for OMG13 (see Figure 24-62), the cells LTC24 up to LTC31 are wired to the OMG13 input lines IN0 to line IN7.
- OMG output line OUT0 is always connected to the input of an I/O or output group with the lowest index. The remaining output lines OUT1 to OUT7 are connected to the I/O or Output lines with ascending index.
Example: for OMG13 (see Figure 24-62), the outputs OUT0 to OUT7 are wired (via OMG03) to input lines 0 to 7 of I/O group 3 (IOG3).
- One input of an I/O or output group can be connected to the output of only one timer cell. This is guaranteed by the OMG control register layout. Otherwise, short circuits and unpredictable behavior would occur. On the other hand, it is permissible for the output of a GTC or LTC to be connected to more than one input of an I/O or output group.

General Purpose Timer Array (GPTA)

The output multiplexer group configuration is based on the following principles:

- Each OMG is referenced with two index variables: n and g (OMG_ng)
- Index n is a group number. Global timer cell groups GTCG[3:0] have the group number 0, Local Timer Cell Groups LTCG[3:0] have the group number 1, and Local Timer Cell Groups LTCG[7:4] have the group number 2.
- Index g indicates the number of an I/O or output group g (g = 0-13_D) to which the outputs of the output multiplexer group OMG_ng are connected. I/O groups IOG0 to IOG6 are assigned to index variable g = 0 to 7 and output groups OG0 to OG6 are assigned to index variable g = 7 to 13.

The output multiplexer logic as seen for programming is shown in **Figure 24-64**. With this logic, always three GTC or LTC group signals are combined to one output line that leads to the input of an I/O or output group. For example, when looking at **Figure 24-62**, each of the eight output multiplexer output lines to I/O group IOG5 is connected via three OMG_n5 (n = 0, 1, 2) with the eight outputs of one GTC group (GTCG1) and two LTC groups (LTCG1 and LTCG5).

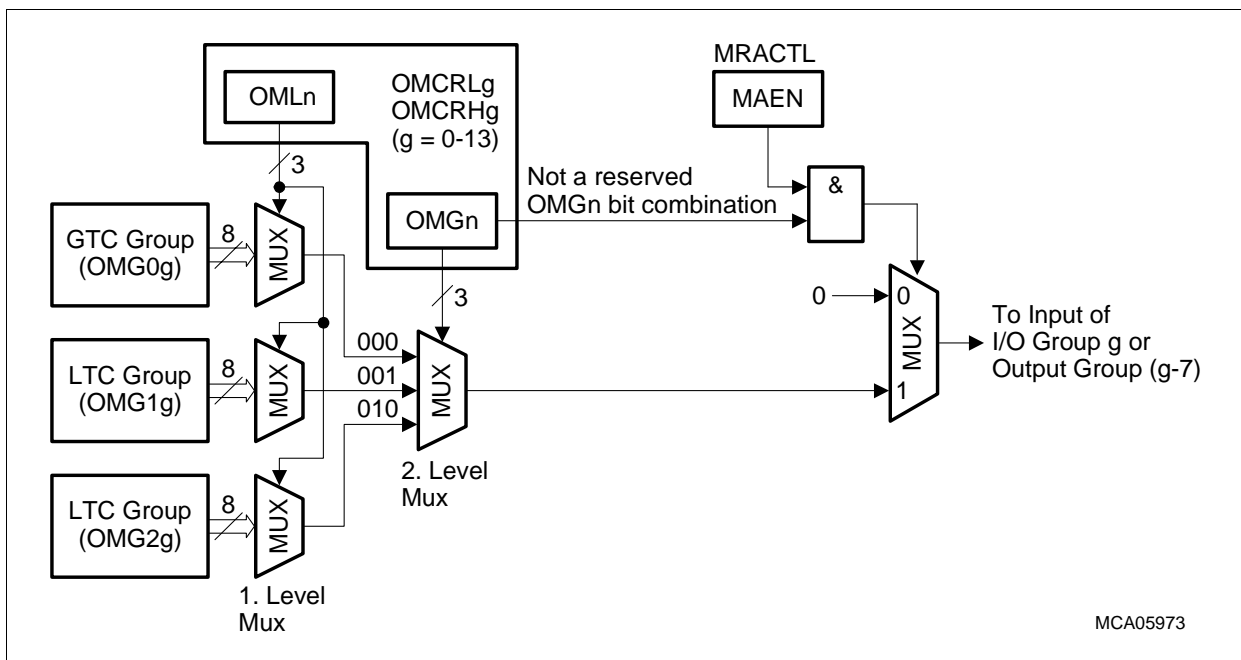


Figure 24-64 Output Multiplexer Group (Programmer's View)

The 1. level multiplexer is built up by three 8:1 multiplexers that are controlled in parallel by bit field OML_n. Bit field OMG_n controls the 2. level multiplexer and connects one of the 1. level multiplexer outputs to output n. The output of the 2. level multiplexer is connected only to the input of an I/O group or output group if bit MRACTL.MAEN is set (multiplexer array enabled) and no reserved bit combination of OMG_n is selected. If one of these conditions is not true, the corresponding OMG output will be held at a low level.

General Purpose Timer Array (GPTA)

Two Output GPTA, OMCRL and OMCRH (see also [Page 24-196](#)), are assigned to each of the I/O or output groups. Therefore, a total of 28 registers control the connections within the output multiplexer of the GPTA module.

The OMCRL registers control the OMG output lines 0 to 3. The OMCRH registers control the OMG output lines 4 to 7. [Table 24-10](#) lists all Output Multiplexer Control Registers with its control functions. Please note that the Output Multiplexer Control Registers are not directly accessible but must be written or read using a FIFO array structure as described on [Page 24-110](#).

Table 24-10 Output Multiplexer Control Register Assignments

I/O Group or Output Group		Controlled by Multiplexer Control Register	Selectable Groups via OMGng
IOG0 IOG0	IN[03:00]/OUT[03:00]	OMCRL0	GTCG0, LTCG0, LTCG4
	IN[07:04]/OUT[07:04]	OMCRH0	
IOG1 IOG0	IN[11:08]/OUT[11:08]	OMCRL1	GTCG1, LTCG1, LTCG5
	IN[15:12]/OUT[15:12]	OMCRH1	
IOG2 IOG0	IN[19:16]/OUT[19:16]	OMCRL2	GTCG2, LTCG2, LTCG6
	IN[23:20]/OUT[23:20]	OMCRH2	
IOG3 IOG0	IN[27:24]/OUT[27:24]	OMCRL3	GTCG3, LTCG3, LTCG7
	IN[31:28]/OUT[31:28]	OMCRH3	
IOG4 IOG0	IN[35:32]/OUT[35:32]	OMCRL4	GTCG0, LTCG0, LTCG4
	IN[39:36]/OUT[39:36]	OMCRH4	
IOG5 IOG0	IN[43:40]/OUT[43:40]	OMCRL5	GTCG1, LTCG1, LTCG5
	IN[47:44]/OUT[47:44]	OMCRH5	
IOG0 IOG0	IN[51:48]/OUT[51:48]	OMCRL6	GTCG2, LTCG2, LTCG6
	IN[55:52]/OUT[55:52]	OMCRH6	
OG0 IOG0	OUT[59:56]	OMCRL7	GTCG3, LTCG3, LTCG7
	OUT[63:60]	OMCRH7	
OG1 IOG0	OUT[67:64]	OMCRL8	GTCG0, LTCG0, LTCG4
	OUT[71:68]	OMCRH8	
OG2 IOG0	OUT[75:72]	OMCRL9	GTCG1, LTCG1, LTCG5
	OUT[79:76]	OMCRH9	
OG3	OUT[83:80]	OMCRL10	GTCG2, LTCG2, LTCG6
	OUT[87:84]	OMCRH10	

General Purpose Timer Array (GPTA)

Table 24-10 Output Multiplexer Control Register Assignments (cont'd)

I/O Group or Output Group		Controlled by Multiplexer Control Register	Selectable Groups via OMGng
OG4	OUT[91:88]	OMCRL11	GTCG3, LTCG3, LTCG7
	OUT[95:92]	OMCRH11	
OG5	OUT[99:96]	OMCRL12	GTCG0, LTCG0, LTCG4
	OUT[103:100]	OMCRH12	
OG6	OUT[107:104]	OMCRL13	GTCG1, LTCG1, LTCG5
	OUT[111:108]	OMCRH13	

General Purpose Timer Array (GPTA)

24.2.4.3 GTC Input Multiplexer Selection

The GTC input multiplexer as shown in **Figure 24-60** and **Figure 24-65** connects the 56 (= 7 × 8) input lines of the I/O groups, the 64 LTC output lines of the eight LTC groups, the six FPC output lines, and two internal input lines INT[1:0] with the 32 (= 4 × 8) LTC input lines, organized into eight LTC groups.

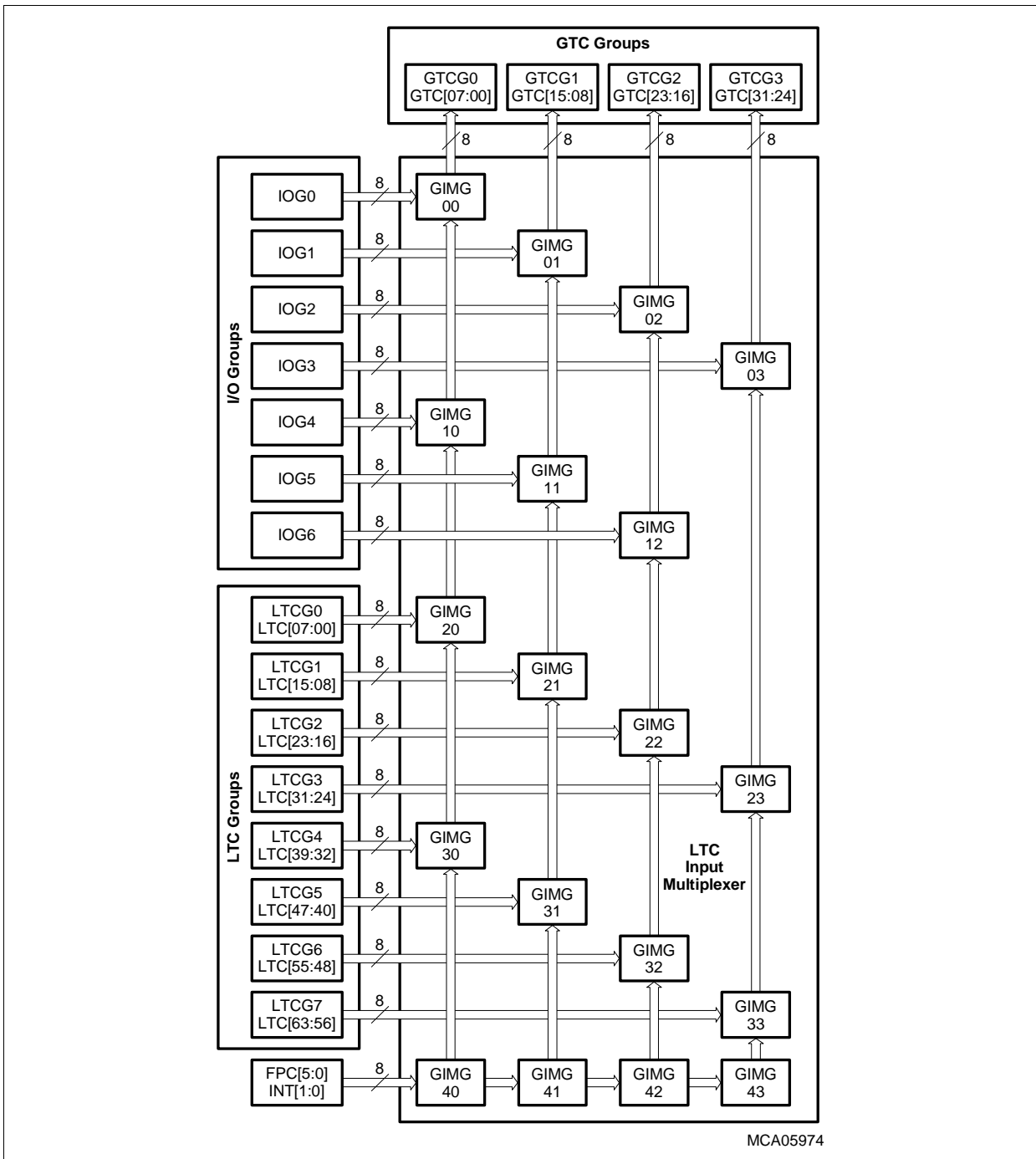


Figure 24-65 GTC Input Multiplexer

General Purpose Timer Array (GPTA)

The GTC input multiplexer contains GTC input Multiplexer Groups (GIMGs) that connect the I/O groups or Local Timer Cells with the input lines of the GTC input lines, organized into four GTC groups with 8 cells each. GTC input Multiplexer Group are grouped into seven IOGs (IOG[6:0]) with eight lines each and eight LTC groups (LTCG[7:0]) with 8 cells each. One special FPC/INT group with eight outputs is established that combines the six FPC outputs and two internal input lines INT[1:0] as a group of GIMGs inputs.

Figure 24-66 shows the logical structure of a GIMG.

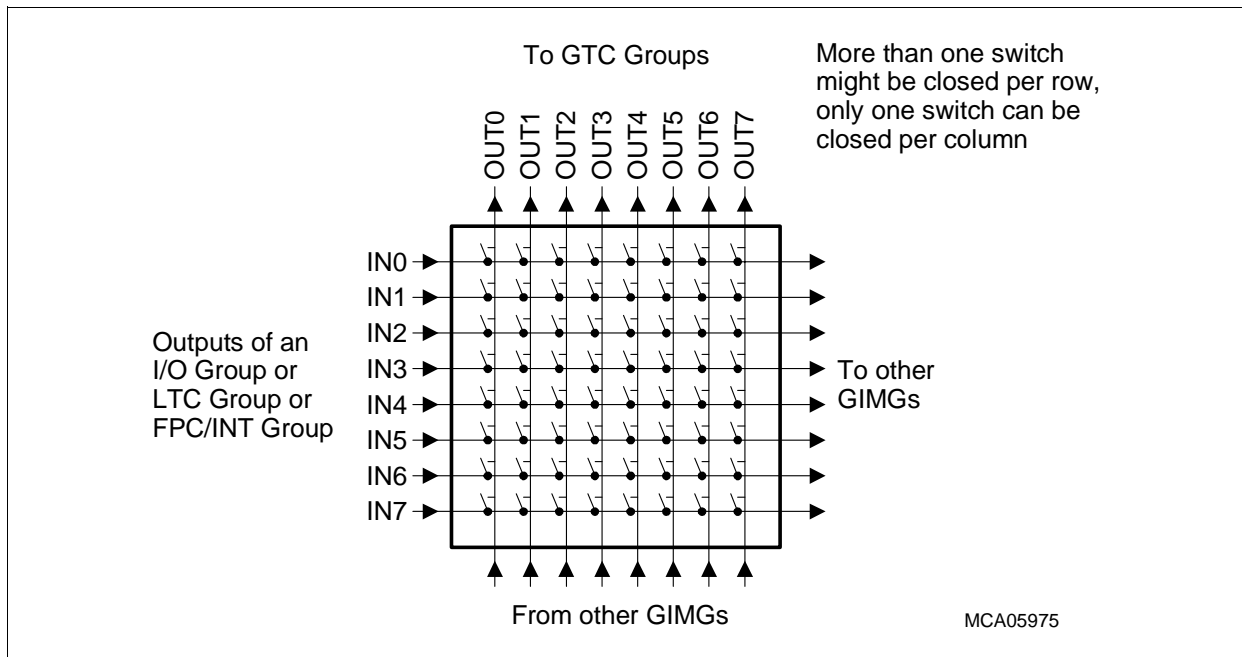


Figure 24-66 GTC Input Multiplexer Group (GIMG) Structure

Rules for connections to GTC Input Multiplexer Group GIMG:

- Within a I/O group or LTC group, the line or the output of the cell with the lowest index number is connected to GIMG input line IN0. The remaining lines, cells or lines of a group are connected to GIMG input lines IN1 to IN7 with ascending index numbers. At the FPC/INT group, FPC[5:0] is connected to IN[5:0] and INT[1:0] is connected to IN[7:6].
Example: for GIMG23 (see [Figure 24-65](#)), the cells LTC24 up to LTC31 are wired to the GIMG23 input lines IN0 to line IN7.
- Multiplexer output OUT0 is always connected to the input of a GTC group with the lowest index. The remaining output lines OUT1 to OUT7 are connected to the GTC inputs with ascending index.
Example: for GIMG23 (see [Figure 24-65](#)), the outputs OUT0 to OUT7 are wired to the inputs of GTC16 to GTC23.
- A GTC input can be connected either to an I/O group output, or to an LTC output, or to an FPC/INT output. This is guaranteed by the GIMG control register layout. Otherwise, short circuits and unpredictable behavior would occur. In contrast, it is

General Purpose Timer Array (GPTA)

permissible for an I/O group output, or an LTC output, or an FPC/INT output to be connected to more than one GTC input.

The GTC input multiplexer group configuration is based on the following principles:

- Each GIMG is referenced with two index variables: n and g (GIMG ng)
- Index n is a group number. I/O groups IOG[3:0] have group number 0, I/O groups IOG[6:4] have group number 1, Local Timer Cell Groups LTCG[3:0] have group number 2, Local Timer Cell Groups LTCG[7:4] have group number 3, and the FPC/INT group has group number 4.
- Index g indicates the number of the GTC group g ($g = 0-3$) to which the outputs of the input multiplexer group GIMG ng are connected.

The GTC input multiplexer logic as seen for programming is shown in [Figure 24-67](#). With this logic, five group signals (from an I/O group, LTC group, or FPC/INT group) are always combined to one output line that leads to the input of a GTC of GTC group g . For example, when looking at [Figure 24-66](#), each of the eight GTC input multiplexer output lines to GTC group GTCG2 is connected via five OMG n 2 ($n = 0-4$) with the eight outputs of two I/O group (IOG2 and IOG6), two LTC groups (LTCG2 and LTCG6), and the FPC/INT group.

General Purpose Timer Array (GPTA)

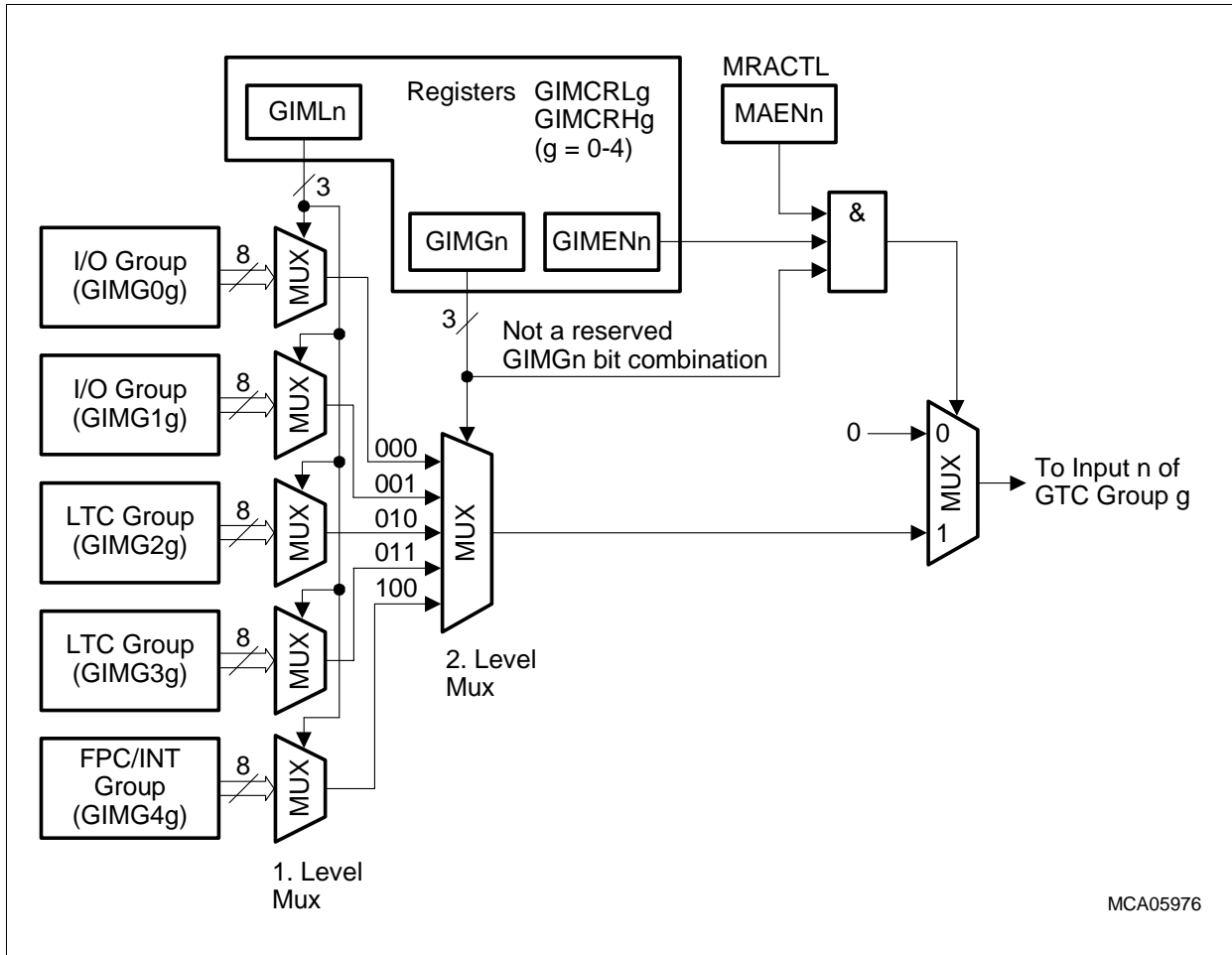


Figure 24-67 GTC Input Multiplexer Group (Programmer's View)

The 1. level multiplexer is built up by five 8:1 multiplexers that are controlled in parallel by bit field GIMLn. Bit field GIMGn controls the 2. level multiplexer and connects one of the 1. level multiplexer outputs to one of the GIMGng outputs. The output of the 2. level multiplexer is only connected to the input of an GTC if bit GIMENn (enable multiplexer connection) is set, and bit MRACTL.AEN is set (multiplexer array enabled), and no reserved bit combination of GIMGn is selected. If one of these conditions is not true, the corresponding GIMG output will be held at a low level.

If one of these bit is not set, the corresponding GTC input will be held at a low level.

Two GTC Input Multiplexer Control Registers, GIMCRL and GIMCRH (see also [Page 24-198](#)), are assigned to each of the GTC groups. Therefore, a total of eight registers control the connections within the GTC input multiplexer of the GPTA module.

The GIMCRL registers control the GIMG output lines 0 to 3 and the GIMCRH registers control the GIMG output lines 4 to 7. [Table 24-11](#) lists all of the GTC Input Multiplexer Control Registers with its control functions. Please note that all GTC Input Multiplexer

General Purpose Timer Array (GPTA)

Control Registers are not directly accessible but must be written or read using a FIFO array structure as described on [Page 24-110](#).

Table 24-11 GTC Input Multiplexer Control Register Assignments

GTC Group and GTCs		Controlled by Multiplexer Control Register	Selectable Groups via GIMGng
GTCC0	GTC[03:00]	GIMCRL0	IOG0, IOG4, LTCC0, LTCC4, FPC/INT
	GTC[07:04]	GIMCRH0	
GTCC1	GTC[11:08]	GIMCRL1	IOG1, IOG5, LTCC1, LTCC5, FPC/INT
	GTC[15:12]	GIMCRH1	
GTCC2	GTC[19:16]	GIMCRL2	IOG2, IOG6, LTCC2, LTCC6, FPC/INT
	GTC[23:20]	GIMCRH2	
GTCC3	GTC[27:24]	GIMCRL3	IOG3, LTCC3, LTCC7, FPC/INT
	GTC[31:28]	GIMCRH3	

General Purpose Timer Array (GPTA)

24.2.4.4 LTC Input Multiplexer Selection

The LTC input multiplexer as shown in **Figure 24-60** and **Figure 24-68** connects the 56 (= 7 × 8) input lines of the I/O groups, the 32 (= 4 × 8) GTC output lines of the GTC groups, the eight clock bus lines, or the four PDL output lines with four internal input lines INT[3:0] with the 64 (= 8 × 8) LTC input lines, organized into eight LTC groups.

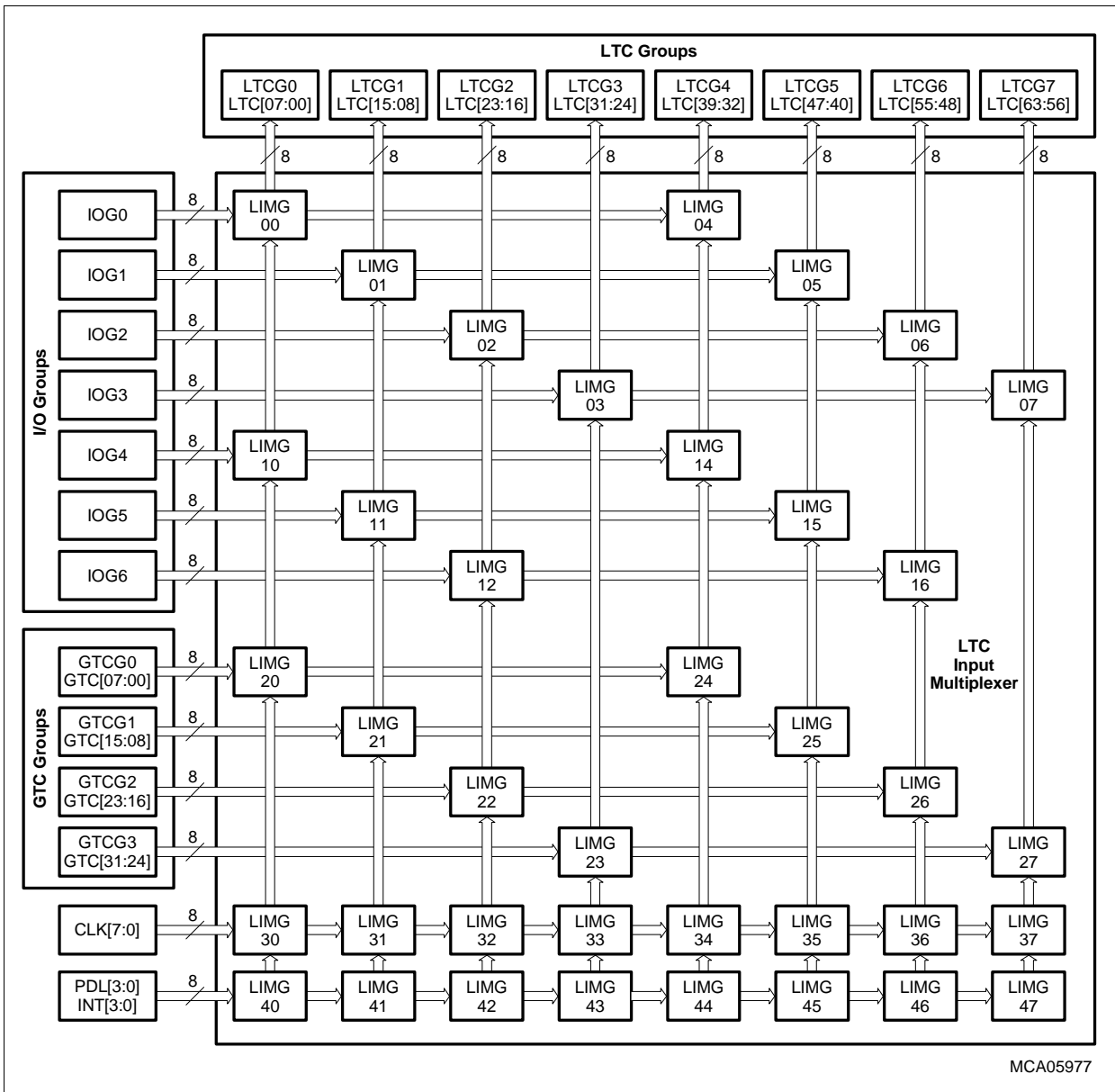


Figure 24-68 LTC Input Multiplexer

General Purpose Timer Array (GPTA)

The LTC input multiplexer contains LTC input Multiplexer Groups (LIMGs) that connect the I/O groups or Global Timer Cells with the input lines of the LTCs, organized into eight LTC groups with 8 cells each. IOGs and GTCs are grouped into seven IOGs (IOG[6:0]) with eight lines each and four GTC groups (GTCCG[3:0]) with 8 cells each. Two special groups are available: a clock group with eight lines representing the clock bus lines CLK[7:0] of the clock distribution unit and a PDL/INT group with eight outputs that combines the four PDL outputs and four internal input lines INT[3:0] as a group of LIMGs inputs.

Figure 24-69 shows the logical structure of a LIMG.

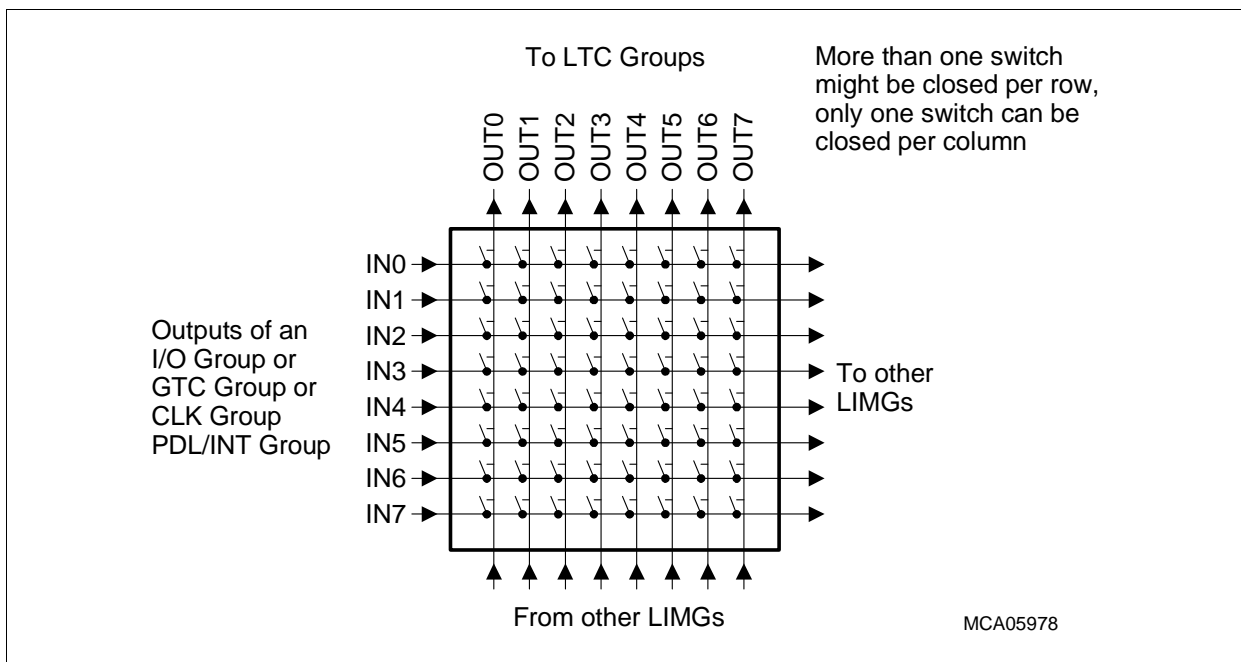


Figure 24-69 LTC Input Multiplexer Group (LIMG) Structure

Rules for connections to LTC Input Multiplexer Group LIMG:

- Within an I/O group or GTC group, the line or the output of the cell with the lowest index number is connected to LIMG input line IN0. The remaining lines, cells or lines of a group are connected to LIMG input lines IN1 to IN7 with ascending index numbers. At the clock group, CLK0 is connected to IN0 and the remaining clock lines are connected to LIMG input lines IN1 to IN7 with ascending index numbers. At the PDL/INT group, PDL[3:0] (see Page 24-20) is connected to IN[3:0] and INT[3:0] is connected to IN[7:4].
Example: for LIMG23 (see Figure 24-68), the cells GTC24 up to GTC31 are wired to the LIMG23 input lines IN0 to line IN7.
- Multiplexer output OUT0 is always connected to the input of an LTC group with the lowest index. The remaining output lines OUT1 to OUT7 are connected to the LTC inputs with ascending index.

General Purpose Timer Array (GPTA)

Example: for LIMG23 (see [Figure 24-68](#)), the outputs OUT0 to OUT7 are wired to the inputs of LTC24 to GTC31.

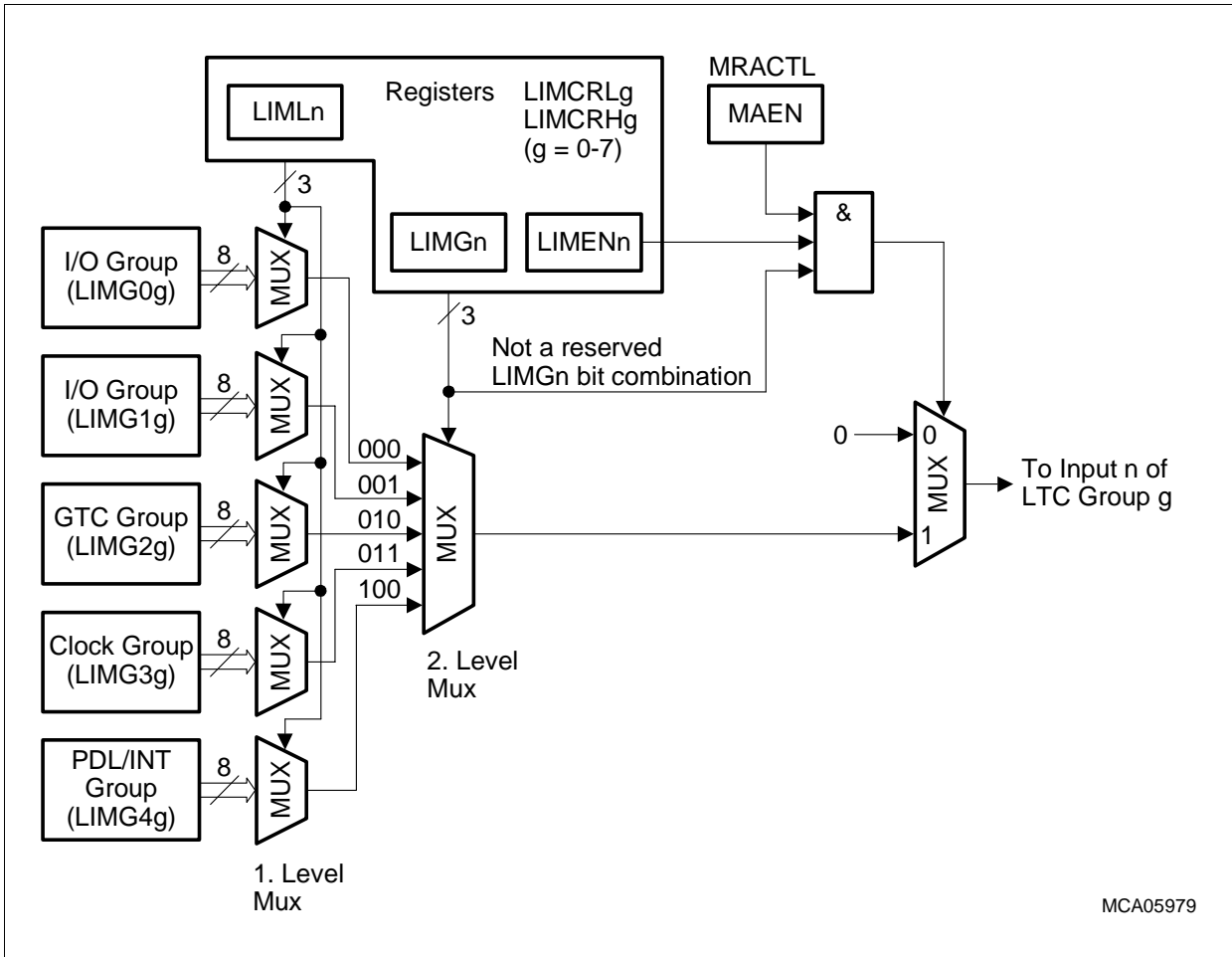
- An LTC input can be connected either to an I/O group output, or to an GTC output, or to a clock bus output, or to an PDL/INT output. This is guaranteed by the LIMG control register layout. Otherwise, short circuits and unpredictable behavior would occur. In contrast, it is permitted that an I/O group output, or an GTC output, or an PDL/INT output is connected to more than one LTC input.

The LTC input multiplexer group configuration is based on the following principles:

- Each LIMG is referenced with two index variables: n and g (LIMG n g)
- Index n is a group number. I/O groups IOG[3:0] have group number 0, I/O groups IOG[6:4] have group number 1, Global Timer Cell Groups GTCG[3:0] have group number 2, clock bus lines CLK[7:0] have group number 3, and the PDL/INT group has group number 4.
- Index g indicates the number of the LTC group g ($g = 0-7$) to which the outputs of the input multiplexer group LIMG n g are connected.

The LTC input multiplexer logic as seen for programming is shown in [Figure 24-70](#). With this logic, five group signals (from an I/O group, GTC group, clock group, or PDL/INT group) are always combined to one output line that leads to the input of an LTC of LTC group g . For example, when looking at [Figure 24-68](#), each of the eight LTC input multiplexer output lines to LTC group LTCG2 is connected via five LIMG n 2 ($n = 0-4$) with the eight outputs of two I/O group (IOG2 and IOG6), one GTC group (GTCG2), the clock group, and the PDL/INT group.

General Purpose Timer Array (GPTA)



MCA05979

Figure 24-70 LTC Input Multiplexer Group (Programmer's View)

The 1. level multiplexer is built up by five 8:1 multiplexers that are controlled in parallel by bit field LIML_n. Bit field LIMG_n controls the 2. level multiplexer and connects one of the 1. level multiplexer outputs to one of the LIMG_n outputs. The output of the 2. level multiplexer is connected only to the input of an LTC if bit LIMEN_n is set (enable multiplexer connection), and bit MRCTL.AEN is set (multiplexer array enabled), and no reserved bit combination of LIMG_n is selected. If one of these conditions is not true, the corresponding LTC input will be held at a low level.

Two LTC Input Multiplexer Control Registers, LIMCRL and LIMCRH (see also [Page 24-202](#)), are assigned to each of the LTC groups. Therefore, a total of sixteen registers control the connections within the LTC input multiplexer of the GPTA module.

The LIMCRL registers control the LIMG output lines 0 to 3 and the LIMCRH registers control the LIMG output lines 4 to 7. [Table 24-12](#) lists all LTC Input Multiplexer Control Registers with its control functions. Please note that all LTC Input Multiplexer Control Registers are not directly accessible but must be written or read using a FIFO array structure as described on [Page 24-110](#).

General Purpose Timer Array (GPTA)

Table 24-12 LTC Input Multiplexer Control Register Assignments

LTC Group and LTCs		Controlled by Register	Selectable Groups via LIMGng
LTCG0	LTC[03:00]	LIMCRL0	IOG0, IOG4, GTCG0, CLOCK, PDL/INT
	LTC[07:04]	LIMCRH0	
LTCG1	LTC[11:08]	LIMCRL1	IOG1, IOG5, GTCG1, CLOCK, PDL/INT
	LTC[15:12]	LIMCRH1	
LTCG2	LTC[19:16]	LIMCRL2	IOG2, IOG6, GTCG2, CLOCK, PDL/INT
	LTC[23:20]	LIMCRH2	
LTCG3	LTC[27:24]	LIMCRL3	IOG3, GTCG3, CLOCK, PDL/INT
	LTC[31:28]	LIMCRH3	
LTCG4	LTC[35:32]	LIMCRL4	IOG0, IOG4, GTCG0, CLOCK, PDL/INT
	LTC[39:36]	LIMCRH4	
LTCG5	LTC[43:40]	LIMCRL5	IOG1, IOG5, GTCG1, CLOCK, PDL/INT
	LTC[47:44]	LIMCRH5	
LTCG6	LTC[51:48]	LIMCRL6	IOG2, IOG6, GTCG2, CLOCK, PDL/INT
	LTC[55:52]	LIMCRH6	
LTCG7	LTC[59:56]	LIMCRL7	IOG3, GTCG3, CLOCK, PDL/INT
	LTC[63:60]	LIMCRH7	

General Purpose Timer Array (GPTA)

24.2.4.5 Multiplexer Register Array Programming

A total of 52 control registers are required to program the configuration of the output multiplexer and the two input multiplexers of the Input/Output Line Sharing Unit. These IOLS control registers are combined into a Multiplexer Register Array FIFO that can only be read or written sequentially. Therefore, the control registers values cannot be accessed directly but must be accessed in a specific sequential order.

Three registers are available for controlling the Multiplexer Register Array:

- Multiplexer Register Array Control Register MRACTL
- Multiplexer Register Array Data In Register MRADIN
- Multiplexer Register Array Data Out Register MRADOUT

Figure 24-71 shows the structure of the multiplexer array FIFO with the arrangement of the multiplexer control registers.

For programming of the multiplexer array FIFO, the following steps must be executed:

1. Disable interconnections of the multiplexer array by writing `MRACTL.MAEN = 0` (default after reset). The multiplexer array is disabled, all cell input lines are driven with 0, and device pins assigned to GPTA I/O lines or output lines are disconnected.
2. Reset the write cycle counter to 0 by writing `MRACTL.WCRES = 1`.
3. Write sequentially the multiplexer control register contents one after the other (52 values) into MRADIN, starting with the register values for `OMCRH13`, `OMCRL13`, ... up to `GIMCRH0`, `GIMCRL0` (see **Figure 24-71**). After the first MRADIN write operation, the content for `OMCRH13` is at FIFO position 1. With each following MRADIN write operation, it becomes shifted one FIFO position upwards. After the 52. MRADIN write operation, the `OMCRH13` value is at its final position. The contents of FIFO position 52 can be read via register MRADOUT. With each MRADIN write operation the write cycle counter `MRACTL.FIFOFILLCNT` is incremented by 1. After all FIFO entries have been written, the FIFO is locked, bit `MRACTL.FIFOFULL` is set, and further MRADIN write operations are discarded until bit `MRACTL.WCRES` is written again with a 0.
4. Enable the multiplexer array by writing `MRACTL.MAEN = 1`. This establishes and enables all programmed interconnections.

To check the FIFO contents, the FIFO can be written a second time. At this check MRADIN is written before MRADOUT is read. This will return the FIFO contents of the first write sequence in the order of `OMCRH13`, `OMCRL13`, ..., `GIMCRH0`, `GIMCRL0`.

Before disabling the multiplexer array FIFO, GPTA output pins that are already enabled as GPTA output should be switched to GPIO function to avoid output spikes. After enabling the multiplexer array FIFO again, the GPTA output can be switched again back to GPTA output function.

Shifting the write data through the FIFO requires a few clock cycles. When new data becomes written before the FIFO is ready to accept them, wait states will be inserted into the write access.

General Purpose Timer Array (GPTA)

If the OMCRLg register bit field OMCn of the multiplexer array is programmed with an invalid (reserved) value, the related outputs will be forced to 0. When the array is disabled (MRACTL.MAEN = 0), all cell inputs and outputs are disconnected from the GPIO lines and are driven with 0.

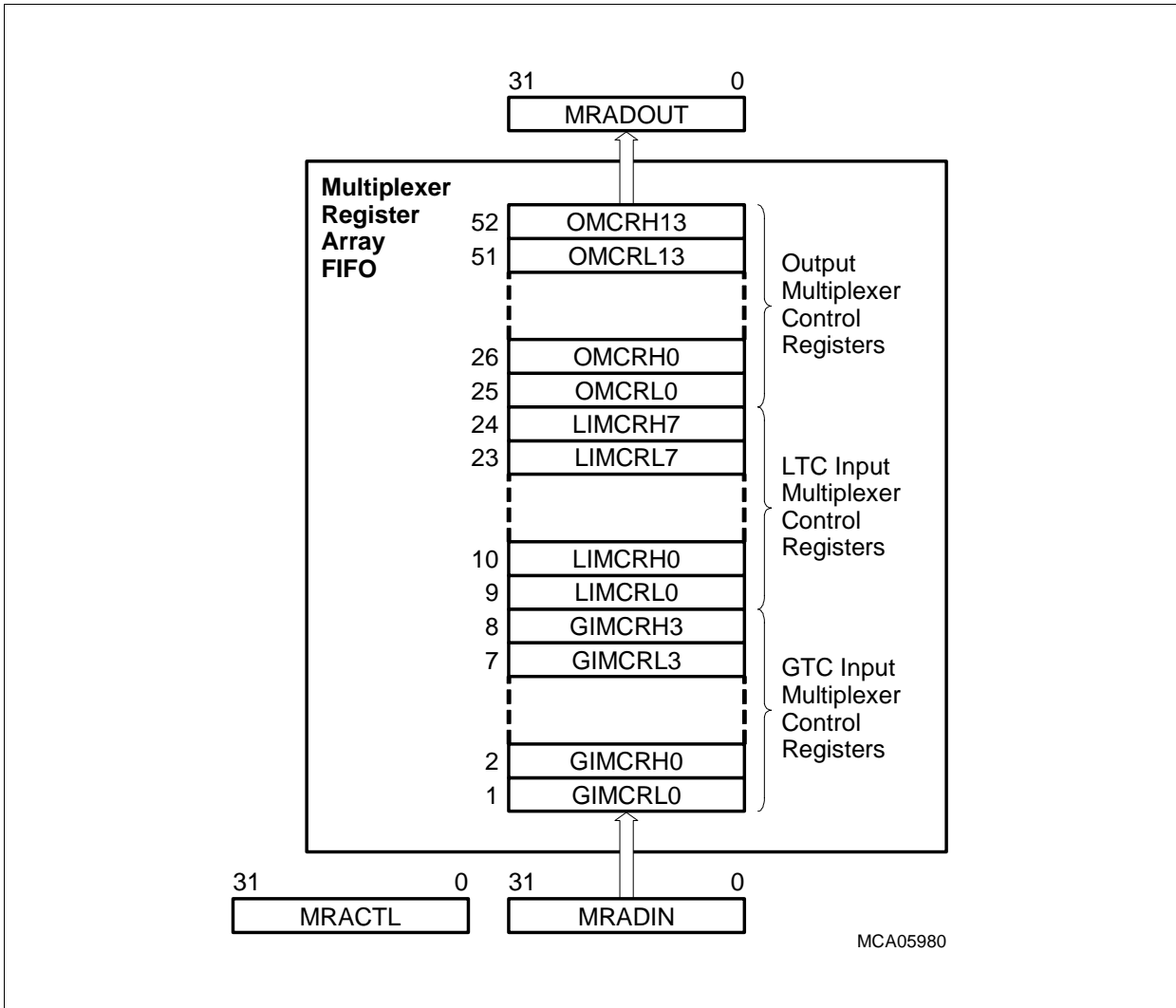


Figure 24-71 GPTA Multiplexer Array Control Register FIFO Structure

General Purpose Timer Array (GPTA)

24.2.5 Interrupt Sharing Unit (IS)

The GPTA provides 111 service request sources. These service request sources are generated by different cell types, as shown in [Table 24-13](#).

Table 24-13 GPTA Number of Service Request Sources

Cell Type	Number of Cells	Number of Service Request Sources/Cell	Total Number of Request Sources
DCM	4	3	12
PLL	1	1	1
GT	2	1	2
GTC	32	1	32
LTC	64	1	64
Sum:			111

To reduce hardware and software overhead, at maximum five request sources are combined together in service request groups. A service request group has up to five service request inputs and one service request output SR_y which is typically connected outside the GPTA kernel with a standard interrupt node y and controlled by its SRC_y register.

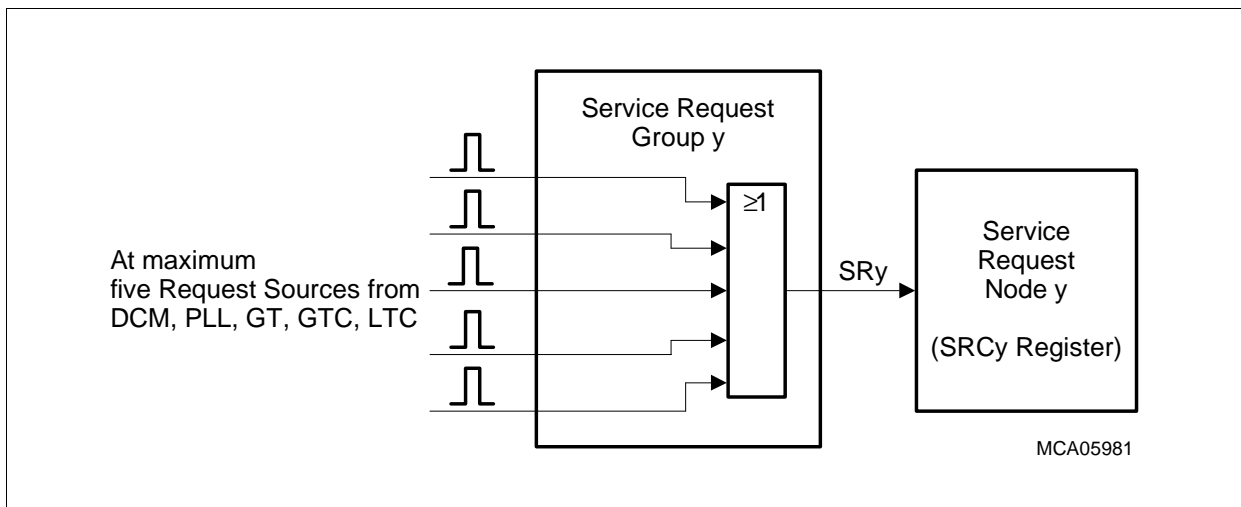


Figure 24-72 Service Request Groups

The bits in the Service Request State Registers ($SRSS_x$ and $SRSC_x$) are service request status flags that are set by hardware (type “h”) when the related event occurs. Each GPTA service request source has its own service request flag. This flag is normally set by hardware but can be set and cleared by software. Each service request status flag can be read twice, at the same bit location in the $SRSC_x$ register and in the $SRSS_x$

General Purpose Timer Array (GPTA)

register, and cleared or set by software when writing to the corresponding request bit in SRSCx or SRSSx. When writing to SRSCx or SRSSx, several request flags can be cleared at once by one write operation. Request flags of bit positions that are written with 0 are not changed. This feature allows fast, simple clearing or setting of request flags without affecting other bits in the same service request state register.

Note that service request flag is always set independently of whether it is enabled or disabled by the related module; but the service request line to the corresponding service request group becomes only active if the corresponding service request is enabled by the related module. Finally, each service request group y must be enabled by the enable flag that is located in SRC register y.

Table 24-14 lists all of the service requests groups with its request sources. Note that service requests of GTCs with an odd index number k can be individually redirected via register SRNR to a service request group that is assigned mainly to four LTCs.

Table 24-14 GPTA Service Request Groups

Service Request Group Number y	Request Source 1	Request Source 2	Request Source 3	Request Source 4	Request Source 5
00	DCM0 rising	DCM0 falling	DCM0 comp.	–	–
01	DCM1 rising	DCM1 falling	DCM1 comp.	–	–
02	DCM2 rising	DCM2 falling	DCM2 comp.	–	–
03	DCM3 rising	DCM3 falling	DCM3 comp.	–	–
04	PLL	–	–	–	–
05	GT0	GT1	–	–	–
06	GTC00	GTC01 ¹⁾	–	–	–
07	GTC02	GTC03 ¹⁾	–	–	–
08	GTC04	GTC05 ¹⁾	–	–	–
09	GTC06	GTC07 ¹⁾	–	–	–
10	GTC08	GTC09 ¹⁾	–	–	–
11	GTC10	GTC11 ¹⁾	–	–	–
12	GTC12	GTC13 ¹⁾	–	–	–
13	GTC14	GTC15 ¹⁾	–	–	–
14	GTC16	GTC17 ¹⁾	–	–	–
15	GTC18	GTC19 ¹⁾	–	–	–
16	GTC20	GTC21 ¹⁾	–	–	–

General Purpose Timer Array (GPTA)

Table 24-14 GPTA Service Request Groups (cont'd)

Service Request Group Number y	Request Source 1	Request Source 2	Request Source 3	Request Source 4	Request Source 5
17	GTC22	GTC23 ¹⁾	–	–	–
18	GTC24	GTC25 ¹⁾	–	–	–
19	GTC26	GTC27 ¹⁾	–	–	–
20	GTC28	GTC29 ¹⁾	–	–	–
21	GTC30	GTC31 ¹⁾	–	–	–
22	LTC00	LTC01	LTC02	LTC03	GTC01 ²⁾
23	LTC04	LTC05	LTC06	LTC07	GTC03 ²⁾
24	LTC08	LTC09	LTC10	LTC11	GTC05 ²⁾
25	LTC12	LTC13	LTC14	LTC15	GTC07 ²⁾
26	LTC16	LTC17	LTC18	LTC19	GTC09 ²⁾
27	LTC20	LTC21	LTC22	LTC23	GTC11 ²⁾
28	LTC24	LTC25	LTC26	LTC27	GTC13 ²⁾
29	LTC28	LTC29	LTC30	LTC31	GTC15 ²⁾
30	LTC32	LTC33	LTC34	LTC35	GTC17 ²⁾
31	LTC36	LTC37	LTC38	LTC39	GTC19 ²⁾
32	LTC40	LTC41	LTC42	LTC43	GTC21 ²⁾
33	LTC44	LTC45	LTC46	LTC47	GTC23 ²⁾
34	LTC48	LTC49	LTC50	LTC51	GTC25 ²⁾
35	LTC52	LTC53	LTC54	LTC55	GTC27 ²⁾
36	LTC56	LTC57	LTC58	LTC59	GTC29 ²⁾
37	LTC60	LTC61	LTC62	LTC63	GTC31 ²⁾

1) Redirection bit SRNR.GTCKR = 0 (k = 01, 03, 05, ... 27, 29, 31).

2) Redirection bit SRNR.GTCKR = 1 (k = 01, 03, 05, ... 27, 29, 31).

24.2.6 Pseudo Code Description of GPTA Kernel Functionality

This section describes the functional algorithms of the GPTA units in a pseudo code language.

24.2.6.1 FPC Algorithm

FPCK_Control_Logic() “to be performed every GPTA clock”

```
switch (FPCK.Mode)
  case PRESCALER_RISING:
    if (FPCK.Rising_Edge) then
      Prescaler()
    endif
    break
  case PRESCALER_FALLING:
    if (FPCK.Falling_Edge) then
      Prescaler()
    endif
    break
  case DELAYED_FILTER_BOTH:
    Delayed_Filter()
    break
  case IMMEDIATE_FILTER_BOTH:
  case IMMEDIATE_FILTER_RISING:
  case IMMEDIATE_FILTER_FALLING:
    Immediate_Filter()
    break
  case MIXED_FILTER_RISING_DELAYED:
    if (FPCK.Signal_Filtered == 0) then
      Delayed_Filter()
    else
      Immediate_Filter()
    endif
    break
  case MIXED_FILTER_RISING_IMMEDIATE:
    if (FPCK.Signal_Filtered == 0) then
      Immediate_Filter()
    else
      Delayed_Filter()
    endif
    break
endswitch
```

General Purpose Timer Array (GPTA)

Prescaler()

```
if (FPCK.Timer >= FPCK.Compare_Value) then
    generate pulse on FPCK.Signal_Output.Transition
    generate pulse on FPCK.Signal_Output.Level
    FPCK.Timer = 0
else
    FPCK.Timer ++
endif
```

General Purpose Timer Array (GPTA)

Delayed_Filter()

```

if (FPCK.Filter_Clock[n]) then
  if (FPCK.Timer >= FPCK.Compare_Value) then
    if (FPCK.Compare_Value == 0) then //by-pass
      if (FPCK.Signal_Output.Level != FPCK.Signal_Input[m]) then
        generate pulse on FPCK.Signal_Output.Transition
        FPCK.Signal_Output.Level = FPCK.Signal_Input[m]
        FPCK.Signal_Filtered = FPCK.Signal_Output.Level
      endif
    else //delay time is over
      generate pulse on FPCK.Signal_Output.Transition
      FPCK.Signal_Output.Level = !FPCK.Signal_Output.Level
      FPCK.Signal_Filtered = FPCK.Signal_Output.Level
    endif
    FPCK.Timer = 0
  else
    if (FPCK.Timer != 0) then //delay time is running
      if (FPCK.Rising_Edge is detected) then //edge detection done at clock input
        FPCK.Rising_Edge_Glitch = 1
      else
        if (FPCK.Falling_Edge is detected) then //edge detection done at clock input
          FPCK.Falling_Edge_Glitch = 1
        endif
      endif
    endif
    if (FPCK.Signal_Output.Level != FPCK.Signal_Input[m])
    then //expected level
      FPCK.Timer ++
    else //unexpected level
      if (FPCK.Timer != 0) then
        if (FPCK.Reset_Timer) then
          FPCK.Timer = 0
        else
          FPCK.Timer --
        endif
      endif
    endif
  endif
endif
endif

```

General Purpose Timer Array (GPTA)

Immediate_Filter()

```
if (FPCK.Filter_Clock[n]) then
  if (FPCK.Timer == 0) then
    if (FPCK.Signal_Output.Level != FPCK.Signal_Input[m]) ) then //change detected
      generate pulse on FPCK.Signal_Output.Transition
      FPCK.Signal_Output.Level = FPCK.Signal_Input[m]
    if ( (FPCK.Compare_Value == 0) or
      ((FPCK.Mode == IMMEDIATE_FILTER_RISING) and !FPCK.Signal_Input[m]) or
      ((FPCK.Mode == IMMEDIATE_FILTER_FALLING) and FPCK.Signal_Input[m]) )
    then //by-pass
      FPCK.Signal_Filtered = FPCK.Signal_Output.Level
    else //start delay time
      FPCK.Timer ++
    endif
  endif
else
  if (FPCK.Timer >= FPCK.Compare_Value) then //delay time is over
    FPCK.Timer = 0
    FPCK.Signal_Filtered = FPCK.Signal_Output.Level
  else //delay time is running
    FPCK.Timer ++
    if (FPCK.Rising_Edge) then
      FPCK.Rising_Edge_Glitch = 1
    else
      if (FPCK.Falling_Edge) then
        FPCK.Falling_Edge_Glitch = 1
      endif
    endif
  endif
endif
endif
endif
endif
```

General Purpose Timer Array (GPTA)

Variables

Input, Local, Output variables of the cell (I, L, O)

Name k = [0 to 5] for FPC m = [0 to 5] for Signal n = [0 to 3] for Clock	Short Name (*FPC	Used (ILO)	Comment
FPCK.Signal_Input[m]	*SINm	I	Signal input selected by FPCK.Input_Source
FPCK.Filter_Clock[n]	*CINn	I	Filter Clock selected by FPCK.Clock_Source
FPCK.Rising_Edge	*RE	L	Signal coming from the edge detect
FPCK.Falling_Edge	*FE	L	Signal coming from the edge detect
FPCK.Signal_Filtered	*SF	L	Filtered output signal (after delay time), initialized to 0 at reset
FPCK.Signal_Output.Transition FPCK.Signal_Output.Level	*SOTk *SOLk	O	Transition/Level of the output signal, initialized to 0 at reset

General Purpose Timer Array (GPTA)

Global variables

Name k = [0 to 5] for FPC	Short Name (*)FPC	Size (bits)	Function
FPCk.Mode	*MODk	3	Selects one of these modes: DELAYED_FILTER_BOTH IMMEDIATE_FILTER_BOTH IMMEDIATE_FILTER_RISING IMMEDIATE_FILTER_FALLING MIXED_FILTER_RISING_DELAYED MIXED_FILTER_RISING_IMMEDIATE PRESCALER_RISING PRESCALER_FALLING
FPCk.Input_Source	*IPSk	3	Selects input signal
FPCk.Clock_Source	*CLKk	2	Selects FPC clock
FPCk.Rising_Edge_Glitch	*REGk	1	Bit is set when rising edge glitch occurs during filtering
FPCk.Falling_Edge_Glitch	*FEGk	1	Bit is set when falling edge glitch occurs during filtering
FPCk.Timer	*TIMk	16	Timer value
FPCk.Reset_Timer	*RTGk	1	Reset timer on glitch in Delayed Filter Mode
FPCk.Compare_Value	*CMPk	16	Compare value

24.2.6.2 PDL-Algorithm

PDLx_Control_Logic() “to be performed every GPTA clock”

```
if (x == 0) then
  S1.Level = FPC0.Signal_Output.Level
  S1.Transition = FPC0.Signal_Output.Transition
  S2.Level = FPC1.Signal_Output.Level
  S2.Transition = FPC1.Signal_Output.Transition
  S3.Level = FPC2.Signal_Output.Level
  S3.Transition = FPC2.Signal_Output.Transition
else //x = 1
  S1.Level = FPC3.Signal_Output.Level
  S1.Transition = FPC3.Signal_Output.Transition
  S2.Level = FPC4.Signal_Output.Level
  S2.Transition = FPC4.Signal_Output.Transition
  S3.Level = FPC5.Signal_Output.Level
  S3.Transition = FPC5.Signal_Output.Transition
endif

if (PDLx.Three_Sensors_Enable) then
  Three_Sensors()
else
  Two_Sensors()
endif

if (PDLx.Mux) then
  PDLx.Signal_Output1.Level = 1
  if (PDLx.Signal_Forward or PDLx.Signal_Backward) then
    PDLx.Signal_Output1.Transition = 1
  else
    PDLx.Signal_Output1.Transition = 0
  endif
else
  PDLx.Signal_Output1.Transition = S1.Transition
  PDLx.Signal_Output1.Level = S1.Level
endif
```

General Purpose Timer Array (GPTA)

Two_Sensors()

```
if ( ( S1.Level and !S2.Level and S1.Transition) or
    ( S1.Level and S2.Level and S2.Transition) or
    (!S1.Level and S2.Level and S1.Transition) or
    (!S1.Level and !S2.Level and S2.Transition) ) then
    generate pulse on PDLx.Signal_Forward
else
    if ( ( S1.Level and S2.Level and S1.Transition) or
        (!S1.Level and S2.Level and S2.Transition) or
        (!S1.Level and !S2.Level and S1.Transition) or
        ( S1.Level and !S2.Level and S2.Transition) ) then
        generate pulse on PDLx.Signal_Backward
    endif
endif
```

```
PDLx.Signal_Output2.Level = S3.Level
PDLx.Signal_Output2.Transition = S3.Transition
```

General Purpose Timer Array (GPTA)

Three_Sensors()

```
if ( ( S1.Level and !S2.Level and S3.Level and S1.Transition) or
    ( S1.Level and !S2.Level and !S3.Level and S3.Transition) or
    ( S1.Level and S2.Level and !S3.Level and S2.Transition) or
    (!S1.Level and S2.Level and !S3.Level and S1.Transition) or
    (!S1.Level and S2.Level and S3.Level and S3.Transition) or
    (!S1.Level and !S2.Level and S3.Level and S2.Transition) ) then
    generate pulse on PDLx.Signal_Forward
else
    if ( ( S1.Level and S2.Level and !S3.Level and S1.Transition) or
        (!S1.Level and S2.Level and !S3.Level and S3.Transition) or
        (!S1.Level and S2.Level and S3.Level and S2.Transition) or
        (!S1.Level and !S2.Level and S3.Level and S1.Transition) or
        ( S1.Level and !S2.Level and S3.Level and S3.Transition) or
        ( S1.Level and !S2.Level and !S3.Level and S2.Transition) ) then
        generate pulse on PDLx.Signal_Backward
    endif
endif

if ( (S1.Level == S2.Level) and (S1.Level == S3.Level) ) then //error
    if (!PDLx.Signal_Output2.Level) then //rising edge
        generate pulse on PDLx.Signal_Output2.Transition
    endif
    PDLx.Signal_Output2.Level = 1
    PDLx.Error = 1
else //no error
    if (PDLx.Signal_Output2.Level) then //falling edge
        generate pulse on PDLx.Signal_Output2.Transition
    endif
    PDLx.Signal_Output2.Level = 0
endif
```

General Purpose Timer Array (GPTA)

Variables

Input, Local, Output variables of the cell (I, L, O)

Name x = [0,1] for PDL k = [0 to 5] for FPC	Short Name (*PDL	Used (ILO)	Comment
FPCk.Signal_Output.Transition FPCk.Signal_Output.Level	SOTk SOLk	I	Transition/Level of signals coming from FPC
S1.Transition, S1.Level S2.Transition, S2.Level S3.Transition, S3.Level	S1T, S1L S2T, S2L S3T, S3L	L	Transition/Level of Local FPC signals
PDLx.Signal_Output1.Transition PDLx.Signal_Output1.Level	SIT0, SIL0 SIT2, SIL2	O	Transition/Level of Output 1 signal going to DCM0/DCM2
PDLx.Signal_Output2.Transition PDLx.Signal_Output2.Level	SIT1, SIL1 SIT3, SIL3	O	Transition/Level of Output 2 signal going to DCM1/DCM3
PDLx.Signal_Forward	*F0 *F1	O	Forward signals to be counted by LTC
PDLx.Signal_Backward	*B0 *B1	O	Backward signals to be counted by LTC

Global variables

Name x = [0,1] for PDL	Short Name (*PDL	Size (bits)	Function
PDLx.Mux	*MUXx	1	Selects PDL speed signal (instead of FPC feed-through signal) for output 1
PDLx.Three_Sensors_Enable	*TSEx	1	Selects 3-sensor option and PDL error signal (instead of FPC feed-through signal) for output 2
PDLx.Error	*ERRx	1	Makes it possible for the software to read PDL error

24.2.6.3 DCM-Algorithm

DCMk_Control_Logic() “to be performed every GPTA clock”

```
Compare()
Add_Clock()
Check_Input()
```

```
Compare()
```

```
if (DCMk.Timer == DCMk.Capcom_Value) then
    trig(DCMk.Service_Request_Compare)
endif
```

```
Add_Clock()
```

```
if (DCMk.Clock_Request) then
    Generate DCMk.Signal_Output
    DCMk.Clock_Request = 0
endif
```

General Purpose Timer Array (GPTA)

Check_Input()

```
if (DCMk.Signal_Input.Transition) then
  if (DCMk.Signal_Input.Level) then //rising edge
    trig(DCMk.Service_Request_Rising)
    if (DCMk.Capture_On_Rising_Edge) then
      DCMk.Capture_Value = DCMk.Timer
    else
      if (DCMk.Capcom_Opposite) then
        DCMk.Capcom_Value = DCMk.Timer
      endif
    endif
    if (DCMk.Clear_On_Rising_Edge) then
      DCMk.Timer = 0
    else DCMk.Timer ++
    endif
    if (DCMk.Clock_On_Rising_Edge) then
      Generate pulse on DCMk.Signal_Output
    endif
  else //falling edge
    trig(DCMk.Service_Request_Falling)
    if (!DCMk.Capture_On_Rising_Edge) then
      DCMk.Capture_Value = DCMk.Timer
    else
      if (DCMk.Capcom_Opposite) then
        DCMk.Capcom_Value = DCMk.Timer
      endif
    endif
    if (DCMk.Clear_On_Falling_Edge) then
      DCMk.Timer = 0
    else DCMk.Timer ++
    endif
    if (DCMk.Clock_On_Falling_Edge) then
      Generate pulse on DCMk.Signal_Output
    endif
  endif
  else DCMk.Timer ++
endif
```

General Purpose Timer Array (GPTA)

Variables

Input, Local, Output variables of the cell (I, L, O)

Name k = [0 to 3] for DCM	Short Name	Used (ILO)	Comment
DCMk.Signal_Input.Transition DCMk.Signal_Input.Level	*SITk *SILk	I	Input of the cell
DCMk.Signal_Output	*SOK	O	Output of the cell
DCMk.Service_Request_Rising	*RTQk	O	Service request on rising edge
DCMk.Service_Request_Falling	*FTQk	O	Service request on falling edge
DCMk.Service_Request_Compare	*CTQk	O	Service request on compare event

Global variables

Name k = [0 to 3] for DCM	Short Name (*)DCM	Size (bits)	Function
DCMk.Capture_On_Rising_Edge	*RCAk	1	Capture into Capture_Value on rising edge
DCMk.Capcom_Opposite	*OCAk	1	Capture into Capcom_Value on opposite edge defined by RCAk
DCMk.Clear_On_Rising_Edge	*RZEK	1	Clear Timer on rising edge
DCMk.Clear_On_Falling_Edge	*FZEK	1	Clear Timer on falling edge
DCMk.Clock_On_Rising_Edge	*RCKk	1	Generate a single clock pulse on rising edge
DCMk.Clock_On_Falling_Edge	*FCKk	1	Generate a single clock pulse on falling edge
DCMk.Clock_Request	*QCKk	1	Generate a single clock pulse immediately
DCMk.Request_Enable_Rising	*RREK	1	Enable request on rising edge
DCMk.Request_Enable_Falling	*FREK	1	Enable request on falling edge
DCMk.Request_Enable_Compare	*CREK	1	Request enable on compare
DCMk.Timer	*TIMk	24	Timer value

General Purpose Timer Array (GPTA)

Name k = [0 to 3] for DCM	Short Name (*)DCM	Size (bits)	Function
DCMk.Capture_Value	*CAV _k	24	Capture value
DCMk.Capcom_Value	*COV _k	24	Capture/compare value

24.2.6.4 PLL-Algorithm

PLL_Control_Logic() “to be performed every GPTA clock”

```

if ( (Pll.Automatic_End) and (Pll.Event) ) then //allow compensation
    Pll.Perform_End = 1
endif

if ( (Pll.Counter_Mtick == 0) and ((Pll.Perform_End) or (!Pll.Automatic_End)) )
then //compensation finished or no automatic compensation
    Pll.Counter_Mtick = Pll.Number_Mtick
    Pll.Perform_End = 0
endif

if ( (Pll.Counter_Mtick != 0) and ((Pll.Perform_End) or (Bit 24 of Pll.Delta)) )
then //output pulse is necessary
    generate pulse on Pll.Signal_Output
    Pll.Counter_Mtick --
    if (Pll.Counter_Mtick == 0) then
        trig(Pll.Service_Request_Trigger)
    endif
endif

if (Bit 24 of Pll.Delta) then //delta is < 0
    Pll.Delta = Pll.Delta + Pll.Reload_Value
    generate pulse on Pll.Signal_Uncomp
else //delta is >= 0
    Pll.Delta = Pll.Delta + (0xFFFF0000 or (Pll.Step))
endif

```

General Purpose Timer Array (GPTA)

Variables

Input, Local, Output variables of the cell (I, L, O)

Name k = [0 to 3] for DCM	Short Name (*PLL)	Used (ILO)	Comment
DCMk.Signal_Output	SOk	I	Input of the cell from DCM
PII.Event	*EVE	L	Input selected by the multiplexer
PII.Signal_Output	*SO	O	Output of the cell
PII.Signal_Uncomp	*SU	O	Uncompensated output of the cell
PII.Service_Request_Trigger	*SQT	O	Service request when Counter reaches zero

Global variables

Name	Short Name (*PLL)	Size (bits)	Function
PII.Mux	*MUX	2	Selects the signal input for PLL
PII.Automatic_End	*AEN	1	Performs the acceleration/ deceleration correction
PII.Perform_End	*PEN	1	Makes it possible to decrement the Counter at full speed
PII.Request_Enable	*REN	1	Allows a request when microtick counter reaches zero
PII.Number_Mtick	*MTI	16	Number of microticks per input signal period
PII.Counter_Mtick	*CNT	16	Microtick counter
PII.Step	*STP	16	Step value, to be added to positive/zero delta register
PII.Reload_Value	*REV	24	Reload value, to be added to negative delta register
PII.Delta	*DTR	25	Delta register

General Purpose Timer Array (GPTA)

24.2.6.5 GT-Algorithm

GTm_Control_Logic() “to be performed every GPTA clock”

```

if (GTm.Run) then
  if (Event on GTm.Clock_In[p] selected by GTm.Clock_Mux) then
    GTm.Timer ++
    if (Overflow of GTm.Timer) then
      GTm.Timer = GTm.Reload_Value
      trig(GTm.Service_Request_Trigger)
    endif
  endif
endif
endif

```

Variables

Input, Local, Output variables of the cell (I, L, O)

Name m = [0, 1] for GT p = [0 to 7] for Clock Bus	Short Name (*GT	Used (ILO)	Comment
GTm.Clock_In[p]	*CINmp	I	Input coming from clock bus
GTm.Timer_Greater_Equal_Comp	TGEm	O	Timer is greater or equal
GTm.Timer_Event	TEVm	O	Signal for timer change
GTm.Service_Request_Trigger	*SQTm	O	Service request line

Global variables

Name m = [0, 1] for GT	Short Name (*GT	Size (bits)	Function
GTm.Run	*RUNm	1	Enables timer
GTm.Scale_Compare	*SCOm	4	Selects compare flag
GTm.Clock_Mux	*MUXm	3	Selects clock from clock bus
GTm.Request_Enable	*RENm	1	Allows a request when timer overflows
GTm.Timer	*TIMm	24	Timer value
GTm.Reload_Value	*REVm	24	Reload value when timer overflows

24.2.6.6 GTC-Algorithm

GTck_Control_Logic() “to be performed every GPTA clock”

```
if (GTck.Cell_Enable) then
  switch (GTck.Mode)
    case CAPTURE_T0:
      Capture(0)
      break
    case CAPTURE_T1:
      Capture(1)
      break
    case COMPARE_T0:
      Compare(0)
      break
    case COMPARE_T1:
      Compare(1)
  endswitch

  if ( (GTck.One_Shot_Mode) and (GTck.Event) ) then
    GTck.Cell_Enable = 0
  endif
endif

Manage_Mux()
```

Capture(m)

```
if (GTck.Signal_Input) then
  trig(GTck.Service_Request_Trigger)
  GTck.X = GTm.Timer
  GTck.Event = 1
else
  GTck.Event = 0
endif
Ck.Event = 0
```

General Purpose Timer Array (GPTA)

Compare(m)

```
if ( ((GTck.X == GTm.Timer) and ((GTck.X_Write_Access) or (GTm.Timer_Event))) or
  ((GTck.Greater_Equal_Select) and (GTck.X_Write_Access)
  and (GTm.Timer_Greater_Equal_Comp)) ) then
  if (GTck.Capture_After_Compare) then
    if (GTck.Capture_Alternate_Timer) then
      GTck.X = GT(!m).Timer
    else
      GTck.X = GTm.Timer
    endif
  endif
  trig(GTck.Service_Request_Trigger)
  GTck.Event = 1
else
  GTck.Event = 0
endif
```

General Purpose Timer Array (GPTA)

Manage_Mux()

```
if ((GTck.Event or GTck.OIA) and GTck.OCM != x00) then //local event
  Set_Data_Out(GTck.Output_Control_Mode.[1:0])
  if (!GTck.Bypass) then //no bypass
    GTck.Output_Mode_Out = GTck.Output_Control_Mode.[1:0]
  else
    if (GTck.Output_Control_Mode.2) then //bypass, input link enabled
      GTck.Output_Mode_Out = GTck.Output_Mode_In
    else //bypass, input link disabled
      GTck.Output_Mode_Out = 00B
    endif
  endif
endif
else //no local event
  if (GTck.Output_Control_Mode.2) then //input link enabled
    Set_Data_Out(GTck.Output_Mode_In)
    GTck.Output_Mode_Out = GTck.Output_Mode_In
  else //input link disabled
    Set_Data_Out(00B)
    GTck.Output_Mode_Out = 00B
  endif
endif

if ( (GTck.Enable_Of_Action) and
  ((GTck.Output_Mode_In.1) or (GTck.Output_Mode_In.0)) ) then
  GTck.Cell_Enable = 1
  GTck.Enable_Of_Action = 0
endif
```

General Purpose Timer Array (GPTA)

Set_Data_Out(mode)

```
switch (mode)
  case 00B: //no change
    break
  case 01B: //toggle
    GTCK.Data_Out = !GTCK.Data_Out
    break
  case 10B: //clear
    GTCK.Data_Out = 0
    break
  case 11B: //set
    GTCK.Data_Out = 1
    break
endswitch
GTCK.Output_State = GTCK.Data_Out
```

General Purpose Timer Array (GPTA)

Variables

Input, Local, Output variables of the cell (I, L, O)

Name k = [0 to 31] for GTC m = [0, 1] for GT	Short Name (*GTC	Used (ILO)	Comment
GTm.Timer_Greater_Equal_Comp	TGEm	I	Timer is greater or equal
GTm.Timer_Event	TEVm	I	Signal for timer change
GTm.Timer	*TIMm	I	Timer value
GTck.Data_In	*DINk	I	Data input from input multiplexer
GTck.Output_Mode_In	*M1Ik *M0Ik	I	Link signals from preceding cell
GTck.X_Write_Access	*XWA	L	Indicates that GTck.X was modified
GTck.Event	*EVE	L	Local event
GTck.Signal_Input	*INS	L	Qualified input signal
GTck.Service_Request_Trigger	*SQSk	O	Service request line
GTck.Data_Out	*DOUk	O	Data output for output multiplexer
GTck.Output_Mode_Out	*M1Ok *M0Ok	O	Link signals to following cell

General Purpose Timer Array (GPTA)

Global variables

Name k = [0 to 31] for GTC	Short Name (*)GTC	Size (bits)	Comment
GTCK.Mode	*MODk	2	Operation mode: CAPTURE_T0, CAPTURE_T1, COMPARE_T0, COMPARE_T1
GTCK.One_Shot_Mode	*OSMk	1	One shot mode
GTCK.Request_Enable	*RENk	1	Allows a request on event
GTCK.Input_Rising_Edge_Select (Capture Mode)	*REDk	1	Selects rising edge of input pin
GTCK.Greater_Equal_Select (Compare Mode)	*GESk	1	Selects >= Compare Mode
GTCK.Input_Falling_Edge_Select (Capture Mode)	*FEDk	1	Selects falling edge of input pin
GTCK.Capture_After_Compare (Compare Mode)	*CACK	1	Selects capture after compare
GTCK.Capture_Alternate_Timer (Compare Mode)	*CATk	1	Capture alternate global timer after compare
GTCK.Bypass	*BYPk	1	Local events bypassed for output link
GTCK.Enable_Of_Action	*EOAk	1	Enables cell on action communicated via link
GTCK.Cell_Enable	*CENk	1	Cell enable state
GTCK.Output_Control_Mode	*OCMk	3	Output control mode
GTCK.Output_Immediate_Action	*OIAk	1	Forces immediate action
GTCK.Output_State	*OUTk	1	Read value of Data_Out
GTCK.X	*Xk	24	Capture/Compare value

24.2.6.7 LTC-Algorithm for Cells 0 to 62

LTck_Control_Logic() “to be performed every GPTA clock”

```
if (LTck.Cell_Enable) then
  switch (LTck.Mode)
    case TIMER_FREE_RUN:
      LTck.Reset_Timer_Bit = 0
      Timer()
      break
    case TIMER_RESET:
      if (LTck.Event_In) then
        LTck.Reset_Timer_Bit = 1
      endif
      Timer()
      break;
    case CAPTURE:
      Capture()
      break
    case COMPARE:
      Compare()
      break
  endswitch
  if ((LTck.One_Shot_Mode) and (LTck.Event)) then
    LTck.Cell_Enable = 0
  endif
endif

Manage_Mux()
```

General Purpose Timer Array (GPTA)

Timer()

```
if ( (LTck.X == 0xFFFF) and (LTck.X_Write_Access) ) then
    //above condition is also true for timer overflow or software reset
    trig(LTck.Service_Request_Trigger)
    LTck.Event = 1
else
    LTck.Event = 0
endif
if (LTck.Signal_Input) then
    if (LTck.Reset_Timer_Bit) then //timer must be reset
        LTck.Reset_Timer_Bit = 0
        LTck.X = 0xFFFF
        if (LTck.Coherent_Update_Enable) then
            LTck.Select_Line_Value = !LTck.Select_Line_Value
            LTck.Coherent_Update_Enable = 0
        endif
    else //timer runs normally
        LTck.X ++
    endif
endif
LTck.Event_Out = LTck.Event
```

General Purpose Timer Array (GPTA)

Capture()

```
if (LTck.Signal_Input) then
    trig(LTck.Service_Request_Trigger)
    LTck.X = LTck.Y_In
    LTck.Event = 1
else
    LTck.Event = 0
endif
LTck.Event_Out = LTck.Event
```

Compare()

```
if ( ((LTck.Select_In) and (LTck.Select_On_High_Level)) or
    ((!LTck.Select_In) and (LTck.Select_On_Low_Level)) ) then //cell is active
    if ( (LTck.X == LTck.Y_In) and
        ((LTck.X_Write_Access) or (LTck.Timer_Event_In)) ) then //event
        trig(LTck.Service_Request_Trigger)
        LTck.Event = 1
    else
        LTck.Event = 0
    endif
    LTck.Event_Out = LTck.Event
else //cell is inactive
    LTck.Event_Out = LTck.Event_In
endif
```

General Purpose Timer Array (GPTA)

Manage_Mux()

```

if ( (LTCh.Mode == TIMER_FREE_RUN) or (LTCh.Mode == TIMER_RESET) ) then
    LTCh.Y_Out = LTCh.X
    if (the timer has been modified) then //increment, reset, software overwrite
        LTCh.Timer_Event_Out = 1
    else
        LTCh.Timer_Event_Out = 0
    endif
    LTCh.Select_Out = LTCh.Select_Line_Value
else //capture mode or compare mode
    LTCh.Y_Out = LTCh.Y_In
    LTCh.Timer_Event_Out = LTCh.Timer_Event_In
    LTCh.Select_Line_Value = LTCh.Select_In
    LTCh.Select_Out = LTCh.Select_In
endif

if (LTCh.Event) then //local event
    Set_Data_Out(LTCh.Output_Control_Mode.[1:0])
    if (!LTCh.Bypass) then //no bypass
        LTCh.Output_Mode_Out = LTCh.Output_Control_Mode.[1:0]
    else
        if (LTCh.Output_Control_Mode.2) then //bypass, input link enabled
            LTCh.Output_Mode_Out = LTCh.Output_Mode_In
        else //bypass, input link disabled
            LTCh.Output_Mode_Out = 00B
        endif
    endif
endif
else //no local event
    if (LTCh.Output_Control_Mode.2) //input link enabled
        Set_Data_Out(LTCh.Output_Mode_In)
        LTCh.Output_Mode_Out = LTCh.Output_Mode_In
    else //input link disabled
        Set_Data_Out(00B)
        LTCh.Output_Mode_Out = 00B
    endif
endif
endif
if ( (LTCh.Enable_Of_Action) and
    ((LTCh.Output_Mode_In.1) or (LTCh.Output_Mode_In.0)) ) then //enable condition
    LTCh.Cell_Enable = 1
    LTCh.Enable_Of_Action = 0
endif

```

General Purpose Timer Array (GPTA)

Set_Data_Out(mode)

```
switch (mode)
  case 00B: //no change
    break
  case 01B: //toggle
    LTCK.Data_Out = !LTCK.Data_Out
    break
  case 10B: //clear
    LTCK.Data_Out = 0
    break
  case 11B: //set
    LTCK.Data_Out = 1
    break
endswitch
LTCK.Output_State = LTCK.Data_Out
```

General Purpose Timer Array (GPTA)

Variables

Input, Local, Output variables of the cell (I, L, O)

Name	Short Name (*)LTC	Used (ILO)	Comment
LTck.Data_In	*DINkp	I	Data input from input multiplexer
LTck.Y_In	*YIk	I	Timer coming from preceding cell
LTck.Output_Mode_In	*M1Ik *M0Ik	I	Link signals coming from preceding cell
LTck.Timer_Event_In	*TIk	I	Signal for timer change from preceding cell
LTck.Event_In	*EIk	I	Signal for event from following cell
LTck.Select_In	*SI	I	Select signal from preceding cell
LTck.X_Write_Access	*XWA	L	Indicates that LTck.X was modified
LTck.Select_Line_Value	*SLV	L	Internal value for select line reset value: 0
LTck.Signal_Input	*INS	L	Qualified input signal for Timer Mode and Capture Mode
LTck.Reset_Timer_Bit	*RTM	L	Flipflop to reset timer on next clock
LTck.Event	*EVE	L	Local event
LTck.Data_Out	*DOUk	O	Data output for output multiplexer
LTck.Service_Request_Trigger	*SQTk	O	Service request line
LTck.Y_Out	*YOK	O	Timer going to following cell
LTck.Output_Mode_Out	*M1Ok *M0Ok	O	Link signals to following cell
LTck.Timer_Event_Out	*TOK	O	Event output to following cell
LTck.Select_Out	*SO	O	Select output to following cell
LTck.Event_Out	*EOK	O	Event output to preceding cell

General Purpose Timer Array (GPTA)

Global variables

Name k = [0 to 62] for LTC	Short Name (*)LTC	Size (bits)	Comment
LTck.Mode	*MODk	2	Operation mode: TIMER, TIMER_RESET, CAPTURE, COMPARE
LTck.One_Shot_Mode	*OSMk	1	One shot mode
LTck.Request_Enable	*RENk	1	Allows a request on event
LTck.Input_Rising_Edge_Select (Timer Mode, Capture Mode)	*REDk	1	Selects rising edge of input pin
LTck.Select_On_Low_Level (Compare Mode)	*SOLk	1	Enables compare on low level of select line
LTck.Input_Falling_Edge_Select (Timer Mode, Capture Mode)	*FEDk	1	Selects falling edge of input pin
LTck.Select_On_High_Level (Compare Mode)	*SOHk	1	Enables compare on high level of select line
LTck.Bypass (Capture Mode, Compare Mode)	*BYPk	1	Local events bypassed for output link
LTck.Enable_Of_Action (Capture Mode, Compare Mode)	*EOAk	1	Enables cell on action communicated via link
LTck.Input_Line_Mode	*ILMk	1	Selects edge input line mode
LTck.Coherent_Update_Enable (Timer Mode)	*CUDk	1	Selects coherent update
LTck.Select_Line_Level (Capture Mode, Compare Mode)	*SLLk	1	Select line level
LTck.Cell_Enable	*CENk	1	Cell enable state
LTck.Output_Control_Mode	*OCMk	3	Output control mode
LTck.Output_Immediate_Action	*OIAk	1	Forces immediate action
LTck.Output_State	*OUTk	1	Read value of Data_Out
LTck.X	*Xk	16	Timer/Capture/Compare value

24.2.6.8 LTC Algorithm for Cell 63

LTC63_Control_Logic() “to be performed every GPTA clock”

Copy()
Compare()

Copy()

```
if (LTC63.Cell_Enable) then
  if (LTC63.Signal_Input) then
    LTC63.X = LTC63.X_Shadow
    trig(LTC63.Service_Request_Trigger)
    if (LTC63.One_Shot_Mode) then
      LTC63.Cell_Enable = 0
    endif
  endif
endif
endif
```

General Purpose Timer Array (GPTA)

Compare()

```
if ( (LTC63.X_Write_Access) or (LTC63.Timer_Event_In) ) then
  if (LTC63.Bit_Rev_Mode) then
    LTC63.Y_Comp = LTC63.Y_Rev
  else
    LTC63.Y_Comp = LTC63.Y_In
  endif
  if ( (LTC63.X > LTC63.Y_Comp) or (LTC63.X == FFFFH) ) then //output must be 1
    LTC63.Data_Out = 1
    LTC63.Event_Out = 0
  else //output must be 0
    if (LTC63.Data_Out == 1) then //falling edge on output
      trig(LTC63.Service_Request_Trigger)
      LTC63.Event_Out = 1
    else
      LTC63.Event_Out = 0
    endif
    LTC63.Data_Out = 0
  endif
  LTC63.Output_State = LTC63.Data_Out
endif
```

General Purpose Timer Array (GPTA)

Variables

Input, Local, Output variables of the cell (I, L, O)

Name	Short Name (*LTC)	Used (ILO)	Comment
LTC63.Data_In	*DIN63	I	Data input from input multiplexer
LTC63.Y_In	*YI63	I	Timer coming from preceding cell
LTC63.Timer_Event_In	*TI63	I	Signal for timer change from preceding cell
LTC63.Y_Rev	*YR	L	Timer coming from preceding cell, bit reversed
LTC63.Y_Comp	*YC	L	Timer actually used for compare
LTC63.X_Write_Access	*XWA	L	Indicates that LTC63.X was modified
LTC63.Signal_Input	*INS	L	Qualified input signal
LTC63.Data_Out	*DOU63	O	Data output for output multiplexer
LTC63.Service_Request_Trigger	*SQT63	O	Service request line
LTC63.Event_Out	*EO63	O	Event output to preceding cell

General Purpose Timer Array (GPTA)

Global variables

Name	Short Name (*)LTC	Size (bits)	Comment
LTC63.Bit_Rev_Mode	*BRM63	1	Bit reverse mode
LTC63.One_Shot_Mode	*OSM63	1	One shot mode for copy
LTC63.Request_Enable	*REN63	2	Allows a request on compare or copy
LTC63.Input_Rising_Edge_Select	*RED63	1	Selects rising edge of input pin
LTC63.Input_Falling_Edge_Select	*FED63	1	Selects falling edge of input pin
LTC63.Input_Line_Mode	*ILM63	1	Selects edge input line mode
LTC63.Cell_Enable	*CEN63	1	Cell enable state for copy
LTC63.Output_State	*OUT63	1	Read value of Data_Out
LTC63.X	*X63	16	Compare value
LTC63.X_Shadow	*XS63	16	Shadow compare value

General Purpose Timer Array (GPTA)

24.2.7 Programming of a GPTA Module

A hierarchical top-down design approach may be used to implement a complex signal processing unit as follows:

- Partitioning the complex signal processing unit into simple function units.
- Implementing each simple function unit by configuring the LTC and/or GTC cells which can be tied together for realizing a common signal operation.
- Implementing necessary signal pre-processing tasks by configuring the FPC, PDL, DCM and PLL cells accordingly.
- Defining and configuring all input/output port pins required as clock source, trigger input or signal output.

Table 24-15 summarizes all of the software tasks to be implemented for getting a GPTA unit into operation.

Table 24-15 Software Tasks Controlling a GPTA Unit

GPTA Shell Initialization	
GPTA Module Clock Enable	
Fractional Divider Setting	
Module Enable	
Configuration of Interrupt Handling	
GPTA Kernel Initialization	
FPC:	PDL:
Selection of Operating Mode (Prescaler, Filter or Feed-Through)	Selection of Operating Mode (Phase Discriminator or Feed-Through)
Input Channel Selection	2- or 3-Sensor Mode Selection
Clock Selection	PLL:
Configuration of Prescaler Factor or Debounce Mode	Selection of Input Channel
DCM:	Estimation of Input Signal Period Width
Selection of Reset Event for Timer	Configuration of Output Signal Frequency
Selection of Trigger Source for Capture Event	Handling of Input Signal Period Length Variation
Selection of Trigger Source for Capcom Register Update	Interrupt Request Enable on End of Output Pulse Generation
Interrupt Request Enable on Input Edge or Compare Event	

General Purpose Timer Array (GPTA)

Table 24-15 Software Tasks Controlling a GPTA Unit (cont'd)

Clock Bus Setup

Selection and Configuration of 8 Clock Sources for GT, GTC and LTC Cells	
GT:	GTC:
Selection of Timer Clock Source	Selection of Operating Mode (Capture or Compare) and Time Base (GT0 or GT1)
Configuration of Timer Width (Reload Value, TGE Flag)	Configuration of Trigger Events for Capture Mode or Selection of a Relational Operator for Compare Mode
Interrupt Request Enable on Timer Overflow	Interrupt Request Enable on Capture or Compare Event
Start Global Timer(s)	Configuration of Data Output triggered by a GTC Event
LTC:	IOLS:
Selection of Operating Mode (Timer, Capture or Compare)	Configuration of the Multiplexer Array to link GTC and LTC data outputs/inputs to external Port Pins or other cells by writing the Multiplexer Register Array FIFO
Selection of Trigger Source for Timer, Capture or Compare Mode	Configuration of Port Output Source
Configuration of Trigger Event for Timer, Capture or Compare Mode	
Interrupt Request Enable on Timer, Capture or Compare Event	
Configuration of Data Output triggered by an LTC Event	
Port Initialization	
Definition of Electrical Port Characteristic	
Configuration of Port Pin Direction (Input or Output)	

General Purpose Timer Array (GPTA)

24.3 GPTA0/1 Kernel Registers

This section describes the kernel registers of the GPTA0 module. The complete and detailed address maps of the GPTA0, GPTA1, and LTCA2 modules is described in [Table 18-20](#) on [Table 18-20](#), in [Table 18-21](#) on [Table 18-21](#), and in [Table 18-22](#) on [Table 18-22](#) of the TC1796 User's Manual System Units part (Volume 1).

GPTA Kernel Register Overview

Control Registers	Data Registers	Interrupt & IOLS Registers	Multiplexer Array FIFO Registers
FPCSTAT	FPCTIMk ¹⁾	SRSCn ⁶⁾	OMRCLg ⁷⁾
FPCCTRk ¹⁾	DCMTIMk ²⁾	SRSSn ⁶⁾	OMRCHg ⁷⁾
PDLCTR	DCMCAV k ²⁾	SRNR	LIMCRLg ⁸⁾
DCMCTRk ²⁾	DCMCOV k ²⁾	MRACTL	LIMCRHg ⁸⁾
PLLCTR	PLLMTI	MRADIN	GIMCRLg ⁹⁾
CKBCTR	PLLSTP	MRADOUT	GIMCRHg ⁹⁾
GTCTRk ³⁾	PLLNT		Identification Register
GTCCTRk ⁴⁾	PLLREV	1) k = 0-5	ID
LTCCTRk ⁵⁾	PLLDR	2) k = 0-3	
	GTTIMk ³⁾	3) k = 0-2	
	GTREVK ³⁾	4) k = 00-31	
	GTCXRk ⁴⁾	5) k = 00-63	
	LTCXRk ⁵⁾	6) n = 0-3	
		7) g = 0-13	
		8) g = 0-7	
		9) g = 0-3	

Note: The Multiplexer Array FIFO registers are not directly accessible ! MCA05982_mod

Figure 24-73 GPTA0 and GPTA1 Kernel Registers

In the TC1796, the registers of the GPTA modules are located in the following address ranges.

Table 24-16 Registers Address Space

Module	Base Address	End Address	Note
GPTA0	F000 1800 _H	F000 1FFF _H	—
GPTA1	F000 2000 _H	F000 27FF _H	—
LTCA2	F000 2800 _H	F000 2FFF _H	—

General Purpose Timer Array (GPTA)

Table 24-17 Registers Overview - GPTA0 and GPTA1 Kernel Registers

Register Short Name	Register Long Name	Offset Address	Description see
ID	Module Identification Register	008 _H	Page 24-154
SRSC0	Service Request State Clear Register 0	010 _H	Page 24-206
SRSS0	Service Request State Set Register 0	014 _H	Page 24-208
SRSC1	Service Request State Clear Register 1	018 _H	Page 24-209
SRSS1	Service Request State Set Register 1	01C _H	Page 24-210
SRSC2	Service Request State Clear Register 2	020 _H	Page 24-211
SRSS2	Service Request State Set Register 2	024 _H	Page 24-212
SRSC3	Service Request State Clear Register 3	028 _H	Page 24-213
SRSS3	Service Request State Set Register 3	02C _H	Page 24-214
SRNR	Service Request Node Redirection Register	030 _H	Page 24-215
MRACTL	Multiplexer Register Array Control Register	038 _H	Page 24-191
MRADIN	Multiplexer Register Array Data In Register	03C _H	Page 24-193
MRADOUT	Multiplexer Register Array Data Out Register	040 _H	Page 24-193
FPCSTAT	Filter and Prescaler Cell Status Register	044 _H	Page 24-155
FPCCTRk	Filter and Prescaler Cell Control Register k (k = 0-5)	048 _H + k × 8	Page 24-156
FPCTIMk	Filter and Prescaler Cell Timer Register k (k = 0-5)	048 _H + k × 8 + 4	Page 24-158
PDLCTR	Phase Discrimination Logic Control Register	078 _H	Page 24-159
DCMCTRk	Duty Cycle Measurement Control Register k (k = 0-3)	080 _H + k × 16	Page 24-161
DCMTIMk	Duty Cycle Measurement Timer Register k (k = 0-3)	080 _H + k × 16 + 4	Page 24-163
DCMCAVk	Duty Cycle Measurement Capture Register k (k = 0-3)	080 _H + k × 16 + 8	Page 24-163
DCMCOVk	Duty Cycle Measurement Capture/Compare Register k (k = 0-3)	080 _H + k × 16 + 12	Page 24-164
PLLCTR	Phase Locked Loop Control Register	0C0 _H	Page 24-165

General Purpose Timer Array (GPTA)

Table 24-17 Registers Overview - GPTA0 and GPTA1 Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Address	Description see
PLLMTI	Phase Locked Loop Micro Tick Register	0C4 _H	Page 24-166
PLLCNT	Phase Locked Loop Counter Register	0C8 _H	Page 24-167
PLLSTP	Phase Locked Loop Step Register	0CC _H	Page 24-167
PLLREV	Phase Locked Loop Reload Register	0D0 _H	Page 24-168
PLLDTR	Phase Locked Loop Delta Register	0D4 _H	Page 24-168
CKBCTR	Clock Bus Control Register	0D8 _H	Page 24-173
GTCTRk	Global Timer Control Register k (k = 0, 1)	0E0 _H + k × 16	Page 24-170
GTREVK	Global Timer Reload Value Register k (k = 0, 1)	0E0 _H + k × 16 + 4	Page 24-172
GTTIMk	Global Timer Register k (k = 0, 1)	0E0 _H + k × 16 + 8	Page 24-171
GTCCTRk	Global Timer Cell Control Register k (k = 00-31)	100 _H + k × 8	Page 24-175 Page 24-177
GTCXRk	Global Timer Cell X Register k (k = 00-31)	100 _H + k × 8 + 4	Page 24-179
LTCCTRk	Local Timer Cell Control Register k (k = 00-62)	200 _H + k × 8	Page 24-180 Page 24-183 Page 24-185
LTCXRk	Local Timer Cell X Register k (k = 00-62)	200 _H + k × 8 + 4	Page 24-190
LTCCTR63	Local Timer Cell Control Register 63	3F8 _H	Page 24-188
LTCXR63	Local Timer Cell X Register 63	3FC _H	Page 24-190

General Purpose Timer Array (GPTA)

Table 24-17 Registers Overview - GPTA0 and GPTA1 Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Address	Description see
OMCRLg	Output Multiplexer Control Register for Lower Half of Group g (g = 0-13)	not directly addressable;	Page 24-194
OMCRLg	Output Multiplexer Control Register for Upper Half of Group g (g = 0-13)	see Page 24-110	Page 24-196
GIMCRLg	Input Multiplexer Control Register for Lower Half of GTC Group g (g = 3-0)		Page 24-198
GIMCRLg	Input Multiplexer Control Register for Lower Half of GTC Group g (g = 0-3)		Page 24-200
LIMCRLg	Input Multiplexer Control Register for Lower Half of LTC Group g (g = 0-7)		Page 24-202
LIMCRLg	Input Multiplexer Control Register for Upper Half of LTC Group g (g = 0-7)		Page 24-204

Bit Protection

Bits with bit protection (this is valid, for example, for all bits in the Service Request State Registers) are not changed during a read-modify-write instruction, for example when hardware sets a request state bit between the read and the write of the read-modify-write sequence. For bit protected bits it is guaranteed that a hardware setting operation always has priority. Thus, no hardware triggered events are lost.

Bits with bit protection are marked in the corresponding bit descriptions.

General Purpose Timer Array (GPTA)

24.3.1 Identification Register

The Module Identification Register ID contains read-only information about the GPTA module version.

GPTA0_ID

GPTA0 Module Identification Register

(008_H)

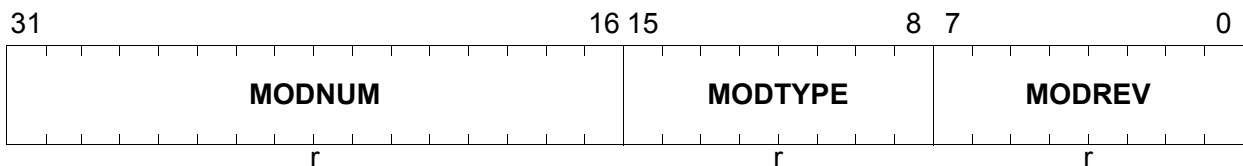
Reset Value: 0029 C0XX_H

GPTA1_ID

GPTA1 Module Identification Register

(008_H)

Reset Value: 0029 C0XX_H



Field	Bits	Type	Description
MODREV	[7:0]	r	Module Revision Number MODREV defines the module revision number. The value of a module revision starts with 01 _H (first revision).
MODTYPE	[15:8]	r	Module Type This bit field defines the module as a 32-bit module: C0 _H
MODNUM	[31:16]	r	Module Number Value This bit field defines the module identification number for the GPTA: 0029 _H

General Purpose Timer Array (GPTA)

24.3.2 FPC Registers

GPTA0_FPCSTAT

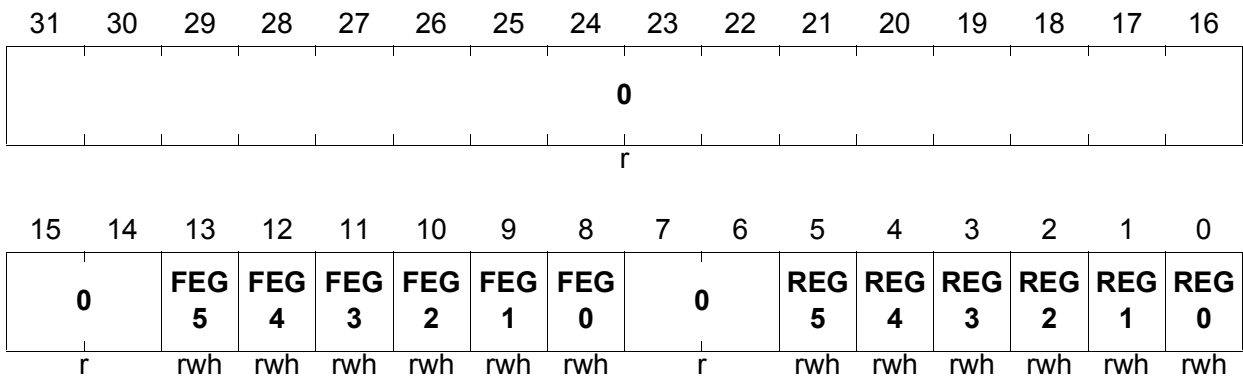
GPTA0 Filter and Prescaler Cell Status Register
(044_H)

Reset Value: 0000 0000_H

GPTA1_FPCSTAT

GPTA1 Filter and Prescaler Cell Status Register
(044_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
REGk (k = 0-5)	k	rwh	Rising Edge Glitch Flag for FPCK 0 _B No rising edge of glitch detected during filtering 1 _B Rising edge of glitch detected during filtering Bits REGk are bit protected (see Section 24.3.2).
FEGk (k = 0-5)	k+8	rwh	Falling Edge Glitch Flag for FPCK 0 _B No falling edge of glitch detected during filtering 1 _B Falling edge of glitch detected during filtering Bits FEGk are bit protected (see Section 24.3.2).
0	[7:6], [31:14]	r	Reserved Read as 0; should be written with 0.

General Purpose Timer Array (GPTA)

GPTA0_FPCCTRk (k = 0-5)

GPTA0 Filter and Prescaler Cell Control Register k

(048_H+k*8_H)

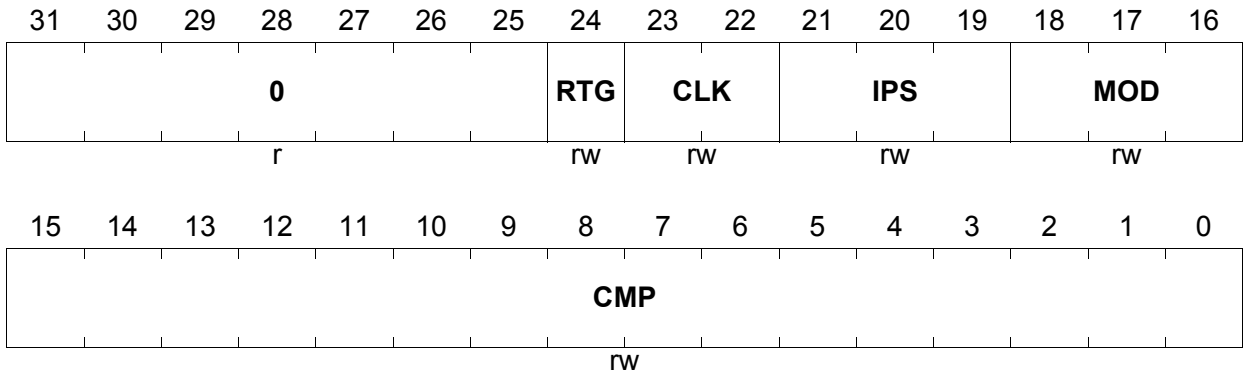
Reset Value: 0000 0000_H

GPTA1_FPCCTRk (k = 0-5)

GPTA1 Filter and Prescaler Cell Control Register k

(048_H+k*8_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CMP	[15:0]	rw	Threshold Value of Filter and Prescaler Cell k CMP is the 16-bit threshold value that is compared with the 16-bit timer value FPCTIMk.TIM.
MOD	[18:16]	rw	Operation Mode Selection for FPCk 000 _B Delayed Debounce Filter Mode on both edges 001 _B Immediate Debounce Filter Mode on both edges 010 _B Rising edge: Immediate Debounce Filter Mode, falling edge: no filtering 011 _B Rising edge: no filtering, falling edge: Immediate Debounce Filter Mode 100 _B Rising edge: Delayed Debounce Filter Mode, falling edge: Immediate Debounce Filter Mode 101 _B Rising edge: Immediate Debounce Filter Mode, falling edge: Delayed Debounce Filter Mode 110 _B Prescaler Mode (triggered on rising edge) 111 _B Prescaler Mode (triggered on falling edge)

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
IPS	[21:19]	rw	<p>Input Line Selection for FPCK IPS determines the signal input used for edge detection.</p> <p>000_B Signal input SINK0 selected 001_B Signal input SINK1 selected 010_B Signal input SINK2 selected 011_B Signal input SINK3 selected 100_B Signal input SINK4 = GPTA module clock f_{GPTA} selected 101_B Signal input SINK5 = preceding FPC output SOLk-1 selected; SIN05 is connected to SOL5 11X_B Reserved</p>
CLK	[23:22]	rw	<p>Clock Selection for FPCK CLK selects the clock signal used for edge detection.</p> <p>00_B Clock input line 0 selected (GPTA module clock f_{GPTA}) 01_B Clock bus line 1 selected (local PLL clock) 10_B Clock bus line 2 selected (prescaled) GPTA module clock f_{GPTA} or PLL clock from other unit or DCM 3 clock 11_B Clock bus line 3 selected DCM 2 clock or PLL clock of other unit or uncompensated PLL clock or uncompensated PLL clock of other unit</p>
RTG	24	rw	<p>Reset Timer for FPCK on Glitch 0_B Timer for FPCK is decremented on glitch 1_B Timer for FPCK is cleared on glitch This bit is effective in Delayed Debounce Filter Mode only.</p>
0	[31:25]	r	<p>Reserved Read as 0; should be written with 0.</p>

General Purpose Timer Array (GPTA)

GPTA0_FPCTIMk (k = 0-5)

GPTA0 Filter and Prescaler Cell Timer Register k

(048_H+k*8_H+4_H)

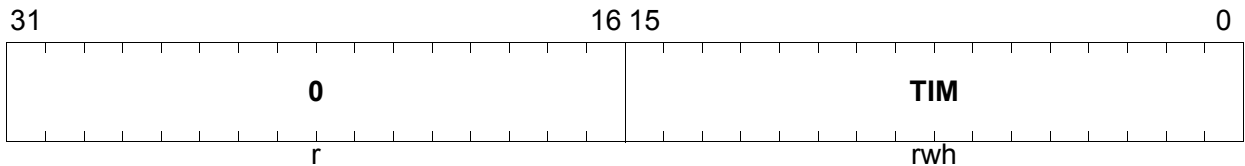
Reset Value: 0000 0000_H

GPTA1_FPCTIMk (k = 0-5)

GPTA1 Filter and Prescaler Cell Timer Register k

(048_H+k*8_H+4_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
TIM	[15:0]	rwh	Timer Value of Filter and Prescaler Cell k
0	[31:16]	r	Reserved Read as 0; should be written with 0.

General Purpose Timer Array (GPTA)

24.3.3 Phase Discriminator Registers

GPTA0_PDLCTR

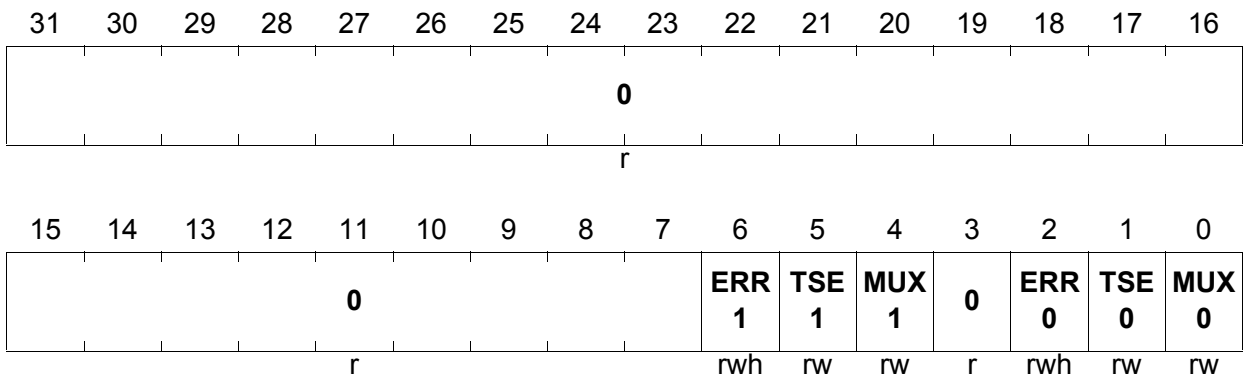
GPTA0 Phase Discrimination Logic Control Register
(078_H)

Reset Value: 0000 0000_H

GPTA1_PDLCTR

GPTA1 Phase Discrimination Logic Control Register
(078_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
MUX0	0	rw	Output Signal Source Selection for PDL0 0 _B DCM0 cell input is driven by fed-through FPC0 output lines 1 _B DCM0 cell input is provided with PDL0 “Forward” and “Backward” pulses
TSE0	1	rw	3-Sensor Mode Enable for PDL0 0 _B PDL0 operates in “2-Sensor Mode” and DCM1 cell input is driven by fed-through FPC2 output lines 1 _B PDL0 operates in “3-Sensor Mode” and DCM1 cell input is provided with PDL0 error information
ERR0	2	rwh	Error Flag for PDL0 0 _B No error has occurred 1 _B Error detected in “3-Sensor Mode”: all PDL0 input signals are simultaneously provided with high or low level Bit ERR0 is bit protected (see Page 24-155).

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
MUX1	4	rw	<p>Output Signal Source Selection for PDL1</p> <p>0_B DCM2 cell input is driven by fed-through FPC3 output lines</p> <p>1_B DCM2 cell input is provided with PDL1 “Forward” and “Backward” pulses</p>
TSE1	5	rw	<p>3-Sensor Mode Enable for PDL1</p> <p>0_B PDL1 operates in “2-Sensor Mode” and DCM3 cell input is driven by fed-through FPC5 output lines</p> <p>1_B PDL1 operates in “3-Sensor Mode” and DCM3 cell input is provided with PDL1 error information</p>
ERR1	6	rwh	<p>Error Flag for PDL1</p> <p>0_B No error has occurred</p> <p>1_B Error detected in “3-Sensor Mode”: all PDL1 input signals are simultaneously provided with high or low level</p> <p>Bit ERR1 is bit protected (see Page 24-155).</p>
0	3, [31:7]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

General Purpose Timer Array (GPTA)

24.3.4 Duty Cycle Measurement Registers

GPTA0_DCMCTR_k (k = 0-3)

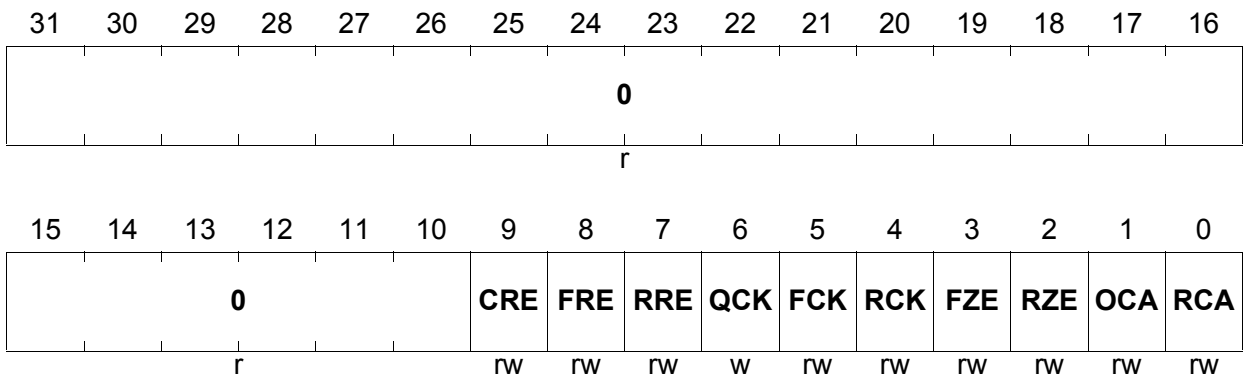
GPTA0 Duty Cycle Measurement Control Register k
(080_H+k*10_H)

Reset Value: 0000 0000_H

GPTA1_DCMCTR_k (k = 0-3)

GPTA1 Duty Cycle Measurement Control Register k
(080_H+k*10_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
RCA	0	rw	Trigger Source Selection for Capture Event 0 _B Timer contents are copied to DCMCAV _k capture register on a falling input signal edge 1 _B Timer contents are copied to capture register on a rising input signal edge
OCA	1	rw	Trigger Source for Capture/Compare Register Update 0 _B Capture/Compare register DCMCOV _k is not affected. 1 _B Timer contents are copied to DCMCOV _k capture/compare register on the opposite edge selected by RCA _k .
RZE	2	rw	Timer Reset on Rising Edge 0 _B Timer is not affected 1 _B Timer is reset on a rising input signal edge
FZE	3	rw	Timer Reset on Falling Edge 0 _B Timer is not affected 1 _B Timer is reset on a falling input signal edge

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
RCK	4	rw	Output Pulse on Rising Edge 0 _B DCM output line is not affected 1 _B DCM output line is provided with a single clock pulse generated on a rising input signal edge
FCK	5	rw	Output Pulse on Falling Edge 0 _B DCM output line is not affected 1 _B DCM output line is provided with a single clock pulse generated on a falling input signal edge
QCK	6	w	Additional Output Pulse Generation 0 _B DCM output line is not affected 1 _B DCM output line is immediately provided with a single clock pulse QCK is always read as 0.
RRE	7	rw	Interrupt Request on Rising Edge 0 _B Interrupt request is not affected 1 _B Interrupt request is set on rising input signal edge
FRE	8	rw	Interrupt Request on Falling Edge 0 _B Interrupt request is not affected 1 _B Interrupt request is set on falling input signal edge
CRE	9	rw	Interrupt Request on Compare Event 0 _B Interrupt request is not affected 1 _B Interrupt request is set when the timer matches capture/compare register DCMCOV _k
0	[31:10]	r	Reserved Read as 0; should be written with 0.

General Purpose Timer Array (GPTA)

GPTA0_DCMTIMk (k = 0-3)

GPTA0 Duty Cycle Measurement Timer Register k

$(080_H+k*10_H+4_H)$

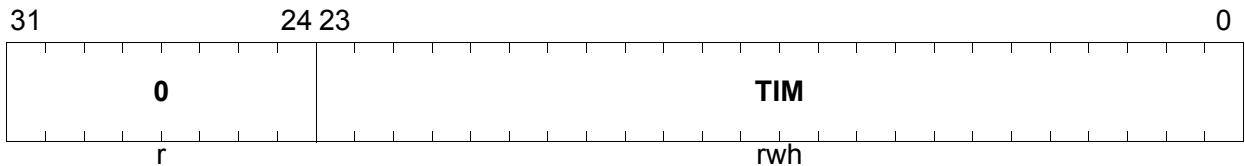
Reset Value: 0000 0000_H

GPTA1_DCMTIMk (k = 0-3)

GPTA1 Duty Cycle Measurement Timer Register k

$(080_H+k*10_H+4_H)$

Reset Value: 0000 0000_H



Field	Bits	Type	Description
TIM	[23:0]	rwh	Timer Value of DCMk
0	[31:24]	r	Reserved Read as 0; should be written with 0.

GPTA0_DCMCAVk (k = 0-3)

GPTA0 Duty Cycle Measurement Capture Register k

(088_H+k*10_H)

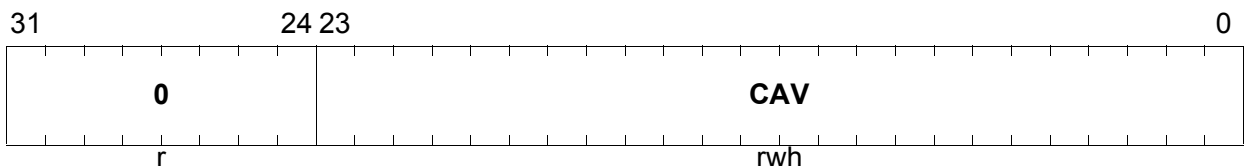
Reset Value: 0000 0000_H

GPTA1_DCMCAVk (k = 0-3)

GPTA1 Duty Cycle Measurement Capture Register k

(088_H+k*10_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CAV	[23:0]	rwh	Capture Value of DCMk
0	[31:24]	r	Reserved Read as 0; should be written with 0.

General Purpose Timer Array (GPTA)

GPTA0_DCMCOV k ($k = 0-3$)

GPTA0 Duty Cycle Measurement Capture/Compare Register k

($08C_H+k*10_H$)

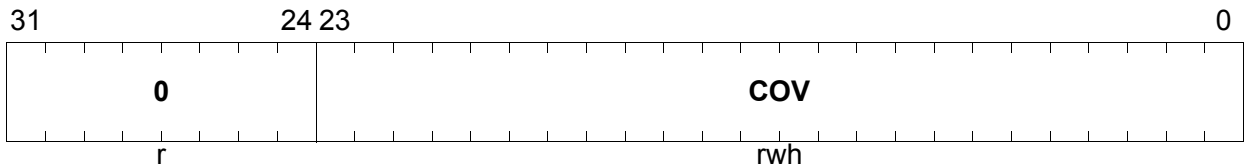
Reset Value: 0000 0000 $_H$

GPTA1_DCMCOV k ($k = 0-3$)

GPTA1 Duty Cycle Measurement Capture/Compare Register k

($08C_H+k*10_H$)

Reset Value: 0000 0000 $_H$



Field	Bits	Type	Description
COV	[23:0]	rwh	Capture/Compare Register Value of DCM k
0	[31:24]	r	Reserved Read as 0; should be written with 0.

General Purpose Timer Array (GPTA)

24.3.5 Digital Phase Locked Loop Registers

GPTA0_PLLCTR

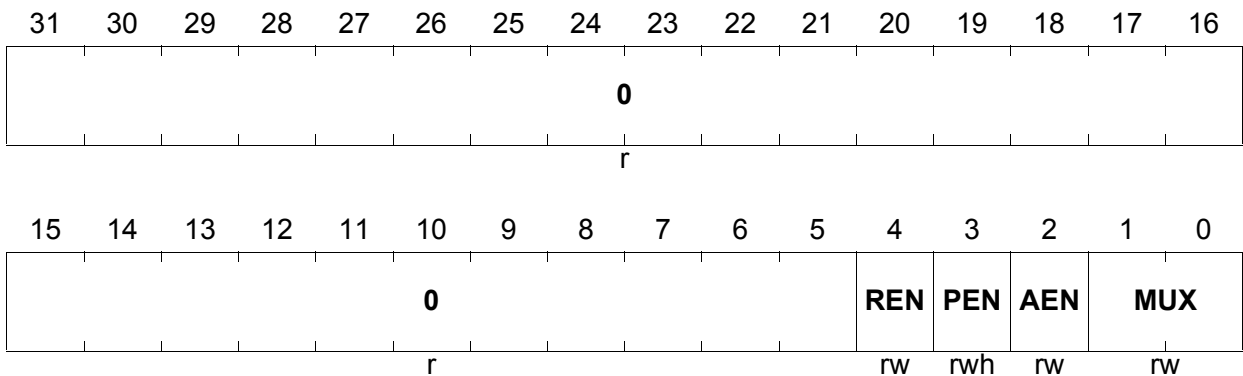
GPTA0 Phase Locked Loop Control Register
(0C0_H)

Reset Value: 0000 0000_H

GPTA1_PLLCTR

GPTA1 Phase Locked Loop Control Register
(0C0_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
MUX	[1:0]	rw	Trigger Input Channel Selection 00 _B DCM0 output is selected as PLL input 01 _B DCM1 output is selected as PLL input 10 _B DCM2 output is selected as PLL input 11 _B DCM3 output is selected as PLL input
AEN	2	rw	Compensation of Input Period Length Variation 0 _B Compensation of input signal's period length variation is disabled 1 _B Compensation of input signal's period length variation (acceleration, deceleration) is requested
PEN	3	rwh	Unexpected Period End Behavior 0 _B Counter decrements with constant frequency 1 _B Counter is allowed to decrement with f_{GPTA} frequency in case of an input signal period length' reduction Programming PEN to 1 immediately changes the microtick counter to decrement with f_{GPTA} frequency. This bit is protected during read-modify-write operations (hardware will win).

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
REN	4	rw	Interrupt Service Request Enable 0 _B Interrupt request is disabled 1 _B An interrupt request is set when the number of remaining output pulses to be generated reaches zero
0	[31:5]	r	Reserved Read as 0; should be written with 0.

GPTA0_PLLMTI

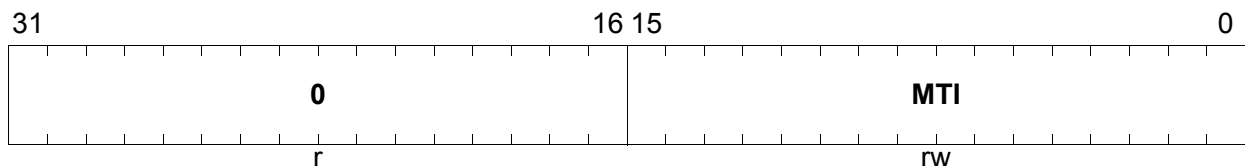
GPTA0 Phase Locked Loop Microtick Register
(0C4_H)

Reset Value: 0000 0000_H

GPTA1_PLLMTI

GPTA1 Phase Locked Loop Microtick Register
(0C4_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
MTI	[15:0]	rw	Microtick Value Number of output pulses to be generated within one input signal period.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

General Purpose Timer Array (GPTA)

GPTA0_PLLSTP

GPTA0 Phase Locked Loop Step Register

(0CC_H)

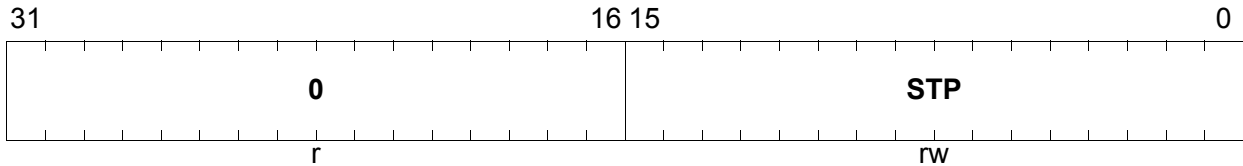
Reset Value: 0000 0000_H

GPTA1_PLLSTP

GPTA1 Phase Locked Loop Step Register

(0CC_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
STP	[15:0]	rw	Step Value Number of output pulses to be generated within one input signal period (2-complement data format).
0	[31:16]	r	Reserved Read as 0; should be written with 0.

GPTA0_PLLCNT

GPTA0 Phase Locked Loop Counter Register

(0C8_H)

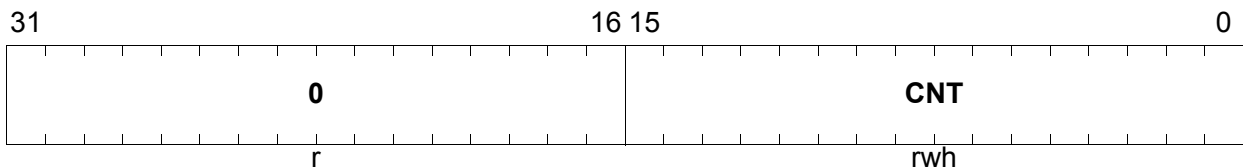
Reset Value: 0000 0000_H

GPTA1_PLLCNT

GPTA1 Phase Locked Loop Counter Register

(0C8_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CNT	[15:0]	rwh	Pulse Counter Counter for the number of remaining output pulses to be generated.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

General Purpose Timer Array (GPTA)

GPTA0_PLLREV

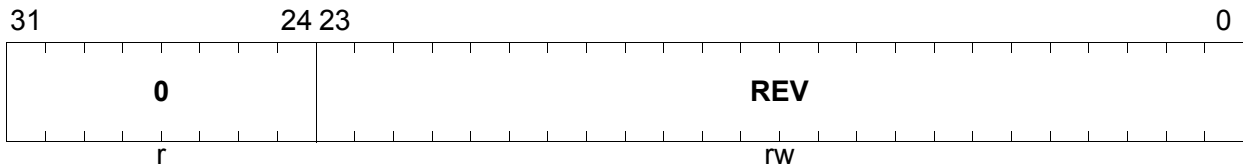
GPTA0 Phase Locked Loop Reload Register
(0D0_H)

Reset Value: 0000 0000_H

GPTA1_PLLREV

GPTA1 Phase Locked Loop Reload Register
(0D0_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
REV	[23:0]	rw	Reload Value Reload value calculated by a subtraction of the number of output pulses to be generated within one input signal period from the input signal's period length (measured in number of GPTA module clocks).
0	[31:24]	r	Reserved Read as 0; should be written with 0.

GPTA0_PLLDTR

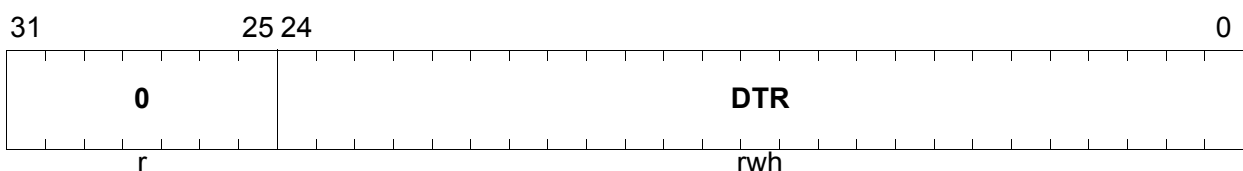
GPTA0 Phase Locked Loop Delta Register
(0D4_H)

Reset Value: 0000 0000_H

GPTA1_PLLDTR

GPTA1 Phase Locked Loop Delta Register
(0D4_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
DTR	[24:0]	rwh	Delta Register Value Internal register used to store intermediate results for output pulse generation. Do not write to this register while PLL is running!

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
0	[31:25]	r	Reserved Read as 0; should be written with 0.

General Purpose Timer Array (GPTA)

24.3.6 Global Timer Registers

GPTA0_GTCTR_k (k = 0-1)

GPTA0 Global Timer Control Register k

(0E0_H+k*10_H)

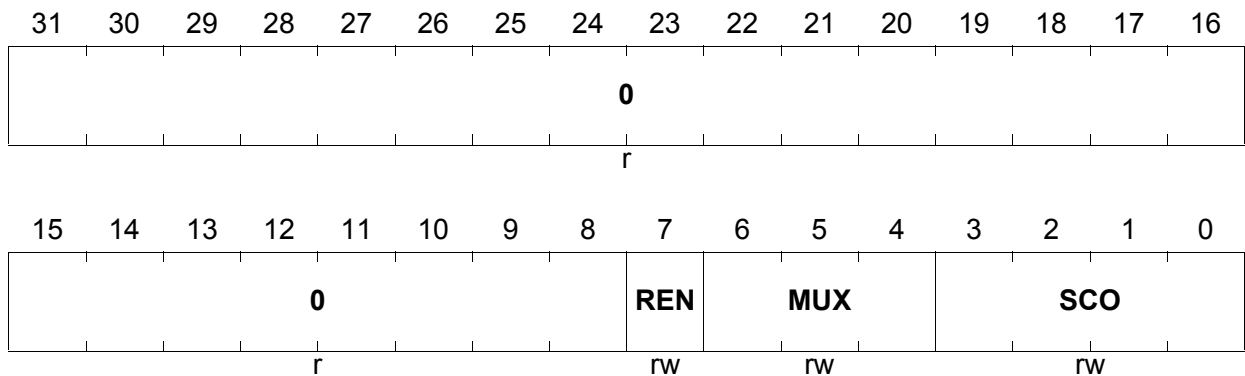
Reset Value: 0000 0000_H

GPTA1_GTCTR_k (k = 0-1)

GPTA1 Global Timer Control Register k

(0E0_H+k*10_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SCO	[3:0]	rw	<p>TGE Flag Source Selection</p> <p>This bit field determines the bit of the operation result “GTk timer value - data bus value” which is used as TGE flag.</p> <p>0000_B 10th bit is used as TGE flag. 0001_B 11th bit is used as TGE flag. ..._B ... 1110_B 24th bit is used as TGE flag. 1111_B 25th bit is used as TGE flag.</p>
MUX	[6:4]	rw	<p>Timer Clock Selection</p> <p>One of eight available clock bus lines is selected as the timer GTk clock.</p> <p>000_B Clock bus line CLK0 selected 001_B Clock bus line CLK1 selected 010_B Clock bus line CLK2 selected 011_B Clock bus line CLK3 selected 100_B Clock bus line CLK4 selected 101_B Clock bus line CLK5 selected 110_B Clock bus line CLK6 selected 111_B Clock bus line CLK7 selected</p>

General Purpose Timer Array (GPTA)

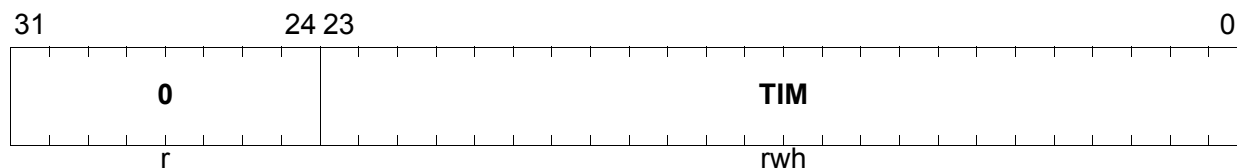
Field	Bits	Type	Description
REN	7	rw	Interrupt Request Enable 0 _B The interrupt request is disabled 1 _B An interrupt request is generated when timer GTk overflows
0	[31:8]	r	Reserved Read as 0; should be written with 0.

GPTA0_GTTIMk (k = 0-1)

GPTA0 Global Timer Register k (0E8_H+k*10_H) **Reset Value: 0000 0000_H**

GPTA1_GTTIMk (k = 0-1)

GPTA1 Global Timer Register k (0E8_H+k*10_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
TIM	[23:0]	rwh	Timer Value of Global Timer k
0	[31:24]	r	Reserved Read as 0; should be written with 0.

General Purpose Timer Array (GPTA)

GPTA0_GTREVK (k = 0-1)

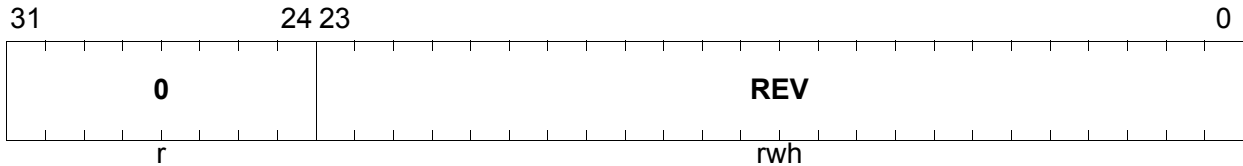
GPTA0 Global Timer Reload Value Register k
(0E4_H+k*10_H)

Reset Value: 0000 0000_H

GPTA1_GTREVK (k = 0-1)

GPTA1 Global Timer Reload Value Register k
(0E4_H+k*10_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
REV	[23:0]	rw	Reload Value of Global Timer k Reload value for timer GTK after an overflow
0	[31:24]	r	Reserved Read as 0; should be written with 0.

General Purpose Timer Array (GPTA)

24.3.7 Clock Bus Register

GPTA0_CKBCTR

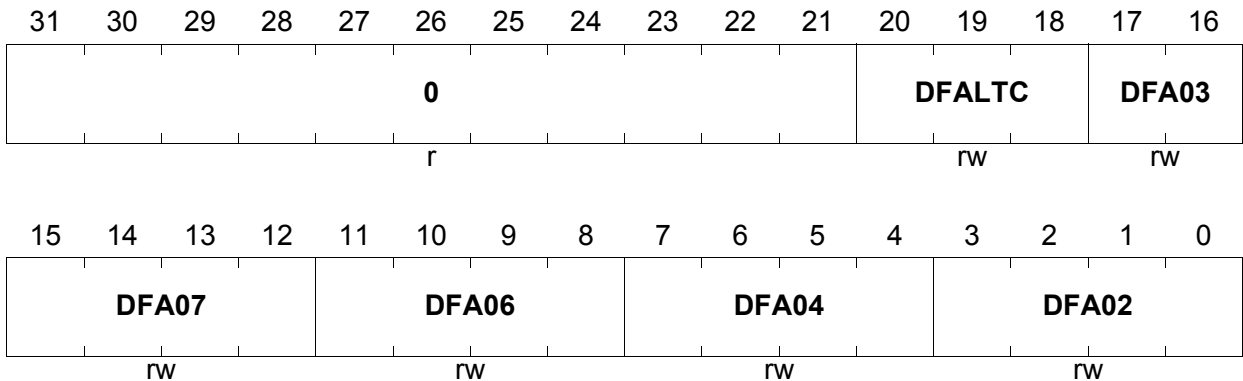
GPTA0 Clock Bus Control Register (0D8_H)

Reset Value: 0000 FFFF_H

GPTA1_CKBCTR

GPTA1 Clock Bus Control Register (0D8_H)

Reset Value: 0000 FFFF_H



Field	Bits	Type	Description
DFA02	[3:0]	rw	<p>Clock Line 2 Driving Source Selection</p> <p>0_D CLK2 is provided with f_{GPTA}</p> <p>1_D CLK2 is provided with f_{GPTA} divided by 2¹</p> <p>..._D ...</p> <p>12_D CLK2 is provided with f_{GPTA} divided by 2¹²</p> <p>13_D CLK2 is provided with f_{GPTA} divided by 2¹³</p> <p>14_D CLK2 is driven by PLL clock of other GPTA unit</p> <p>15_D CLK2 is driven by DCM3 output</p>
DFA04	[7:4]	rw	<p>Clock Line 4 Driving Source Selection</p> <p>0_D CLK4 is provided with f_{GPTA}</p> <p>1_D CLK4 is provided with f_{GPTA} divided by 2¹</p> <p>..._D ...</p> <p>13_D CLK4 is provided with f_{GPTA} divided by 2¹³</p> <p>14_D CLK4 is provided with f_{GPTA} divided by 2¹⁴</p> <p>15_D CLK4 is driven by DCM1 output</p>
DFA06	[11:8]	rw	<p>Clock Line 6 Driving Source Selection</p> <p>0_D CLK6 is provided with f_{GPTA}</p> <p>1_D CLK6 is provided with f_{GPTA} divided by 2¹</p> <p>..._D ...</p> <p>13_D CLK6 is provided with f_{GPTA} divided by 2¹³</p> <p>14_D CLK6 is provided with f_{GPTA} divided by 2¹⁴</p> <p>15_D CLK6 is driven by FPC1 output</p>

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
DFA07	[15:12]	rw	Clock Line 7 Driving Source Selection 0 _D CLK7 is provided with f_{GPTA} 1 _D CLK7 is provided with f_{GPTA} divided by 2^1 ... _D ... 13 _D CLK7 is provided with f_{GPTA} divided by 2^{13} 14 _D CLK7 is provided with f_{GPTA} divided by 2^{14} 15 _D CLK7 is driven by FPC4 output
DFA03	[17:16]	rw	Clock Line 3 Driving Source Selection 0 _D CLK3 is driven by DCM2 output 1 _D CLK3 is driven by PLL clock of other GPTA unit 2 _D CLK3 is driven by uncompensated PLL clock 3 _D CLK3 is driven by uncompensated PLL clock of other GPTA unit
DFALTC	[20:18]	rw	Dividing Factor for LTC Prescaler Clock Selection The LTCPRE clock is provided with the GPTA module clock f_{GPTA} divided by 2^{DFALTC} . 0 _D LTCPRE clock is f_{GPTA} 1 _D LTCPRE clock is f_{GPTA} divided by 2^1 ... _D ... 6 _D LTCPRE clock is f_{GPTA} divided by 2^6 7 _D LTCPRE clock is f_{GPTA} divided by 2^7
0	[31:21]	r	Reserved Read as 0; should be written with 0.

General Purpose Timer Array (GPTA)

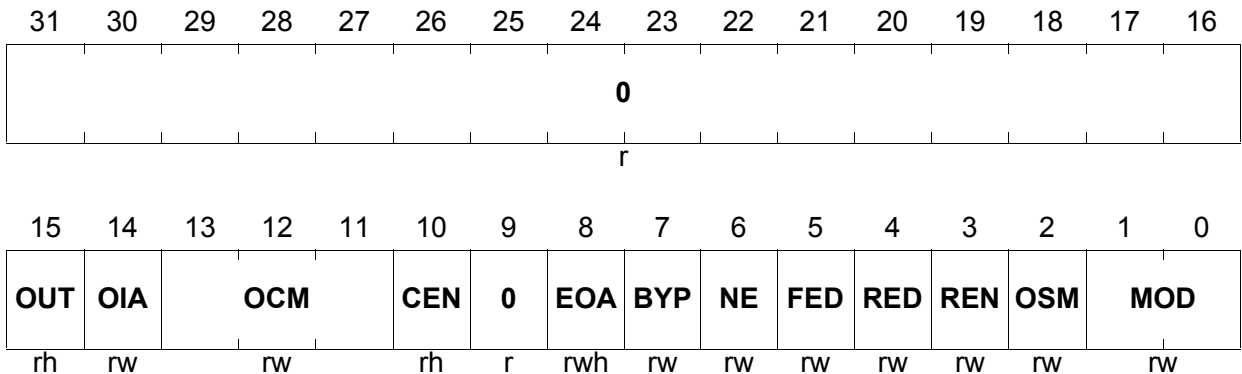
24.3.8 Global Timer Cell Registers

GPTA0_GTCCTR_k (k = 00-31)

GPTA0 Global Timer Cell Control Register k [Capture Mode]
(100_H+k*8_H) **Reset Value: 0000 0000_H**

GPTA1_GTCCTR_k (k = 00-31)

GPTA1 Global Timer Cell Control Register k [Capture Mode]
(100_H+k*8_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
MOD	[1:0]	rw	Mode Control Bits 00 _B GTCK operates in Capture Mode hooked to GT0. 01 _B GTCK operates in Capture Mode hooked to GT1. 10 _B GTCK operates in Compare Mode hooked to GT0. 11 _B GTCK operates in Compare Mode hooked to GT1.
OSM	2	rw	One Shot Mode Enable 0 _B GTCK is continuously enabled. 1 _B GTCK is enabled for one event only.
REN	3	rw	Interrupt Request Enable 0 _B Service request is disabled. 1 _B Service request line SQSk is activated when a capture or compare event has occurred.
RED	4	rw	Input Rising Edge Select 0 _B Capture event is not triggered by a rising edge. 1 _B Capture event is triggered by a rising edge on the GTCKIN input line.
FED	5	rw	Input Falling Edge Select 0 _B Capture event is not triggered by a falling edge. 1 _B Capture event is triggered by a falling edge on the GTCKIN input line.

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
NE	6	rw	Not Effective Reserved
BYP	7	rw	Bypass 0_B M0O/M1O lines are affected either by M0I/M1I lines or by OCM0/OCM1 bits. 1_B M0O/M1O lines are affected only by M0I/M1I lines. <i>Note: OCM2 must be set in any case to enable reaction on M0I/M1I changes.</i>
EOA	8	rwh	Enable On Action 0_B GTCK is enabled for local events. 1_B GTCK is disabled for local events. On an event on the communication link via M0I/M1I lines, EOA will be cleared and local events will be enabled. This bit is protected during read-modify-write operations (hardware will win).
CEN	10	rh	Cell Enable 0_B GTCK is currently disabled for local events. 1_B GTCK is currently enabled for local events.
OCM	[13:11]	rw	Output Control Mode Select $X00_B$ Current state of GTCKOUT output line is hold. $X01_B$ Current state of GTCKOUT output line is toggled. $X10_B$ GTCKOUT output line is forced to 0. $X11_B$ GTCKOUT output line is forced to 1. $0XX_B$ GTCKOUT output line state is set by an internal GTCK event only. $1XX_B$ GTCKOUT output line state is affected by an internal GTCK event and/or by an operation occurred in an adjacent GTCn (n = less or equal k) and reported by the M1I, M0I interface lines.
OIA	14	rw	Output Immediate Action 0_B No immediate action required. 1_B Action defined by OCM must be performed immediately. Reading bit OIA always returns 0.
OUT	15	rh	Output State 0_B GTCKOUT output line is 0. 1_B GTCKOUT output line is 1.

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
0	9, [31:16]	r	Reserved Read as 0; should be written with 0.

GPTA0_GTCCTRk (k = 00-31)

GPTA0 Global Timer Cell Control Register k [Compare Mode]

(100_H+k*8_H)

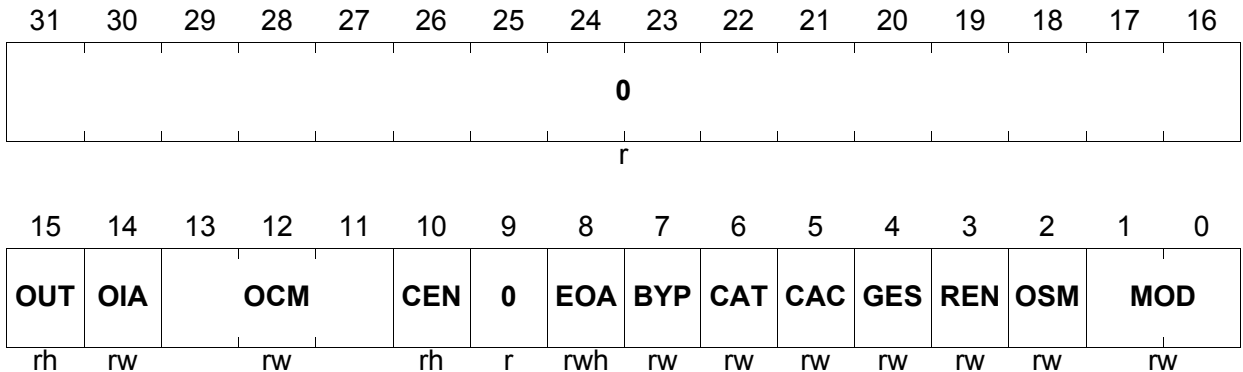
Reset Value: 0000 0000_H

GPTA1_GTCCTRk (k = 00-31)

GPTA1 Global Timer Cell Control Register k [Compare Mode]

(100_H+k*8_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
MOD	[1:0]	rw	Mode Control Bits 00 _B GTCK operates in Capture Mode hooked to GT0. 01 _B GTCK operates in Capture Mode hooked to GT1. 10 _B GTCK operates in Compare Mode hooked to GT0. 11 _B GTCK operates in Compare Mode hooked to GT1.
OSM	2	rw	One Shot Mode Enable 0 _B GTCK is continuously enabled. 1 _B GTCK is enabled for one event only.
REN	3	rw	Interrupt Request Enable 0 _B Service request is disabled. 1 _B Service request line SQSk is activated when a capture or compare event has occurred.
GES	4	rw	Greater Equal Select 0 _B An “equal” compare is selected. 1 _B A “greater equal” compare is required.

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
CAC	5	rw	<p>Capture after Compare Select</p> <p>0_B Capture after compare is disabled.</p> <p>1_B After a compare event, the contents of the associated Global Timer as selected by MOD or (depending on control bit CAT) the contents of the alternate Global Timer are copied to the capture/compare register GTCXRk.</p>
CAT	6	rw	<p>Capture Alternate Timer</p> <p>0_B The Global Timer as selected by MOD is captured, if enabled by control bit CAC = 1.</p> <p>1_B The alternate Global Timer is captured.</p>
BYP	7	rw	<p>Bypass</p> <p>0_B M0O/M1O lines are affected either by M0I/M1I lines or by OCM0/OCM1 bits.</p> <p>1_B M0O/M1O lines are affected only by M0I/M1I lines.</p> <p><i>Note: OCM2 must be set in any case to enable reaction on M0I/M1I changes.</i></p>
EOA	8	rwh	<p>Enable On Action</p> <p>0_B GTCK is enabled for local events.</p> <p>1_B GTCK is disabled for local events. On an event on the communication link via M0I/M1I lines, EOA will be cleared and local events will be enabled.</p> <p>This bit is protected during read-modify-write operations (hardware will win).</p>
CEN	10	rh	<p>Cell Enable</p> <p>0_B GTCK is currently disabled for local events.</p> <p>1_B GTCK is currently enabled for local events.</p>
OCM	[13:11]	rw	<p>Output Control Mode Select</p> <p>X00_B Current state of GTCKOUT output line is hold.</p> <p>X01_B Current state of GTCKOUT output line is toggled.</p> <p>X10_B GTCKOUT output line is forced to 0.</p> <p>X11_B GTCKOUT output line is forced to 1.</p> <p>0XX_B GTCKOUT output line state is set by an internal GTCK event only.</p> <p>1XX_B GTCKOUT output line state is affected by an internal GTCK event and/or by an operation occurred in an adjacent GTCn (n = less or equal k) and reported by the M1I, M0I interface lines.</p>

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
OIA	14	rw	Output Immediate Action 0_B No immediate action required. 1_B Action defined by OCM must be performed immediately. Reading bit OIA always returns 0.
OUT	15	rh	Output State 0_B GTCKOUT output line is 0. 1_B GTCKOUT output line is 1.
0	9, [31:16]	r	Reserved Read as 0; should be written with 0.

GPTA0_GTCXRk (k = 00-31)

GPTA0 Global Timer Cell X Register k

$(104_H + k * 8_H)$

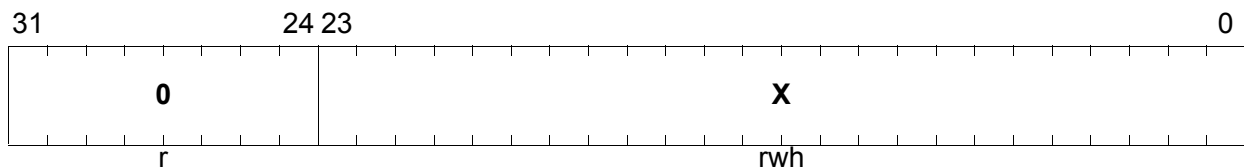
Reset Value: 0000 0000_H

GPTA1_GTCXRk (k = 00-31)

GPTA1 Global Timer Cell X Register k

$(104_H + k * 8_H)$

Reset Value: 0000 0000_H



Field	Bits	Type	Description
X	[23:0]	rwh	Capture/Compare Register Contents of GTCK
0	[31:24]	r	Reserved Read as 0; should be written with 0.

Note: GTCXRk is write-protected when control bits CAC and OSM are set to 1 (“capture after compare” in Single Shot Mode). Write protection is activated, when the value of the selected GT timer matches and/or exceeds the capture/compare register contents. Write protection is released after a software access to register GTCXRk.

General Purpose Timer Array (GPTA)

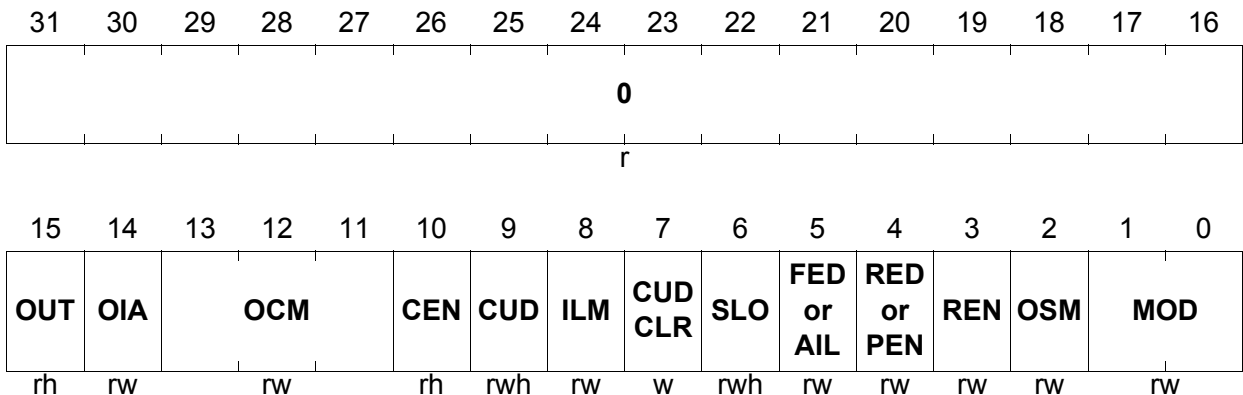
24.3.9 Local Timer Cell Registers

LTCCTR_k (k = 00-62)

Local Timer Cell Control Register k [Timer Mode]

(200_H+k*8_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
MOD	[1:0]	rw	Mode Control Bits 00 _B LTCK operates in Capture Mode. 01 _B LTCK operates in Compare Mode. 10 _B LTCK operates in Free-Running Timer Mode. 11 _B LTCK operates in Reset Timer Mode.
OSM	2	rw	One Shot Mode Enable 0 _B LTCK is continuously enabled. 1 _B LTCK is enabled for one event only.
REN	3	rw	Request Enable 0 _B Service request is disabled. 1 _B Service request SQSk is activated when a <ul style="list-style-type: none"> - capture event has occurred - compare event has occurred - timer overflow has happened depending on the operation mode selected by bit field MOD.
RED	4	rw	ILM = 0: Input Rising Edge Select 0 _B Timer is not updated by a rising edge. 1 _B Timer is updated by a rising edge on the LTCKIN input line.

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
PEN	4	rw	<p>ILM = 1: LTC Prescaler Enable</p> <p>0_B LTC Prescaler Mode is disabled.</p> <p>1_B LTC Prescaler Mode with LTC prescaler clock LTCPRE is enabled.</p>
FED	5	rw	<p>ILM = 0: Input Falling Edge Select</p> <p>0_B Timer is not updated by a falling edge.</p> <p>1_B Timer is updated by a falling edge on the LTCKIN input line.</p>
AIL	5	rw	<p>ILM = 1: Active Input Level Select</p> <p>0_B Input signal is active high.</p> <p>1_B Input signal is active low.</p>
SLO	6	rwh	<p>Select Line Output</p> <p>0_B State of select line output SO is 0.</p> <p>1_B State of select line output SO is 1.</p> <p>SLO is bit protected (see Page 24-155).</p>
CUDCLR	7	w	<p>Coherent Update Disable</p> <p>0_B No effect.</p> <p>1_B Coherent update disabled (bit CUD is cleared). If bits CUD and CUDCLR are both written with 1, bit CUD will be set. CUDCLR is always read as 0.</p>
ILM	8	rw	<p>Input Line Mode</p> <p>0_B Input line is operating in Edge Sensitive Mode.</p> <p>1_B Input line is operating in Level Sensitive Mode.</p> <p>In case of full speed GPTA module clock selection as input clock, Level Sensitive Mode must be selected. In this case the Edge Sensitive Mode will not produce any event.</p>
CUD	9	rwh	<p>Coherent Update Enable</p> <p>0_B Select output SO is not toggled on timer reset overflow.</p> <p>1_B Select output SO is toggled on next timer reset overflow.</p> <p>When CUD is set by software (writing CUD and CUDCLR both with 1), it remains set until the next timer reset overflow (LTCK reset event) occurs and is cleared by hardware afterwards. CUD can be cleared by software by writing bit CUDCLR with 1 and CUD with 0.</p>

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
CEN	10	rh	Cell Enable 0 _B LTck is currently disabled for local events. 1 _B LTck is currently enabled for local events.
OCM	[13:11]	rw	Output Control Mode Select X00 _B Current state of LTckOUT output line is hold. X01 _B Current state of LTckOUT output line is toggled. X10 _B LTckOUT output line is forced to 0. X11 _B LTckOUT output line is forced to 1. 0XX _B LTckOUT output line state is set by an internal LTck event only. 1XX _B LTckOUT output line state is affected by an internal LTck event and/or by an operation occurred in an adjacent LTck cell (reported by M11/M0I interface lines).
OIA	14	rw	Output Immediate Action 0 _B No immediate action required. 1 _B Action defined by bit field OCM must be performed immediately. OIA is always read as 0.
OUT	15	rh	Output State 0 _B LTckOUT output line is 0. 1 _B LTckOUT output line is 1.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

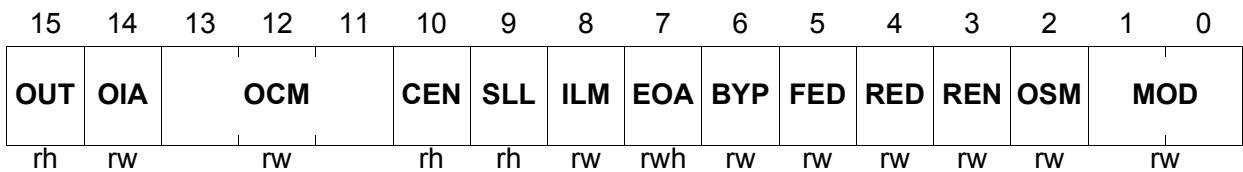
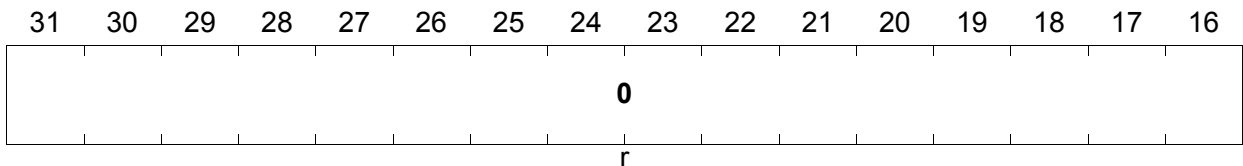
General Purpose Timer Array (GPTA)

LTCCTR_k (k = 00-62)

Local Timer Cell Control Register k [Capture Mode]

(200_H+k*8_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
MOD	[1:0]	rw	Mode Control Bits 00 _B LTCK operates in Capture Mode. 01 _B LTCK operates in Compare Mode. 10 _B LTCK operates in Free-Running Timer Mode. 11 _B LTCK operates in Reset Timer Mode.
OSM	2	rw	One Shot Mode Enable 0 _B LTCK is continuously enabled. 1 _B LTCK is enabled for one event only.
REN	3	rw	Request Enable 0 _B Service request is disabled. 1 _B Service request SQSk is activated when a - capture event has occurred - compare event has occurred - timer overflow has happened depending on the operation mode selected by bit field MOD.
RED	4	rw	Input Rising Edge Select 0 _B Capture event is not triggered by a rising edge. 1 _B Capture event is triggered by a rising edge on the LTCKIN input line.
FED	5	rw	Input Falling Edge Select 0 _B Capture event is not triggered by a falling edge. 1 _B Capture event is triggered by a falling edge on the LTCKIN input line.

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
BYP	6	rw	<p>Bypass</p> <p>0_B M00/M10 lines are affected either by M0I/M1I lines or by OCM0/OCM1 bits.</p> <p>1_B M00/M10 lines are affected only by M0I/M1I lines.</p> <p>This bit is cleared if mode is switched to Timer Mode. OCM2 must be set in any case to enable reaction on M0I/M1I change.</p>
EOA	7	rwh	<p>Enable On Action</p> <p>0_B LTck is enabled for local events.</p> <p>1_B LTck is disabled for local events. On an event on the communication link via M0I/M1I lines, EOA will be cleared and local events will be enabled. EOA is bit protected (see Section 24.3.2). EOA is cleared if mode is switched to Timer Mode.</p>
ILM	8	rw	<p>Input Line Mode</p> <p>0_B Input line is operating in Edge Sensitive Mode.</p> <p>1_B Input line is operating in Level Sensitive Mode. In case of full speed GPTA module clock selection as input clock, Level Sensitive Mode must be selected. In this case the Edge Sensitive Mode will not produce any event.</p>
SLL	9	rh	<p>Capture & Compare Mode: Select Line Level</p> <p>0_B Current state of select input SI is 0.</p> <p>1_B Current state of select input SI is 1.</p>
CEN	10	rh	<p>Cell Enable</p> <p>0_B LTck is currently disabled for local events.</p> <p>1_B LTck is currently enabled for local events.</p>
OCM	[13:11]	rw	<p>Output Control Mode Select</p> <p>X00_B Current state of LTckOUT output line is hold.</p> <p>X01_B Current state of LTckOUT output line is toggled.</p> <p>X10_B LTckOUT output line is forced to 0.</p> <p>X11_B LTckOUT output line is forced to 1.</p> <p>0XX_B LTckOUT output line state is set by an internal LTck event only.</p> <p>1XX_B LTckOUT output line state is affected by an internal LTck event and/or by an operation occurred in an adjacent LTck cell (reported by M1I/M0I interface lines).</p>

General Purpose Timer Array (GPTA)

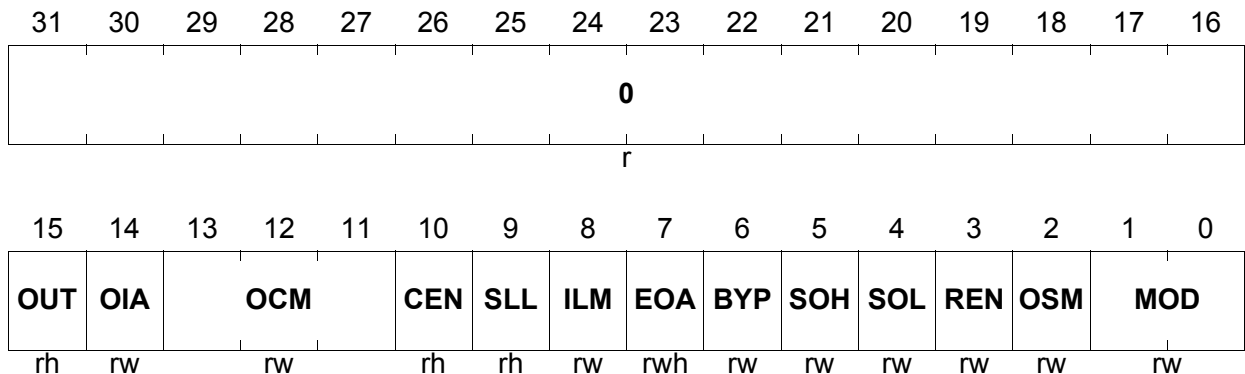
Field	Bits	Type	Description
OIA	14	rw	Output Immediate Action 0_B No immediate action required. 1_B Action defined by bit field OCM must be performed immediately. OIA is always read as 0.
OUT	15	rh	Output State 0_B LTCKOUT output line is 0. 1_B LTCKOUT output line is 1.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

LTCCTRk (k = 00-62)

Local Timer Cell Control Register k [Compare Mode]

(200_H+k*8_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
MOD	[1:0]	rw	Mode Control Bits 00_B LTCK operates in Capture Mode. 01_B LTCK operates in Compare Mode. 10_B LTCK operates in Free-Running Timer Mode. 11_B LTCK operates in Reset Timer Mode.
OSM	2	rw	One Shot Mode Enable 0_B LTCK is continuously enabled. 1_B LTCK is enabled for one event only.

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
REN	3	rw	<p>Request Enable</p> <p>0_B Service request is disabled.</p> <p>1_B Service request SQSk is activated when a</p> <ul style="list-style-type: none"> - capture event has occurred - compare event has occurred - timer overflow has happened <p>depending on the operation mode selected by bit field MOD.</p>
SOL	4	rw	<p>Compare Mode: Select Output Low</p> <p>0_B Compare is deactivated or on high level.</p> <p>1_B Compare operation is enabled by a low level on select input SI¹.</p>
SOH	5	rw	<p>Compare Mode: Select Output High</p> <p>0_B Compare is deactivated or on high level.</p> <p>1_B Compare operation is enabled by a high level on select input SI¹.</p>
BYP	6	rw	<p>Bypass</p> <p>0_B M00/M10 lines are affected either by M0I/M1I lines or by OCM0/OCM1 bits.</p> <p>1_B M00/M10 lines are affected only by M0I/M1I lines.</p> <p>This bit is cleared if mode is switched to Timer Mode. OCM2 must be set in any case to enable reaction on M0I/M1I change.</p>
EOA	7	rwh	<p>Enable On Action</p> <p>0_B LTCK is enabled for local events.</p> <p>1_B LTCK is disabled for local events. On an event on the communication link via M0I/M1I lines, EOA will be cleared and local events will be enabled.</p> <p>EOA is bit protected (see Section 24.3.2). EOA is cleared if mode is switched to Timer Mode.</p>
ILM	8	rw	<p>Input Line Mode</p> <p>0_B Input line is operating in Edge Sensitive Mode.</p> <p>1_B Input line is operating in Level Sensitive Mode.</p> <p>In case of full speed GPTA module clock selection as input clock, Level Sensitive Mode must be selected. In this case the Edge Sensitive Mode will not produce any event.</p>

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
SLL	9	rh	Select Line Level 0 _B Current state of select input SI is 0. 1 _B Current state of select input SI is 1.
CEN	10	rh	Cell Enable 0 _B LTCK is currently disabled for local events. 1 _B LTCK is currently enabled for local events.
OCM	[13:11]	rw	Output Control Mode Select X00 _B Current state of LTCKOUT output line is hold. X01 _B Current state of LTCKOUT output line is toggled. X10 _B LTCKOUT output line is forced to 0. X11 _B LTCKOUT output line is forced to 1. 0XX _B LTCKOUT output line state is set by an internal LTCK event only. 1XX _B LTCKOUT output line state is affected by an internal LTCK event and/or by an operation occurred in an adjacent LTCK cell (reported by M1I/M0I interface lines).
OIA	14	rw	Output Immediate Action 0 _B No immediate action required. 1 _B Action defined by bit field OCM must be performed immediately. OIA is always read as 0.
OUT	15	rh	Output State 0 _B LTCKOUT output line is 0. 1 _B LTCKOUT output line is 1.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

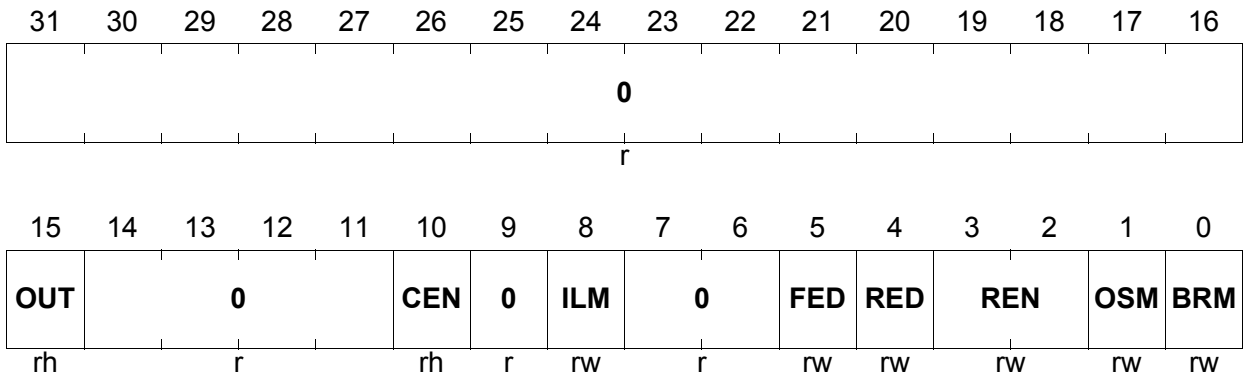
1) To enable Compare Mode in all cases, SOL and SOH bits must be set to 1.

General Purpose Timer Array (GPTA)

LTCCTR63

Local Timer Cell Control Register 63 (3F8_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
BRM	0	rw	Bit Reversal Mode Control 0 _B Compare uses normal sequence of local input data bus (YI) bits. 1 _B Compare uses reversed sequence of local input data bus (YI) bits.
OSM	1	rw	One Shot Mode Enable for Shadow Register Copy 0 _B Shadow register copy is continuously enabled. 1 _B Shadow register copy is enabled for one event only.
REN	[3:2]	rw	Request Enable 00 _B Service request SQT63 is disabled. 01 _B Service request SQT63 is generated when a compare event has occurred. 10 _B Service request SQT63 is generated when a shadow register copy event has occurred. 11 _B Reserved.
RED	4	rw	Rising Edge Select for Shadow Register Copy 0 _B Shadow register copy is not triggered by a rising edge on the LTC63IN input line. 1 _B Shadow register copy is triggered by a rising edge on the LTC63IN input line.
FED	5	rw	Falling Edge Select for Shadow Register Copy 0 _B Shadow register copy is not triggered by a falling edge on the LTC63IN input line. 1 _B Shadow register copy is triggered by a falling edge on the LTC63IN input line.

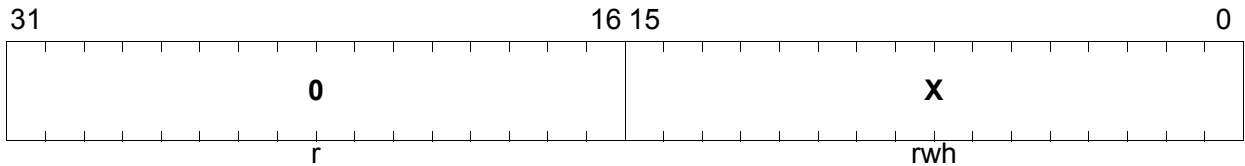
General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
ILM	8	rw	Shadow Register Copy Input Line Mode 0 _B LTC63IN is operating in Edge Sensitive Mode. 1 _B LTC63IN is operating in Level Sensitive Mode.
CEN	10	rh	Enable for Shadow Register Copy 0 _B Shadow register copy is currently disabled. 1 _B Shadow register copy is currently enabled.
OUT	15	rh	Output State 0 _B LTC63OUT output line is 0. 1 _B LTC63OUT output line is 1.
0	[7:6], 9, [14:11], [31:16]	r	Reserved Read as 0; should be written with 0.

General Purpose Timer Array (GPTA)

LTCXRk (k = 00-62)

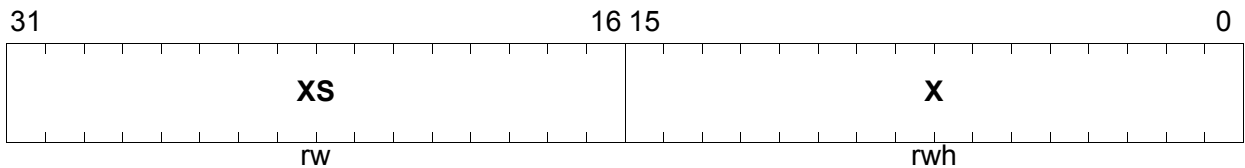
Local Timer Cell X Register k $(204_H + k \cdot 8_H)$ **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
X	[15:0]	rwh	Local Timer Data Register Value
0	[31:16]	r	Reserved Read as 0; should be written with 0.

LTCXR63

Local Timer Cell X Register 63 $(3FC_H)$ **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
X	[15:0]	rwh	Compare Register Value
XS	[31:16]	rw	Shadow Register Value

General Purpose Timer Array (GPTA)

24.3.10 I/O Sharing Unit Registers

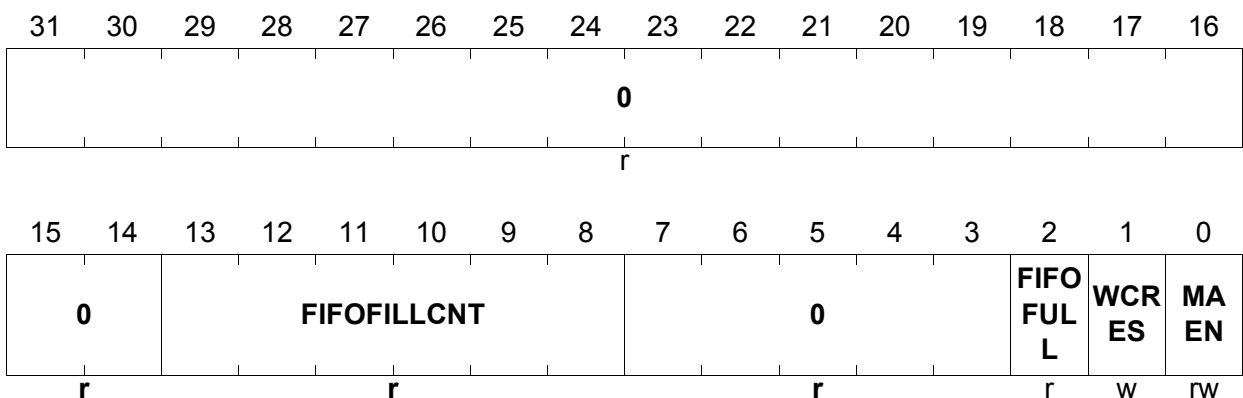
The three registers MRACTL, MRADIN, and MRADOUT are used to write data to and read data from the GPTA Multiplexer Register Array FIFO. The Multiplexer Register Array FIFO controls the operation of the Input/Output Line Sharing Unit (see [Section 24.2.4.5](#)).

The Multiplexer Register Array Control register controls the operation of the Multiplexer Register Array FIFO.

MRACTL

Multiplexer Register Array Control Register (038_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
MAEN	0	rw	<p>Multiplexer Array Enable</p> <p>Bit field MAEN enables/disables the programming and the interconnections of the multiplexer array.</p> <p>0_B Multiplexer array is disabled; all cell inputs are driven with 0, GPTA I/O lines (pins) are disconnected and FIFO writing is enabled.</p> <p>1_B Multiplexer array is enabled; all cell and I/O line interconnections are established as previously programmed and FIFO writing is disabled.</p>
WCRES	1	w	<p>Write Count Reset</p> <p>Writing WCRES with 1 while the array is disabled (MAEN = 0), resets the write cycle counter to zero and the FIFO written sequentially (initialized). WCRES is always read as 0.</p>

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
FIFOFULL	2	r	FIFO Full Status 0 _B FIFO not completely written (write access to MRADIN allowed). 1 _B FIFO completely written (write access to MRADIN ignored). Must be re-enabled by WCRES = 0 before array can be re-initialized.
FIFOFILLCNT	[13:8]	r	FIFO Fill Count This bit field shows the current contents of the write cycle counter.
0	[7:3], [31:14]	r	Reserved Read as 0; should be written with 0.

General Purpose Timer Array (GPTA)

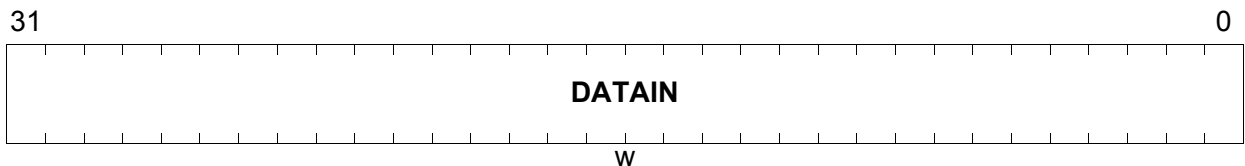
The Multiplexer Register Array Data In MRADIN register is used to **write** data to the Multiplexer Register Array FIFO. The Multiplexer Register Array Data Out MRADOUT register is used to **read** data from the Multiplexer Register Array FIFO.

Note: For correct operation, the MRADIN and MRADOUT registers must be always read or written 32-bit wide. 8-bit and 16-bit accesses are ignored but without any bus error!

MRADIN

Multiplexer Register Array Data In Register
(03C_H)

Reset Value: 0000 0000_H

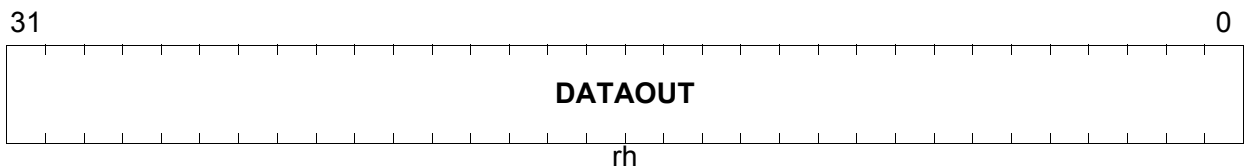


Field	Bits	Type	Description
DATAIN	[31:0]	w	FIFO Write Data This register contains the FIFO write data as defined for the Output Multiplexer Control Registers, the GTC Input Multiplexer Control Registers, or the LTC Input Multiplexer Control Registers.

MRADOUT

Multiplexer Register Array Data Out Register
(040_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
DATAOUT	[31:0]	rh	FIFO Read Data This register contains the FIFO read data as assigned for the Output Multiplexer Control Registers, the GTC Input Multiplexer Control Registers, or the LTC Input Multiplexer Control Registers.

General Purpose Timer Array (GPTA)

24.3.11 Multiplexer Control Registers

These registers are not directly accessible and can be written and read only via the multiplexer register array FIFO (see [Page 24-110](#)).

Output Multiplexer Control Registers

Two registers, OMCRL and OMCRH, are assigned to each I/O Group IOG[6:0] and each Output Group OG[6:0]. OMCRL[6:0]/OMCRH[6:0] are assigned to IOG[6:0] and OMCRL[13:7]/OMCRH[13:7] are assigned to OG[6:0].

OMCRL controls the connections of group pins 0 to 3. OMCRH controls the connections of group pins 4 to 7.

OMCRL_g (g = 0-13)

Output Multiplexer Control Register for Lower Half of Group g

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	OMG3		0	OML3		0	OMG2		0	OML2					
r	rw		r	rw		r	rw		r	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	OMG1		0	OML1		0	OMG0		0	OML0					
r	rw		r	rw		r	rw		r	rw					

Field	Bits	Type	Description
OML0, OML1, OML2, OML3	[2:0], [10:8], [18:16], [26:24]	rw	Multiplexer Line Selection This bit field selects the input line of a OMG that can be selected by bit field OMG _n for OMG output n. 000 _B OMG input IN0 selected 001 _B OMG input IN1 selected 010 _B OMG input IN2 selected 011 _B OMG input IN3 selected 100 _B OMG input IN4 selected 101 _B OMG input IN5 selected 110 _B OMG input IN6 selected 111 _B OMG input IN7 selected

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
OMG0, OMG1, OMG2, OMG3	[6:4], [14:12], [22:20], [30:28]	rw	<p>Multiplexer Group Selection</p> <p>This bit field determines the OMGng which is connected to input n of I/O Group g or Output group g-7.</p> <p>X00_B OMG0g selected X01_B OMG1g selected X10_B OMG2g selected</p> <p>All other combinations are reserved. If a reserved combination of OMGn value is selected, the corresponding OMG output is forced to 0 level. For compatibility reasons, OMGn[2] = 0 should be used (as value for X) for OMGn bit field programming.</p>
0	3, 7, 11, 15, 19, 23, 27, 31	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

General Purpose Timer Array (GPTA)

OMCRHg (g = 0-13)

Output Multiplexer Control Register for Upper Half of Output Group g

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	OMG7			0	OML7			0	OMG6			0	OML6		
r	rw			r	rw			r	rw			r	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	OMG5			0	OML5			0	OMG4			0	OML4		
r	rw			r	rw			r	rw			r	rw		

Field	Bits	Type	Description
OML4, OML5, OML6, OML7	[2:0], [10:8], [18:16], [26:24]	rw	Multiplexer Line Selection This bit field selects the input line of a OMG that can be selected by bit field OMGn for OMG output n. 000 _B OMG input IN0 selected 001 _B OMG input IN1 selected 010 _B OMG input IN2 selected 011 _B OMG input IN3 selected 100 _B OMG input IN4 selected 101 _B OMG input IN5 selected 110 _B OMG input IN6 selected 111 _B OMG input IN7 selected
OMG4, OMG5, OMG6, OMG7	[6:4], [14:12], [22:20], [30:28]	rw	Multiplexer Group Selection This bit field determines the OMGng which is connected to input n of I/O Group g or Output group g-7. X00 _B OMG0g selected X0X _B OMG1g selected X10 _B OMG2g selected All other combinations are reserved. If a reserved combination of OMGn value is selected, the corresponding OMG output is forced to 0 level. For compatibility reasons, OMGn[2] = 0 should be used (as value for X) for OMGn bit field programming.

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
0	3, 7, 11, 15, 19, 23, 27, 31	r	Reserved Read as 0; should be written with 0.

General Purpose Timer Array (GPTA)

GTC Input Multiplexer Control Registers

Two registers, GIMCRL and GIMCRH, are assigned to each GTCG[3:0]. GIMCRL controls the connections of cells 0 to 3 in a GTC Group. GIMCRH controls the connections of cells 4 to 7 in a GTC Group.

Note: These registers are not directly accessible and can be written and read only via the multiplexer register array FIFO (see [Section 24.2.4.5](#)).

GIMCRLg (g = 0-3)

Input Multiplexer Control Register for Lower Half of GTC Group g

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GIM EN3	GIMG3			0	GIML3			GIM EN2	GIMG2			0	GIML2		
r	rw			r	rw			r	rw			r	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GIM EN1	GIMG1			0	GIML1			GIM EN0	GIMG0			0	GIML0		
rw	rw			r	rw			rw	rw			r	rw		

Field	Bits	Type	Description
GIML0, GIML1, GIML2, GIML3	[2:0], [10:8], [18:16], [26:24]	rw	<p>Multiplexer Line Selection</p> <p>This bit field selects the input line of a GIMG that can be selected by bit field GIMGn for GIMG output n.</p> <p>000_B LIMG input IN0 selected 001_B LIMG input IN1 selected 010_B LIMG input IN2 selected 011_B LIMG input IN3 selected 100_B LIMG input IN4 selected 101_B LIMG input IN5 selected 110_B LIMG input IN6 selected 111_B LIMG input IN7 selected</p>

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
GIMG0, GIMG1, GIMG2, GIMG3	[6:4], [14:12], [22:20], [30:28]	rw	Multiplexer Group Selection This bit field determines the GIMG _n g which is connected to input n of GTC group g. 000 _B GIMG0g selected 001 _B GIMG1g selected (reserved for g = 3) 010 _B GIMG2g selected 011 _B GIMG3g selected 100 _B GIMG4g selected All other combinations are reserved.
GIMEN0, GIMEN1, GIMEN2, GIMEN3	7, 15, 23, 31	rw	Enable Multiplexer Connection 0 _B Input n is not connected to any line. 1 _B Input n is connected to the line defined by GIML _n and GIMG _n .
0	3, 11, 19, 27	r	Reserved Read as 0; should be written with 0.

General Purpose Timer Array (GPTA)

GIMCRHg (g = 0-3)

Input Multiplexer Control Register for Upper Half of GTC Group g

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GIMEN7	GIMG7			0	GIML7			GIMEN6	GIMG6			0	GIML6		
r	rw			r	rw			r	rw			r	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GIMEN5	GIMG5			0	GIML5			GIMEN4	GIMG4			0	GIML4		
r	rw			r	rw			r	rw			r	rw		

Field	Bits	Type	Description
GIML4, GIML5, GIML6, GIML7	[2:0], [10:8], [18:16], [26:24]	rw	<p>Multiplexer Line Selection</p> <p>This bit field selects the input line of a GIMG that can be selected by bit field GIMGn for GIMG output n.</p> <p>000_B LIMG input IN0 selected 001_B LIMG input IN1 selected 010_B LIMG input IN2 selected 011_B LIMG input IN3 selected 100_B LIMG input IN4 selected 101_B LIMG input IN5 selected 110_B LIMG input IN6 selected 111_B LIMG input IN7 selected</p>
GIMG4, GIMG5, GIMG6, GIMG7	[6:4], [14:12], [22:20], [30:28]	rw	<p>Multiplexer Group Selection</p> <p>This bit field determines the GIMGng which is connected to input n of GTC group g.</p> <p>000_B GIMG0g selected 001_B GIMG1g selected (reserved for g = 3) 010_B GIMG2g selected 011_B GIMG3g selected 100_B GIMG4g selected All other combinations are reserved.</p>

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
GIMEN4, GIMEN5, GIMEN6, GIMEN7	7, 15, 23, 31	rw	Enable Multiplexer Connection 0 _B Input n is not connected to any line. 1 _B Input n is connected to the line defined by GIMLn and GIMGn.
0	3, 11, 19, 27	r	Reserved Read as 0; should be written with 0.

General Purpose Timer Array (GPTA)

LTC Input Multiplexer Control Registers

Two registers, LIMCRL and LIMCRH, are assigned to each LTC group. LIMCRL controls the connections of LTC group cells with index 0 to 3. LIMCRH controls the connections of LTC group cells with index 4 to 7.

Note: These registers are not directly accessible and can be written and read only via the multiplexer register array FIFO (see [Section 24.2.4.5](#)).

LIMCRLg (g = 0-7)

Input Multiplexer Control Register for Lower Half of LTC Group g

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LIM EN3	LIMG3			0	LIML3			LIM EN2	LIMG2			0	LIML2		
r	rw			r	rw			r	rw			r	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LIM EN1	LIMG1			0	LIML1			LIM EN0	LIMG0			0	LIML0		
r	rw			r	rw			r	rw			r	rw		

Field	Bits	Type	Description
LIML0, LIML1, LIML2, LIML3	[2:0], [10:8], [18:16], [26:24]	rw	<p>Multiplexer Line Selection</p> <p>This bit field selects the input line of a LIMG that can be selected by bit field LIMGn for LIMG output n.</p> <p>000_B LIMG input IN0 selected 001_B LIMG input IN1 selected 010_B LIMG input IN2 selected 011_B LIMG input IN3 selected 100_B LIMG input IN4 selected 101_B LIMG input IN5 selected 110_B LIMG input IN6 selected 111_B LIMG input IN7 selected</p>

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
LIMG0, LIMG1, LIMG2, LIMG3	[6:4], [14:12], [22:20], [30:28]	rw	Multiplexer Group Selection This bit field determines the LIMG _n g which is connected to input n of LTC group g. 000 _B LIMG0g selected 001 _B LIMG1g selected (reserved for g = 3, 7) 010 _B LIMG2g selected 011 _B LIMG3g selected 100 _B LIMG4g selected All other combinations are reserved.
LIMEN0, LIMEN1, LIMEN2, LIMEN3	7, 15, 23, 31	rw	Enable Multiplexer Connection 0 _B Input n is not connected to any line. 1 _B Input n is connected to the line defined by LIML _n and LIMG _n .
0	3, 11, 19, 27	r	Reserved Read as 0; should be written with 0.

General Purpose Timer Array (GPTA)

LIMCRHg (g = 0-7)

Input Multiplexer Control Register for Upper Half of LTC Group g

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LIM EN7	LIMG7			0	LIML7			LIM EN6	LIMG6			0	LIML6		
r	rw			r	rw			r	rw			r	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LIM EN5	LIMG5			0	LIML5			LIM EN4	LIMG4			0	LIML4		
r	rw			r	rw			r	rw			r	rw		

Field	Bits	Type	Description
LIML4, LIML5, LIML6, LIML7	[2:0], [10:8], [18:16], [26:24]	rw	Multiplexer Line Selection This bit field selects the input line of a LIMG that can be selected by bit field LIMGn for LIMG output n. 000 _B LIMG input IN0 selected 001 _B LIMG input IN1 selected 010 _B LIMG input IN2 selected 011 _B LIMG input IN3 selected 100 _B LIMG input IN4 selected 101 _B LIMG input IN5 selected 110 _B LIMG input IN6 selected 111 _B LIMG input IN7 selected
LIMG4, LIMG5, LIMG6, LIMG7	[6:4], [14:12], [22:20], [30:28]	rw	Multiplexer Group Selection This bit field determines the LIMGng which is connected to input n of LTC group g. 000 _B LIMG0g selected 001 _B LIMG1g selected (reserved for g = 3, 7) 010 _B LIMG2g selected 011 _B LIMG3g selected 100 _B LIMG4g selected All other combinations are reserved.

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
LIMEN4, LIMEN5, LIMEN6, LIMEN7	7, 15, 23, 31	rw	Enable Multiplexer Connection 0 _B Input n is not connected to any line. 1 _B Input n is connected to the line defined by LIMLn and LIMGn.
0	3, 11, 19, 27	r	Reserved Read as 0; should be written with 0.

General Purpose Timer Array (GPTA)

24.3.12 Service Request Registers

The bits in the Service Request State Registers are service request status flags that are set by hardware (type “h”) when the related event occurs. Each service request status flag can be read twice (in SRSCx register and in SRSSx register, x = 0-3) and cleared or set by software when writing to the specific request bit in SRSCx or SRSSx.

The service request status flags can be cleared by software when writing a 1 to the corresponding bit location in the SRSCx registers. Writing a 0 has no effect.

The service request status flags can be set by software when writing a 1 to the corresponding bit location in the SRSSx registers. Writing a 0 has no effect.

GPTA0_SRSC0

GPTA0 Service Request State Clear Register 0

(010_H)

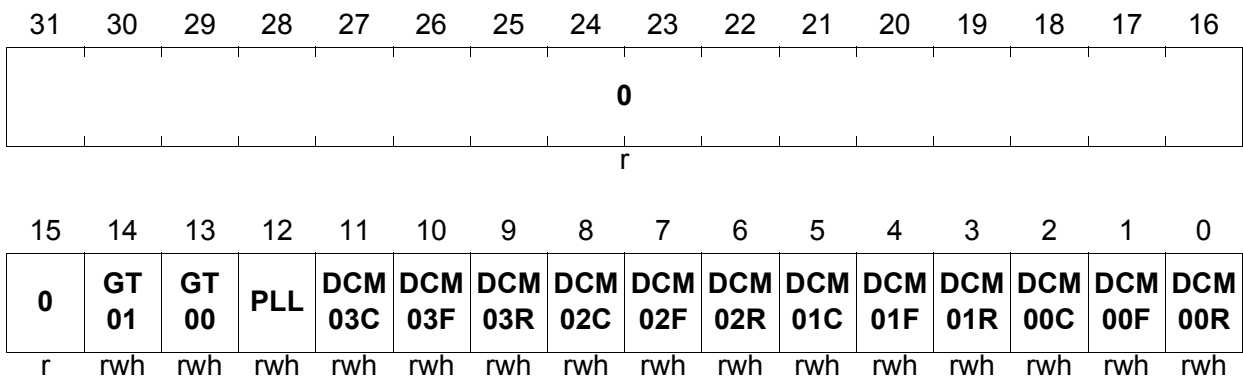
Reset Value: 0000 0000_H

GPTA1_SRSC0

GPTA1 Service Request State Clear Register 0

(010_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
DCM00R, DCM01R, DCM02R, DCM03R	0, 3, 6, 9	rwh	DCMk¹⁾ Rising Edge Event Service Request State 0 _B No service is requested. 1 _B Service is requested due to a rising edge detected on the DCMk input signal line.
DCM00F, DCM01F, DCM02F, DCM03F	1, 4, 7, 10	rwh	DCMk¹⁾ Falling Edge Event Service Request State 0 _B No service is requested. 1 _B Service is requested due to a falling edge detected on the DCMk input signal line.

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
DCM00C, DCM01C, DCM02C, DCM03C	2, 5, 8, 11	rwh	DCMk¹⁾ Compare Event Service Request State 0 _B No service is requested. 1 _B Service is requested due to a compare event occurred in DCMk unit (k = 0-3).
PLL	12	rwh	Counter Service Request State for PLL 0 _B No service is requested. 1 _B Service is requested because the counter for the number remaining output pulses decremented to 0.
GT00	13	rwh	GT0 Timer Service Request State 0 _B No service is requested. 1 _B Service is requested due to a GT0 timer overflow.
GT01	14	rwh	GT1 Timer Service Request State 0 _B No service is requested. 1 _B Service is requested due to a GT1 timer overflow.
0	[31:15]	r	Reserved Read as 0; should be written with 0.

1) k = 0-3; k = 0 refers to DCM00R, DCM00P, or DCM00C; k = 1 refers to DCM01R, DCM01P, or DCM01C; k = 2 refers to DCM02R, DCM02P, or DCM02C; k = 3 refers to DCM03R, DCM03P, or DCM03C.

General Purpose Timer Array (GPTA)

GPTA0_SRSS0

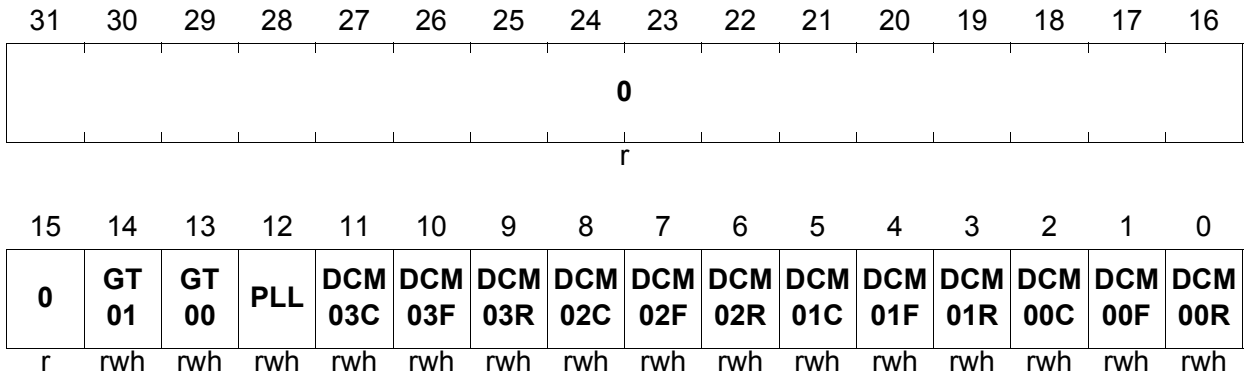
GPTA0 Service Request State Set Register 0
(014_H)

Reset Value: 0000 0000_H

GPTA1_SRSS0

GPTA1 Service Request State Set Register 0
(014_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
DCM00R, DCM01R, DCM02R, DCM03R	0, 3, 6, 9	rwh	DCMk¹⁾ Rising Edge Event Service Request State 0 _B No service is requested. 1 _B Service is requested due to a rising edge detected on DCMk input signal line.
DCM00F, DCM01F, DCM02F, DCM03F	1, 4, 7, 10	rwh	DCMk¹⁾ Falling Edge Event Service Request State 0 _B No service is requested. 1 _B Service is requested due to a falling edge detected on DCMk input signal line.
DCM00C, DCM01C, DCM02C, DCM03C	2, 5, 8, 11	rwh	DCMk¹⁾ Compare Event Service Request State 0 _B No service is requested. 1 _B Service is requested due to a compare event occurred in DCMk cell.
PLL	12	rwh	Counter Service Request State for PLL 0 _B No service is requested 1 _B Service is requested because the counter for the number remaining output pulses decremented to 0.
GT00	13	rwh	GT0 Timer Service Request State 0 _B No service is requested. 1 _B Service is requested due to a GT0 timer overflow.

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
GT01	14	rwh	GT1 Timer Service Request State 0 _B No service is requested. 1 _B Service is requested due to a GT1 timer overflow.
0	[31:15]	r	Reserved Read as 0; should be written with 0.

1) k = 0-3; k = 0 refers to DCM00R, DCM00P, or DCM00C; k = 1 refers to DCM01R, DCM01P, or DCM01C; k = 2 refers to DCM02R, DCM02P, or DCM02C; k = 3 refers to DCM03R, DCM03P, or DCM03C.

GPTA0_SRSC1

GPTA0 Service Request State Clear Register 1
(018_H)

Reset Value: 0000 0000_H

GPTA1_SRSC1

GPTA1 Service Request State Clear Register 1
(018_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
GTck (k = 00-31)	k	rwh	GTck Capture/Compare Service Request State 0 _B No service is requested. 1 _B Service is requested due to a capture or compare event occurred in GTck.

General Purpose Timer Array (GPTA)

GPTA0_SRSS1

GPTA0 Service Request State Set Register 1
(01C_H)

Reset Value: 0000 0000_H

GPTA1_SRSS1

GPTA1 Service Request State Set Register 1
(01C_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC	GTC
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
GTck (k = 00-31)	k	rwh ¹⁾	GTck Capture/Compare Service Request State 0 _B No service is requested. 1 _B Service is requested due to a capture or compare event occurred in GTck.

1) Writing a one to a cleared bit sets the bit. All other write operations have no effect.

General Purpose Timer Array (GPTA)

SRSC2

Service Request State Clear Register 2

(020_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
LTck (k = 00-31)	k	rwh	LTck Timer/Capture/Compare Service Request State 0 _B No service is requested. 1 _B Service is requested due to a timer overflow, capture, or compare event that occurred in LTck.

General Purpose Timer Array (GPTA)

SRSS2

Service Request State Set Register 2

(024_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
LTck (k = 00-31)	k	rwh	LTck Timer/Capture/Compare Service Request State 0 _B No service is requested. 1 _B Service is requested due to a timer overflow, capture, or compare event that occurred in LTck.

General Purpose Timer Array (GPTA)

SRSC3

Service Request State Clear Register 3

(028_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
LTCK (k = 32-63)	k-32	rwh	LTCK Timer/Capture/Compare Service Request State 0 _B No service is requested. 1 _B Service is requested due to a timer overflow, capture, or compare event that occurred in LTCK.

General Purpose Timer Array (GPTA)

SRSS3

Service Request State Set Register 3

(02C_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC	LTC
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
LTck (k = 32-63)	k-32	rwh	LTck Timer/Capture/Compare Service Request State 0 _B No service is requested. 1 _B Service is requested due to a timer overflow, capture, or compare event that occurred in LTck.

General Purpose Timer Array (GPTA)

Node Redirection Register

The Service Request Node Redirection Register allows that GTC service requests of GTCs with an odd index number k can be individually redirected via register SRNR to a service request group that is assigned mainly to four LTCs. More details are provided on [Page 24-113](#).

GPTA0_SRNR

GPTA0 Service Request Node Redirection Register

(030_H)

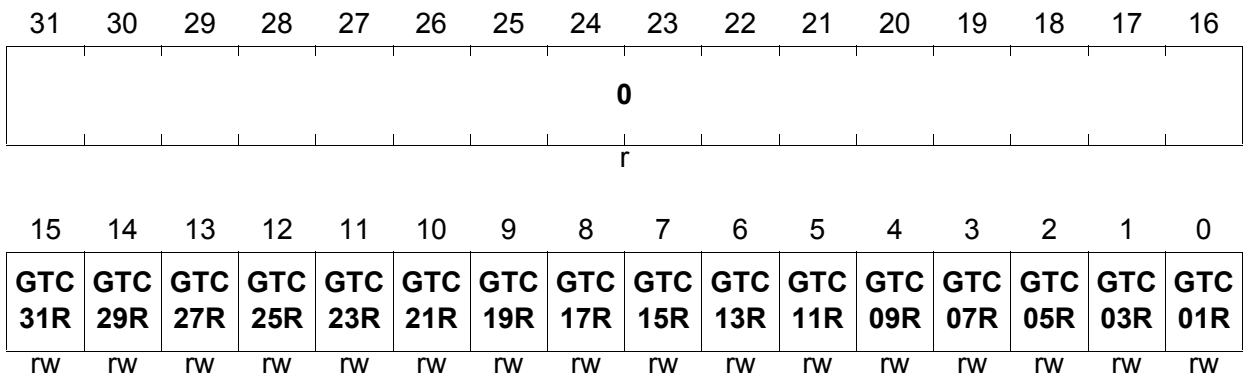
Reset Value: 0000 0000_H

GPTA1_SRNR

GPTA1 Service Request Node Redirection Register

(030_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
GTC01R, GTC03R, GTC05R, GTC07R, GTC09R, GTC11R, GTC13R, GTC15R, GTC17R, GTC19R, GTC21R, GTC23R, GTC25R, GTC27R, GTC29R, GTC31R	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15	rw	Global Timer Cell k Redirection 0 _B No redirection of GTC service requests. 1 _B Redirection of GTC service request to LTC service request groups (see Page 24-113).

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
0	[31:16]	r	Reserved Read as 0; should be written with 0.

General Purpose Timer Array (GPTA)

24.4 LTCA2 Kernel Description

The Local Timer Cell Array 2 (LTCA2) is a module that contains a signal generation unit with 64 Local Timer Cells. These LTCs have identical functionality as the LTCs which are available in the signal generation unit of the GPTA module.

The Local Timer Cells (LTC00 to LTC62) can be configured to operate in different modes: Capture Mode, Compare Mode, Free-Running Timer Mode, Reset Timer Mode, and One Shot Mode. Adjacent cells may be combined to operate on the same pin, thus generating complex waveforms. One LTC (LTC63) can be used for special compare modes.

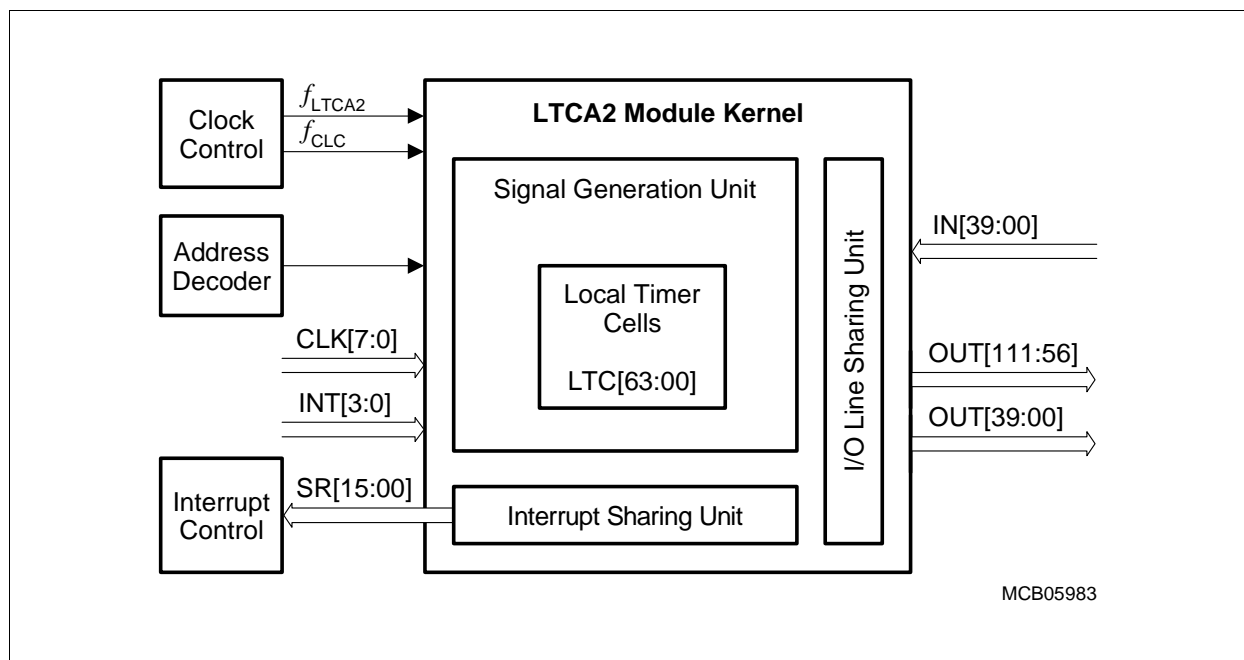


Figure 24-74 LTCA2 Kernel Block Diagram

The LTCA2 Module Kernel contains a Signal Generation Unit that contains all Local Timer Cells including an I/O Line Sharing Unit that controls the LTC connections to the I/O lines and output lines. I/O lines are supposed to be connected to I/O port lines while the output lines are typically connected to a MSC interface that is especially able to control external power devices via a serial connection. The LTCs can be further connected to input signals of an external clock bus and input lines coming e.g. from other on-chip peripheral units.

The clock control unit generates two module clocks that are required for LTCA2 operation. An address decoder generates the select signals for the LTC registers. Service requests, coming from the LTCs, are able to generate interrupts via the Interrupt Sharing Unit. The interrupts are handled by the external interrupt logic.

General Purpose Timer Array (GPTA)

24.4.1 Local Timer Cell (LTC00 to LTC62)

LTC00 to LTC62 are fully identical with the Local Timer Cells in the GPTA0 and GPTA1 module. Its functionalities are described in sections “Local Timer Cell (LTC00 to LTC62)” on Page 24-67 and “Local Timer Cell LTC63” on Page 24-79.

24.4.2 Input/Output Line Sharing Unit (IOLS)

The I/O Line Sharing Unit allows the inputs and outputs of the LTCA2 unit to be routed with high flexibility between I/O lines, output lines, clock inputs, and other on-chip peripherals. The LTCA2 module provides a total of 40 input lines and 96 output lines, that are connected to five I/O groups IOG[4:0] and seven output groups OG[6:0].

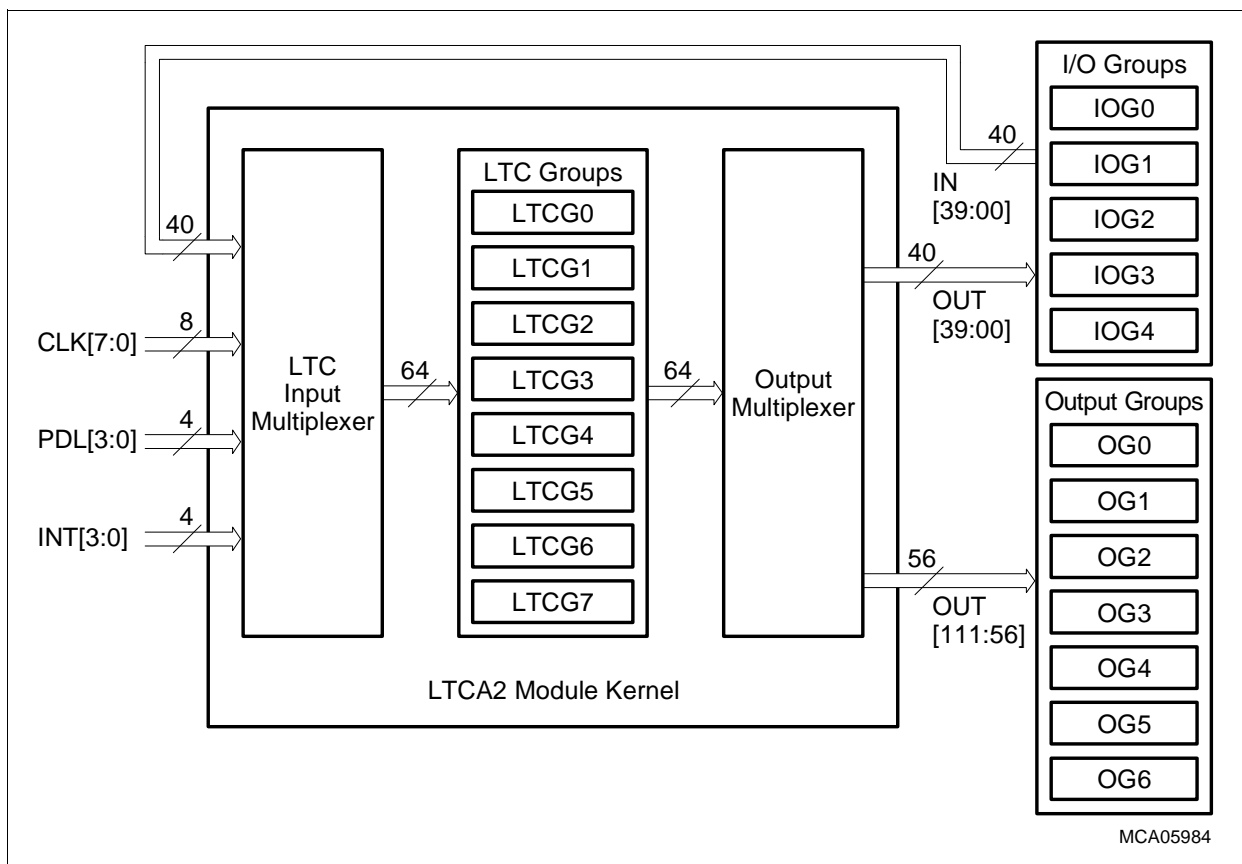


Figure 24-75 Input/Output Line Sharing Unit Overview

The LTCA2 I/O Line Sharing Unit makes the following two selections:

- LTC output multiplexer selection
- LTC input multiplexer selection

To choose these selection, the input and output lines of the related cells are integrated into groups with eight parts, each. There are I/O groups, output groups, LTC groups, and a PDL/INT group.

General Purpose Timer Array (GPTA)

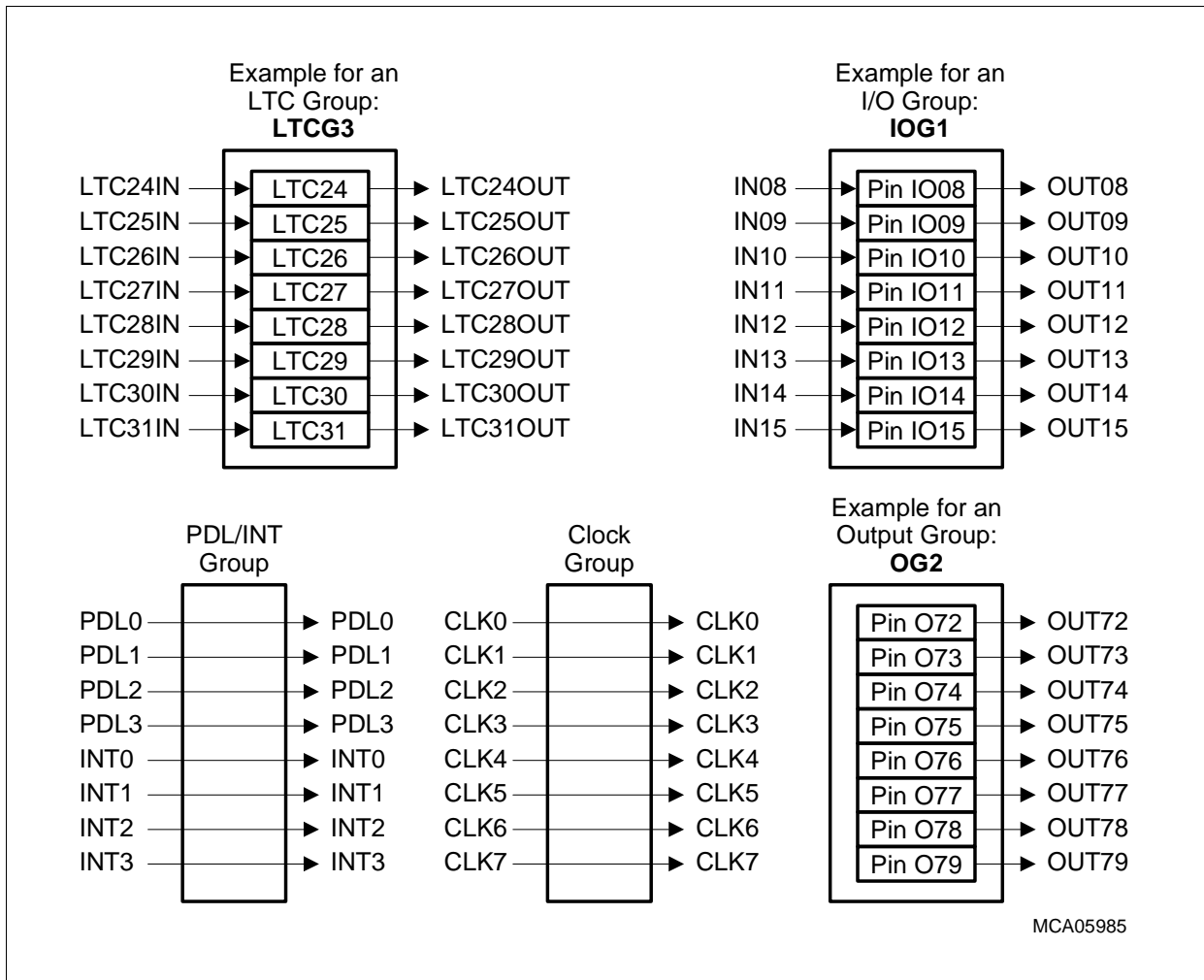


Figure 24-76 Groups Definitions for I/O Line Sharing Unit

An **LTC group** combines eight LTC cells with its input and output lines. This results in eight LTC groups, LTCG0 to LTCG7.

An **I/O group** combines eight LTCA2 I/O lines connected to bi-directional device pins with its input and output lines. This results in five I/O groups, IOG0 to IOG4, supporting 40 I/O lines.

An **output group** combines eight LTCA2 output lines connected to device pins as an output. This results in seven output groups, OG0 to OG6, supporting 56 output lines.

The **PDL/INT group** is a logical group that combines the LTCA2 module inputs PDL[3:0] together with the inputs INT[3:0].

The **clock group** is a logical group that combines the eight LTCA2 module clock inputs CLK[7:0].

General Purpose Timer Array (GPTA)

Table 24-18 Group to I/O Line/Cell Assignment

Group/Module	Cell/Line	Input	Output
LTC Groups			
LTCG0	LTC[07:00]	LTC[07:00]IN	LTC[07:00]OUT
LTCG1	LTC[15:08]	LTC[15:08]IN	LTC[15:08]OUT
LTCG2	LTC[23:16]	LTC[23:16]IN	LTC[23:16]OUT
LTCG3	LTC[31:24]	LTC[31:24]IN	LTC[31:24]OUT
LTCG4	LTC[39:32]	LTC[39:32]IN	LTC[39:32]OUT
LTCG5	LTC[47:40]	LTC[47:40]IN	LTC[47:40]OUT
LTCG6	LTC[55:48]	LTC[55:48]IN	LTC[55:48]OUT
LTCG7	LTC[63:56]	LTC[63:56]IN	LTC[63:56]OUT
I/O Groups			
IOG0	–	IN[07:00]	OUT[07:00]
IOG1	–	IN[15:08]	OUT[15:08]
IOG2	–	IN[23:16]	OUT[23:16]
IOG3	–	IN[31:24]	OUT[31:24]
IOG4	–	IN[39:32]	OUT[39:32]
Output Groups			
OG0	–	–	OUT[63:56]
OG1	–	–	OUT[71:64]
OG2	–	–	OUT[79:72]
OG3	–	–	OUT[87:80]
OG4	–	–	OUT[95:88]
OG5	–	–	OUT[103:96]
OG6	–	–	OUT[111:104]
Clock Group			
–	–	CLK[7:0]	CLK[7:0]
PDL/INT Groups			
PDL[1:0] PDL Bus	–	PDL[3:0]	PDL[3:0]
External Input [3:0]	–	INT[3:0]	INT[3:0]

General Purpose Timer Array (GPTA)

24.4.2.1 Output Multiplexer

The output multiplexer shown in **Figure 24-75** and **Figure 24-77** connects the 32 GTC output lines and the 64 LTC output lines with the I/O groups (5 × 8 = 40 output lines) and the output groups (7 × 8 = 56 output lines).

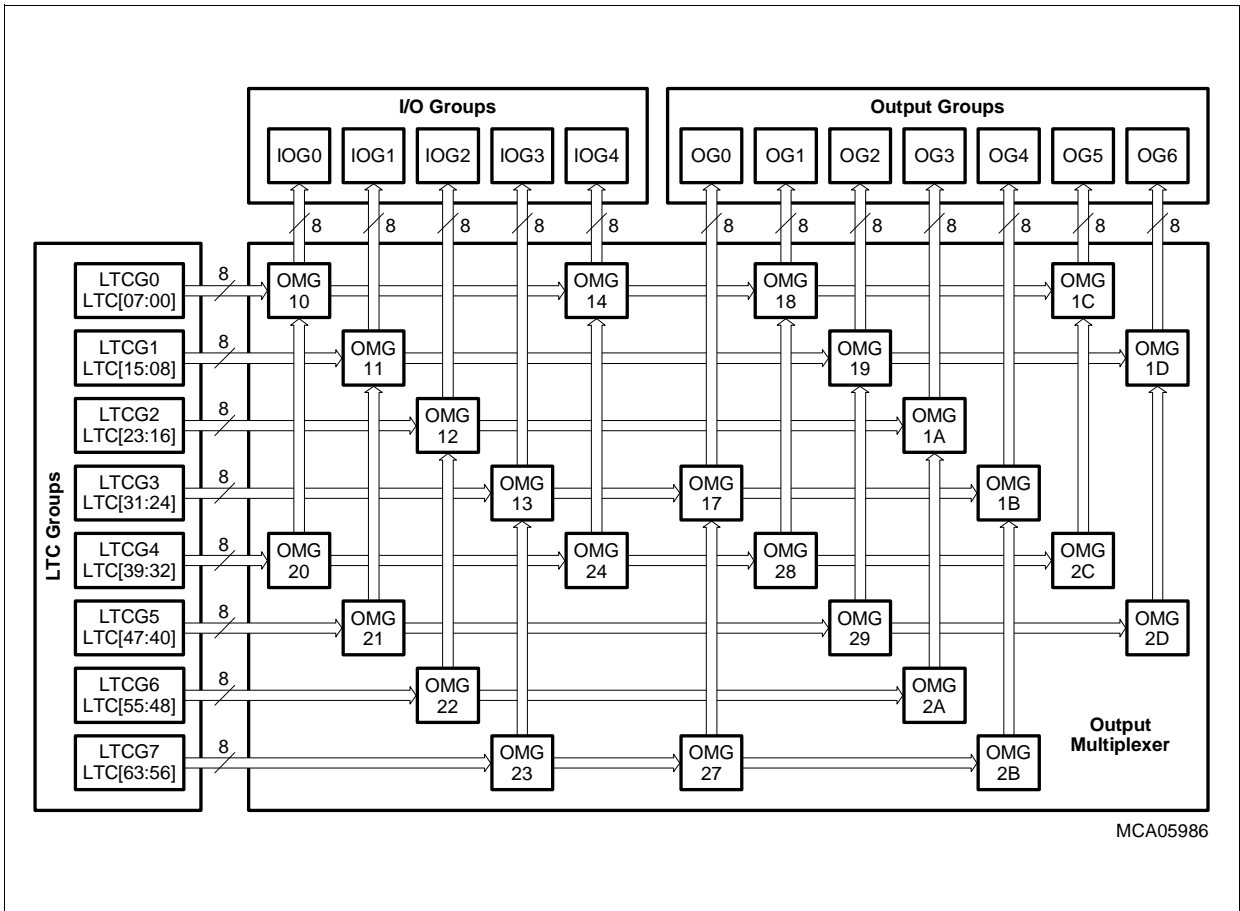


Figure 24-77 Output Multiplexer of LTCA2

The output multiplexer contains Output Multiplexer Groups (OMGs) that connect the Local Timer Cells with the input lines of the I/O groups and output groups. The LTCs are grouped into eight LTC groups (LTCG[7:0]) with 8 cells each. In the same way, I/O groups and output groups are grouped into 12 groups (five I/O groups and seven output groups) with 8 lines each.

General Purpose Timer Array (GPTA)

Figure 24-78 shows the logical structure of an OMG.

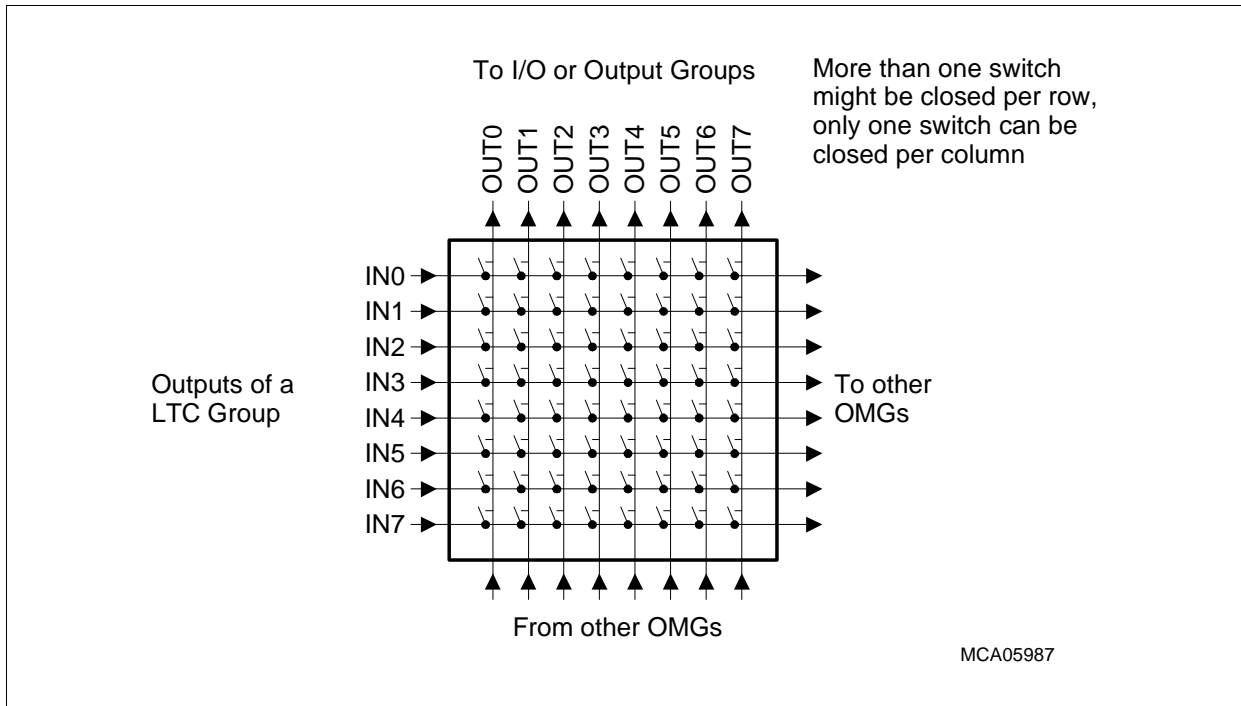


Figure 24-78 Output Multiplexer Group (OMG) Structure

Rules for connections to Output Multiplexer Group OMG:

- OMG output line OUT0 is always connected to the input of an I/O or output group with the lowest index. The remaining output lines OUT1 to OUT7 are connected to the I/O or Output lines with ascending index.
Example: for OMG13 (see Figure 24-77), the outputs OUT0 to OUT7 are wired to input lines 0 to 7 of I/O group 3 (IOG3).
- One input of an I/O or output group can be connected only to the output of one timer cell. This is guaranteed by the OMG control register layout. Otherwise, short circuits and unpredictable behavior would occur. On the other hand, it is permitted that for the output of an LTC cell to be connected to more than one input of an I/O or output group.

The output multiplexer group configuration is based on the following principles:

- Each OMG is referenced with two index variables: n and g (OMGng)
- Index n is a group number. Local timer cell groups LTCG[3:0] have the group number 1, and Local Timer Cell Groups LTCG[7:4] have the group number 2.
- Index g indicates the number of an I/O or output group g ($g = 0-13_D$) to which the outputs of the output multiplexer group OMGng are connected. I/O groups OG0 to OG4 are assigned to index variable $g = 0$ to 4 and output groups OG0 to OG6 are assigned to index variable $g = 7$ to 13. Index $g = 5, 6$ is not available.

General Purpose Timer Array (GPTA)

The output multiplexer logic as seen for programming is shown in [Figure 24-79](#). With this logic, three GTC or LTC group signals are always combined to one output line that leads to the input of an I/O or output group. For example, when looking at [Figure 24-77](#), each of the eight output multiplexer output lines to I/O group IOG4 is connected via two OMGn4 (n = 1, 2) with the eight outputs of two LTC groups (LTG0 and LTG4).

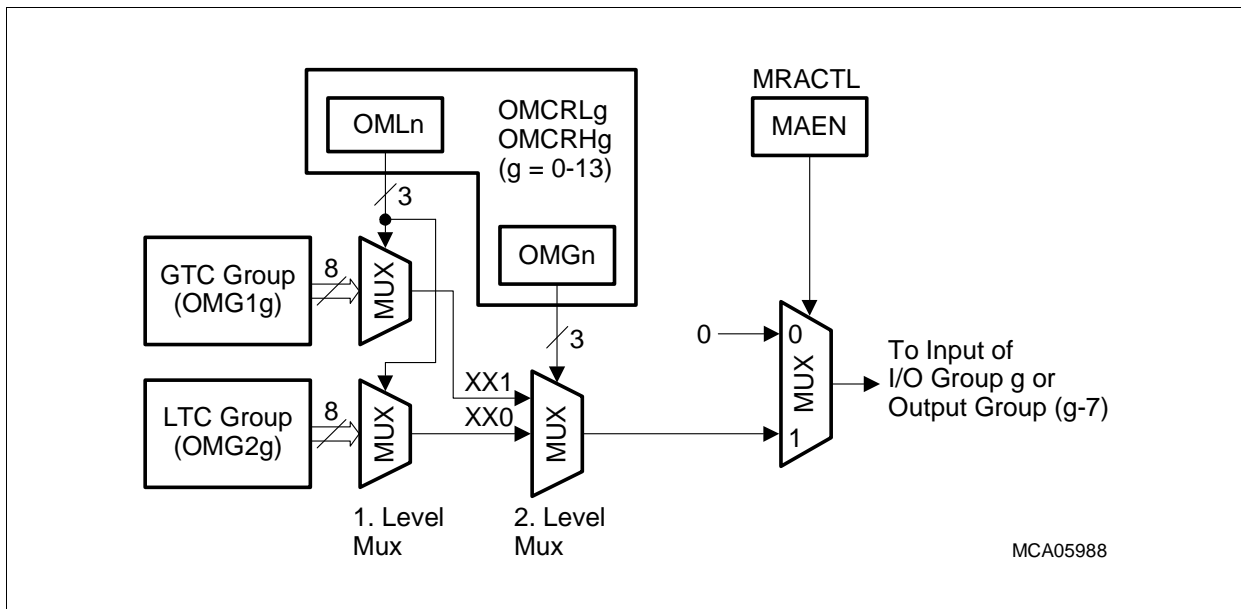


Figure 24-79 OMG Multiplexer (Programmer's View)

The 1. level multiplexer is built up by two 8:1 multiplexers that are controlled in parallel by bit field OMLn. Bit field OMGn controls the 2. level multiplexer and connects one of the 1. level multiplexer outputs to output n. The output of the 2. level multiplexer is only connected only to the input of an I/O group or output group if bit MRACTL.MAEN (multiplexer array enabled) is set. If MRACTL.MAEN = 0, the corresponding OMG output will be held at a low level.

Two Output Multiplexer Control Registers, OMCRL and OMCRH (see also [Page 24-239](#)), are assigned to each of the I/O or output groups. Therefore, in total 24 registers control the connections within the output multiplexer of the LTCA2 module. The OMCRL registers control the OMG output lines 0 to 3 and the OMCRH registers control the OMG output lines 4 to 7. [Table 24-19](#) lists all Output Multiplexer Control Registers with its control functions. Please note that all Output Multiplexer Control Registers are not directly accessible but must be written or read using the FIFO array structure as described on [Page 24-229](#).

General Purpose Timer Array (GPTA)

Table 24-19 Output Multiplexer Control Register Assignments

I/O Group or Output Group		Controlled by Multiplexer Control Register	Selectable Groups via OMGng
IOG0	IN[03:00]/OUT[03:00]	OMCRL0	GTCG0, LTCG0, LTCG4
	IOG0	OMCRH0	
IOG1	IN[11:08]/OUT[11:08]	OMCRL1	GTCG1, LTCG1, LTCG5
	IN[15:12]/OUT[15:12]	OMCRH1	
IOG2	IN[19:16]/OUT[19:16]	OMCRL2	GTCG2, LTCG2, LTCG6
	IN[23:20]/OUT[23:20]	OMCRH2	
IOG3	IN[27:24]/OUT[27:24]	OMCRL3	GTCG3, LTCG3, LTCG7
	IN[31:28]/OUT[31:28]	OMCRH3	
IOG4	IN[35:32]/OUT[35:32]	OMCRL4	GTCG0, LTCG0, LTCG4
	IN[39:36]/OUT[39:36]	OMCRH4	
OG0	OUT[59:56] ¹⁾	OMCRL7	GTCG3, LTCG3, LTCG7
	OUT[63:60]	OMCRH7	
OG1	OUT[67:64]	OMCRL8	GTCG0, LTCG0, LTCG4
	OUT[71:68]	OMCRH8	
OG2	OUT[75:72]	OMCRL9	GTCG1, LTCG1, LTCG5
	OUT[79:76]	OMCRH9	
OG3	OUT[83:80]	OMCRL10	GTCG2, LTCG2, LTCG6
	OUT[87:84]	OMCRH10	
OG4	OUT[91:88]	OMCRL11	GTCG3, LTCG3, LTCG7
	OUT[95:92]	OMCRH11	
OG5	OUT[99:96]	OMCRL12	GTCG0, LTCG0, LTCG4
	OUT[203:100]	OMCRH12	
OG6	OUT[107:104]	OMCRL13	GTCG1, LTCG1, LTCG5
	OUT[111:108]	OMCRH13	

1) OUT[55:40] is not available.

General Purpose Timer Array (GPTA)

24.4.2.2 LTC Input Multiplexing Scheme

The LTC input multiplexer as shown in **Figure 24-77** and **Figure 24-80** connects the 40 (= 5 × 8) input lines of the I/O groups, the eight clock bus input lines, or the four PDL input lines PDL[3:0] and the four internal input lines INT[3:0] with the 64 (= 8 × 8) LTC inputs, organized in eight LTC groups.

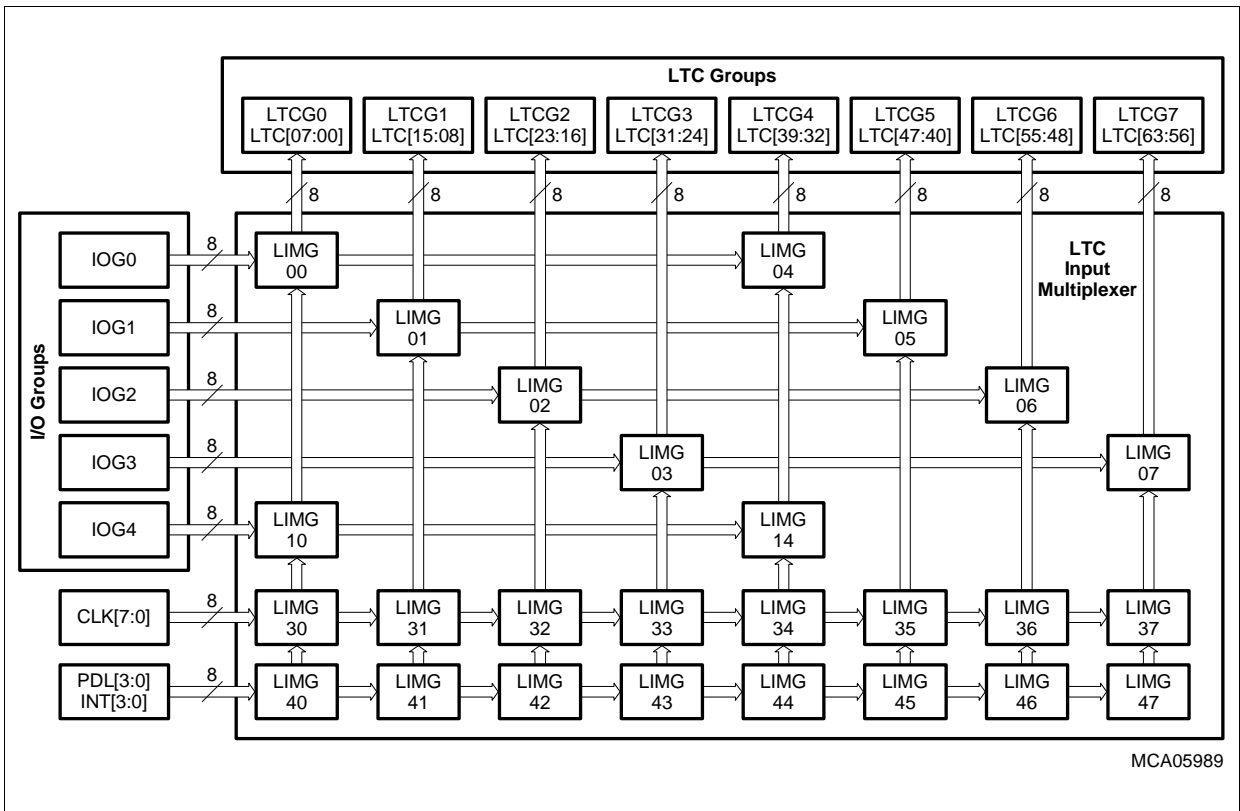


Figure 24-80 LTC Input Multiplexer of LTCA2

General Purpose Timer Array (GPTA)

The LTC input multiplexer contains LTC input Multiplexer Groups (LIMGs) that connect the I/O groups or the clock, PDL, or INT inputs to the input lines of the LTCs, organized in eight LTC groups with 8 cells each. IOGs are grouped into five IOGs (IOG[4:0]) with eight lines each. Two special groups are available, a clock group with eight lines representing the clock bus inputs CLK[7:0] and a PDL/INT group with eight outputs that combines the four PDL inputs and the four inputs INT[3:0] as a group of LIMGs inputs.

Figure 24-81 shows the logical structure of a LIMG.

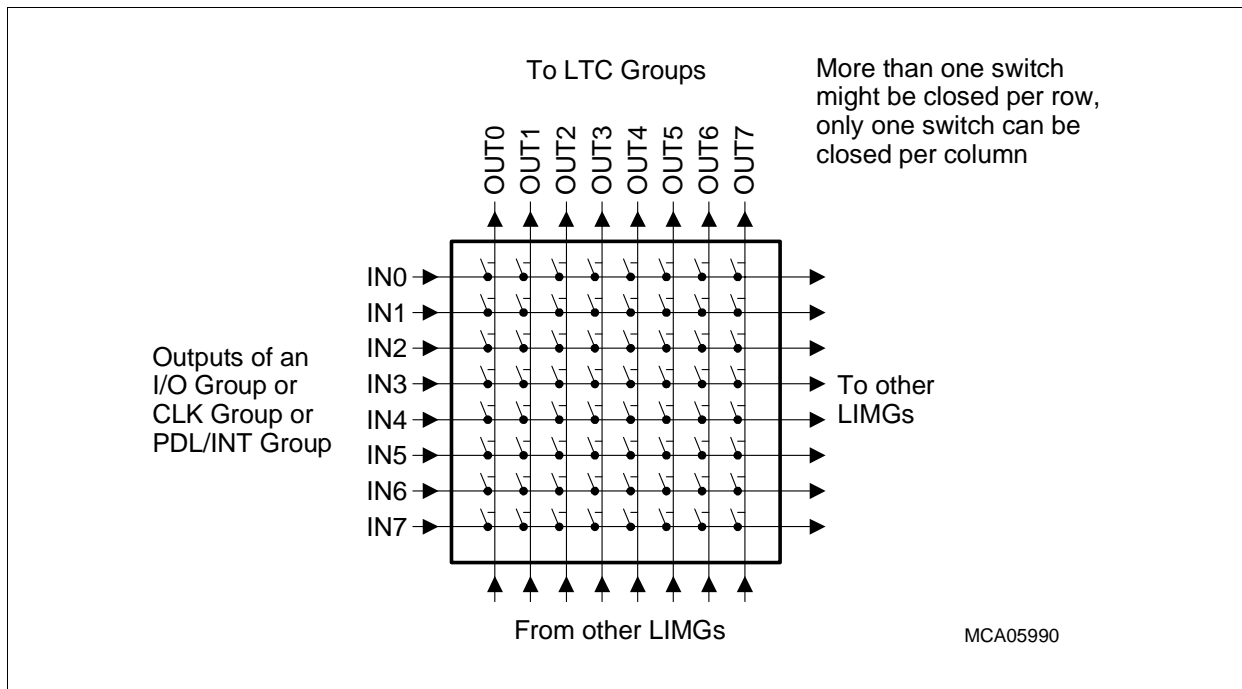


Figure 24-81 LTC Input Multiplexer Group (LIMG) Structure

Rules for connections to LTC Input Multiplexer Group LIMG:

- Within a I/O group, the line or the output of the cell with the lowest index number is connected to LIMG input line IN0. The remaining lines, cells or lines of a group are connected to LIMG input lines IN1 to IN7 with ascending index numbers. At the clock group, CLK0 is connected to IN0 and the remaining clock lines are connected to LIMG input lines IN1 to IN7 with ascending index numbers. At the PDL/INT group, PDL[3:0] (see [Page 24-20](#)) are connected to IN[3:0] and INT[3:0] are connected to IN[7:4].
Example: for LIMG04 (see [Figure 24-80](#)), the I/O lines of IOG0 (IN00 up to IN07) are wired to its input lines IN0 to line IN7.
- Multiplexer output OUT0 of a LIMG is always connected to the input of an LTC group with the lowest index. The remaining output lines OUT1 to OUT7 are connected to the LTC inputs with ascending index.
Example: for LIMG04 (see [Figure 24-80](#)), the outputs OUT0 to OUT7 are wired to the inputs of LTC32 to GTC39.

General Purpose Timer Array (GPTA)

- An LTC input can be connected either to an I/O group output, or to a clock bus output, or to an PDL/INT output. This is guaranteed by the LIMG control register layout. Otherwise, short circuits and unpredictable behavior would occur. In contrast, it is permitted that an I/O group output, or to a clock bus output, or a PDL/INT output is connected to more than one LTC input.

The LTC input multiplexer group configuration is based on the following principles:

- Each LIMG is referenced with two index variables: n and g (LIMGn_g)
- Index n is a group number. I/O groups IOG[3:0] have group number 0, I/O group IOG4 has group number 1, clock bus lines CLK[7:0] have group number 3, and the PDL/INT group has group number 4.
- Index g indicates the number of the LTC group g (g = 0-7) to which the outputs of the input multiplexer group LIMGn_g are connected.

The LTC input multiplexer logic as seen for programming is shown in **Figure 24-82**. With this logic, four group signals (from I/O groups, clock group, or PDL/INT group) are always combined to one output line that leads to an LTC input of LTC group g. For example, when looking at **Figure 24-80**, each of the eight LTC input multiplexer output lines to LTC group LTCG2 is connected via three LIMGn₂ (n = 1, 3, 4) to the eight outputs of I/O group IOG2, the clock group, and the PDL/INT group.

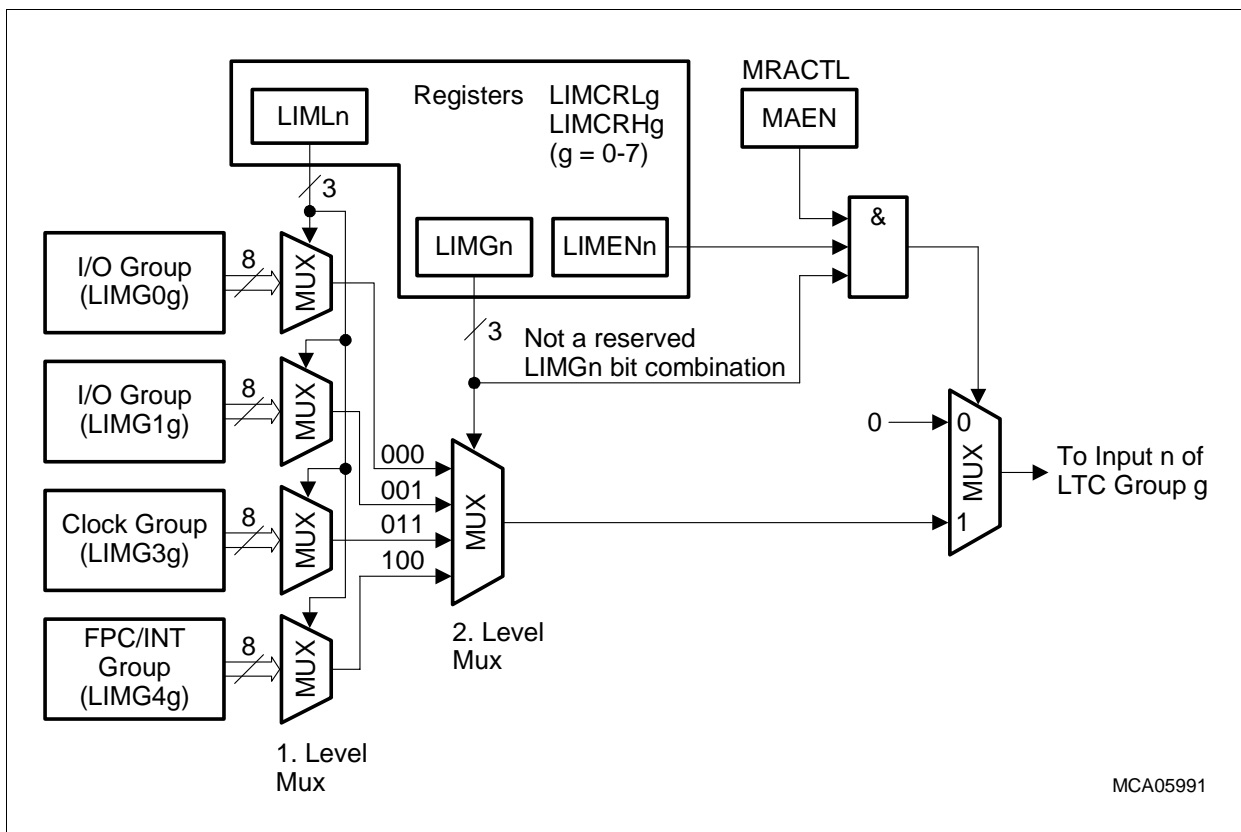


Figure 24-82 LTC Input Multiplexer (Programmer's View)

General Purpose Timer Array (GPTA)

The 1. level multiplexer is built up by four 8:1 multiplexers that are controlled in parallel by bit field LIMLn. Bit field LIMGn controls the 2. level multiplexer and connects one of the 1. level multiplexer outputs to one of the LIMGng outputs. The output of the 2. level multiplexer is connected only to the input of an LTC if bit LIMENn is set (enable multiplexer connection), and bit MRACTL.AEN is set (multiplexer array enabled), and no reserved bit combination of LIMGn is selected. If one of these conditions is not true, the corresponding LTC input will be held at a low level.

Two LTC Input Multiplexer Control Registers, LIMCRL and LIMCRH (see also [Page 24-243](#)), are assigned to each of the LTC groups. Therefore, in total sixteen registers control the connections within the LTC input multiplexer of the LTCA2 module.

The LIMCRL registers control the LIMG output lines 0 to 3 and the LIMCRH registers control the LIMG output lines 4 to 7. [Table 24-20](#) lists all LTC Input Multiplexer Control Registers with its control functions. Please note that all LTC Input Multiplexer Control Registers are not directly accessible but must be written or read using a FIFO array structure as described on [Page 24-229](#).

Table 24-20 LTC Input Multiplexer Control Register Assignments

LTC Group and LTCs		Controlled by Register	Selectable Groups via LIMGn
LTCG0	LTC[03:00]	LIMCRL0	IOG0, IOG4, CLOCK, PDL/INT
	LTC[07:04]	LIMCRH0	
LTCG1	LTC[11:08]	LIMCRL1	IOG1, CLOCK, PDL/INT
	LTC[15:12]	LIMCRH1	
LTCG2	LTC[19:16]	LIMCRL2	IOG2, CLOCK, PDL/INT
	LTC[23:20]	LIMCRH2	
LTCG3	LTC[27:24]	LIMCRL3	IOG3, CLOCK, PDL/INT
	LTC[31:28]	LIMCRH3	
LTCG4	LTC[35:32]	LIMCRL4	IOG0, IOG4, CLOCK, PDL/INT
	LTC[39:36]	LIMCRH4	
LTCG5	LTC[43:40]	LIMCRL5	IOG1, CLOCK, PDL/INT
	LTC[47:44]	LIMCRH5	
LTCG6	LTC[51:48]	LIMCRL6	IOG2, CLOCK, PDL/INT
	LTC[55:52]	LIMCRH6	
LTCG7	LTC[59:56]	LIMCRL7	IOG3, CLOCK, PDL/INT
	LTC[63:60]	LIMCRH7	

General Purpose Timer Array (GPTA)

24.4.2.3 Multiplexer Register Array Programming

A total of 40 control registers are required to program the configuration of the output multiplexer and the LTC input multiplexer of the Input/Output Line Sharing Unit. These IOLS control registers are combined into a Multiplexer Register Array FIFO that can only be read or written sequentially. Therefore, the control registers values cannot be accessed directly but must be accessed in a specific sequential order.

Three registers are available for controlling the Multiplexer Register Array:

- Multiplexer Register Array Control Register MRACTL
- Multiplexer Register Array Data In Register MRADIN
- Multiplexer Register Array Data Out Register MRADOUT

Figure 24-83 shows the structure of the multiplexer array FIFO with the arrangement of the multiplexer control registers.

For programming of the multiplexer array FIFO, the following steps must be executed:

1. Disable interconnections of the multiplexer array by writing `MRACTL.MAEN = 0` (default after reset). The multiplexer array is disabled, all cell input lines are driven with 0, and device pins assigned to LTCA2 I/O lines or output lines are disconnected.
2. Reset the write cycle counter to 0 by writing `MRACTL.WCRES = 1`.
3. Write sequentially the multiplexer control register contents one after the other (40 values) into MRADIN, starting with the register values for `OMCRH13`, `OMCRL13`, ... up to `LIMCRH0`, `LIMCRL0` (see **Figure 24-83**). After the first MRADIN write operation, the contents for `OMCRH13` is at FIFO position 1. With each following MRADIN write operation, it becomes shifted one FIFO position upwards. After the 40. MRADIN write operation, the `OMCRH13` value is at its final position. The contents of FIFO position 40 can be read via register MRADOUT. With each MRADIN write operation the write cycle counter `MRACTL.FIFOFILLCNT` is incremented by 1. After all FIFO entries have been written, the FIFO is locked, bit `MRACTL.FIFOFULL` is set, and further MRADIN write operations are discarded until bit `MRACTL.WCRES` is written again with a 0.
4. Enable the multiplexer array by writing `MRACTL.MAEN = 1`. This establishes and enables all programmed interconnections.

To check the FIFO contents, the FIFO can be written a second time. At this check MRADIN is written before MRADOUT is read. This will return the FIFO contents of the first write sequence in the order of `OMCRH13`, `OMCRL13`, ..., `LIMCRH0`, `LIMCRL0`.

Before disabling the multiplexer array FIFO, LTCA2 output pins that are already enabled as LTCA2 output should be switched to GPIO function to avoid output spikes. After enabling the multiplexer array FIFO again, the LTCA2 output can be switched again back to LTCA2 output function.

Shifting the write data through the FIFO requires a few clock cycles. When new data becomes written before the FIFO is ready to accept them, wait states will be inserted into the write access.

General Purpose Timer Array (GPTA)

If the OMCRLg register bit field OMCn of the multiplexer array is programmed with an invalid (reserved) value, the related outputs will be forced to 0. When the array is disabled (MRACTL.MAEN = 0), all cell inputs and outputs are disconnected from the GPIO lines and are driven with 0.

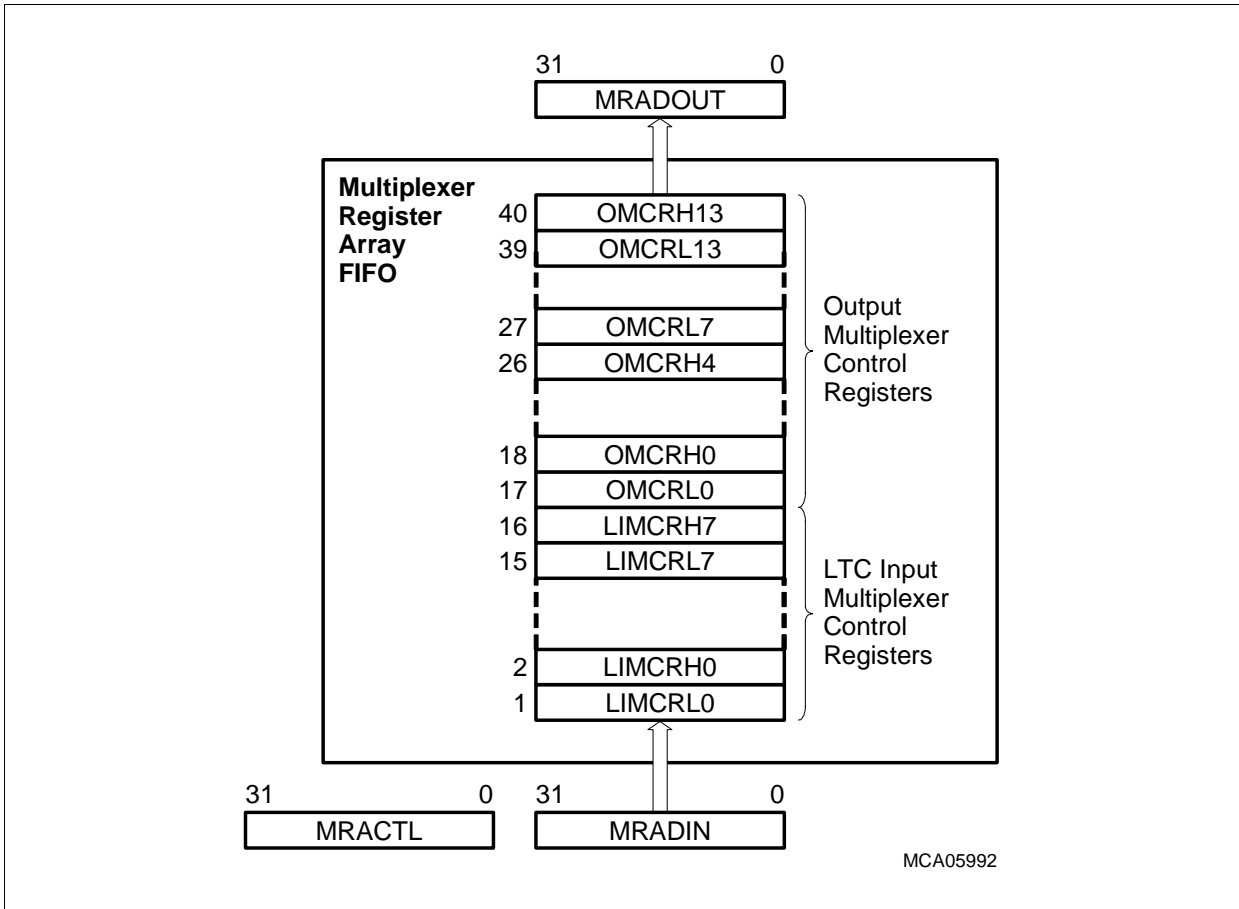


Figure 24-83 LTCA2 Multiplexer Array Control Register FIFO Structure

General Purpose Timer Array (GPTA)

24.4.3 Interrupt Sharing Unit (IS)

The LTCA2 provides 16 service request sources. These service request sources are generated by the LTCs.

Table 24-21 LTCA2 Number of Service Request Sources

Cell Type	Number of Cells	Number of Service Request Sources/Cell	Total Number of Request Sources
LTC	64	1	16

To reduce hardware and software overhead, four request sources are combined together in service request groups. A service request group y ($y = 00-15$) has four service request inputs and one service request output SR_y which is typically connected outside the LTCA2 kernel to a standard interrupt node y and controlled by its SR_{Cy} register.

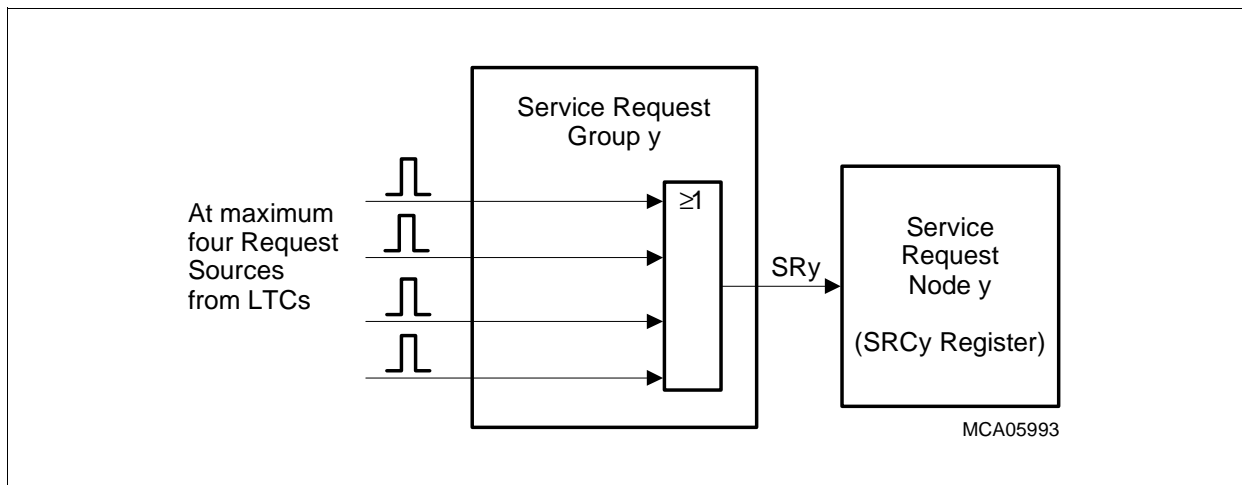


Figure 24-84 Service Request Groups

The bits in the Service Request State Registers ($SRSS_x$ and $SRSC_x$, $x = 3, 2$) are service request status flags that are set by hardware (type "h") when the related event occurs. Each LTCA2 service request source has its own service request flag. This flag is normally set by hardware but can be set and cleared by software. Each service request status flag can be read twice, at the same bit location in $SRSC_x$ register and in $SRSS_x$ register, and cleared or set by software when writing to the corresponding request bit in $SRSC_x$ or $SRSS_x$. When writing to $SRSC_x$ or $SRSS_x$, several flags can be cleared at once by one write operation. Flags written with 0 are not influenced. This feature allows fast, simple clearing or setting of request flags without affecting other bits in the same register.

Note that the service request flag is always set by the service request event even if the corresponding service request is disabled in the interrupt node.

General Purpose Timer Array (GPTA)

Table 24-22 lists the interrupt requests used by LTCA2.

Table 24-22 LTCA2 Service Request Groups

Service Request Group Number	Source 1	Source 2	Source 3	Source 4
00	LTC00	LTC01	LTC02	LTC03
01	LTC04	LTC05	LTC06	LTC07
02	LTC08	LTC09	LTC10	LTC11
03	LTC12	LTC13	LTC14	LTC15
04	LTC16	LTC17	LTC18	LTC19
05	LTC20	LTC21	LTC22	LTC23
06	LTC24	LTC25	LTC26	LTC27
07	LTC28	LTC29	LTC30	LTC31
08	LTC32	LTC33	LTC34	LTC35
09	LTC36	LTC37	LTC38	LTC39
10	LTC40	LTC41	LTC42	LTC43
11	LTC44	LTC45	LTC46	LTC47
12	LTC48	LTC49	LTC50	LTC51
13	LTC52	LTC53	LTC54	LTC55
14	LTC56	LTC57	LTC58	LTC59
15	LTC60	LTC61	LTC62	LTC63

General Purpose Timer Array (GPTA)

24.5 LTCA2 Kernel Registers

This section describes the LTCA2 kernel registers. Some of the kernel registers (SRSCn, SRSSn, LTCCTRk, and LTCXRk) are already described as GPTA kernel register at the pages as referenced in column “Description see” of [Table 24-23](#). The multiplexer array FIFO registers are LTCA2 specific and therefore defined on the next pages.

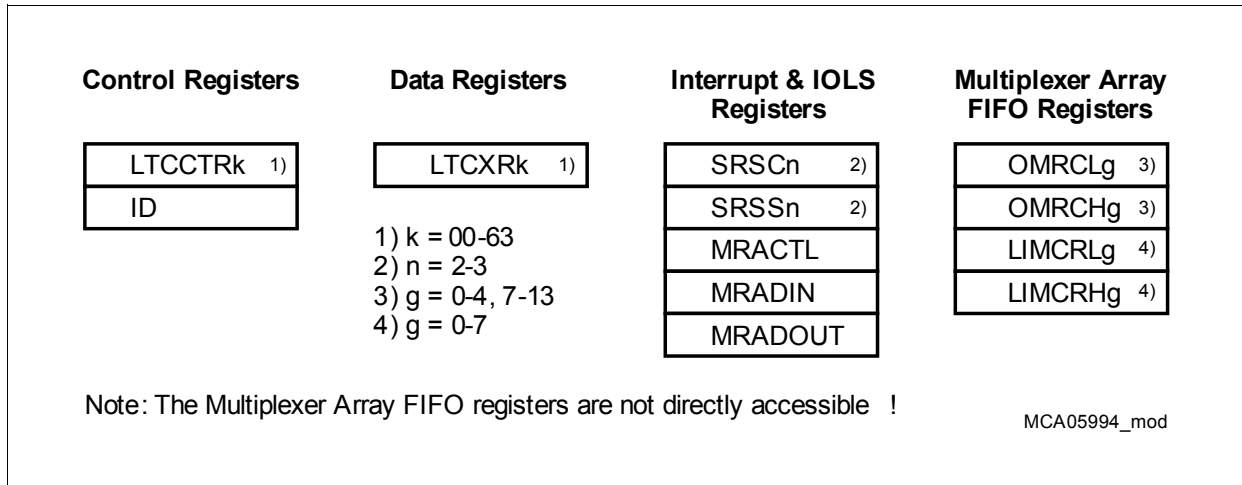


Figure 24-85 LTCA2 Kernel Registers

Table 24-23 LTCA2 Kernel Registers

Register Short Name	Register Long Name	Offset Address	Description see
ID	Module Identification Register	008 _H	Page 24-211
SRSC2	Service Request State Clear Register 2	020 _H	Page 24-211
SRSS2	Service Request State Set Register 2	024 _H	Page 24-212
SRSC3	Service Request State Clear Register 3	028 _H	Page 24-213
SRSS3	Service Request State Set Register 3	02C _H	Page 24-214
MRACTL	Multiplexer Register Array Control Register	038 _H	Page 24-236
MRADIN	Multiplexer Register Array Data In Register	03C _H	Page 24-238
MRADOUT	Multiplexer Register Array Data Out Register	040 _H	Page 24-238
LTCCTRk	Local Timer Cell Control Register k (k = 00-62)	200 _H + k × 8	Page 24-180

General Purpose Timer Array (GPTA)

Table 24-23 LTCA2 Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Address	Description see
LTCXRk	Local Timer Cell X Register k (k = 00-62)	$200_H + k \times 8 + 4$	Page 24-190
LTCCTR63	Local Timer Cell Control Register 63	$3F8_H$	Page 24-188
LTCXR63	Local Timer Cell X Register 63	$3FC_H$	Page 24-190
OMCRLg	Output Multiplexer Control Register for Lower Half of Group g (g = 0-4, 7-13)	not directly addressable	Page 24-239
OMCRLg	Output Multiplexer Control Register for Upper Half of Group g (g = 0-4, 7-13)	see Page 24-229	Page 24-241
LIMCRLg	Input Multiplexer Control Register for Lower Half of LTC Group g (g = 0-7)		Page 24-243
LIMCRLg	Input Multiplexer Control Register for Upper Half of LTC Group g (g = 0-7)		Page 24-245

24.5.1 Bit Protection

Bits with bit protection (this is valid, for example, for all bits in the Service Request State Registers) are not changed during a read-modify-write instruction, that is when hardware sets a request state bit between the read and the write of the read-modify-write sequence. For bit protected bits it is guaranteed that a hardware setting operation always has priority. Thus, no hardware triggered events are lost.

24.5.2 Service Request Registers

See [“SRSC2” on Page 24-211](#).

24.5.3 Local Timer Cell Registers

See [“LTCCTRk \(k = 00-62\)” on Page 24-180](#).

General Purpose Timer Array (GPTA)

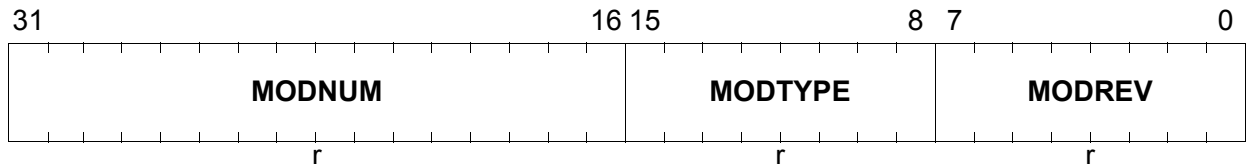
24.5.4 Module Identification Register

LTCA2_ID

LTCA2 Module Identification Register

(008_H)

Reset Value: 002A C0XX_H



Field	Bits	Type	Description
MODREV	[7:0]	r	Module Revision Number MODREV defines the module revision number. The value of a module revision starts with 01 _H (first revision).
MODTYPE	[15:8]	r	Module Type This bit field defines the module as a 32-bit module: C0 _H
MODNUM	[31:16]	r	Module Number Value This bit field defines the module identification number for the LTCA2: 002A _H

General Purpose Timer Array (GPTA)

24.5.5 I/O Sharing Unit Registers

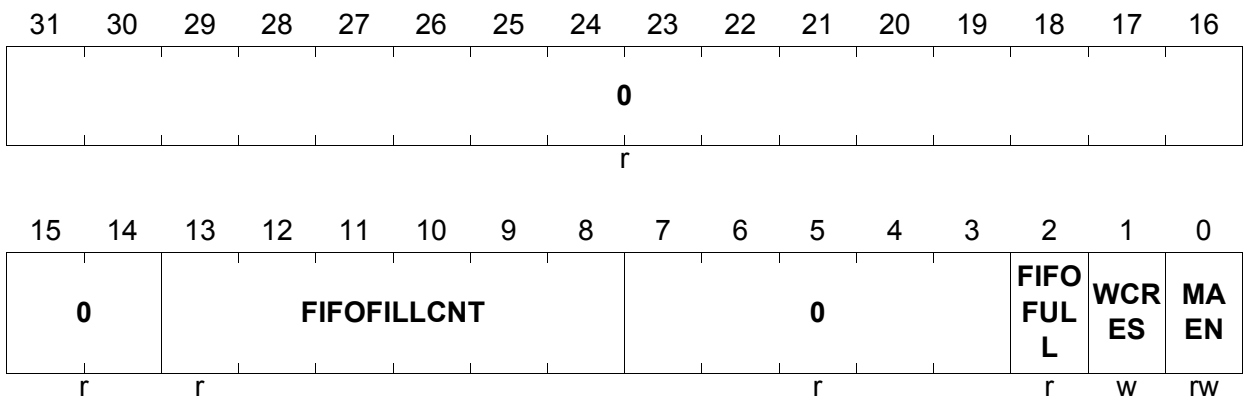
The three registers MRACTL, MRADIN, and MRADOUT are used to write data to and read data from the LTCA2 Multiplexer Register Array FIFO. The Multiplexer Register Array FIFO controls the operation of the Input/Output Line Sharing Unit (see [“Input/Output Line Sharing Unit \(IOLS\)” on Page 24-90](#)).

The Multiplexer Register Array Control register controls the operation of the Multiplexer Register Array FIFO.

MRACTL

Multiplexer Register Array Control Register (038_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
MAEN	0	rw	<p>Multiplexer Array Enable</p> <p>Bit field MAEN enables/disables the programming and the interconnections of the multiplexer array.</p> <p>0_B Multiplexer array is disabled; all cell inputs are driven with 0, LTCA2 I/O lines (pins) are disconnected and FIFO writing is enabled.</p> <p>1_B Multiplexer array is enabled; all cell and I/O line interconnections are established as previously programmed and FIFO writing is disabled.</p>
WCRES	1	w	<p>Write Count Reset</p> <p>Writing WCRES with 1 while the array is disabled (MAEN = 0), resets the write cycle counter to zero and the FIFO written sequentially (initialized). WCRES is always read as 0.</p>

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
FIFOFULL	2	r	FIFO Full Status 0 _B FIFO not completely written (write access to MRADIN allowed). 1 _B FIFO completely written (write access to MRADIN ignored). Must be re-enabled via WCRES before array can be re-initialized.
FIFOFILLCNT	[13:8]	r	FIFO Fill Count This bit field shows the current contents of the write cycle counter.
0	[7:3], [31:14]	r	Reserved Read as 0; should be written with 0.

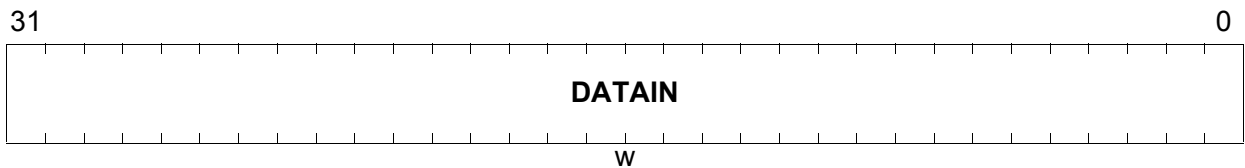
General Purpose Timer Array (GPTA)

The Multiplexer Register Array Data In register is used to **write** data to the Multiplexer Register Array FIFO. The Multiplexer Register Array Data Out register is used to **read** data from the Multiplexer Register Array FIFO.

MRADIN

Multiplexer Register Array Data In Register
(03C_H)

Reset Value: 0000 0000_H

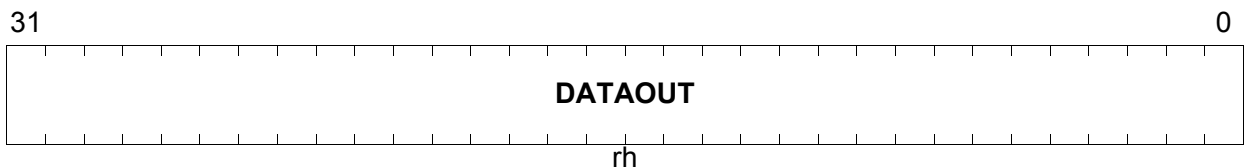


Field	Bits	Type	Description
DATAIN	[31:0]	w	FIFO Write Data This register contains the FIFO write data as defined for the Output Multiplexer Control Registers and the LTC Input Multiplexer Control Registers.

MRADOUT

Multiplexer Register Array Data Out Register
(040_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
DATAOUT	[31:0]	rh	FIFO Read Data This register contains the FIFO read data as assigned for the Output Multiplexer Control Registers and the LTC Input Multiplexer Control Registers.

Note: For correct operation, the MRADIN and MRADIN registers must be always read or written 32-bit wide. 8-bit and 16-bit accesses are ignored without any bus error!

General Purpose Timer Array (GPTA)

24.5.6 Multiplexer Control Registers

Note: These registers are not directly accessible and can be written and read only via the multiplexer register array FIFO.

24.5.6.1 Output Multiplexer Control Registers

Two registers, OMCRL and OMCRH, are assigned to each I/O Group IOG[4:0] and each Output Group OG[6:0]. OMCRL[4:0]/OMCRH[4:0] are assigned to IOG[4:0] and OMCRL[13:7]/OMCRH[13:7] are assigned to OG[6:0].

OMCRL controls the connections of group pins 0 to 3. OMCRH controls the connections of group pins 4 to 7.

OMCRL_g (g = 0-4, 7-13)

Output Multiplexer Control Register for Lower Half of Pin Group g

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	OMG3			0	OML3			0	OMG2			0	OML2		
r	rw			r	rw			r	rw			r	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	OMG1			0	OML1			0	OMG0			0	OML0		
r	rw			r	rw			r	rw			r	rw		

Field	Bits	Type	Description
OML0, OML1, OML2, OML3	[2:0], [10:8], [18:16], [26:24]	rw	Multiplexer Line Selection This bit field selects the input line of a OMG that can be selected by bit field OMG _n for OMG output n. 000 _B 000 _B OMG input IN0 selected 001 _B OMG input IN1 selected 010 _B OMG input IN2 selected 011 _B OMG input IN3 selected 100 _B OMG input IN4 selected 101 _B OMG input IN5 selected 110 _B OMG input IN6 selected 111 _B OMG input IN7 selected

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
OMG0, OMG1, OMG2, OMG3	[6:4], [14:12], [22:20], [30:28]	rw	<p>Multiplexer Group Selection</p> <p>This bit field determines the OMG_ng which is connected to input n of I/O Group g or Output group g-7.</p> <p>XX1_B OMG1g selected XX0_B OMG2g selected</p> <p>All other combinations are reserved. If a reserved combination of OMG_n value is selected, the corresponding OMG output is forced to 0 level. For compatibility reasons, the values 001_B (for XX1_B) and 010_B (for XX0_B) should be used for OMG_n bit field programming.</p>
0	3, 7, 11, 15, 19, 23, 27, 31	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

General Purpose Timer Array (GPTA)

OMCRHg (g = 0-13)

Output Multiplexer Control Register for Upper Half of Pin Group g

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	OMG7			0	OML7			0	OMG6			0	OML6		
r	rw			r	rw			r	rw			r	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	OMG5			0	OML5			0	OMG4			0	OML4		
r	rw			r	rw			r	rw			r	rw		

Field	Bits	Type	Description
OML4, OML5, OML6, OML7	[2:0], [10:8], [18:16], [26:24]	rw	<p>Multiplexer Line Selection</p> <p>This bit field selects the input line of a OMG that can be selected by bit field OMGn for OMG output n.</p> <p>000_B OMG input IN0 selected 001_B OMG input IN1 selected 010_B OMG input IN2 selected 011_B OMG input IN3 selected 100_B OMG input IN4 selected 101_B OMG input IN5 selected 110_B OMG input IN6 selected 111_B OMG input IN7 selected</p>
OMG4, OMG5, OMG6, OMG7	[6:4], [14:12], [22:20], [30:28]	rw	<p>Multiplexer Group Selection</p> <p>This bit field determines the OMGng which is connected to input n of I/O Group g or Output group g-7.</p> <p>XX1_B OMG1g selected XX0_B OMG2g selected</p> <p>All other combinations are reserved. If a reserved combination of OMGn value is selected, the corresponding OMG output is forced to 0 level. For compatibility reasons, the values 001_B (for XX1_B) and 010_B (for XX0_B) should be used for OMGn bit field programming.</p>

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
0	3, 7, 11, 15, 19, 23, 27, 31	r	Reserved Read as 0; should be written with 0.

Note: In the LTCA2 module, registers OMCRL5, OMCRH5, OMCRL6, and OMCRH6 are not available (see also [Figure 24-83](#)).

General Purpose Timer Array (GPTA)

24.5.6.2 LTC Input Multiplexer Control Registers

Two registers, LIMCRL and LIMCRH, are assigned to each LTC group. LIMCRL controls the connections of LTC group cells with index 0 to 3. LIMCRH controls the connections of LTC group cells with index 4 to 7.

LIMCRL_g (g = 0-7)

Input Multiplexer Control Register for Lower Half of LTC Group g

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LIM EN3	LIMG3			0	LIML3			LIM EN2	LIMG2			0	LIML2		
r	rw			r	rw			r	rw			r	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LIM EN1	LIMG1			0	LIML1			LIM EN0	LIMG0			0	LIML0		
r	rw			r	rw			r	rw			r	rw		

Field	Bits	Type	Description
LIML0, LIML1, LIML2, LIML3	[2:0], [10:8], [18:16], [26:24]	rw	<p>Multiplexer Line Selection</p> <p>This bit field selects the input line of a LIMG that can be selected by bit field LIMG_n for LIMG output n.</p> <p>000_B LIMG input IN0 selected 001_B LIMG input IN1 selected 010_B LIMG input IN2 selected 011_B LIMG input IN3 selected 100_B LIMG input IN4 selected 101_B LIMG input IN5 selected 110_B LIMG input IN6 selected 111_B LIMG input IN7 selected</p>

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
LIMG0, LIMG1, LIMG2, LIMG3	[6:4], [14:12], [22:20], [30:28]	rw	<p>Multiplexer Group Selection</p> <p>This bit field determines the LIMG_ng which is connected to input n of LTC group g.</p> <p>000_B LIMG0g selected 001_B LIMG1g selected 011_B LIMG3g selected 100_B LIMG4g selected</p> <p>All other combinations are reserved. If a reserved combination of LIMG_n is selected, or if LIMEN_n = 0, the corresponding LIMG output is forced to 0 level.</p>
LIMEN0, LIMEN1, LIMEN2, LIMEN3	7, 15, 23, 31	rw	<p>Enable Multiplexer Connection</p> <p>0_B Input n is not connected to any line. 1_B Input n is connected to the line defined by LIML_n and LIMG_n.</p>
0	3, 11, 19, 27	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

General Purpose Timer Array (GPTA)

LIMCRHg (g = 0-7)

Input Multiplexer Control Register for Upper Half of LTC Group g

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LIM EN7	LIMG7			0	LIML7			LIM EN6	LIMG6			0	LIML6		
r	rw			r	rw			r	rw			r	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LIM EN5	LIMG5			0	LIML5			LIM EN4	LIMG4			0	LIML4		
r	rw			r	rw			r	rw			r	rw		

Field	Bits	Type	Description
LIML4, LIML5, LIML6, LIML7	[2:0], [10:8], [18:16], [26:24]	rw	<p>Multiplexer Line Selection</p> <p>This bit field selects the input line of a LIMG that can be selected by bit field LIMGn for LIMG output n.</p> <p>000_B LIMG input IN0 selected 001_B LIMG input IN1 selected 010_B LIMG input IN2 selected 011_B LIMG input IN3 selected 100_B LIMG input IN4 selected 101_B LIMG input IN5 selected 110_B LIMG input IN6 selected 111_B LIMG input IN7 selected</p>
LIMG4, LIMG5, LIMG6, LIMG7	[6:4], [14:12], [22:20], [30:28]	rw	<p>Multiplexer Group Selection</p> <p>This bit field determines the LIMGng which is connected to input n of LTC group g.</p> <p>000_B LIMG0g selected 001_B LIMG1g selected 011_B LIMG3g selected 100_B LIMG4g selected</p> <p>All other combinations are reserved. If a reserved combination of LIMGn is selected, or if LIMENn = 0, the corresponding LIMG output is forced to 0 level.</p>

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
LIMEN4, LIMEN5, LIMEN6, LIMEN7	7, 15, 23, 31	rw	Enable Multiplexer Connection 0 _B Input n is not connected to any line. 1 _B Input n is connected to the line defined by LIMLn and LIMGn.
0	3, 11, 19, 27	r	Reserved Read as 0; should be written with 0.

24.6 GPTA Module Implementation

This section describes the GPTA interfaces as implemented in TC1796 with the clock control, port and Micro Second Channel connections, interrupt control, and address decoding.

24.6.1 Interconnections of the GPTA0/GPTA1/LTCA2 Modules

The following items are described in this section:

- GPTA module (kernel) external registers
- Port control and connections
 - I/O port line assignment
 - I/O function selection
 - Pad driver characteristics selection
 - Emergency control of GPTA outputs
- On-chip connections
 - Clock bus connections
 - MSC controller connections
 - FADC connections
 - MultiCAN, SCU, and DMA connections
 - SCU connections (ADC, DMA)
- Module clock generation
- Interrupt registers
- GPTA address map

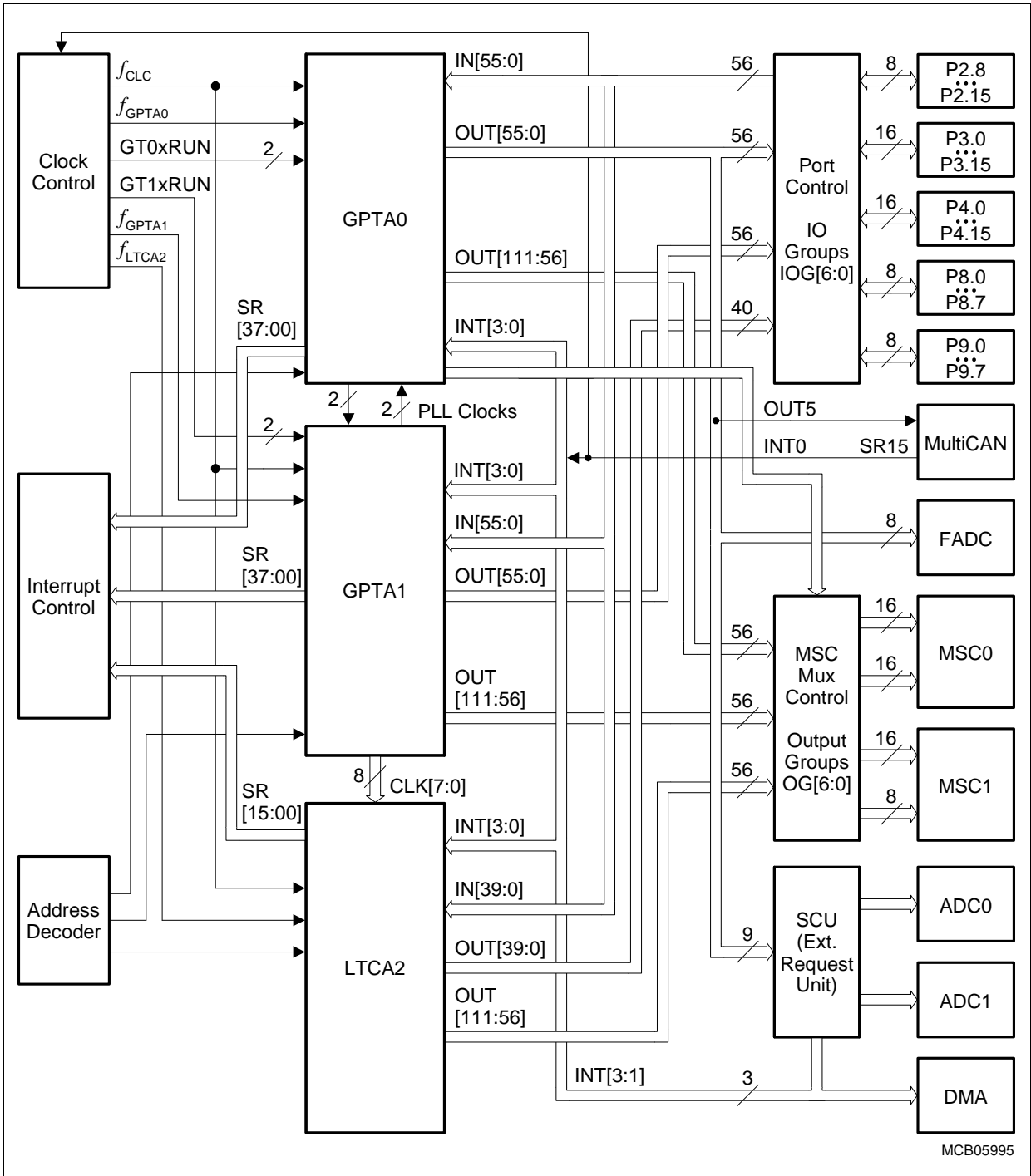
Figure 24-86 shows the TC1796 specific implementation details and interconnections of the modules GPTA0, GPTA1 and LTCA2. The modules are supplied by clock control and address decoding logic.

Each GPTA0/1 module has 56 input signals and 112 output signals which can be connected to 56 port pins and 56 MSC interface lines. Additional four inputs for internal connections (coming from the SCU) and 38 service request outputs are provided.

The LTCA2 module has 40 input signals and 96 output signals that can be connected to the same 56 port pins as GPTA0 and GPTA1 and 56 MSC interface lines. Additional four inputs for internal connections (coming from the SCU) and 16 service request outputs are provided.

Additional connections are described in the following sections.

General Purpose Timer Array (GPTA)



MCB05995

Figure 24-86 Block Diagram of GPTA Implementation

General Purpose Timer Array (GPTA)

24.6.2 GPTA Module External Registers

Figure 24-87 summarizes the GPTA module related external registers that are required for GPTA0, GPTA1, and LTCA2 programming. These registers are referenced and (some of it) described in detail in the following sub-sections.

Clock Control Registers	Port Control Registers	MSC Multiplexer Control Registers	Interrupt Registers
GPTA0_CLC	P2_IOCRk 1)	GPTA0_MMXCTR00	GPTA0_SRCK 4)
GPTA0_FDR	P3_IOCRk 2)	GPTA0_MMXCTR01	GPTA1_SRCK 4)
GPTA0_EDCTR	P4_IOCRk 2)	GPTA0_MMXCTR10	LTCA2_SRCK 5)
GPTA0_DBGCTR	P8_IOCRk 3)	GPTA0_MMXCTR11	
	P9_IOCRk 3)		
	P2_PDR		
	P3_PDR		
	P4_PDR		
	P8_PDR		
	P9_PDR		
	P2_ESR		
	P3_ESR		
	P4_ESR		
	P8_ESR		
	P9_ESR		

1) k = 8, 12
 2) k = 0, 4, 8, 12
 3) k = 0, 4
 4) k = 00-37
 5) k = 00-15

MCA05996

Figure 24-87 GPTA Implementation-Specific Special Function Registers

General Purpose Timer Array (GPTA)

24.6.3 Port Control and Connections

This sections describes the I/O control and selections of the GPTA0/GPTA1/LTCA2 modules.

24.6.3.1 I/O Group to Port Line Assignment

In the TC1796, the seven I/O groups of GPTA0 and GPTA1 with their input lines IN[55:0] and output lines OUT[55:0] are assigned to seven 8-bit port groups as shown in **Figure 24-88**. Within an 8-bit I/O group, the IN/OUT line with lowest index number is assigned to the port line with the lowest index number. The remaining lines are assigned linearly with increasing index numbers. For example, P3.5 is assigned to IN13/OUT13.

In the LTCA2, only 40 input/output lines are available. Therefore, the LTCA2 does not have IOG5 and IOG6 and therefore it has no connections to Port 8 or Port 9 pins.

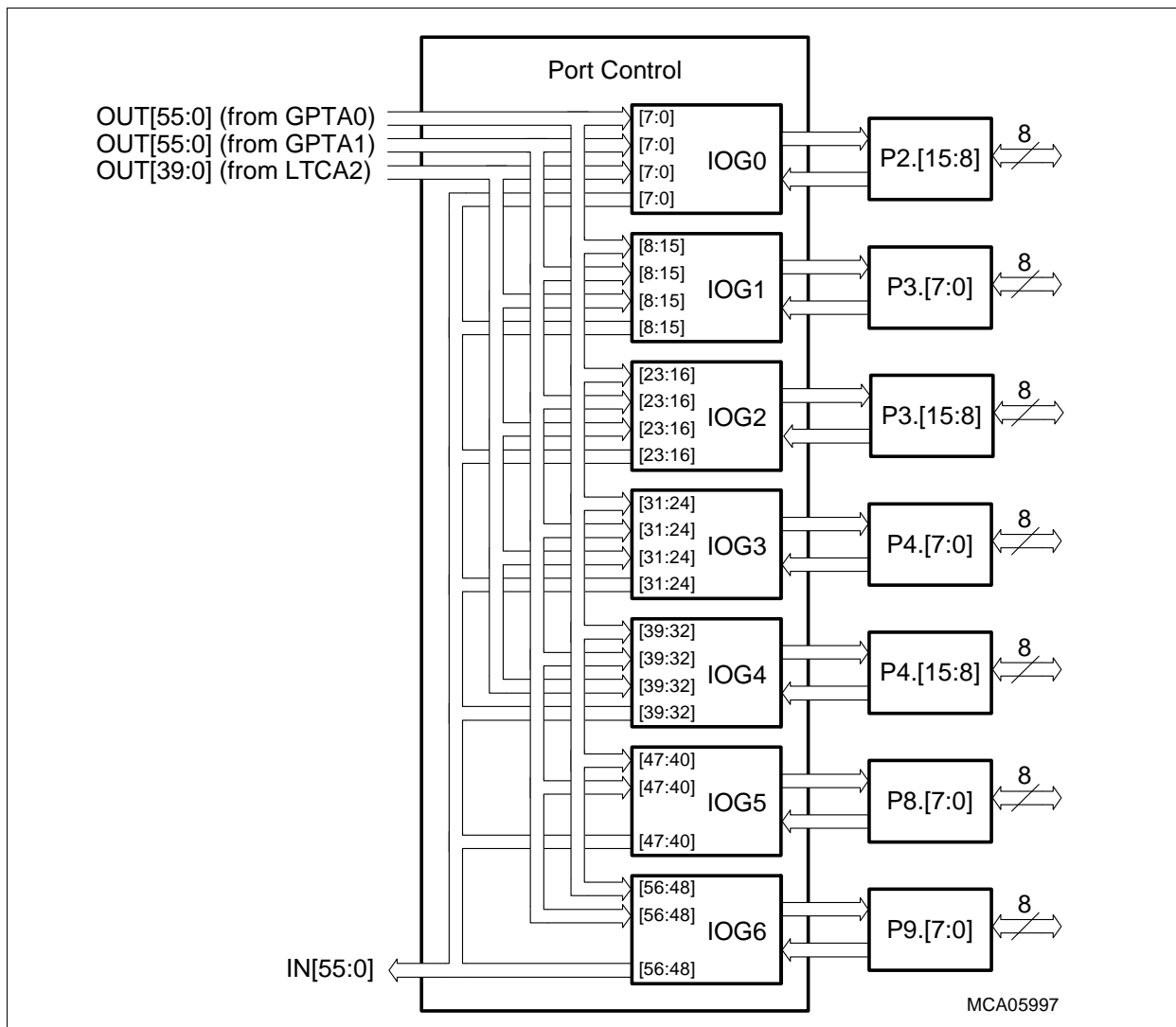


Figure 24-88 I/O Group to Port Assignment

General Purpose Timer Array (GPTA)

The interconnections between the GPTA0/GPTA1/LTCA2 modules and the port I/O lines are controlled in the port logics. The following port control operations selections must be executed:

- Input/output function selection (IOCR registers)
- Pad driver characteristics selection for outputs (PDR registers)

24.6.3.2 Input/Output Function Selection

The port input/output control registers contain bit fields that select the digital output and input driver characteristics such as port direction (input/output), pull-up/down device or open-drain selection for outputs, and alternate output selections. The I/O lines for the GPTA0/GPTA1/LTCA2 modules are controlled by the port input/output control registers for Port 2, Port 3, Port 4, Port 8, and Port 9. Each of the input/output control registers controls four port lines using a 4-bit wide bit field PCx (definitions see [Table 24-25](#)). [Table 24-24](#) shows which of the input/output control register bit field is related to a specific GPTA0/GPTA1/LTCA2 module I/O line. Note that input P2.9/IN1 has special connections (see [Page 24-263](#)).

Table 24-24 IOCR Assignment for GPTA Port Lines

Port	Port Lines for GPTA	GPTA I/O Lines		Controlled by IOCR Register
		Input	Output	
Port 2	P2.[11:8]	IN[3:0] ¹⁾	OUT[3:0]	P2_IOCR8
	P2.[15:12]	IN[7:4]	OUT[7:4]	P2_IOCR12
Port 3	P3.[3:0]	IN[11:8]	OUT[11:8]	P3_IOCR0
	P3.[7:4]	IN[15:12]	OUT[15:12]	P3_IOCR4
	P3.[11:8]	IN[19:16]	OUT[19:16]	P3_IOCR8
	P3.[15:12]	IN[23:20]	OUT[23:20]	P3_IOCR12
Port 4	P4.[3:0]	IN[27:24]	OUT[27:24]	P4_IOCR0
	P4.[7:4]	IN[31:28]	OUT[31:28]	P4_IOCR4
	P4.[11:8]	IN[35:32]	OUT[35:32]	P4_IOCR8
	P4.[15:12]	IN[39:36]	OUT[39:36]	P4_IOCR12
Port 8	P8.[3:0]	IN[43:40]	OUT[43:40]	P8_IOCR0
	P8.[7:4]	IN[47:44]	OUT[47:44]	P8_IOCR4
Port 9	P9.[3:0]	IN[51:48]	OUT[51:48]	P9_IOCR0
	P9.[7:4]	IN[55:52]	OUT[55:52]	P9_IOCR4

1) There is a special connection provided for GPTA input line IN1 (see [Page 24-263](#)).

General Purpose Timer Array (GPTA)

Bit field PCx (x is the number of a port line) in the input/output control register IOCRy (y is the number of the first port line controlled by an IOCR) must be programmed according [Table 24-25](#).

Table 24-25 PCx Coding

PCx[3:0]	I/O	Output Characteristics	Selected Pull-up/Pull-down/ Selected Output Function	Comment
0X00 _B	Input	–	No pull device connected	–
0X01 _B			Pull-down device connected	
0X10 _B ¹⁾			Pull-up device connected	
0X11 _B			No pull device connected	
1000 _B	Output	Push-pull	General purpose output selected	
1001 _B			GPTA0 output (ALT1) selected	
1010 _B			GPTA1 output (ALT2) selected	
1011 _B			LTCA2 output (ALT3) selected ²⁾	
1100 _B		Open-drain	General purpose output selected	
1101 _B			GPTA0 output (ALT1) selected	
1110 _B			GPTA1 output (ALT2) selected	
1111 _B			LTCA2 output (ALT3) selected ²⁾	

1) This bit field value is default after reset.

2) Not applicable at Port 8 and Port 9 for LTCA2.

A port line that is programmed as input can be used by the GPTA0, GPTA1, or LTCA2 module simultaneously as input.

Port lines selected as GPTA0, GPTA1, or LTCA2 output are forced to a 0 level if the related multiplexer array in the I/O Line Sharing Unit is disabled or if a reserved combination of an OMGN value is selected. Therefore, no glitches and spikes can occur during the programming of the related multiplexer array.

General Purpose Timer Array (GPTA)

24.6.3.3 Pad Driver Characteristics Selection

The output driver strength and the slew rate of GPTA output lines is controlled by 3-bit wide PDX bit fields located in the pad driver mode registers PDR. These bit fields determine the driver strength and the slew rate for a group of port output lines. [Table 24-26](#) shows which of the pad driver register bit field is related to a specific GPTA0/GPTA1/LTCA2 module I/O line group.

Table 24-26 PDR Assignment for GPTA Port Lines

Port	Pad Class	PDR Register PDx Bit Field	Controlled Port Lines	Related GPTA Output Lines
Port 2	A1	P2_PDR.PD1	P2.[15:8]	OUT[7:0]
Port 3	A1	P3_PDR.PD0	P3.[7:0]	OUT[15:8]
		P3_PDR.PD1	P3.[15:8]	OUT[23:16]
Port 4	A2	P4_PDR.PD0	P4.[7:0]	OUT[31:24]
	A1	P4_PDR.PD1	P4.[15:8]	OUT[39:32]
Port 8	A1	P8_PDR.PD0	P8.1, P8.4, P8.6, P8.7	OUT41, OUT44, OUT46, OUT47
	A2	P8_PDR.MLI1	P8.0, P8.2, P8.3, P8.5	OUT40, OUT42, OUT43, OUT45
Port 9	A2	P9_PDR.PDMSC0	P9.[7:4]	OUT[55:52]
		P9_PDR.PDMSC1	P9.[3:0]	OUT[51:48]

PDx Selection Table

Table 24-27 Pad Driver Mode Mode Selection (Class A1/A2 Pads)

Pad Class	PDx Bit Field	Driver Strength	Signal Transitions
A1	XX0 _B	Medium driver	–
	XX1 _B	Weak driver	–

General Purpose Timer Array (GPTA)

Table 24-27 Pad Driver Mode Mode Selection (Class A1/A2 Pads) (cont'd)

Pad Class	PDx Bit Field	Driver Strength	Signal Transitions
A2	000 _B	Strong driver	Sharp edge ¹⁾
	001 _B		Medium edge ¹⁾
	010 _B		Soft edge ¹⁾
	011 _B	Weak driver	–
	100 _B	Medium driver	Sharp edge
	101 _B		Medium edge
	110 _B		Soft edge
	111 _B	Weak driver	–

1) In strong driver mode, the output driver characteristics of class A2 pads can be additionally controlled by the temperature compensation logic.

General Purpose Timer Array (GPTA)

24.6.3.4 Emergency Control of GPTA Output Ports Lines

Port lines connected to GPTA output pins can be selectively switched into an Emergency Mode. In this mode, GPTA output pins react immediately to an active input signal P10.1 (HWCFG1) and drive a logic level that has been programmed in the port output register. As a result, in Emergency Mode a GPTA output pin drives a predefined value instead of the corresponding logic level that is provided on the related GPTA module output line.

All GPTA pins at Port 2, Port 3, Port 4, Port 8, and Port 9 are connected to one common emergency stop signal that is generated in the System Control Unit of the TC1796. More details about the generation of this emergency stop signal are described in the “System Control Unit” chapter of the TC1796 System Units User’s Manual.

The emergency stop signal always controls 8-bit groups of port lines. The enable function is controlled for each pin by bits EN_y (y = number of port line) which are located in the Px_ESR (x = port number) registers. When the emergency stop signal generated in the SCU becomes active and bit Px_ESR.EN_y set, output line Px.y is set to the value of register Px_OUT.Py (emergency enabled). Output Px.y is not affected by the emergency stop signal when bit Px_OUT.Py is reset (emergency disabled).

When the emergency stop signal is released, Pin x.y is switched back to the previously selected GPTA output function without reprogramming the related port registers.

The emergency stop enable bits EN_y are implemented only for output pins at Port 2, Port 3, Port 4, Port 8, and Port 9 that can be connected to the GPTA module.

Table 24-28 Emergency Control for GPTA Port Output Lines

Port	ESR Register	ESR Enable Bits	GPTA Output Lines
Port 2	P2_ESR	EN[15:8]	OUT[7:0]
Port 3	P3_ESR	EN[15:0]	OUT[23:8]
Port 4	P4_ESR	EN[15:0]	OUT[39:24]
Port 8	P8_ESR	EN[7:0]	OUT[47:40]
Port 9	P9_ESR	EN[7:0]	OUT[55:48]

General Purpose Timer Array (GPTA)

24.6.4 On-Chip Connections

This section describes all on-chip interconnections of the GPTA0, GPTA1 and LTCA2 modules except the connections to I/O ports (see [Section 24.6.3](#)).

24.6.4.1 Clock Bus Connections

The clock bus signals generated in the GPTA0 and GPTA1 clock distribution unit and PLL clocks are interconnected between the GPTA0, GPTA1 and LTCA2 modules as shown in [Figure 24-89](#).

The GPTA0 clock bus drives Global Timers (GTs), Global Timer Cells (GTCs), and Local Timer Cells (LTCs) of the GPTA0 module only. The GPTA1 clock bus drives its GTs, GTCs, and LTCs **and** the LTCs of the LTCA2 module.

Each GPTA0 and GPTA1 clock distribution unit has two pairs for PLL clock inputs, an input pair for the local PLL clocks and an input pair for module-external PLL clocks. The GPTA0 and GPTA1 modules are capable to use the PLL clocks from each other.

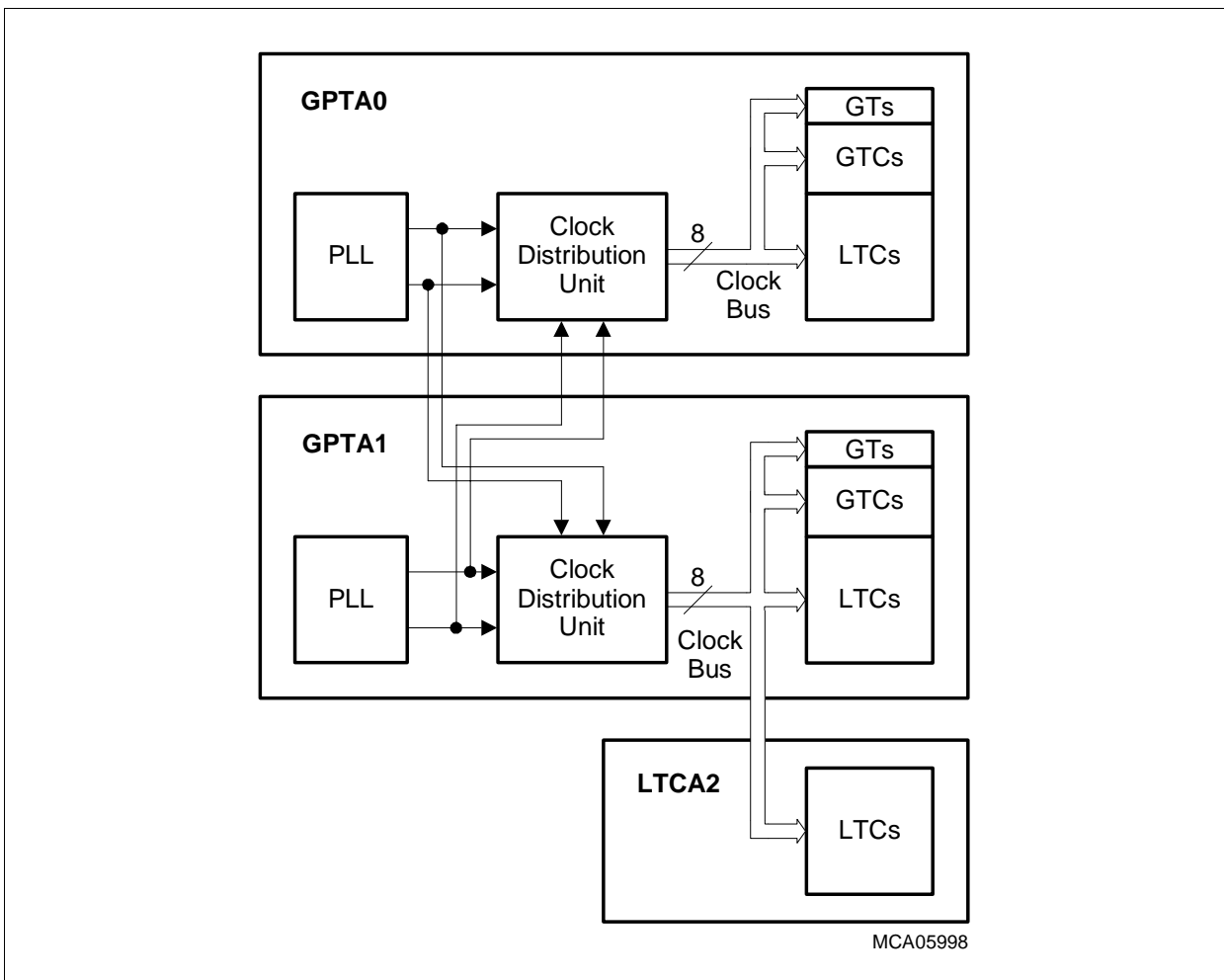


Figure 24-89 Clock Bus Connections of GPTA0/GPTA1/LTCA2

General Purpose Timer Array (GPTA)

24.6.4.2 MSC Controller Connections

The MSC interfaces (MSC0 and MSC1) provide a serial communication link typically used to connect power switches or other peripheral devices. Each output multiplexers of GPTA0, GPTA1, and LTCA2 generates $7 \times 8 = 56$ output lines $OUT[111:56]$ grouped into seven output groups. These 56 output lines $OUT[111:56]$ are wired to 32 MSC0 inputs and 24 MSC1 inputs. **Figure 24-90** shows the interconnections among the MSC0/1 modules and the three GPTA modules. The source for each line of the ALTINL/ALTINH MSC input buses can be selected via a multiplexer connected either to GPTA0 output x, GPTA1 output x, or LTCA2 output OUT_x ($x = 56-111$).

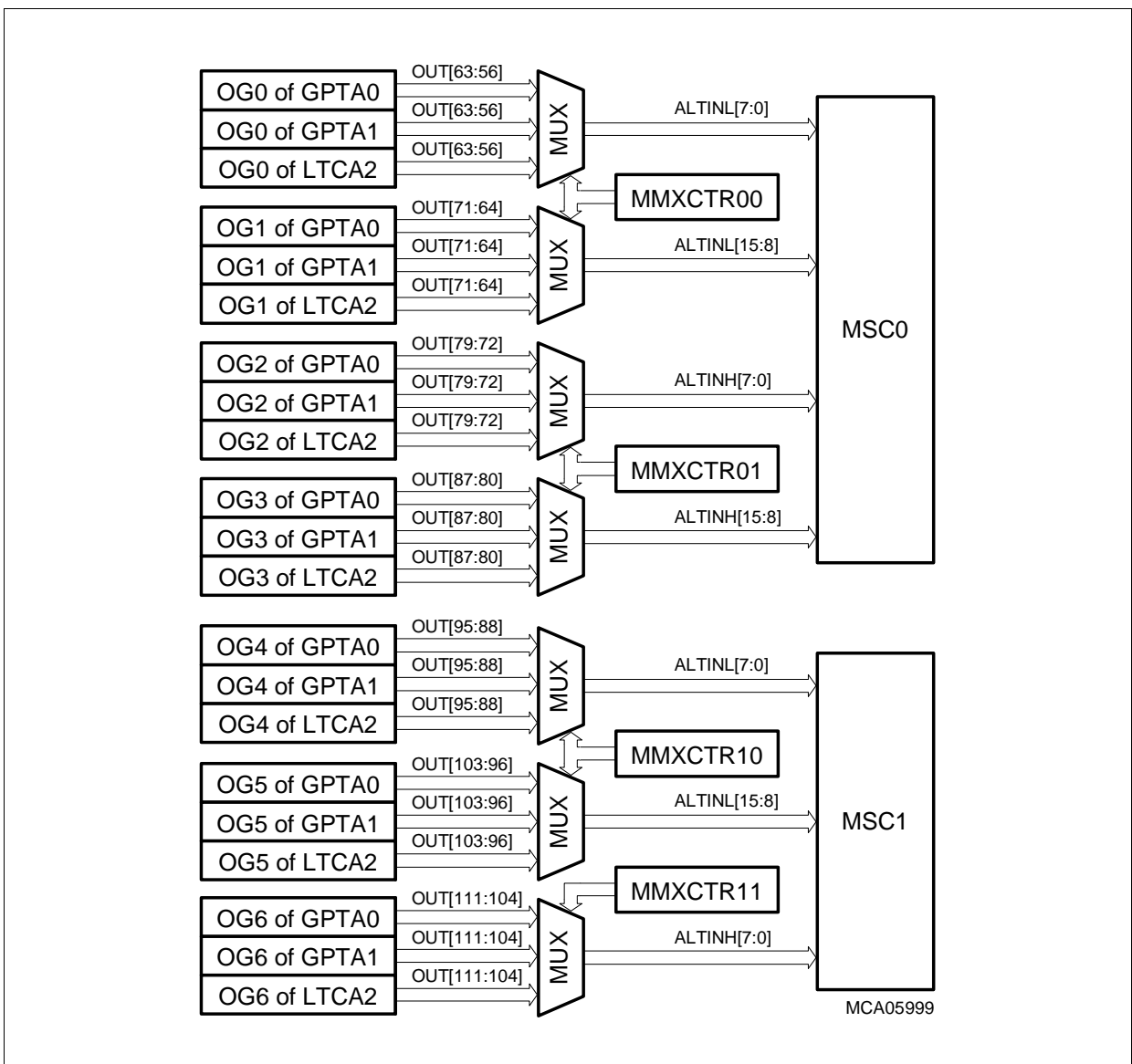


Figure 24-90 GPTA-to-MSC Multiplexer

General Purpose Timer Array (GPTA)

The multiplexer selection is controlled by four multiplexer control registers that are logically assigned to the GPTA0 module address range (but described in this section). Note that eight ALTINH inputs (ALTINH[15:8]) of the MSC1 module are not connected.

Table 24-29 shows the GPTA-to-MSC interconnection assignment.

Note: Table 24-29 also shows the assignment of the GPTA module's seven OGx output group lines OGx.y to the output signals OUT[111:56].

Table 24-29 GPTA0/GPTA1/LTCA2 to MSC Interconnection Assignment

MSC0 Input Line	Assigned GPTA0/GPTA1/LTCA2 Output Line	MSC1 Input Line	Assigned GPTA0/GPTA1/LTCA2 Output Line
ALTINL.0	OUT56 / OG0.0	ALTINL.0	OUT88 / OG4.0
ALTINL.1	OUT57 / OG0.1	ALTINL.1	OUT89 / OG4.1
ALTINL.2	OUT58 / OG0.2	ALTINL.2	OUT90 / OG4.2
ALTINL.3	OUT59 / OG7.3	ALTINL.3	OUT91 / OG4.3
ALTINL.4	OUT60 / OG0.4	ALTINL.4	OUT92 / OG4.4
ALTINL.5	OUT61 / OG0.5	ALTINL.5	OUT93 / OG4.5
ALTINL.6	OUT62 / OG0.6	ALTINL.6	OUT94 / OG4.6
ALTINL.7	OUT63 / OG0.7	ALTINL.7	OUT95 / OG4.7
ALTINL.8	OUT64 / OG1.0	ALTINL.8	OUT96 / OG5.0
ALTINL.9	OUT65 / OG1.1	ALTINL.9	OUT97 / OG5.1
ALTINL.10	OUT66 / OG1.2	ALTINL.10	OUT98 / OG5.2
ALTINL.11	OUT67 / OG1.3	ALTINL.11	OUT99 / OG5.3
ALTINL.12	OUT68 / OG1.4	ALTINL.12	OUT100 / OG5.4
ALTINL.13	OUT69 / OG1.5	ALTINL.13	OUT101 / OG5.5
ALTINL.14	OUT70 / OG1.6	ALTINL.14	OUT102 / OG5.6
ALTINL.15	OUT71 / OG1.7	ALTINL.15	OUT103 / OG5.7
ALTINH.0	OUT72 / OG2.0	ALTINH.0	OUT104 / OG6.0
ALTINH.1	OUT73 / OG2.1	ALTINH.1	OUT105 / OG6.1
ALTINH.2	OUT74 / OG2.2	ALTINH.2	OUT106 / OG6.2
ALTINH.3	OUT75 / OG2.3	ALTINH.3	OUT107 / OG6.3
ALTINH.4	OUT76 / OG2.4	ALTINH.4	OUT108 / OG6.4
ALTINH.5	OUT77 / OG2.5	ALTINH.5	OUT109 / OG6.5
ALTINH.6	OUT78 / OG2.6	ALTINH.6	OUT110 / OG6.6

General Purpose Timer Array (GPTA)

Table 24-29 GPTA0/GPTA1/LTCA2 to MSC Interconnection Assignment (cont'd)

MSC0 Input Line	Assigned GPTA0/GPTA1/LTCA2 Output Line	MSC1 Input Line	Assigned GPTA0/GPTA1/LTCA2 Output Line
ALTINH.7	OUT79 / OG2.7	ALTINH.7	OUT111 / OG6.7
ALTINH.8	OUT80 / OG3.0	–	–
ALTINH.9	OUT81 / OG3.1	–	–
ALTINH.10	OUT82 / OG3.2	–	–
ALTINH.11	OUT83 / OG3.3	–	–
ALTINH.12	OUT84 / OG3.4	–	–
ALTINH.13	OUT85 / OG3.5	–	–
ALTINH.14	OUT86 / OG3.6	–	–
ALTINH.15	OUT87 / OG3.7	–	–

GPTA-to-MSC Multiplexer Control Registers

Four registers are required for GPTA-to-MSC multiplexer control:

- **GPTA0_MMXCTR00** controls the interconnections of GPTA0, GPTA1 and LTCA2 to the MSC0 ALTINL[15:0] inputs.
- **GPTA0_MMXCTR01** controls the interconnections of GPTA0, GPTA1 and LTCA2 to the MSC0 ALTINH[15:0] inputs.
- **GPTA0_MMXCTR10** controls the interconnections of GPTA0, GPTA1 and LTCA2 to the MSC1 ALTINL[15:0] inputs.
- **GPTA0_MMXCTR11** controls the interconnections of GPTA0, GPTA1 and LTCA2 to the MSC1 ALTINH[7:0] inputs.

For each of the ALTINL/ALTINH inputs of MSC0 and MSC1, the 2-bit bit fields in these registers determine which module (GPTA0, GPTA1 or LTCA2) output is selected.

General Purpose Timer Array (GPTA)

GPTA0_MMXCTR00

GPTA-to-MSC Multiplexer Control Register 00
(700_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Field	Bits	Type	Description
MUXn (n = 0-15)	[2*n+1:2*n]	rW	<p>Multiplexer Control for MSC0 Inputs ALTINL.n</p> <p>00_B GPTA0 output OUT[56+n] selected and connected to ALTINL.n</p> <p>01_B GPTA1 output OUT[56+n] selected and connected to ALTINL.n</p> <p>10_B LTCA2 output OUT[56+n] selected and connected to ALTINL.n</p> <p>11_B Reserved</p>

GPTA0_MMXCTR01

GPTA-to-MSC Multiplexer Control Register 01
(704_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	MUX	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Field	Bits	Type	Description
MUXn (n = 0-15)	[2*n+1:2*n]	rW	<p>Multiplexer Control for MSC0 Inputs ALTINH.n</p> <p>00_B GPTA0 output OUT[72+n] selected and connected to ALTIN1.n</p> <p>01_B GPTA1 output OUT[72+n] selected and connected to ALTIN1.n</p> <p>10_B LTCA2 output OUT[72+n] selected and connected to ALTIN1.n</p> <p>11_B Reserved</p>

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
0	[31:16]	r	Reserved Read as 0; should be written with 0.

General Purpose Timer Array (GPTA)

24.6.4.3 GPTA Connections with SCU, MultiCAN, FADC, DMA, Ports

The GPTA0, GPTA1, and LTCA2 sub-modules of the TC1796 GPTA module have several on-chip interconnections with the SCU, MultiCAN, FADC, DMA modules.

Figure 24-91 shows these interconnections.

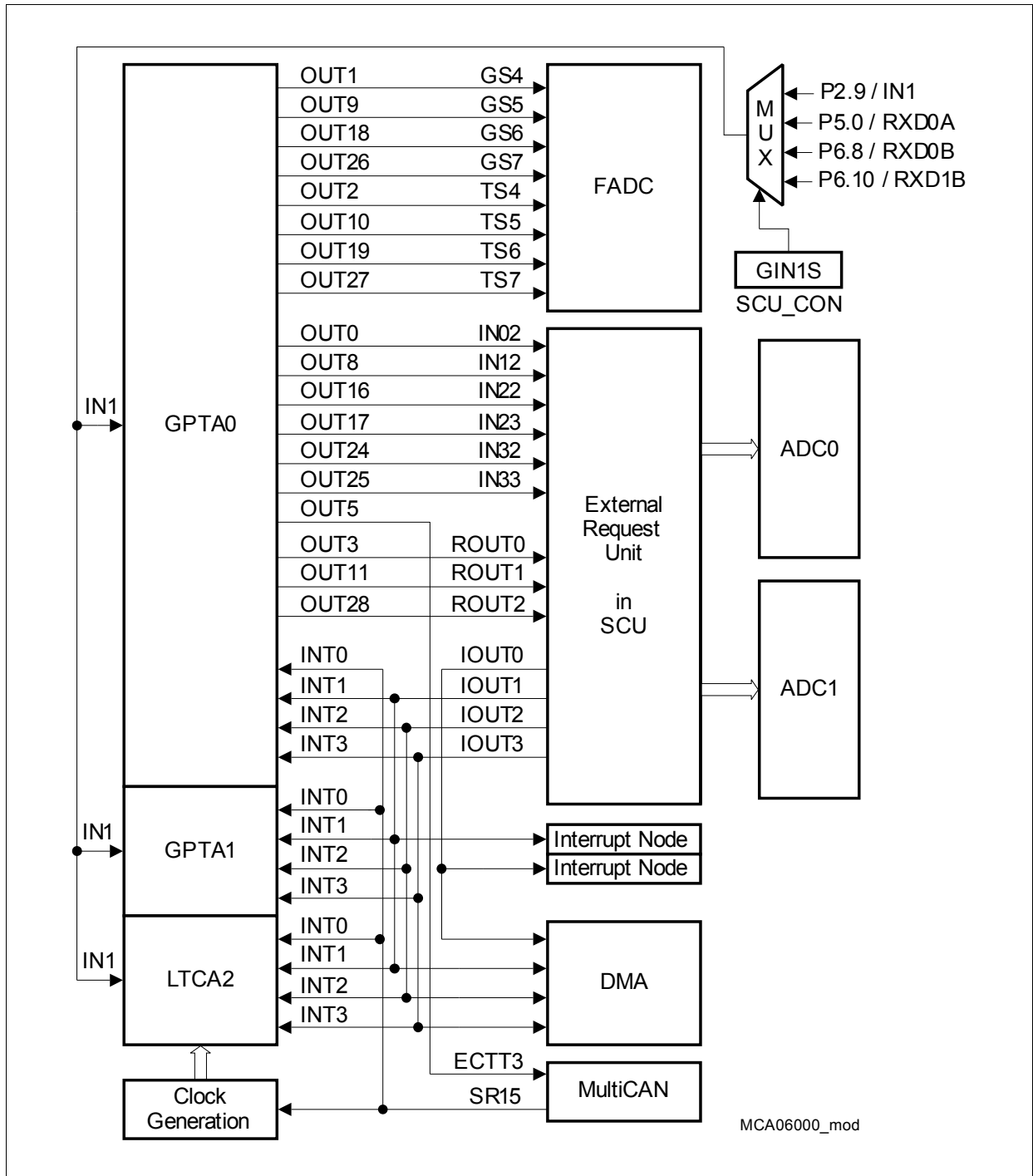


Figure 24-91 Connections of GPTA with On-Chip Modules

General Purpose Timer Array (GPTA)

System Control Unit

The SCU contains the external request unit (ERU), which is especially responsible for controlling requests coming from the MSC modules, port pins, or from the GPTA0 module and passing these request to AD converters, DMA controller, or interrupt nodes.

MultiCAN Connections

The MultiCAN controller has two connections to the GPTA modules:

- MultiCAN service request output SR15 is connected to the INT0 input of GPTA0, GPTA1, and LTCA2.
- GPTA0 output line OUT5 is connected to the external time trigger input ECTT3 of the MultiCAN module.

DMA Controller

The external request unit generates four DMA request output signals (IOUT[3:0]) that can be activated via port pins, the MSC clock outputs, or the six GPTA output lines. Three of these four DMA request output signals are connected to the GPTA0/GPTA1/LTCA2 internal inputs INT[3:1]. These connections allow, for example, GTC or LTC events in the GPTA modules to be triggered by a request coming from a port pin or from the MSC clock.

FADC Connections

As shown in [Figure 24-91](#), eight GPTA0 output lines are connected as trigger input signals or gating input signals to the channel trigger logic of the FADC. Thus dedicated GPTA0 outputs can generate trigger events or act as gating signals for FADC channels.

Port Connections of Input IN1

The input lines IN1 of the GPTA0/GPTA1/LTCA2 modules are connected to the output of a 4-to-1 multiplexer. This multiplexer is controlled by bit field SCU_CON.GIN1S and allows the GPTA input lines IN1 to be connected to one out of four port input lines. This feature especially allows the baud rates of an ASC0 or ASC1 receiver input signal to be measured by timers of the GPTA.

Table 24-30 GPTA Input Line IN1 Connections

SCU_CON.GIN1S	GPTA1 Input IN1 Connected to
00 _B	P2.9 / IN1 (default after reset)
01 _B	P5.0 / RXD0A
10 _B	P6.8 / RXD0B
11 _B	P6.10 / RXD1B

General Purpose Timer Array (GPTA)

24.6.5 Module Clock Generation

As shown in **Figure 24-92**, the clock signals for the GPTA0, GPTA1, and LTCA2 modules are generated and controlled by one clock generation unit. This clock generation unit is responsible for the enable/disable control, the clock frequency adjustment, and the debug clock control. The unit includes four registers:

- **Clock Control Register GPTA0_CLC** (see [Page 24-267](#)), responsible for the generation of the control clock f_{CLC} that is used by each of the three kernel modules as general control tasks.
- **Fractional Divider Register GPTA0_FDR** (see [Page 24-268](#)), responsible for the frequency control of the module timer clock f_{GPTA} .
- **Clock Enable/Disable Control Register GPTA0_EDCTR** (see [Page 24-270](#)), responsible for the enable/disable control of the three module timer clocks f_{GPTA0} , f_{GPTA1} , and f_{LTCA2} , and for the run control for the Global Timers in GPTA0 and GPTA1.
- **Debug Clock Control Register GPTA0_DBGCTR** (see [Page 24-271](#)), responsible for the module timer clock control in Debug Mode.

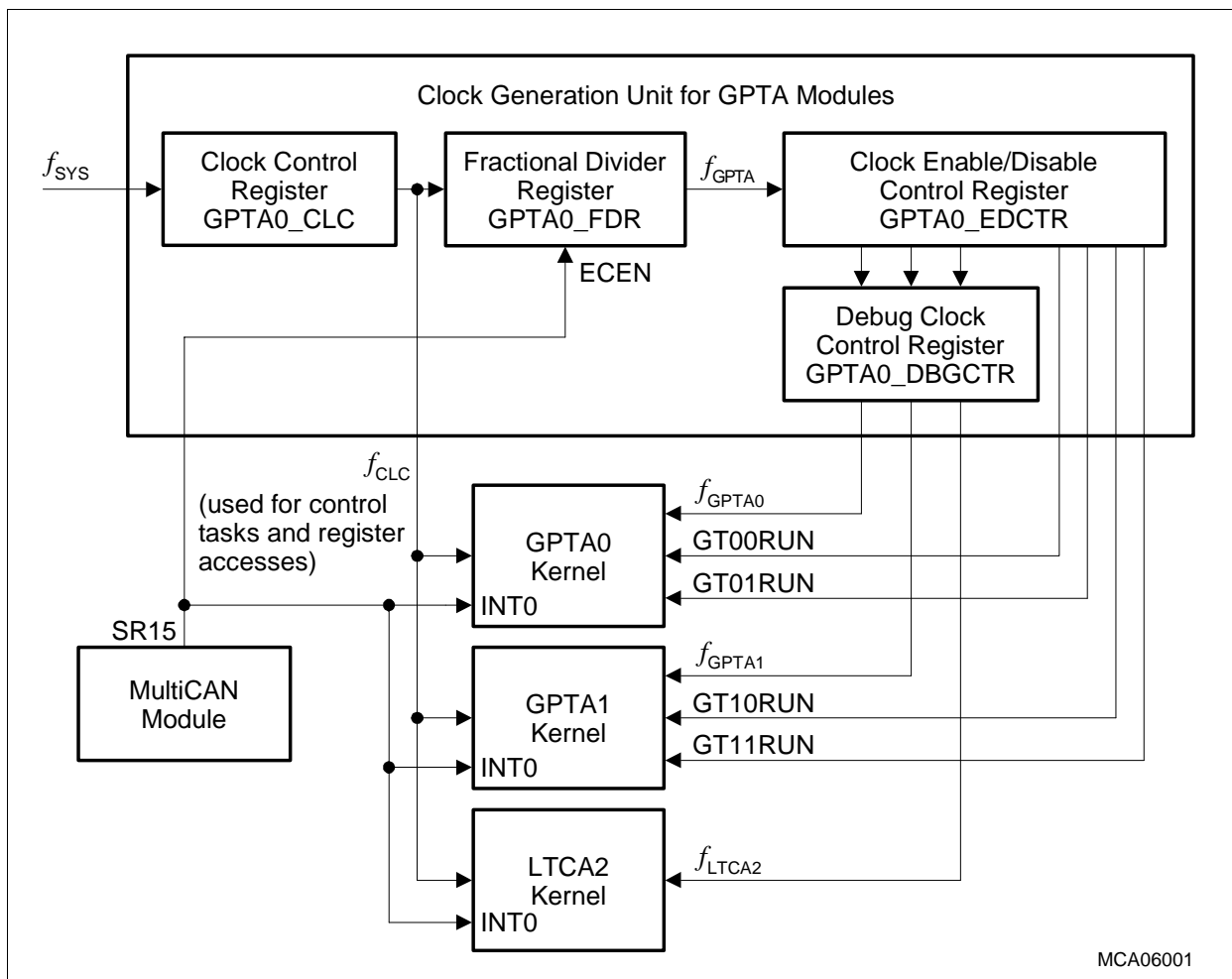


Figure 24-92 GPTA Modules Clock Generation

General Purpose Timer Array (GPTA)

Note: Registers GPTA0_CLC, GPTA0_FDR, GPTA0_EDCTR and GPTA0_DBGCTR are located in the address space of GPTA0.

The module clock f_{CLC} is used inside the GPTA module kernels for control purposes such as clocking of control logic and register operations. The frequency of f_{CLC} is identical to the system clock frequency f_{SYS} . The clock control registers GPTA0_CLC make it possible to enable/disable f_{CLC} under certain conditions.

The three separate module timer clocks f_{GPTA0} , f_{GPTA1} , and f_{LTCA2} are used inside the GPTA module kernels as input clocks for the timers. All three module timer clocks have the same frequency as f_{GPTA} (as selected through register GPTA0_FDR) and can be enabled/disabled separately each through register GPTA0_ECDTR.

Attention: If f_{GPTA0} , f_{GPTA1} , and f_{LTCA2} are disabled by the enable bits in register GPTA0_ECDTR, f_{CLC} keeps on running. In this case, that means that register accesses to the GPTA modules are possible.

The frequency of f_{GPTA} is defined by:

$$f_{GPTA} = f_{SYS} \times \frac{1}{n} \quad \text{with } n = 1024 - \text{FDR.STEP or} \quad (24.6)$$

$$f_{GPTA} = f_{SYS} \times \frac{n}{1024} \quad \text{with } n = 0-1023 \quad (24.7)$$

Note: The upper formula applies to normal divider mode of the fractional divider (GPTA0_FDR.DM = 01_B). The lower formula applies to fractional divider mode (GPTA0_FDR.DM = 10_B).

The debug clock control register additionally makes it possible to control the timer clocks f_{GPTA0} , f_{GPTA1} , and f_{LTCA2} for debug purposes on basis of a clock counter.

If the debug clock feature is enabled (GPTA0_DBGCTR.DBGCEN = 1) and bit GPTA0_DBGCTR.DBGCST is set, the timer clocks f_{GPTA0} , f_{GPTA1} , and f_{LTCA2} will be activated in parallel for as many clock cycles as have been programmed into bit field GPTA0_DBGCTR.CLKCNT. When the debug clock feature becomes enabled, bit field CLKCNT counts down and stops counting at 0000_H. Bit DBGCST is again cleared by hardware after the programmed number of clock pulses has been issued. This feature makes it possible to single step the GPTA modules with a programmable timer clock granularity.

Attention: If the frequencies of the module timer clocks f_{GPTA0} , f_{GPTA1} , or f_{LTCA2} are configured to be smaller than the control clock f_{CLC} (as programmed in register GPTA0_FDR) or even disabled (as programmed in register GPTA0_EDCTR), an action initiated by a write access to a module register could be significantly delayed, because the register write access is clocked by f_{CLC} and the register content is evaluated by hardware using the slower or disabled module timer clocks f_{GPTA0} , f_{GPTA1} , or f_{LTCA2} .

General Purpose Timer Array (GPTA)

24.6.5.1 Clock Control Registers

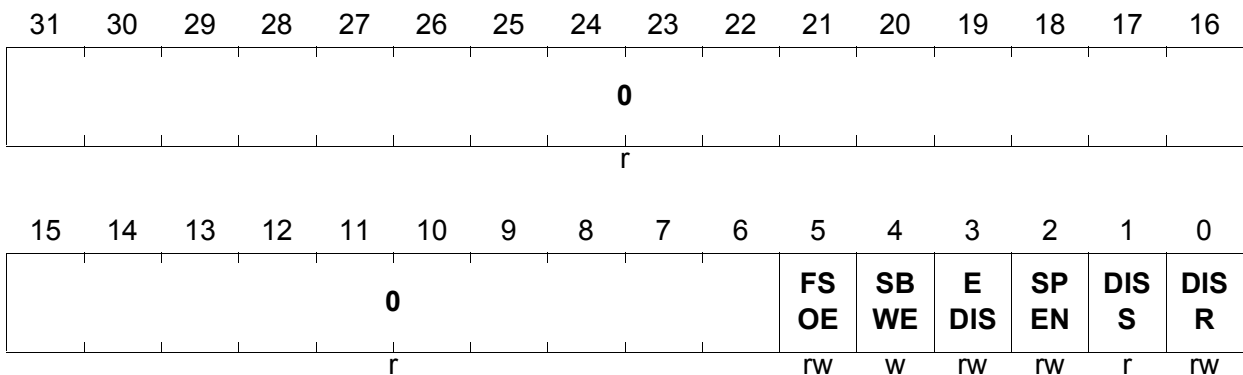
The clock control register makes it possible to control (enable/disable) the module control clock f_{CLC} . The clock signal f_{CLC} is used by the GPTA0, GPTA1, and LTCA2 as a clock for internal control operations but not for timer purposes.

GPTA0_CLC

GPTA Clock Control Register

(000_H)

Reset Value: 0000 0003_H



Field	Bits	Type	Description
DISR	0	rw	Module Disable Request Bit Used for enable/disable control of the module.
DISS	1	r	Module Disable Status Bit Bit indicates the current status of the module.
SPEN	2	rw	Module Suspend Enable for OCDS Used to enable the suspend mode.
EDIS	3	rw	External Request Disable Used to control the external clock disable request.
SBWE	4	w	Module Suspend Bit Write Enable for OCDS Determines whether SPEN and FSOE are write-protected.
FSOE	5	rw	Fast Switch Off Enable Used for fast clock switch off in suspend mode.
0	[31:6]	r	Reserved Read as 0; should be written with 0.

Note: After a hardware reset operation, the f_{CLC} clock is disabled (DISS set). Therefore, the GPTA modules clock generation is completely disabled.

General Purpose Timer Array (GPTA)

Note: Additional details on the clock control register functionality are described in section “Clock Control Register CLC” on Page 3-24 of the TC1796 User’s Manual System Units part (Volume 1).

The fractional divider register controls the clock frequency of the module timer clock f_{GPTA} . The clock frequency of f_{GPTA0} , f_{GPTA1} , and f_{LTCA2} is identical to the one of f_{GPTA} .

GPTA0_FDR

GPTA Fractional Divider Register (00C_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIS CLK	EN HW	SUS REQ	SUS ACK	0		RESULT									
rwh	rw	rh	rh	r		rh									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DM		SC		SM	0		STEP								
rw		rw		rw	r		rw								

Field	Bits	Type	Description
STEP	[9:0]	rw	Step Value Reload or addition value for RESULT.
SM	11	rw	Suspend Mode SM selects between granted or immediate suspend mode.
SC	[13:12]	rw	Suspend Control This bit field determines the behavior of the fractional divider in suspend mode.
DM	[15:14]	rw	Divider Mode This bit field selects normal divider mode, fractional divider mode, and off-state.
RESULT	[25:16]	rh	Result Value Bit field for the addition result.
SUSACK	28	rh	Suspend Mode Acknowledge Indicates state of SPNDACK signal.
SUSREQ	29	rh	Suspend Mode Request Indicates state of SPND signal.

General Purpose Timer Array (GPTA)

Field	Bits	Type	Description
ENHW	30	rw	Enable Hardware Clock Control Controls operation of ECEN input and DISCLK bit.
DISCLK	31	rwh	Disable Clock Hardware controlled disable for f_{OUT} signal.
0	10, [27:26]	rw	Reserved Read as 0; should be written with 0.

*Note: Further details on the fractional divider register functionality are described in section **“Fractional Divider Operation”** on **Page 3-29** of the TC1796 User’s Manual System Units part (Volume 1).*

General Purpose Timer Array (GPTA)

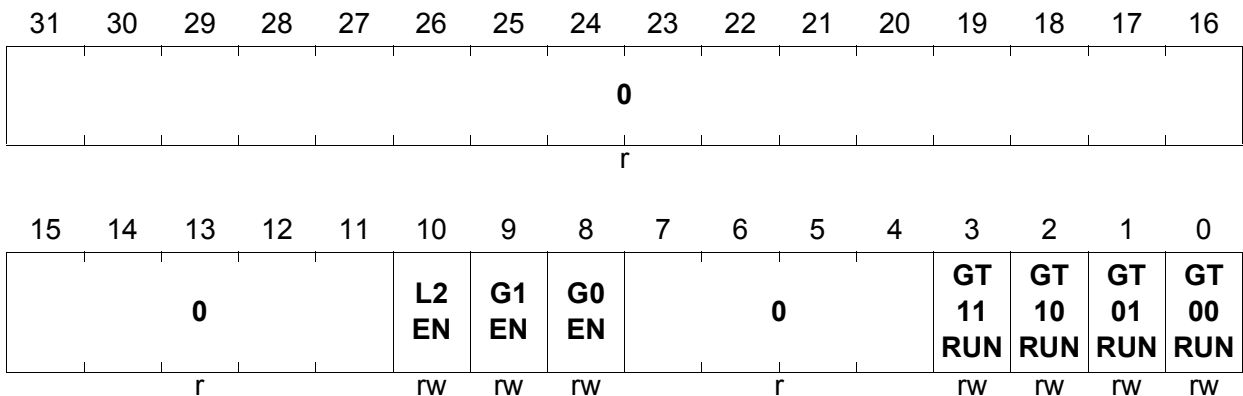
The clock enable/disable control register controls two functions: clock enable/disable control for each Global Timer in the GPTA0/GPTA1 modules and enable/disable control for the module timer clocks, separately for each clock f_{GPTA0} , f_{GPTA1} , and f_{LTCA2} .

GPTA0_EDCTR

GPTA Clock Enable/Disable Control Register

(400_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
GT00RUN	0	rw	GPTA0 Global Timer 0 Run Control 0 _B GPTA0 Global Timer 0 clock is stopped. 1 _B GPTA0 Global Timer 0 clock is started/running.
GT01RUN	1	rw	GPTA0 Global Timer 1 Run Control 0 _B GPTA0 Global Timer 1 clock is stopped. 1 _B GPTA0 Global Timer 1 clock is started/running.
GT10RUN	2	rw	GPTA1 Global Timer 0 Run Control 0 _B GPTA1 Global Timer 0 clock is stopped. 1 _B GPTA1 Global Timer 0 clock is started/running.
GT11RUN	3	rw	GPTA1 Global Timer 1 Run Control 0 _B GPTA1 Global Timer 1 clock is stopped. 1 _B GPTA1 Global Timer 1 clock is started/running.
G0EN	8	rw	GPTA0 Timer Clock Enable 0 _B GPTA0 timer clock f_{GPTA0} is disabled. 1 _B GPTA0 timer clock f_{GPTA0} is enabled.
G1EN	9	rw	GPTA1 Timer Clock Enable 0 _B GPTA1 timer clock f_{GPTA1} is disabled. 1 _B GPTA1 timer clock f_{GPTA1} is enabled.

General Purpose Timer Array (GPTA)

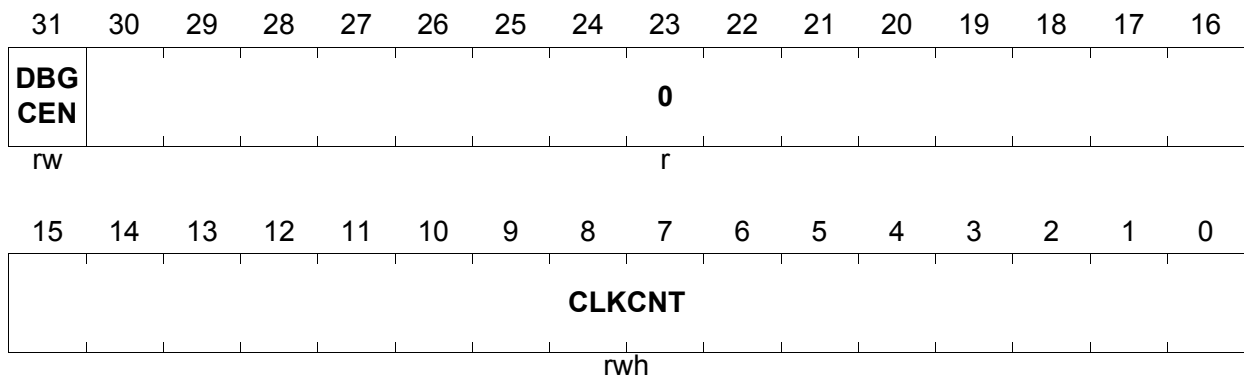
Field	Bits	Type	Description
L2EN	10	rw	LTCA2 Timer Clock Enable 0 _B LTCA2 timer clock f_{LTCA2} is disabled. 1 _B LTCA2 timer clock f_{LTCA2} is enabled.
0	[7:4], [31:11]	r	Reserved Read as 0; should be written with 0.

The debug clock control register makes it possible to control the module timer clocks f_{GPTA0} , f_{GPTA1} , and f_{LTCA2} for debug purposes on the basis of a clock counter.

GPTA0_DBGCTR

GPTA Debug Clock Control Register (004_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CLKCNT	[15:0]	rwh	Debug Clock Count This bit field determines the number of clock pulses to be issued when the debug clock feature is enabled (DBG CEN = 1). CLKCNT counts down to 0000 _H and stops when the debug clock feature is enabled.
DBG CEN	31	rw	Debug Clock Enable 0 _B The debug clock feature is disabled. The module timer clocks are always enabled. 1 _B The debug clock feature is enabled. If a non-zero value is written to bit field CLKCNT the related number of clock pulses is issued at f_{GPTA0} , f_{GPTA1} , and f_{LTCA2} .
0	[30:16]	r	Reserved Read as 0; should be written with 0.

General Purpose Timer Array (GPTA)

24.6.6 Limits of Cascading GTCs and LTCs

As shown on [Page 24-57](#) and [Page 24-70](#), a maximum of 32 GTCs and a maximum of 64 LTCs can be cascaded. In the TC1796, however cascading of GTCs and LTCs is limited under certain conditions.

If the LTCs are running with the maximum module timer clock of $f_{GPTA} = f_{SYS} = 75$ MHz, a maximum of 20 GTCs and 20 LTCs can be connected together. If the module timer clock f_{GPTA} is reduced, the number of LTCs that can be cascaded increases accordingly. Only the integer part of the divider ratio as selected by the GPTA0_FDR fractional divider register determines the maximum number of cascaded GTCs and LTCs.

Table 24-31 Limits of Cascading GTCs and LTCs

f_{SYS}	Selected Clock Divider Ratio ¹⁾	Max. Number of Cascaded GTCs/LTCs
75 MHz	$1 \leq f_{SYS}/f_{GPTA} < 2$	20 GTCs, 20 LTCs
	$2 \leq f_{SYS}/f_{GPTA} < 3$	no limits for GTCs, 40 LTCs
	$3 \leq f_{SYS}/f_{GPTA}$	no limits for GTCs and LTCs

1) Selected by the GPTA0_FDR fractional divider register.

General Purpose Timer Array (GPTA)

24.6.7 Interrupt Registers

Each of the service request outputs of the GPTA0, GPTA1 and LTCA2 module kernels is able to generate an interrupt and is controlled by an interrupt service request control register GPTA_SRCk. Therefore, the following interrupt service request control registers are available:

- GPTA0: GPTA0_SRC[37:00]
- GPTA1: GPTA1_SRC[37:00]
- LTCA2: LTCA2_SRC[15:00]

GPTA0_SRCk (k = 00-37)

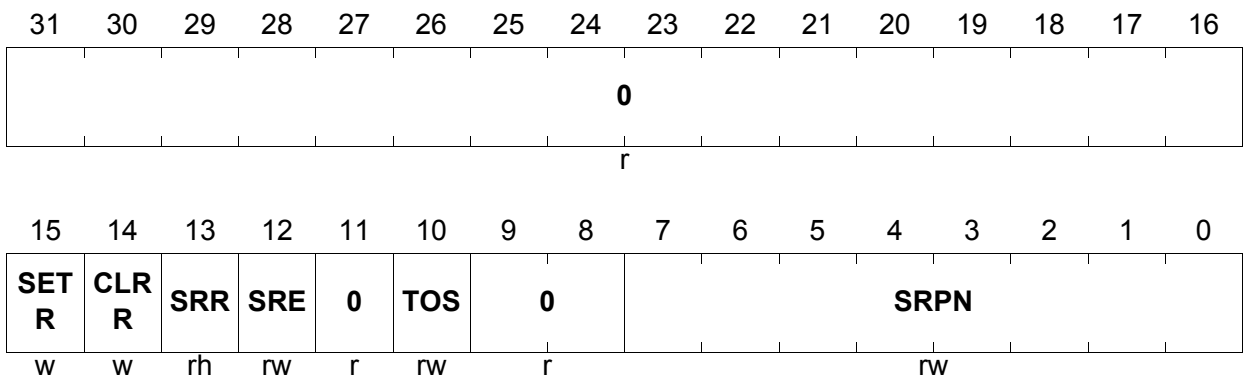
GPTA0 Interrupt Service Request Control Register k
(7FC_H-k*4_H) Reset Value: 0000 0000_H

GPTA1_SRCk (k = 00-37)

GPTA1 Interrupt Service Request Control Register k
(7FC_H-k*4_H) Reset Value: 0000 0000_H

LTCA2_SRCk (k = 00-15)

LTCA2 Interrupt Service Request Control Register k
(7FC_H-k*4_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
SRPN	[7:0]	rw	Service Request Priority Number
TOS	10	rw	Type of Service Control
SRE	12	rw	Service Request Enable
SRR	13	rh	Service Request Flag
CLRR	14	w	Request Clear Bit
SETR	15	w	Request Set Bit
0	[9:8], 11, [31:16]	r	Reserved Read as 0; should be written with 0.

General Purpose Timer Array (GPTA)

Note: Additional details on service request nodes and the service request control registers are described on [Page 14-3](#) of the TC1796 User's Manual System Units part (Volume 1).

24.6.8 GPTA Register Address Map

The GPTA0 and GPTA1 register map shown in [Figure 24-93](#). The LTCA2 register map shown in [Figure 24-94](#).

General Purpose Timer Array (GPTA)

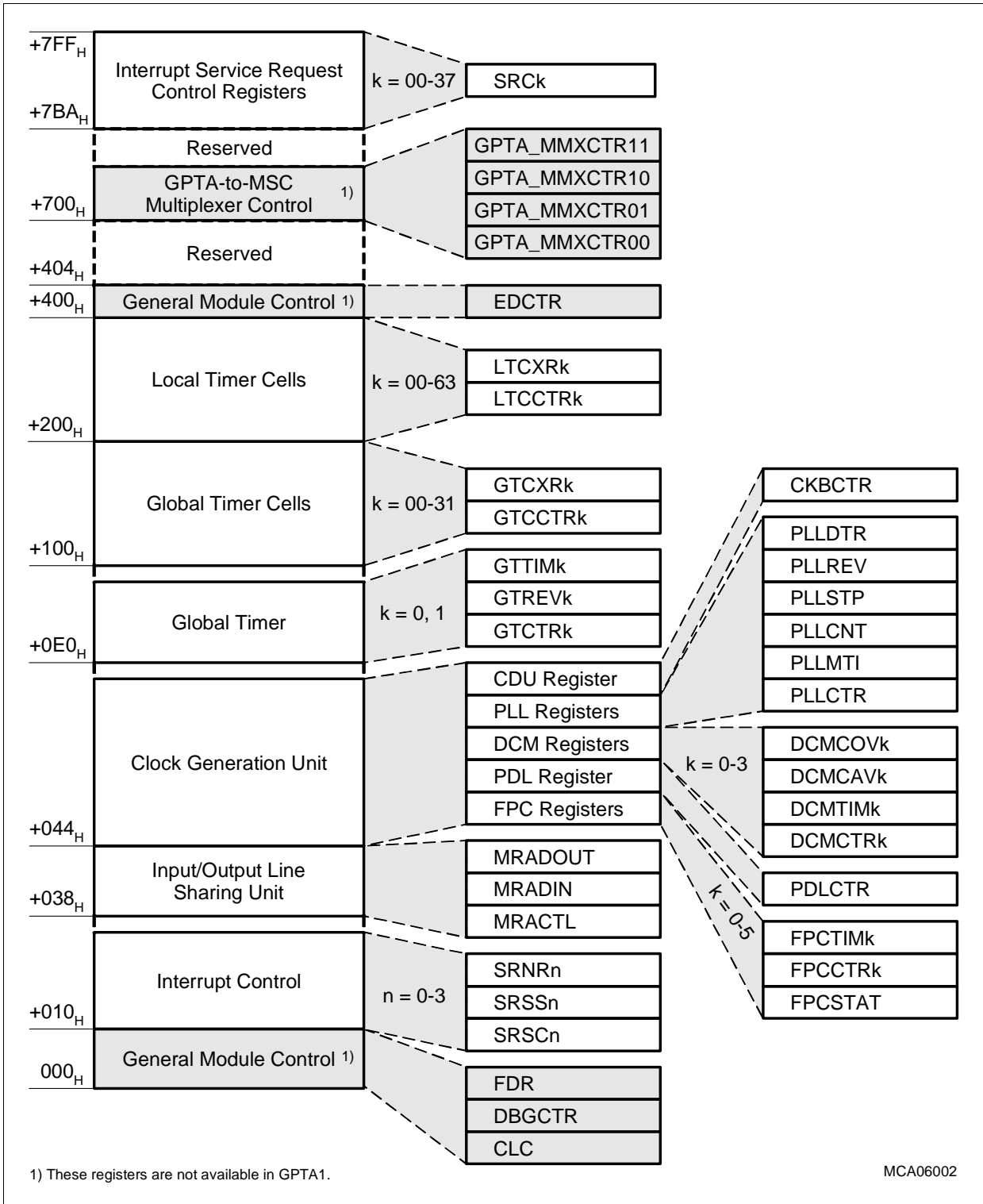


Figure 24-93 GPTA0/GPTA1 Register Map

General Purpose Timer Array (GPTA)

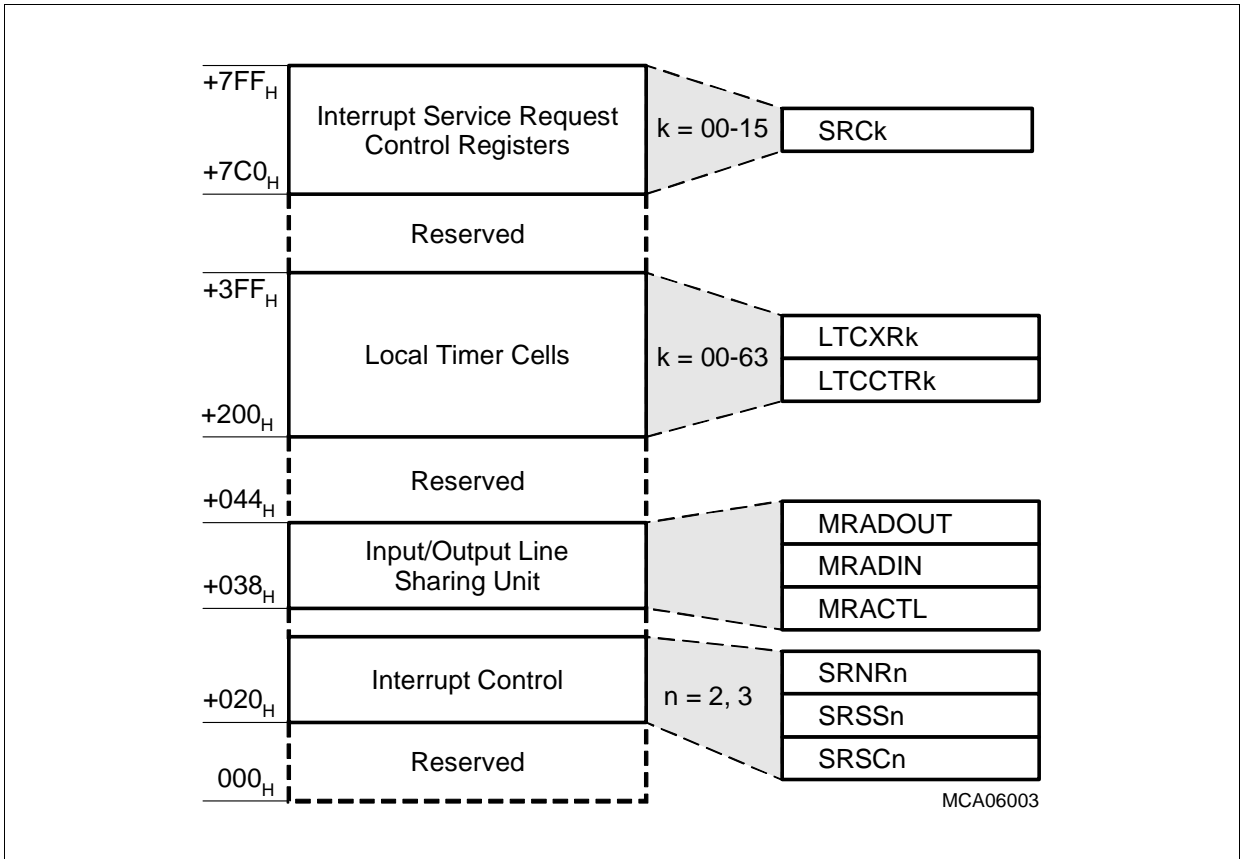


Figure 24-94 LTCA2 Register Map

25 Analog-to-Digital Converter (ADC)

The TC1796 contains two medium-speed Analog-to-Digital Converters (ADC0 and ADC1) with identical functionality and a third fast Analog-to-Digital Converter (FADC). ADC0 and ADC1 provide 2-3 μs conversion time @ 10-bit resolution and are intended primarily for single-ended signals. They offer a very flexible and comprehensive control for monitoring a large number of relatively slow signals, including synchronous conversion of both ADCs.

The FADC offers very fast conversion rates (280 ns @ 10-bit resolution) thus allowing sampling of high-frequency signals. For slow and mid-range frequency signal heavy oversampling can be performed to avoid the use of expensive filters.

The FADC is described in [Chapter 26](#). This chapter describes in detail the two Analog-to-Digital converters ADC0 and ADC1. It contains the following sections:

- Functional description of the ADC Kernel for ADC0 and ADC1 (see [Page 25-1](#))
- Register descriptions for all ADC Kernel specific registers (see [Page 25-57](#))
- TC1796 implementation-specific details and registers of the ADC0/ADC1 modules, including port connections and control, service request control, address decoding, and clock control (see [Page 25-100](#)).

Note: The ADC Kernel register names described in [Section 25.2](#) are referenced in the TC1796 User's Manual with the module name prefix "ADC0_" for the ADC0 interface and "ADC1_" for the ADC1 interface.

25.1 ADC Kernel Description

The two on-chip ADC modules of the TC1796 are analog-to-digital converters with 8-bit, 10-bit, or 12-bit resolution including sample & hold functionality. The A/D converters operate by the method of successive approximation. A multiplexer selects up to 32 analog inputs that can be connected to the 16 conversion channels in each ADC module. An automatic self-calibration adjusts the ADC modules to changing temperatures or process variations.

Features

- 8-bit, 10-bit, 12-bit A/D conversion
- Conversion time below 2.5 μs @ 10-bit resolution
- Extended channel status information on request source
- Successive approximation conversion method
- Total Unadjusted Error (TUE) of ± 2 LSB @ 10-bit resolution
- Integrated sample & hold functionality
- Direct control of up to 16(32) analog input channels per ADC
- Dedicated control and status registers for each analog channel
- Powerful conversion request sources
- Selectable reference voltages for each channel

Analog-to-Digital Converter (ADC)

- Programmable sample and conversion timing schemes
- Limit checking
- Flexible ADC module service request control unit
- Synchronization of the two on-chip A/D Converters
- Automatic control of external analog multiplexers
- Equidistant samples initiated by timer
- External trigger and gating inputs for conversion requests
- Power reduction and clock control feature

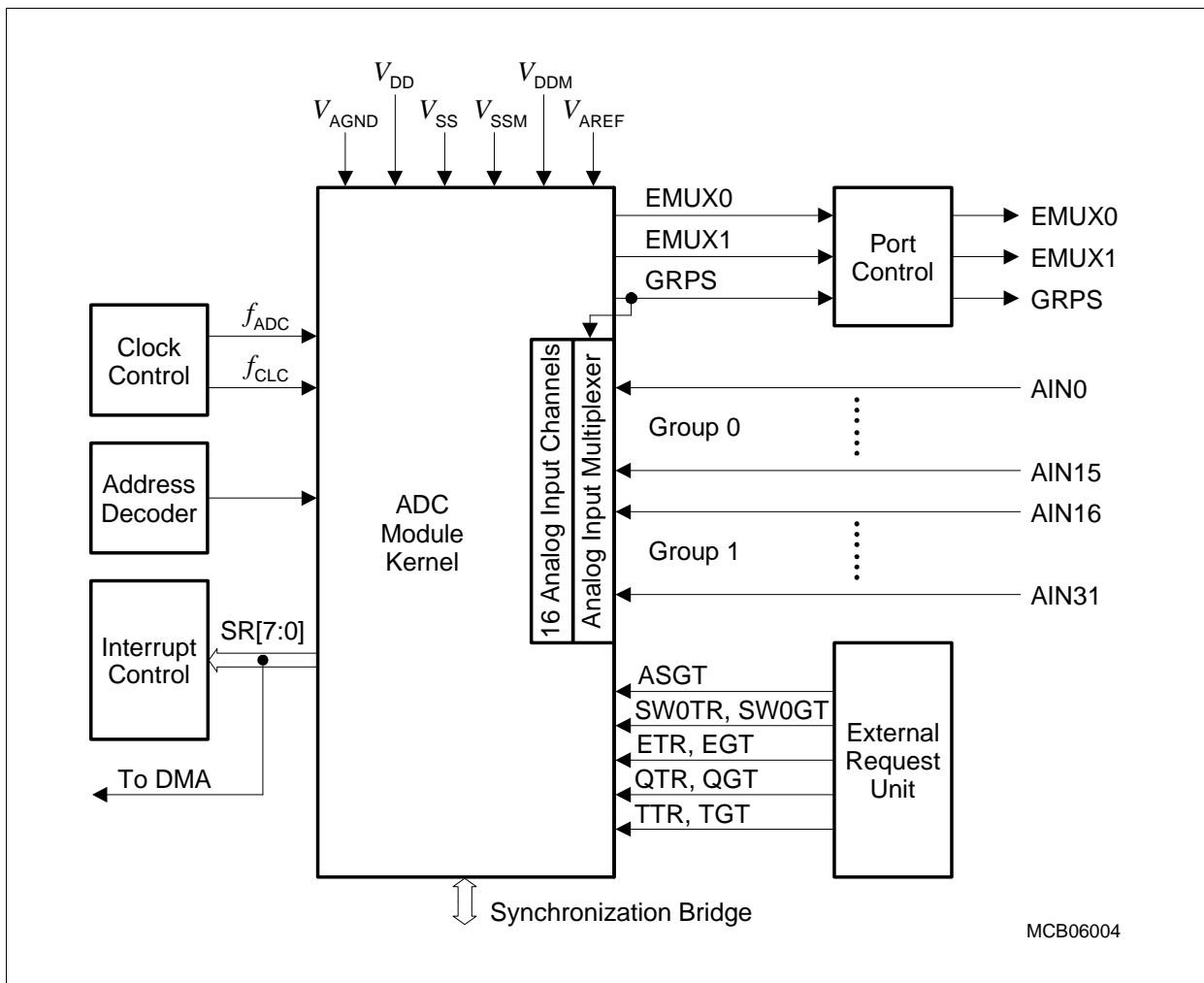


Figure 25-1 General Block Diagram of the ADC Module

As shown in **Figure 25-1**, each of the two ADC modules, ADC0 and ADC1 has 16 analog input channels. An analog multiplexer selects the input line for the analog input channels from among 32 analog inputs. Additionally, an external analog multiplexer can be used for analog input extension. External Clock control, address decoding, and service request (interrupt) control are managed outside the ADC module kernel. A synchronization bridge is used for synchronization of two ADC modules. External trigger

Analog-to-Digital Converter (ADC)

conditions are controlled by an External Request Unit. This unit generates the control signals for auto-scan control (ASGT), software trigger control (SW0TR, SW0GT), the event trigger control (ETR, EGT), queue control (QTR, QGT), and timer trigger control (TTR, TGT).

Figure 25-2 provides a more detailed block diagram of the ADC kernel with its main functional units.

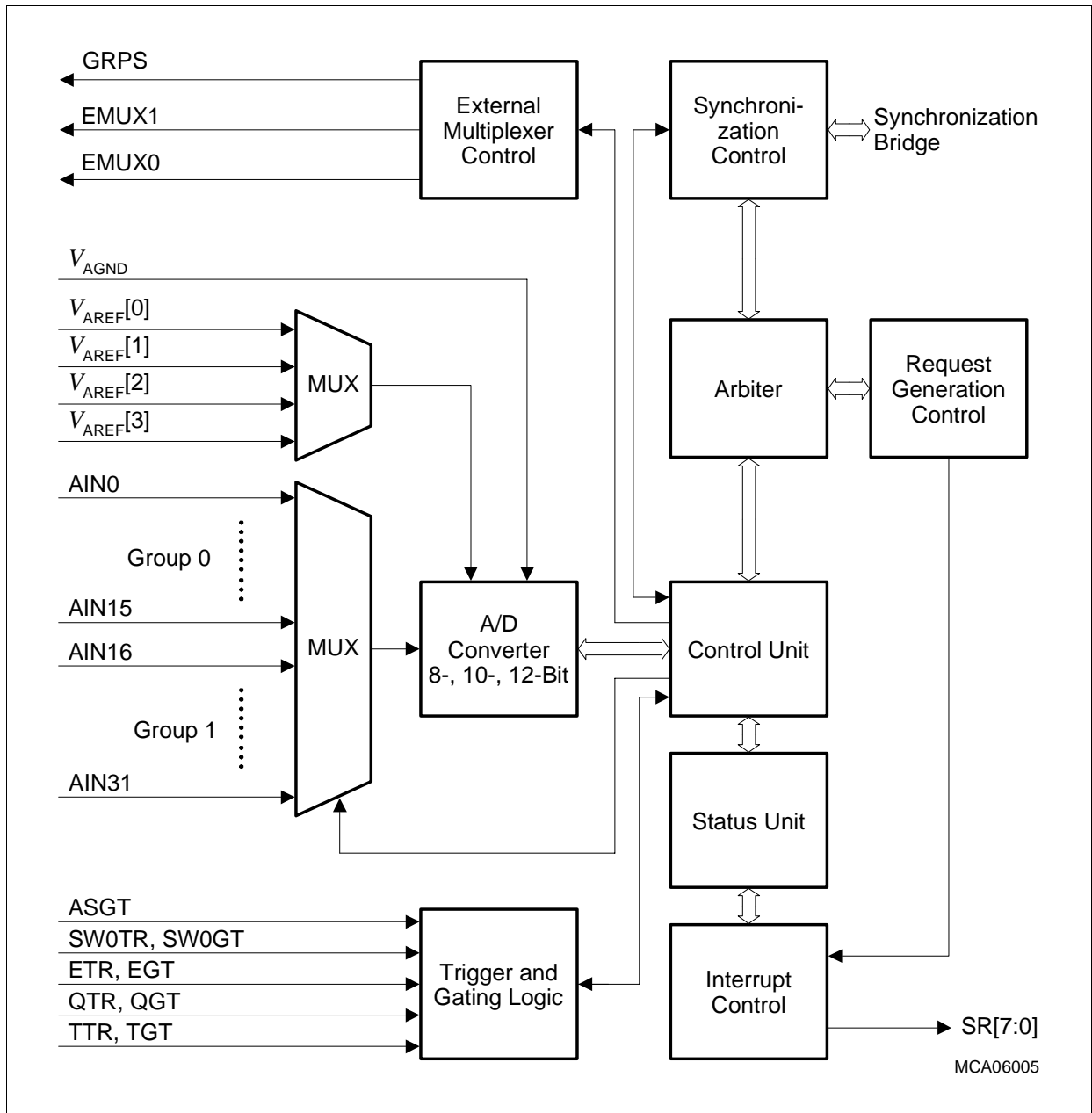


Figure 25-2 Functional Units of the ADC Kernel

25.1.1 Analog Input Connections

The 32 analog inputs AIN[31:0] of the ADC module are connected to the 16 analog input channels of the TC1796 via analog input multiplexers. These analog input multiplexers are shown in **Figure 25-3**.

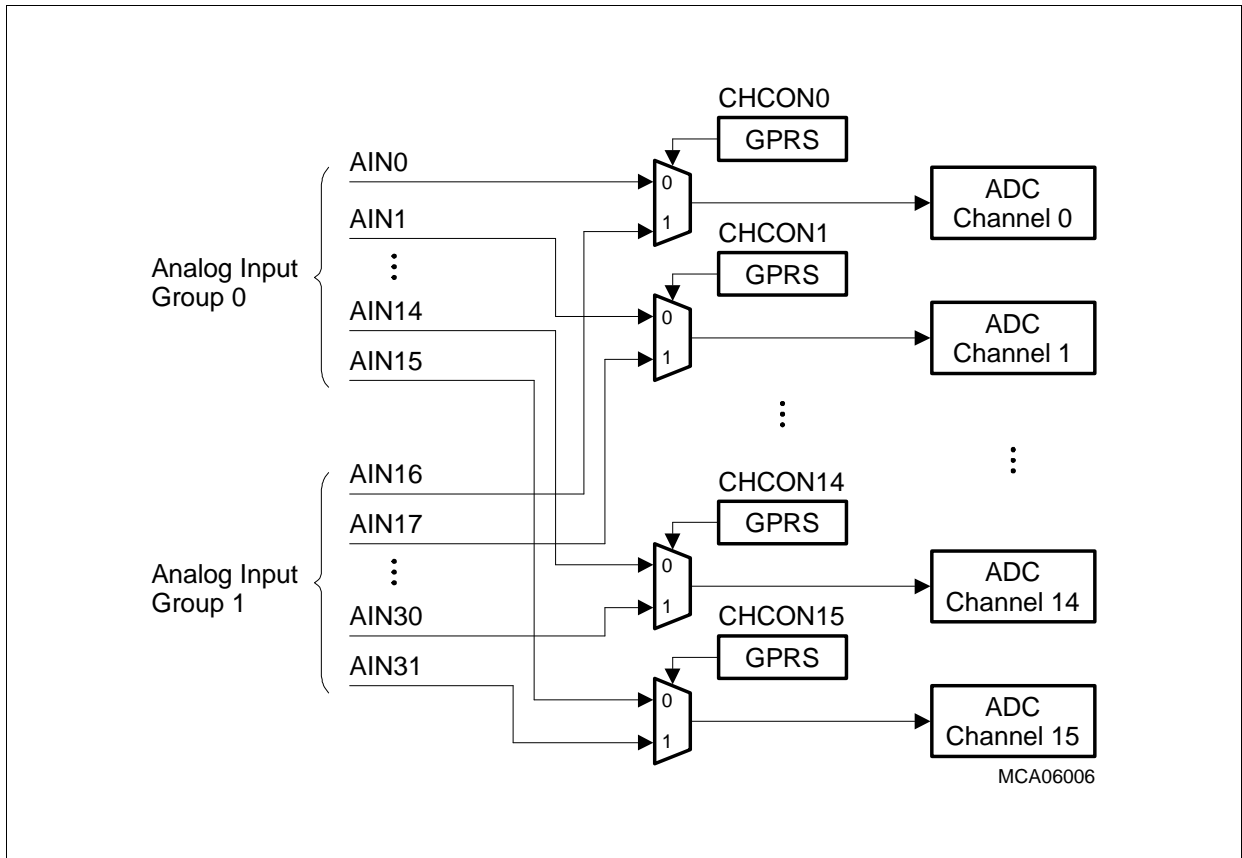


Figure 25-3 Analog Input Multiplexer Configuration

Each ADC channel n ($n = 0-15$) can be connected to two analog inputs: AIN[n] and AIN[$n+16$]. The selection is made for ADC channel n by the channel control register group select bit CHCONn.GRPS. With CHCONn.GRPS = 0, an analog input of analog input group 0 is selected. With CHCONn.GRPS = 1, an analog input of analog input group 1 is selected.

25.1.2 Conversion Request Sources

The ADC module control logic provides effective methods to request and arbitrate conversions. Conversion requests for one or more analog channels can be triggered by hardware as well as by software to provide maximum flexibility in requesting A/D conversions. Up to six individual configurable conversion request sources are implemented to issue A/D conversion requests.

In principle, the conversion request sources can be assigned either to the group of parallel conversion request sources or the group of sequential conversion request sources. A global overview of parallel and sequential conversion request sources and detailed descriptions of each source are provided in the following sections.

25.1.2.1 Parallel Conversion Request Sources

Parallel conversion request sources generate one or more conversion request at a time for the analog channels. [Table 25-1](#) shows the available parallel conversion request sources including the associated control and status signals.

Table 25-1 Parallel Conversion Request Sources

Request Source	Control Register	Conversion Req. Pending Register	Arbitration Participation Flag	Source Arbitration Level
Timer	TTC	TCRP	AP.TP	SAL.SALT
External Event	EXTC	EXCRP	AP.EXP	SAL.SALEX
Software	REQ0	SW0CRP	AP.SW0P	SAL.SALSW0
Auto-Scan	SCN	ASCRP	AP.ASP	SAL.SALAS

A parallel conversion request source is controlled by a control register, a request pending register, an arbitration participation flag, and the source arbitration level.

Each parallel conversion control register contains 16 bits defining whether a conversion request is activated for a specific channel. The contents of the parallel conversion control register are loaded into the conversion request pending register on request source specific trigger events. If at least one bit is set in the conversion request pending register, the arbitration participation flag is set for this source. This informs the arbiter to include this parallel conversion request source into arbitration.

If a parallel conversion request source is the arbitration winner, a conversion is started for the conversion request within the conversion request pending register with the highest channel number. Starting a conversion causes the conversion request bit in the conversion request pending register to be cleared by the arbiter. If a currently running conversion initiated by the parallel source is cancelled, the arbiter restores the corresponding conversion request bit in the conversion request pending registers for this

Analog-to-Digital Converter (ADC)

channel. If all pending conversion requests are processed, the arbitration participation flag of this parallel source is cleared.

The bits of the conversion request pending register can be cleared globally under software control by clearing the arbitration participation flag for this source.

25.1.2.2 Sequential Conversion Request Sources

Sequential conversion request sources generate only one conversion request at a time for an analog channel. The settings for this conversion are derived from the control register of the sequential source. [Table 25-2](#) shows the available sequential conversion request sources including the associated control and status blocks.

Table 25-2 Sequential Conversion Request Sources

Request Source	Control Register	Back-Up Register	Arbitration Participation Flag	Source Arbitration Level
Channel Injection	CHIN	not accessible via Bus	AP.CHP	SAL.SALCHIN
Queue	QR	not accessible via Bus	AP.QP	SAL.SALQ

A sequential conversion request source consists of a control register, a back-up register, an arbitration participation flag, and the source arbitration level.

Each sequential conversion control register contains a conversion request bit, the channel number to be converted, two control bits for the external multiplexer settings, one bit for the analog input channel group selection, and control information to select the resolution of the channel. Setting the conversion request bit causes the arbitration participation flag to be set. This informs the arbiter to include the sequential conversion request source into arbitration. If this sequential source is the arbitration winner, a conversion is started for the analog channel specified within the request register. The settings of the external multiplexer and the resolution of the ADC are also derived from this conversion request control register.

Starting a conversion causes the conversion request bit to be cleared by the arbiter. The arbitration participation flag is automatically cleared if the conversion request register and the back-up register contains no valid request.

If a currently running conversion initiated by a sequential source is cancelled, the arbiter restores the conversion information (conversion request bit, external multiplexer setting, analog input channel group selection, conversion resolution) in the back-up registers for this channel. If the back-up register contains valid conversion information, the arbiter reads from the back-up register instead from the conversion request control register. Thus, the previously cancelled conversion participates in arbitration again. A new

Analog-to-Digital Converter (ADC)

conversion request generated in the meantime via the conversion request register will be performed after the request in the back-up register has been served.

The request bit in the control register and in the back-up register can be cancelled under software control. Clearing the arbitration participation bit clears either the request bit in the request register (if the back-up register contains no request) or the request bit in the back-up register if (the back-up register contains a valid request).

25.1.2.3 Parallel Conversion Request Source “Timer”

Periodic samples can be achieved by timer generated conversion requests. A programmable timer serves as trigger source. Service request generation logic as well as the arbitration lock mechanism are provided to ensure periodical sampling without jitter. The block diagram of the timer and its control and status blocks are shown in [Figure 25-4](#).

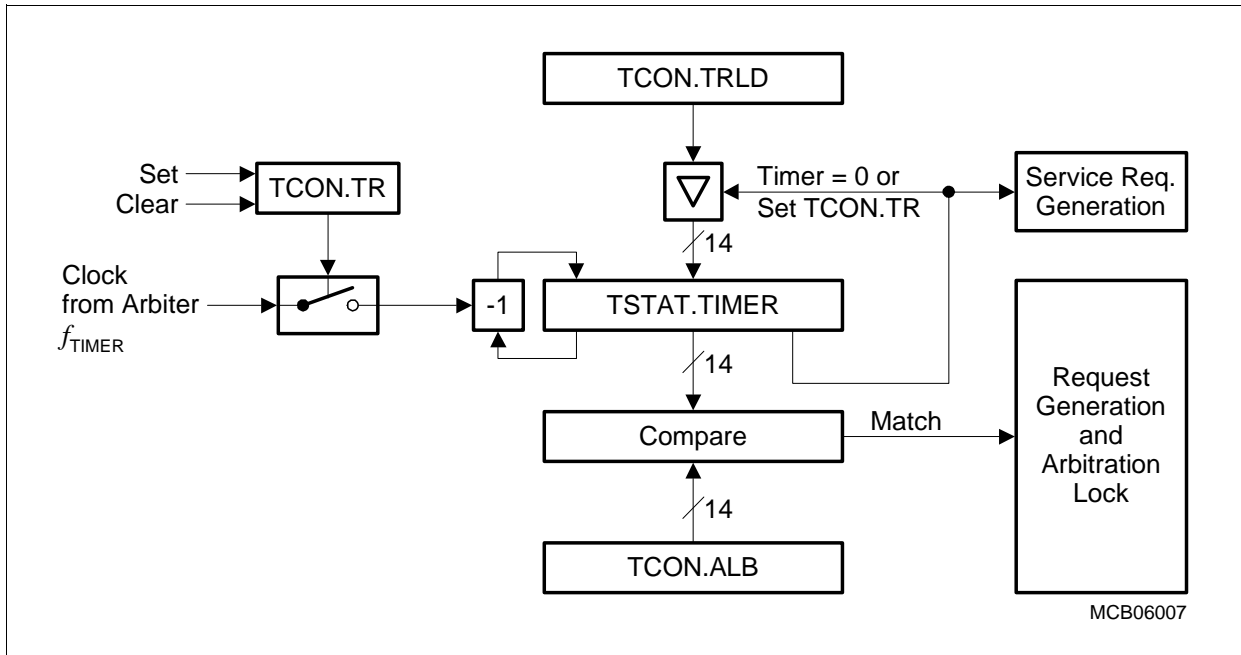


Figure 25-4 Block Diagram of Conversion Request Source “Timer”

When the timer run bit is set, the timer is clocked with f_{TIMER} , which is derived from the arbiter. This synchronizes the timer on the arbiter for jitter-free sampling. If the timer run bit **TCON.TR** becomes set, the timer bit field **TSTAT.TIMER** is loaded with the timer reload value **TCON.TRLD**. With each clock cycle of f_{TIMER} , the timer register is decremented and compared to the arbitration-lock-boundary value **TCON.ALB**. If the value of the timer register is equal to the value of the arbitration-lock-boundary, the arbitration lock bit **STAT.AL** is set (see [Section 25-6](#)). This arbitration-lock mechanism can be used to generate samples without being delayed by a currently running conversion. When the timer = 0, the arbitration is unlocked, the timer register is reloaded, the arbitration lock bit is cleared, the timer related service request status flag (**MSS1.MSRT**) is set, and a trigger pulse is sent to the conversion request source “Timer”.

Analog-to-Digital Converter (ADC)

The timer period t_{TPERIOD} can be specified within the range from microseconds up to milliseconds according to the following equation.

$$t_{\text{TPERIOD}} = \text{TRL D} \times \frac{20}{f_{\text{ADC}}} \quad (25.1)$$

Figure 25-5 shows the control and status of the conversion request source “Timer”.

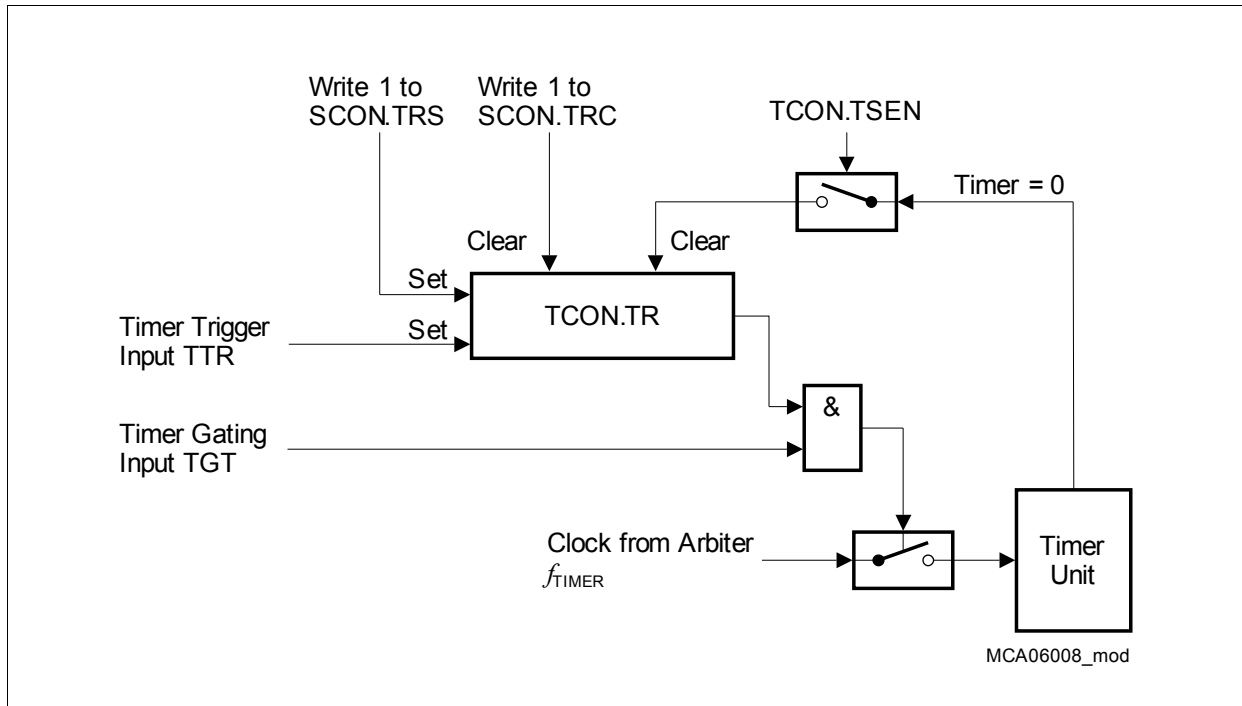


Figure 25-5 Conversion Request Source “Timer”

Up to sixteen analog input channels can be controlled by the conversion request source “Timer”. Setting request bit(s) in the timer trigger control register TTC enables the generation of a conversion request for this analog input channel(s) by the timer. If timer = 0, the contents of the timer trigger control register TTC are loaded into the timer conversion request pending register TCRP. This triggers conversion requests for the selected channel(s) if the gating input line TGT = 1. When the timer gating line TGT = 0, the pending conversion requests do not take part in the arbitration cycle. The timer can also be started by a pulse on the timer trigger input line TTR. The TTR and TGT input connections depend on the product specific implementation of the ADC module. They are described on Page 25-114.

The contents of the timer conversion request pending register and the arbitration lock bit are logically OR-ed. If bit STAT.AL or at least one bit is set in the timer conversion request pending register, the arbitration participation flag AP.TP is set. This informs the arbiter to include the conversion request source “Timer” in the arbitration.

Analog-to-Digital Converter (ADC)

If “Timer” is the arbitration winner, a conversion is started for the conversion request within register TCRP with the highest channel number. Starting a conversion causes the conversion request bit in register TCRP to be cleared by the arbiter. If a currently running timer initiated conversion is cancelled, the arbiter sets the corresponding conversion request bit in registers TCRP for this channel. If all pending conversion requests are processed, the arbiter clears the arbitration participation flag AP.TP. The contents of register TCRP can be cleared globally under software control by clearing the timer arbitration participation flag AP.TP.

The arbitration-lock mechanism provides the means to start timer triggered conversion requests without being delayed by a currently running conversion. **Figure 25-6** shows this method in detail.

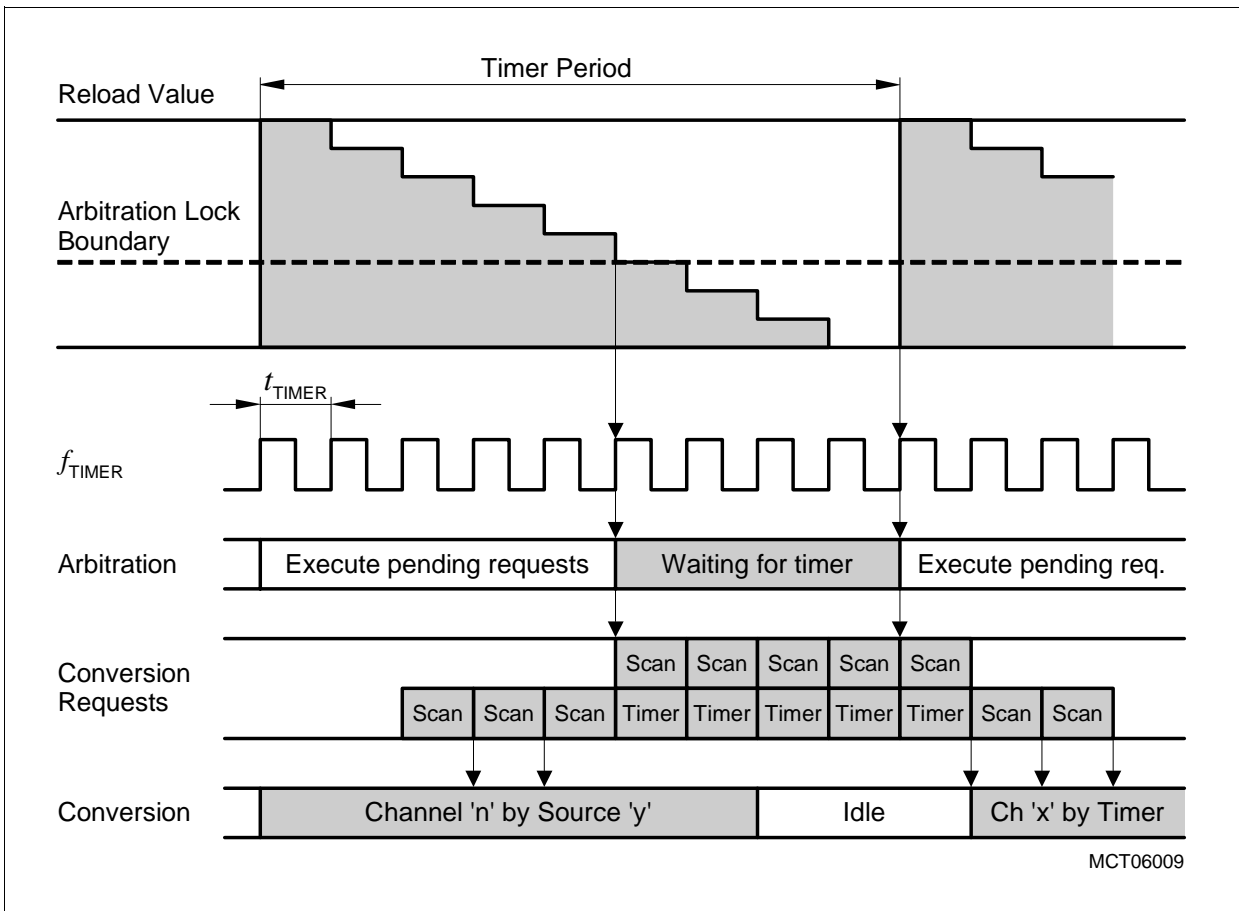


Figure 25-6 Arbitration Lock Mechanism

The arbitration must be locked before the timer is 0 in order to ensure that the running conversion has been finished and no new conversion will be started in the meantime. When the arbitration is locked, lower prioritized conversion request source than the “Timer” are blocked from performing requested conversions.

Analog-to-Digital Converter (ADC)

In the example of **Figure 25-6**, the conversion request source “Auto Scan” has triggered conversion request(s) that are not served according to a currently running conversion and the locked arbitration. On timer = 0, the conversion requested by the timer is started (it is assumed that the “Timer” is programmed to a higher priority than the “Auto Scan”). Arbitration Lock Mode is enabled by setting bit field TCON.ALB to any value greater than zero.

The value of the arbitration lock boundary is also used to specify the time t_{LOCK} for which the arbitration is locked. Running in Arbitration Lock Mode, the current value of the timer register is compared to the arbitration lock boundary. Note that the arbitration will always be locked if the reload value is selected to be equal to or less than the arbitration lock boundary. On a compare match, the arbitration logic is locked (STAT.AL = 1), while timer = 0 removes the arbitration lock. Bit STAT.AL is either cleared if timer = 0 or after clearing bit TCON.TR.

25.1.2.4 Parallel Conversion Request Source “External Event”

Externally triggered conversion requests are mandatory for various control applications. The conversion request source “External Event” receives the trigger line ETR from the external world via the interrupt input unit. **Figure 25-7** shows the structure of the conversion request source “External Event”.

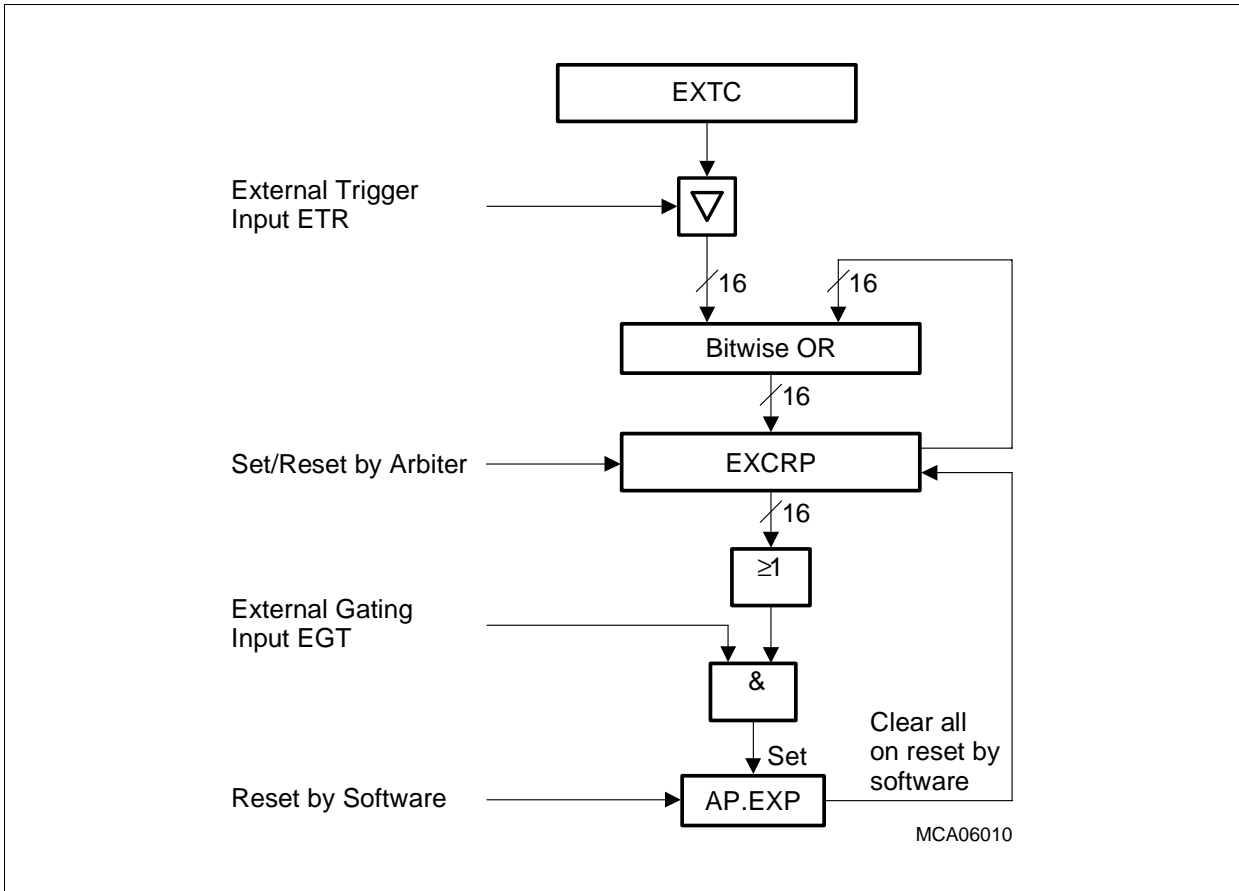


Figure 25-7 Conversion Request Source “External Event”

Up to sixteen individually selectable analog input channels in the external trigger control register EXTC can be controlled by the conversion request source “External Event”. Setting request bit EXTC.ETCH_n enables the generation of a conversion request for analog channel *n* on external events. A trigger issued on an external event loads the contents of EXTC into the external conversion request pending register EXCRP. This triggers conversion requests for the selected channel(s) while the external gating input EGT = 1. When the external gating line EGT = 0, the pending external conversion requests do not take part in the arbitration cycle.

If an external event is detected (input ETR = 1), the contents of the external trigger control register EXTC are loaded into the external conversion request pending register. “Load” means that the contents of EXTC and the contents of external conversion request pending register EXCRP are bit-wise OR-ed, as shown in [Figure 25-7](#).

The ETR and EGT input connections depend on the product specific implementation of the ADC module. They are described on [Page 25-114](#).

If at least one bit is set in register EXCRP, the arbitration participation flag AP.EXP is set. This informs the arbiter to include the conversion request source “External Event” into arbitration. If “External Event” is the arbitration winner, a conversion is started for the

Analog-to-Digital Converter (ADC)

conversion request within register EXCRP with the highest channel number. Starting a conversion causes the conversion request bit in register EXCRP to be cleared by the arbiter. If a currently running “External Event” initiated conversion is cancelled, the arbiter sets the corresponding conversion request bit in register EXCRP for this channel. If all pending conversion requests are processed, the arbitration participation flag AP.EXP is cleared.

The contents of register EXCRP can be reset globally under software control by clearing the “External Event” arbitration participation flag. Note that conversion requests caused by external trigger pulses are lost if the flag for this channel is already set in the external conversion request pending register.

25.1.2.5 Parallel Conversion Request Source “Software”

The conversion request source “Software” makes it possible to generate conversion request under software control, as shown in [Figure 25-8](#).

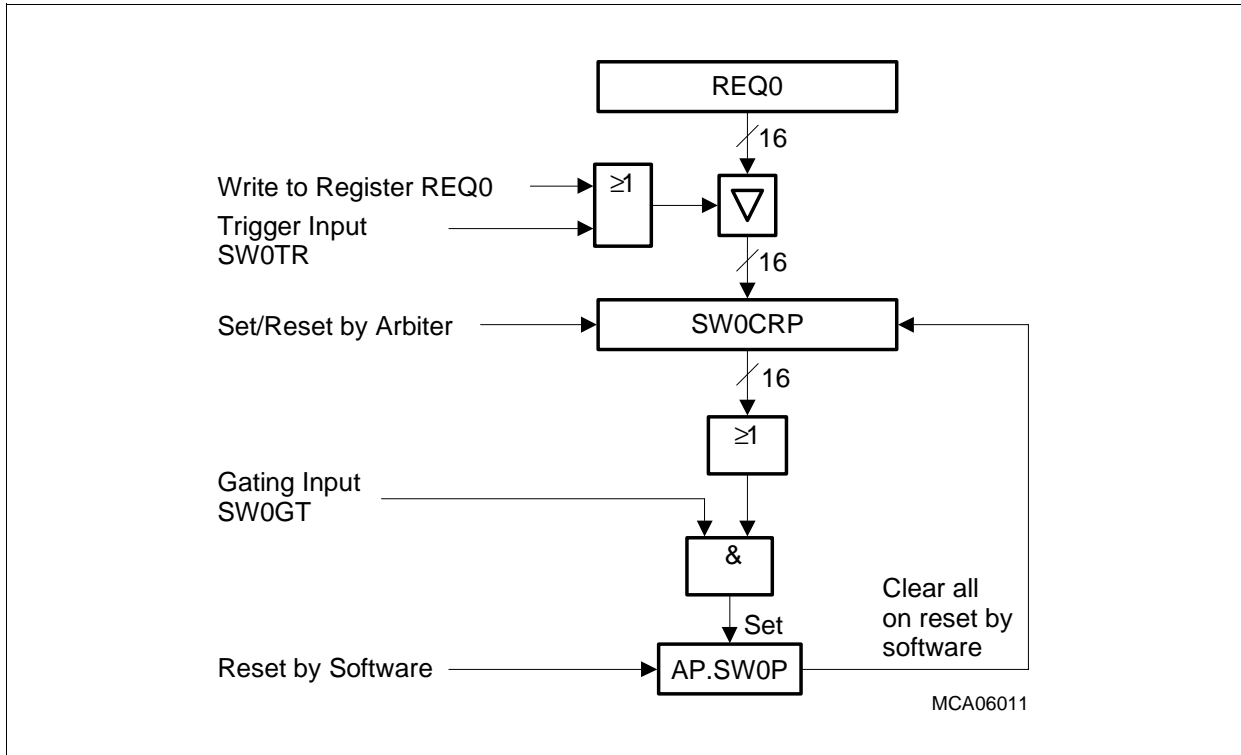


Figure 25-8 Conversion Request Source “Software”

One or more request bits can be set at a time by software, resulting in a conversion request for the designated analog channel(s). Writing to the register REQ0 automatically loads its contents into register SW0CRP. The contents of REQ0 remains unchanged after a load operation. A trigger pulse (SW0TR = 1) also leads to this load operation.

If at least one bit is set in REQ0 when the software gating input SW0GT = 1, the arbitration participation flag AP.SW0P is set. This informs the arbiter to include the conversion request source “Software” in the arbitration. If “Software” is the arbitration winner, a conversion is started for the conversion request within register SW0CRP with the highest channel number. Starting a conversion causes the conversion request bit in register SW0CRP to be cleared by the arbiter. If a currently running “Software” initiated conversion is cancelled, the arbiter sets the corresponding conversion request bit in registers SW0CRP for this channel. If all pending conversion requests are processed, the arbitration participation flag AP.SW0P becomes 0.

The contents of register SW0CRP can be reset under software control either bit-wise by writing a 0 to the corresponding bit position in register REQ0 or globally by clearing the “Software” arbitration participation flag.

The SW0TR and SW0GT input connections depend on the product specific implementation of the ADC module. They are described in [Section 25.3.6.2](#).

25.1.2.6 Parallel Conversion Request Source “Auto-Scan”

The conversion request source “Auto-scan” allows continuous conversions of a selectable group of analog channels with almost zero software effort in generating and controlling these conversion requests. Auto-scan provides a single conversion sequence mode as well as continuous conversion sequence mode. Each analog channel can be configured individually to participate in an auto-scan sequence.

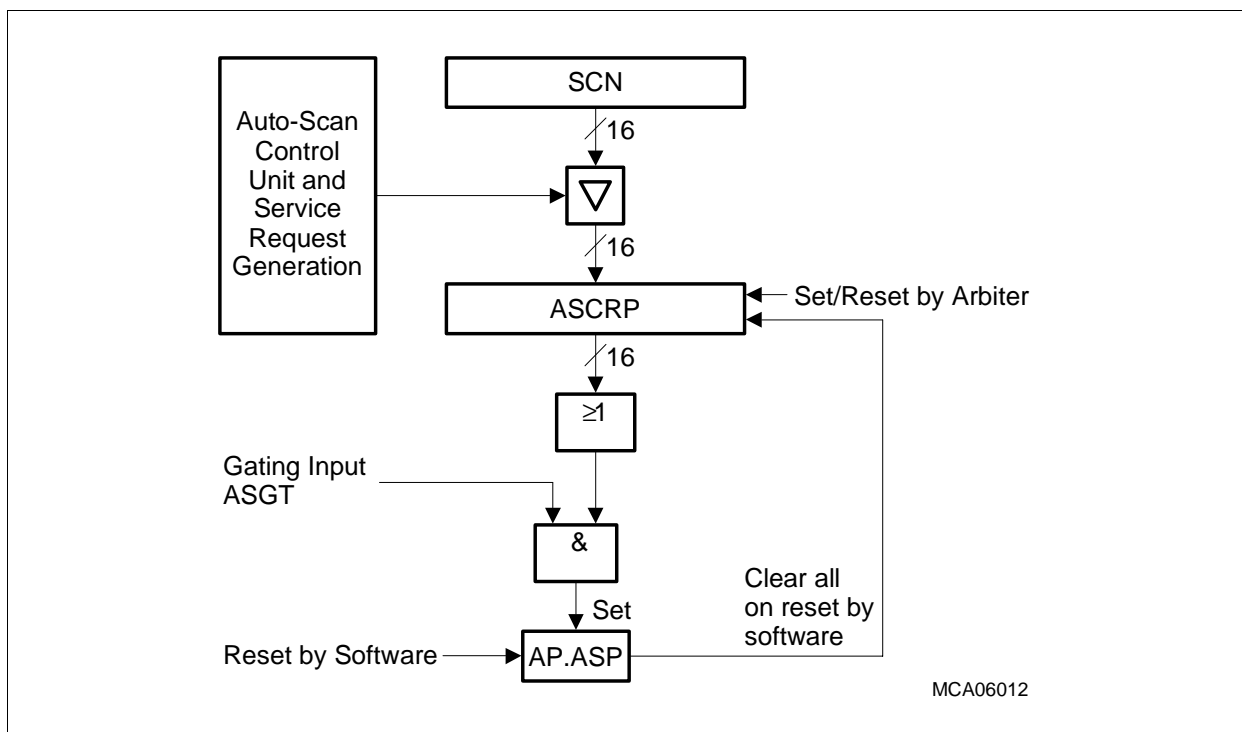


Figure 25-9 Conversion Request Source “Auto-Scan”

The group of analog channels to be auto-scanned is determined by setting the corresponding channel request flags SRQ_n in the auto-scan conversion request register **SCN**. The auto-scan sequence is started by programming an auto-scan mode in the A/D Converter control register (bit field **CON.SCNM**). Selecting an auto-scan mode loads the contents of the auto-scan conversion request register **SCN** into the auto-scan conversion request pending register **ASCRP**.

If at least one bit is set in the auto-scan conversion request pending register when the auto-scan gating line $ASGT = 1$, the arbitration participation flag **AP.ASP** is set. This informs the arbiter to include the conversion request source “Auto-scan” into the arbitration. If “Auto-scan” is the arbitration winner, a conversion is started for the conversion request within register **ASCRP** with the highest channel number. Pending conversion requests for the auto-scan channels are processed in the sequence from the

Analog-to-Digital Converter (ADC)

highest to the lowest channel number. Starting a conversion causes the conversion request pending flag in register ASCRP to be cleared.

The auto-scan sequence is complete if the channel with the lowest number selected to be auto-scanned has been converted (all bits in register ASCRP are cleared).

In single conversion sequence mode, the bit field CON.SCNM is automatically cleared and the conversion request source “Auto-scan” is disabled.

In continuous conversion sequence mode, the conversion request source “Auto-scan” automatically requests a new auto-scan sequence. Results previously stored in the specific channel status register(s) will be overwritten. Continuous auto-scan sequence is performed until auto-scan is stopped under software control.

If a currently running “Auto-scan” initiated conversion is cancelled, the arbiter sets the corresponding conversion request bit in register ASCRP for this channel.

After the conversion of the last channel within an auto-scan sequence has been finished, the source service request flag MSS1.MSRAS is set. Auto-scan service requests can be generated only if the auto-scan service request is enabled by SRNP.ENPAS = 1.

The ASGT input connection depends on the product specific implementation of the ADC module. It is described on [Page 25-114](#).

The auto-scan control functionality is described in [Table 25-3](#), [Table 25-4](#) and [Table 25-5](#). This description of the auto-scan includes the actions to be performed on changes in the auto-scan mode or the channels to be auto-scanned as well as clearing the auto-scan arbitration participation flag.

Analog-to-Digital Converter (ADC)

Table 25-3 describes the action to be performed on a change of bit field CON.SCNM.

Table 25-3 Change of Auto-Scan Mode Bit Field CON.SCNM

Value of CON.SCNM		Action
Current Value of CON.SCNM	Value after Write Action to CON.SCNM	
00 _B	00 _B	No action
	01 _B	Load SCN contents into register ASCRP, set bit AP.ASP, and start single auto-scan sequence if at least one channel is specified in register SCN to participate in auto-scan mode; otherwise, clear bit field CON.SCNM.
	10 _B	Load SCN contents into register ASCRP, set bit AP.ASP, and start continuous auto-scan sequence if at least one channel is specified in register SCN to participate in auto-scan mode; otherwise, clear bit field CON.SCNM.
	11 _B	Clear bit field CON.SCNM
01 _B	00 _B	Finish currently performed auto-scan sequence and generate a service request (if enabled) at the end of the sequence.
	01 _B	Continue to perform auto-scan sequence and generate a service request (if enabled) at the end of the sequence.
	10 _B	Finish currently performed auto-scan conversion, generate a service request (if enabled) if this is the last channel of the auto-scan sequence. Load SCN contents into register ASCRP and start a continuous auto-scan sequence.
	11 _B	Clear bit field CON.SCNM, finish auto-scan sequence, and generate service request (if enabled) at the end of the sequence.

Analog-to-Digital Converter (ADC)

Table 25-3 Change of Auto-Scan Mode Bit Field CON.SCNM (cont'd)

Value of CON.SCNM		Action
Current Value of CON.SCNM	Value after Write Action to CON.SCNM	
10 _B	00 _B	Finish auto-scan sequence and generate service request (if enabled) at the end of the sequence.
	01 _B	Finish currently performed auto-scan conversion and generate a service request (if enabled) at the end of the conversion if this was the last channel of the sequence. Load SCN contents into register ASCRP and start single auto-scan sequence.
	10 _B	Continue to perform continuous auto-scan sequence and generate a service request (if enabled) at the end of the sequence. Load SCN contents into register ASCRP and start continuous auto-scan sequence.
	11 _B	Clear bit field CON.SCNM and finish auto-scan sequence. Generate a service request (if enabled) at the end of the sequence.

Analog-to-Digital Converter (ADC)

Table 25-4 shows the actions to be taken on a change of the auto-scan conversion request register SCN.

Table 25-4 Change of the Auto-Scan Conversion Request Register SCN

Value of SCN		Action
Current Value of SCN	Value after Write Action to SCN	
≠0	0000 _H	Bit field CON.SCNM is cleared independently from the auto-scan mode. Finish currently performed auto-scan sequence and generate a service request (if enabled) if this was the last channel of the sequence. No new auto-scan sequence is started.
	≠0	In case of CON.SCNM = 00_B, 01_B, or 11_B: Clear bit field CON.SCNM. Finish currently performed auto-scan sequence and generate a service request (if enabled) if this was the last channel of the sequence. No new auto-scan sequence is started.
		In case of CON.SCNM = 10_B: Finish the currently performed auto-scan conversion and generate a service request is (if enabled) if this was the last channel of the sequence. Start a new continuous auto-scan sequence.

Analog-to-Digital Converter (ADC)

Table 25-5 shows the actions to be taken on a change of the auto-scan arbitration participation flag.

Table 25-5 Change of the Auto-Scan Arbitration Participation Flag AP.ASP

Current Value of ASP	Write to ASP	Action
0	0	No action
	1	No action
1	0	In case of bit field CON.SCNM = 00_B, 01_B, or 11_B: Bit field SCNM is cleared. Finish currently performed auto-scan conversion. Generate a service request (if enabled) if this was the last channel of auto-scan sequence.
		In case of bit field CON.SCNM = 10_B: Finish currently performed auto-scan conversion and generate a service request (if enabled) if this was the last channel of auto-scan sequence. Start new continuous auto-scan sequence
	1	Don't care

25.1.2.7 Sequential Conversion Request Source “Channel Injection”

The conversion request source “Channel Injection” generates sequential conversion requests for analog channels either with Wait-Inject or Cancel-Inject-Repeat functionality. “Channel Injection” consists of the channel injection control register, the back-up register, and the channel injection arbitration participation flag.

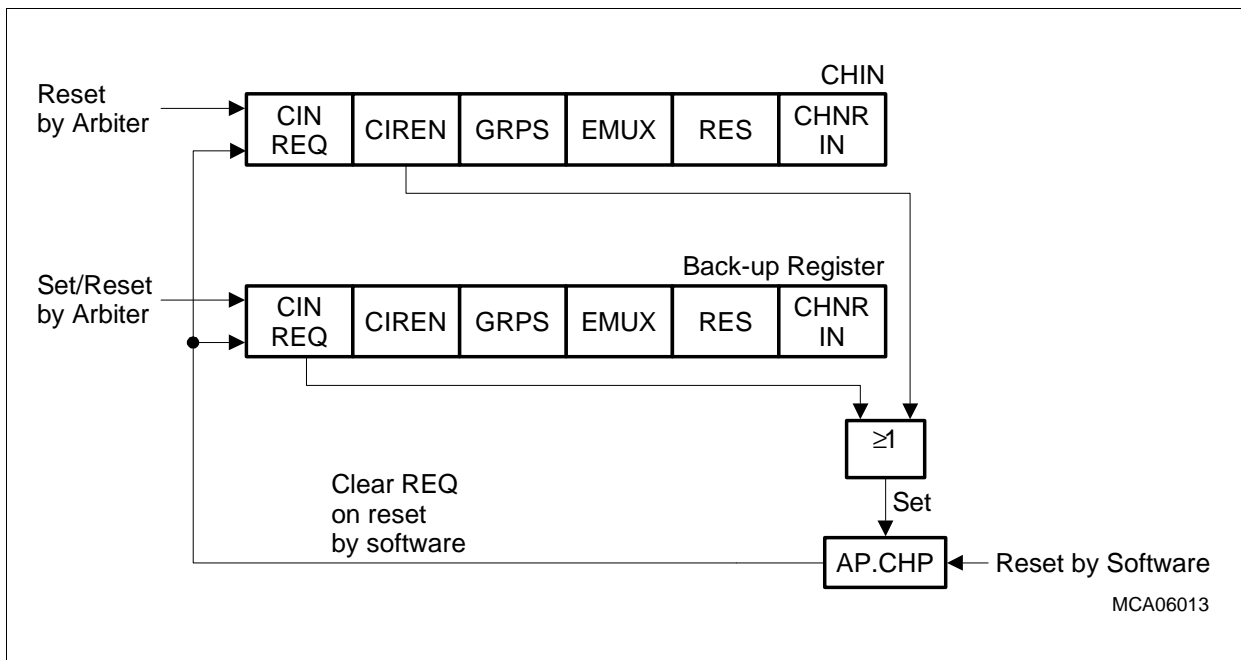


Figure 25-10 Conversion Request Source Channel Injection

The channel injection control register CHIN contains a conversion request bit CINREQ, a control bit CIREN for selecting the cancel inject repeat feature, control information for analog input group selection (GRPS) and external multiplexer settings (EMUX), control information (RES) for resolution selection, and the channel number (CHNRIN) of the ADC channel to be converted.

Setting the channel injection request bit CINREQ causes the arbitration participation flag AP.CHP to be set. This informs the arbiter to include the conversion request source “Channel Injection” into arbitration. If “Channel Injection” is the arbitration winner, a conversion is started for the analog channel specified in the channel injection control register CHIN. The settings of the external multiplexer and the resolution of the ADC are also taken from this register.

Starting a conversion causes the channel injection request bit CINREQ to be cleared. The channel injection arbitration participation flag AP.CHP is automatically cleared if the channel injection control register and the back-up register contain no valid request.

If a currently running conversion initiated by “Channel Injection” is cancelled, the arbiter restores the conversion information in a back-up register for this channel. In this context, conversion information refers to the conversion request bit, the setting for the external

Analog-to-Digital Converter (ADC)

multiplexer, and the settings of the ADC's resolution. If the back-up register contains valid conversion information, the arbiter reads from the back-up register instead from the channel injection control register. Thus, the previously cancelled conversion participates in arbitration once again. A new conversion requested via the conversion request control register will be performed after the request in the back-up register is served.

The request bit of the channel injection control register and the backup register can be cleared under software control. Clearing the arbitration participation bit clears either the request bit in the request register (if the back-up register contains no request) or the request bit in the back-up register (if the back-up register contains a valid request).

As already mentioned, "Channel Injection" generates sequential conversion requests for analog channels either with the Inject-Wait or the Cancel-Inject-Repeat functionality.

- **Channel Injection with Inject-Wait** provides the means to wait until the current conversion with higher priority is finished before the requested conversion is injected.
- **Channel Injection with Cancel-Inject-Repeat** "Cancels" a currently performed conversion, "Injects" the requested conversion, and finally "Repeats" the previously cancelled conversion. The Cancel-Inject-Repeat feature is enabled if bit CHIN.CIREN is set. When using this feature, the currently performed conversion is cancelled if its source arbitration level is lower than the source arbitration level of channel injection. If a currently performed conversion is cancelled, a new request is generated for this conversion. Thus, the previously cancelled conversion participates in the arbitration again.

The following examples provide an overview on the behavior of the conversion request source "Channel Injection".

Figure 25-11 shows the functionality of conversion requests generated by "Channel Injection" with Inject-Wait feature. The conversion requested with a source-arbitration-level of 'L3' waits until the currently performed conversion with a source-arbitration-level of 'L1' is finished. The second channel injection request is delayed until both conversions requested with a source-arbitration-level of 'L2' are finished.

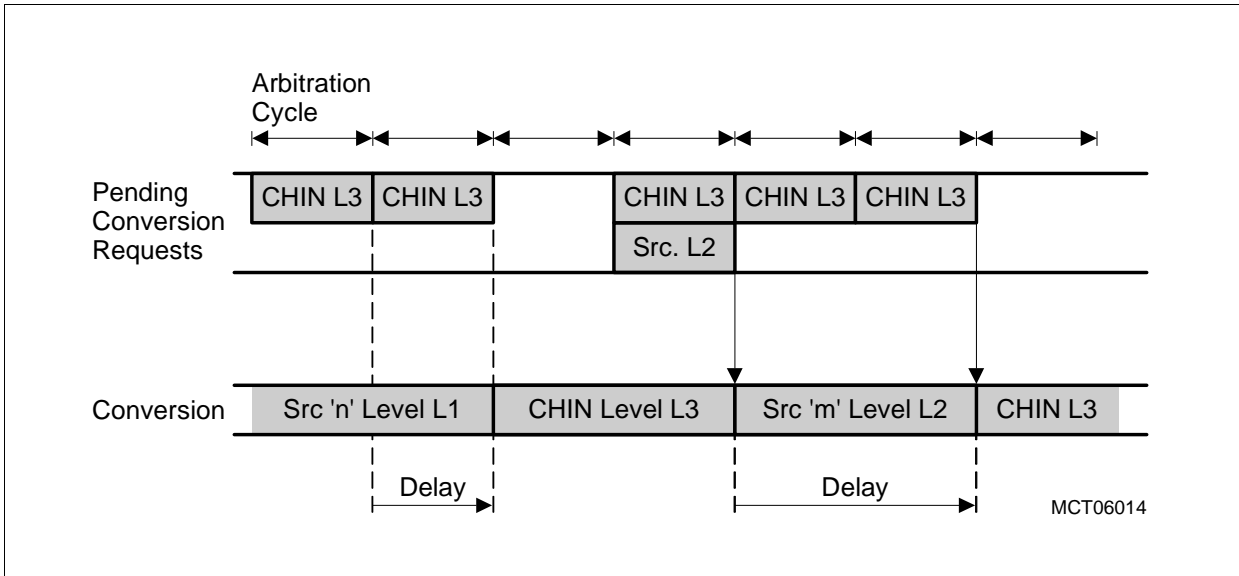


Figure 25-11 Channel Injection with Inject-Wait

Figure 25-12 shows the behavior of conversion requests generated by “Channel Injection” using the Cancel-Inject-Repeat feature. In the first case, the currently performed conversion is cancelled, because its source arbitration level of ‘L2’ is below the source arbitration level of ‘L1’ of “Channel Injection”. A new conversion request is generated for the cancelled conversion in order to restart this cancelled conversion later. This new request participates in arbitration and will be selected for repetition due to its priority level. The second injection request with a source arbitration level of ‘L4’ is delayed, even if the Cancel-Inject-Repeat feature is enabled.

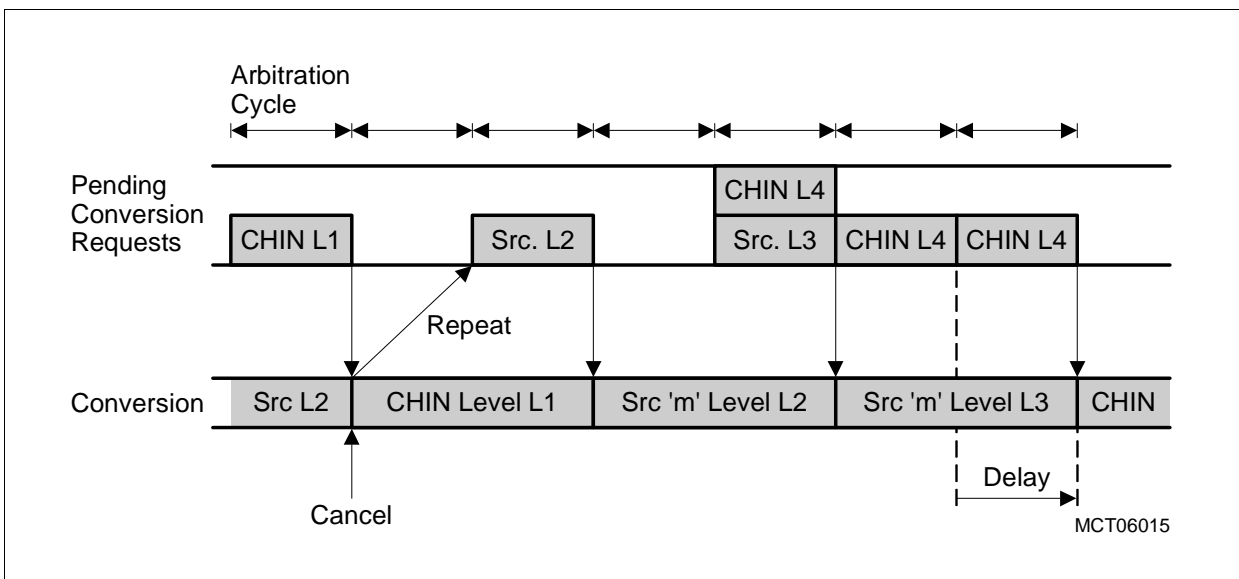


Figure 25-12 Channel Injection with Cancel-Inject-Repeat Feature

Analog-to-Digital Converter (ADC)

Figure 25-13 shows the cooperation of conversions requested by “Channel Injection” and conversions triggered by “Timer” running in Arbitration Lock Mode. First, a conversion is requested by “Channel Injection” with a source arbitration level of ‘L3’ using the Cancel-Inject-Repeat feature, during which the arbitration is locked by the timer. This request is delayed until the timer triggered conversion is finished or until “Channel Injection” is programmed to a higher priority than the timer. Second, a conversion is requested by “Channel Injection” with a source arbitration level of ‘L1’ with the Cancel-Inject-Repeat feature selected, during which the arbitration is locked by the timer. In this case, the arbitration lock is not taken into account because the timer was programmed on source arbitration level ‘L2’. Even a currently running timer triggered conversion would have been cancelled and participates in arbitration anew.

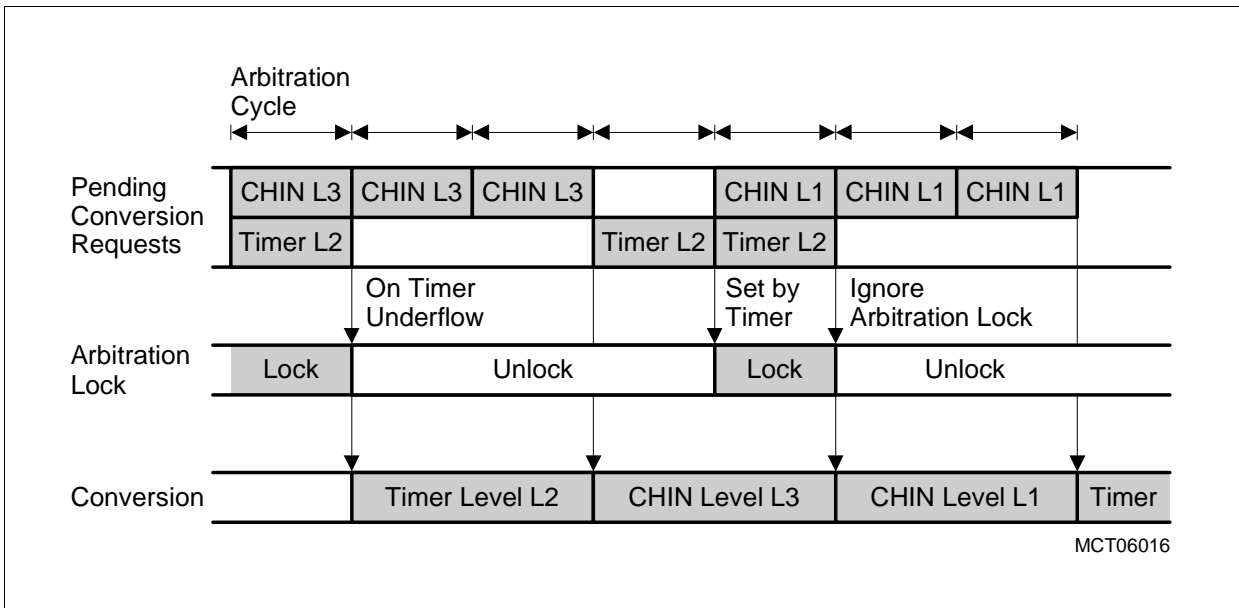


Figure 25-13 Channel Injection and Timer Triggered Conversion

25.1.2.8 Sequential Conversion Request Source “Queue”

The conversion request source “Queue” with its queue storage block is designed to handle and store burst transfers of conversion requests. Dedicated queue filling-state control logic can be used to request the next burst transfer of data while the queue’s filling level is below a predefined level.

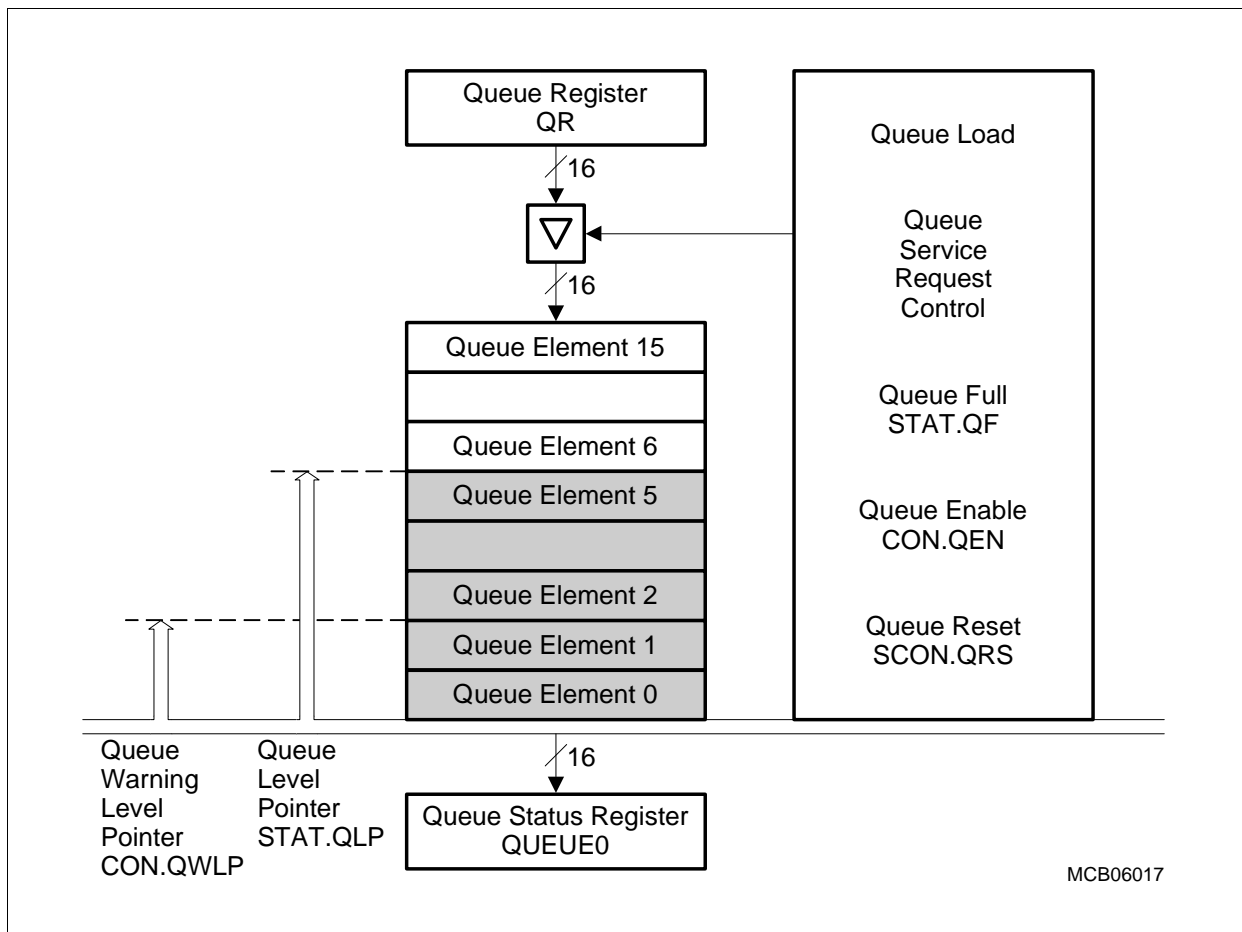


Figure 25-14 Queue Storage Block Diagram

As shown in [Figure 25-14](#), the queue consists of a queue register QR, sixteen queue elements, queue status register QUEUE0, and the queue control logic.

The queue control logic includes the queue load logic, a queue level pointer, a queue warning limit pointer, the queue based service request control block, as well as control and status flags to monitor and control the queue state.

The queue register, the queue status register, and each of the sixteen queue elements contain a valid bit (V), an analog input group selection bit (GRPS), external multiplexer control bits (EMUX), A/D conversion resolution control bits (RES), and the channel number for which a conversion should be started (CHNR).

Analog-to-Digital Converter (ADC)

The queue is automatically filled by writing valid data to the queue register QR. Valid data means that at least the V bit is set, while “zero” is a valid option for the external multiplexer setting, the resolution control bit field and the channel number. Valid data in the queue register (QR.V, QR.GRPS, QR.EMUX, QR.RES and QR.CHNR data) is then copied to the next empty queue element determined by the queue level pointer STAT.QLP. The queue load operation causes the valid bit in the queue register to be cleared automatically. Any software access to the queue register is denied during this copy operation. No queue load is performed if the queue state is full (STAT.QF is set) and the queue register contains valid data.

As shown in [Figure 25-14](#), queue elements zero to five contain valid data; therefore, the queue register’s contents are copied to queue element six.

The queue level pointer STAT.QLP indicates the number of valid queue elements. It is incremented after a queue load operation. It is decremented after a queue based conversion is started or after the queue participation flag is cleared. STAT.QLP is cleared after a queue reset operation by setting the queue reset bit. Note that there are sixteen valid queue elements in the queue if the queue level pointer is $0F_H$ and the queue full bit is set.

The queue warning limit pointer CON.QWLP can be used to generate service requests, based on a queue element state change. The value of the queue warning limit pointer must be programmed with a value “n” in order to focus on a state change from valid to invalid of queue element “n”. A queue based service request can be triggered in this case, thus requesting the next transfer of data to the queue. If the queue element specified by (CON.QWLP)+1 becomes invalid after a conversion, the module service request flag MSS1.MSRQR is automatically set. The service request destination node pointer (PQR) must be configured and enabled (ENPQR) in order to trigger a service request node assigned to the queue.

The conversion request source “Queue” consists of the queue status register QUEUE0, a backup register, and a queue arbitration participation flag AP.QP, as shown in [Figure 25-15](#). The contents of queue element number zero are represented in the queue status register QUEUE0. Therefore, set/clear actions of the valid-bit of the queue status register QUEUE0 are also performed on queue element zero.

If at least one queue element contains valid data, this (these) valid bit(s) cause(s) the queue arbitration participation flag to be set. This informs the arbiter to include the conversion request source “Queue” into arbitration. If “Queue” is the arbitration winner, a conversion is started for the analog channel specified within the queue status register. The settings of the external multiplexer and the resolution of the A/D Converter are also derived from this register.

Starting a queue based conversion causes the valid bit of the queue status register QUEUE0 to be cleared by the arbiter. The contents of all queue elements containing valid data slides one step down. For example, queue element one contains valid data, this data “slides down” to queue element zero. Queue based conversion requests are

Analog-to-Digital Converter (ADC)

generated for the control information of register QUEUE0, if the queue is enabled (bit CON.QEN = 1), the queue status register contains valid data (QUEUE0.V is set) and the queue gating line QGT = 1. The arbitration participation flag is automatically cleared if all queue elements, the queue status register (remember: QUEUE0 represents the contents of queue element number zero) and the back-up register contain no valid request.

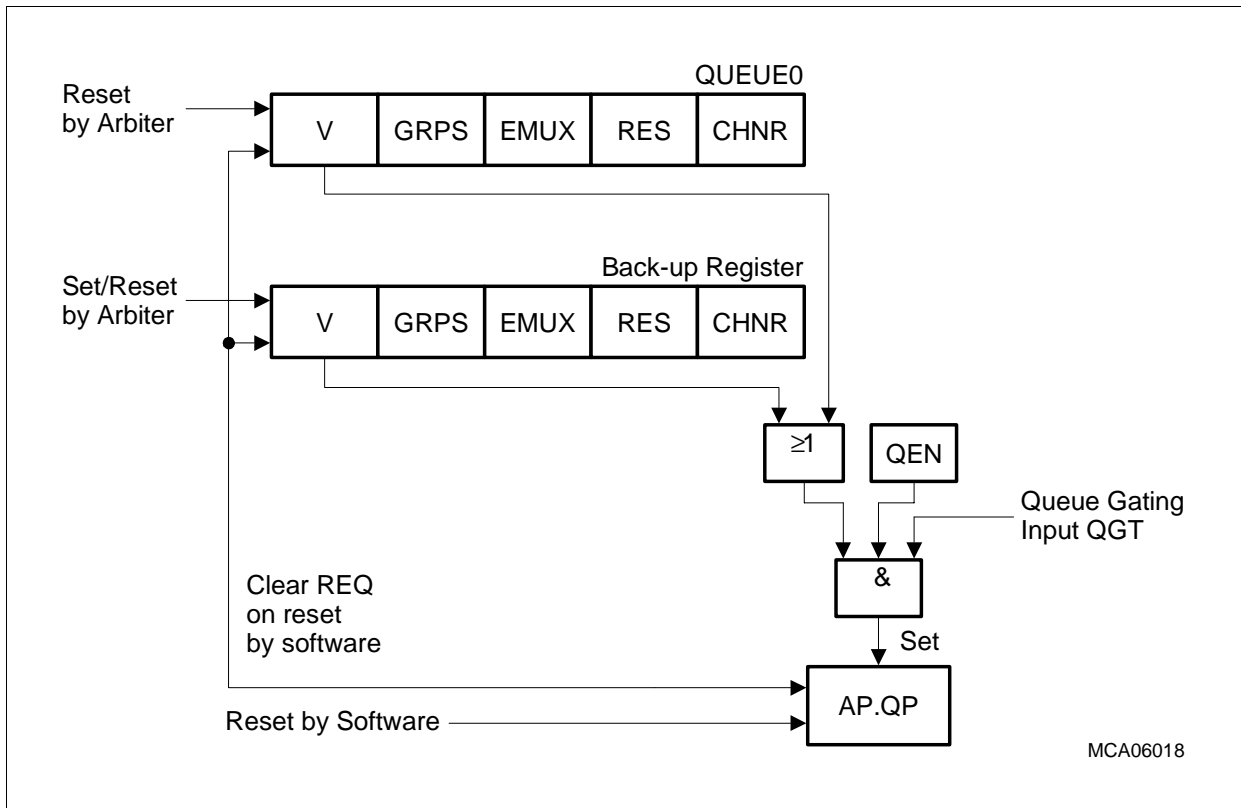


Figure 25-15 Conversion Request Source "Queue"

If a currently running conversion initiated by "Queue" is cancelled, the arbiter restores the conversion information in the back-up for this channel. In this context, conversion information refers to the conversion request bit, the setting for the external multiplexer and the settings of the A/D Converter's resolution. If the back-up register contains valid conversion information, the arbiter reads from the back-up register instead of the queue status register. Thus, the previously cancelled conversion participates in arbitration once again. A conversion requested via the queue storage block (register QUEUE0) will be performed after the request in the back-up register is served.

The valid bit (V bit) of the queue status register and the back-up register can be cancelled under software control. Clearing the queue arbitration participation bit clears either the valid bit in the queue status register (if the back-up register contains no request) or the request bit in the back-up register (if the back-up register contains a valid request). If the valid bit of the queue status register is cleared, a slide operation is

Analog-to-Digital Converter (ADC)

performed equal to the slide operation after starting a queue based conversion. The enable bit CON.QEN can be set by hardware if the queue trigger request line QTR = 1. The QTR and QGT input connections depend on the product specific implementation of the ADC module. They are described in [Section 25.3.6.2](#).

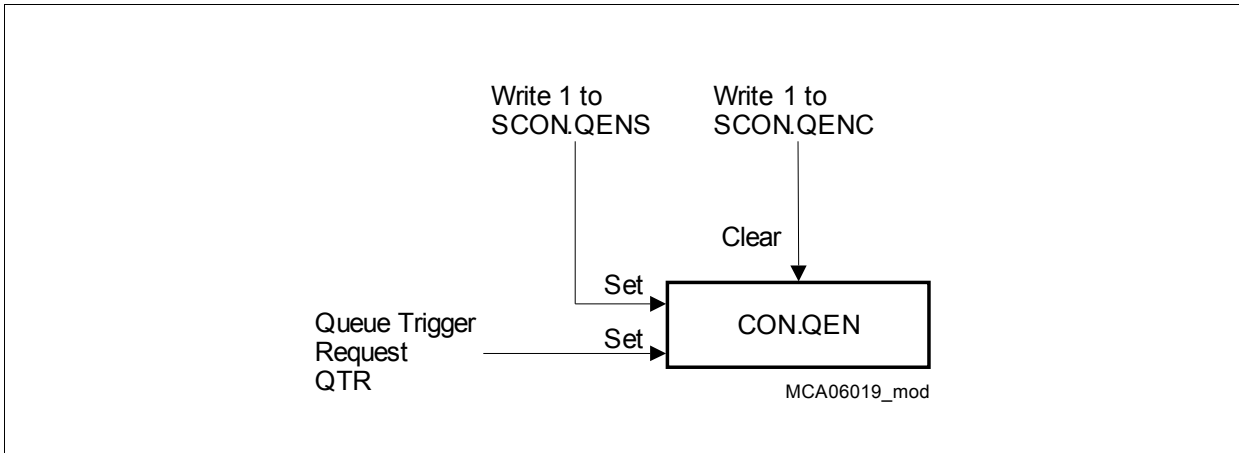


Figure 25-16 Modification of Bit CON.QEN

25.1.3 Conversion Request Arbitration

Because several conversion request sources can generate conversion requests at the same time, an arbitration mechanism is implemented in order to detect the conversion request source and channel with the highest priority. **Figure 25-17** shows the arbitration scheme with the associated controls.

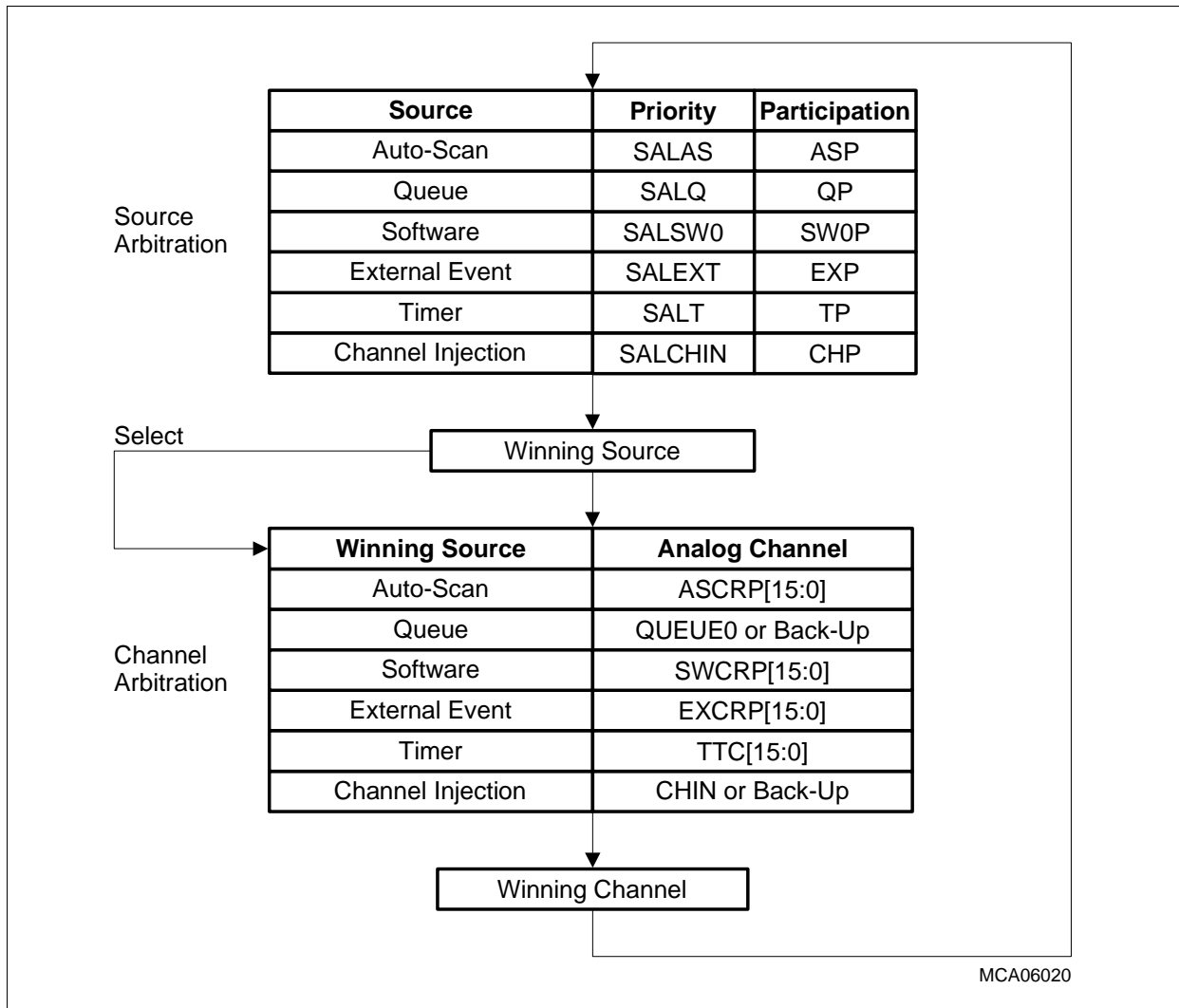


Figure 25-17 Arbitration

Arbitration of pending conversion requests is performed according to the following two stage prioritization algorithm:

- **Source arbitration** is the first stage in the arbitration algorithm. Starting with the conversion request source “Auto-Scan” up to “Channel Injection”, each source is checked if its arbitration participation flag is set. If the participation flag is set and its priority is higher than the priority of the other selected sources, that source is the winner of the arbitration.

Analog-to-Digital Converter (ADC)

- **Channel arbitration** follows after source arbitration. For the winning source, channel arbitration is performed. Within the second stage of the arbitration algorithm, the pending conversion request with the highest priority is detected. If a parallel source is the winning source, the flag representing the highest channel number within the conversion request pending register is determined. If a sequential source is the winning source, the channel in the request register or in the back-up register is determined. Note that a pending request in the back-up register is preferred.

The arbitration result consists of the winning source and channel number. A start of conversion can occur, if the A/D Converter is idle or if the arbitration winner has permission to cancel a currently running conversion. After the conversion has started, the corresponding pending conversion request is automatically reset. Attempt to start a conversion for this arbitration result will be repeated until either the start is successful (other conversion is currently running) or a new result (source and channel number with a higher priority) was arbitrated.

25.1.3.1 Source Arbitration Level

The priority of each conversion request source can be programmed individually in the corresponding bit fields of the source arbitration level register SAL. The priority of a source is named as source arbitration level and it determines the order in which pending conversion requests from different sources are performed. A low number of the source-arbitration-level represents a high priority and vice versa.

After initialization, an individual source arbitration level is assigned to each source. "Channel Injection" has the highest priority, while "Auto-Scan" has the lowest priority. These predefined priority levels can be reprogrammed to adapt the ADC's functionality to the requirements of the application.

It is recommended that source arbitration levels should be reprogrammed while no conversion request is pending, as any modification of the source arbitration level register immediately affects the arbitration scheme. Each source should have an individual priority level. Nevertheless, if several conversion request sources have been programmed to the same priority level, the first detected source within this group of identical levels is taken into account.

25.1.3.2 Arbitration Participation Flags

Each source has an arbitration participation flag located in the arbitration participation register AP. An arbitration participation flag set to 1 indicates, that at least one conversion request has been generated by this source and that this source participates in the arbitration.

An arbitration participation flag is automatically cleared by hardware if no conversion request is pending for this source (if all requested conversions have been started).

Analog-to-Digital Converter (ADC)

The arbitration participation flag can also be cleared under software control. Writing a 0 to the corresponding flag clears the arbitration participation flag. All bits in the corresponding conversion request pending register are cleared if a participation flag of a parallel source is cleared under software control. If a participation flag of a sequential source is cleared, the following action is performed:

- **ONLY** the request bit of the back-up register is cleared, **if the back-up register contains valid data**. The request bit of the corresponding conversion request register (CHIN or QUEUE0) is **not** cleared in this case.
- **OR** the request bit of the corresponding conversion request register (CHIN or QUEUE0) is cleared, **if the backup register does not contain valid data**.

Note: Writing a 1 to a participation bit has no effect.

25.1.3.3 Cancel Functionality

Channel Injection and Synchronized Injection have the ability to cancel a currently running conversion. If a conversion is cancelled, one of the following actions are performed:

- If a conversion initiated by a parallel source is cancelled, the conversion request flag is automatically set again in the corresponding conversion request pending register.
- If a conversion initiated by a sequential source is cancelled, the control information (such as resolution, external multiplexer information, etc.) of the cancelled conversion is rescued into the backup register (for example: queue based conversion is cancelled, so the queue backup register receives the control information of the cancelled conversion).

Afterwards, the request participates in the arbitration anew and will be served according to its source-arbitration level.

25.1.3.4 Clear of Pending Conversion Requests

This feature can be used to save conversion time by handling more than one conversion request at the same time.

Clear of Pending Conversion Requests in Parallel Sources

If several conversion requests are pending for the same analog channel and a conversion for this analog channel has been started, all pending conversion requests of **parallel** sources can be cancelled for this analog channel by the arbiter (CON.CPR = 1). For example, if timer, software and auto-scan trigger a conversion request for the same analog channel, only one conversion is started for this analog channel. The other two pending conversion requests will be automatically cancelled by the arbiter. Note that the conversion will be started for the arbitration winner, the source with the highest priority. The conversion result is valid for all parallel sources which requested this channel. A service request is generated only for the source that caused the processed conversion.

Individual Clear of Pending Conversion Requests

If several conversion requests are pending for the same analog channel, this channel will be converted several times until all pending conversion requests are performed. This is the default setting after reset (CON.CPR = 0).

25.1.3.5 Arbitration and Synchronized Injection

The master of a Synchronized Injection provides no separated source for this feature. The behavior of a Synchronized Injection is specified by the original requesting source. In the slave A/D Converter, a request for a Synchronized Injection **always** has the highest priority. A request for a synchronized request in a slave module does not participate in the arbitration cycle. This synchronized request is immediately set as the arbitration winner. This request remains until it is served or it is cancelled by the master.

25.1.3.6 Arbitration Lock

If the timer runs in Arbitration Lock Mode and the current timer value TSTAT.TIMER is equal to or below the arbitration lock boundary, the arbitration lock bit STAT.AL is set. Setting the arbitration lock bit also sets the timer participation flag. In this way, the timer source can participate in the arbitration cycle without any pending request. Such an arbitration participation by the timer without a pending request denies all currently pending sources that have a source-arbitration-level below the timer source as arbitration winner. All sources with a source-arbitration-level greater than the timer source retain their possibility to win the arbitration. If the timer wins the arbitration without a pending request, no conversion will be started for this arbitration winner. This case can occur if bit AP.TP is set while no bit is set in register TCRP. This feature can be used to guarantee that no conversions can be started for lower prioritized sources.

Note: The timer participation flag is also set by any pending timer conversion request in register TCRP.

- 4. If any source has the same source-arbitration-level as the timer source, the result of the arbitration cycle depends on the position of this source compared to the timer source. If this source is checked before the timer source, this can be the arbitration winner. If this source is checked after the timer source, this source cannot be the arbitration winner.*

25.1.4 Clock Circuit

The clock divider blocks shown in **Figure 25-18** determine the clock frequencies in the ADC module and the conversion and sample timing.

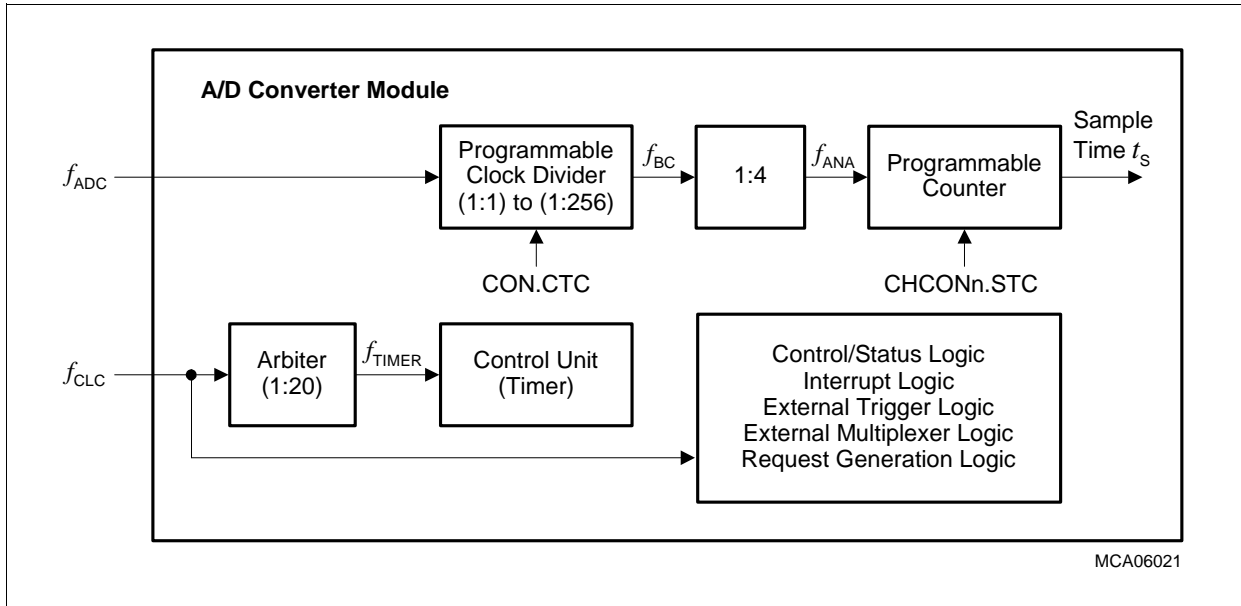


Figure 25-18 Clock Control Structure

The following definitions for the A/D Converter clocks are used in this chapter:

- f_{CLC} : Module control clock
- f_{ADC} : Module timing clock
- f_{BC} : Basic operating clock
- f_{ANA} : Internal A/D Converter clock
- f_{TIMER} : Arbiter clock

The conversion time is composed of the sample time, the time for the successive approximation, and the calibration time. **Table 25-6** shows the conversion time t_C based on the sample time t_S , basic operating clock frequency f_{BC} and the module timing clock f_{ADC} ($t_{BC} = 1 / f_{BC}$, $t_{ADC} = 1 / f_{ADC}$).

Table 25-6 Conversion Time t_C

A/D Converter Resolution	Conversion Time t_C
8-bit	$t_S + 40 t_{BC} + 2 t_{ADC}$
10-bit	$t_S + 48 t_{BC} + 2 t_{ADC}$
12-bit	$t_S + 56 t_{BC} + 2 t_{ADC}$

Note: The TC1796 basic operating clock frequency f_{BC} influences the maximum allowable internal resistance of the used reference voltage supply.

25.1.4.1 Conversion Principles

After reset, a power-up calibration is automatically performed in order to correct gain and offset errors of the A/D Converter. The ongoing power-up calibration is indicated in the A/D Converter status register by an activated calibrate bit STAT.CAL. To achieve best calibration results, the reference voltages as well as the supply voltages must be stable during the power-up calibration.

When a conversion is started, the capacitances of the converter are loaded first via the respective analog input channel to the analog input voltage. The time to load the capacitances is referred to as sample time t_S . The sample phase is indicated by an activated status bit STAT.SMPL in the A/D Converter status register. Next, the sampled voltage is converted to a digital value. Finally, an internal self calibration adapts the analog converter module to changing temperatures and device tolerances. The conversion and calibration phase is indicated by the busy signal STAT.BUSY, which goes inactive at the end of the calibration phase.

Note: During the power-up calibration, no conversion should be started.

25.1.4.2 Conversion Timing Control (CTC and CPS)

The module clock f_{ADC} is generated in the ADC Clock Generation unit (see [Page 25-103](#)). The A/D Converter's basic operating clock frequency f_{BC} is derived from f_{ADC} via the programmable clock divider CTC, which provides dividing factors from 1:1 to 1:256.

$$f_{BC} = \frac{f_{ADC}}{CTC + 1} \quad (25.2)$$

The A/D Converter's basic operating clock frequency f_{BC} must not exceed 40 MHz. It must also not drop below 2 MHz.

The internal A/D Converter clock frequency f_{ANA} is a quarter of the basic operating clock frequency f_{BC} (min. 0.5 MHz, max. 10 MHz). The internal A/D Converter clock f_{ANA} is related to f_{ADC} according to the following equation:

$$f_{ANA} = \frac{f_{BC}}{4} = \frac{1}{4} \times \frac{f_{ADC}}{CTC + 1} \quad (25.3)$$

With the clock control bit field CON.CTC, the internal A/D Converter clock f_{ANA} can be adjusted to different module timing clock frequencies f_{ADC} in order to optimize the performance of the TC1796 A/D Converter. Note that CON.CTC may be changed during a conversion, but will be evaluated after the currently performed conversion is finished.

Table 25-7 Conversion Timing Control

CON.CTC	f_{BC}	t_{BC}	f_{ANA}	t_{ANA}
00 _H	f_{ADC}	$1 / f_{ADC}$	$f_{ADC} / 4$	$4 / f_{ADC}$
01 _H	$f_{ADC} / 2$	$2 / f_{ADC}$	$f_{ADC} / 8$	$8 / f_{ADC}$
02 _H	$f_{ADC} / 3$	$3 / f_{ADC}$	$f_{ADC} / 12$	$12 / f_{ADC}$
03 _H	$f_{ADC} / 4$	$4 / f_{ADC}$	$f_{ADC} / 16$	$16 / f_{ADC}$
...
FF _H	$f_{ADC} / 256$	$256 / f_{ADC}$	$f_{ADC} / 1024$	$1024 / f_{ADC}$

25.1.4.3 Sample Timing Control

The sample time control determines the duration of the sample phase of a conversion, that is, the period during which the channel input capacitance is charged/discharged by the selected analog signal source. The duration of the sample phase is programmed individually for each channel via sample time control bit field CHCONn.STC. Any modification of CHCONn.STC will be evaluated after the currently performed conversion is terminated.

The sample time t_S depends on the basic operating clock f_{BC} and the programmable value of bit field CHCONn.STC. The sample time t_S is selected in periods of $t_{BC} = 1 / f_{BC}$ within the range from $8 \times t_{BC}$ up to $1028 \times t_{BC}$.

The sample time t_S is calculated according to the following equation:

$$t_S = 4 \times (\text{STC} + 2) \times t_{BC} \quad (25.4)$$

Table 25-8 shows the selectable values of CON.STC and the resulting ADC basic operating clock f_{BC} and sample time t_S .

Table 25-8 Sample Time Control

CHCONn.STC	Sample Time t_S
00 _H	$8 \times t_{BC}$
01 _H	$12 \times t_{BC}$
02 _H	$16 \times t_{BC}$
03 _H	$20 \times t_{BC}$
...	...
FF _H	$1028 \times t_{BC}$

Note: The duration of the sample phase influences the maximum allowable internal resistance of the respective analog input signal source.

25.1.4.4 Power-Up Calibration Time

When the A/D Converter becomes clocked after a reset operation, a power-up calibration is automatically performed in order to correct gain and offset errors of the A/D Converter. A running power-up calibration is indicated by status flag STAT.CAL. To achieve best calibration results, the reference voltages as well as the supply voltages must be stable during the power-up calibration. The first A/D conversion can be started after the power-up calibration is finished (STAT.CAL = 0).

The power-up calibration takes 3840 clock cycles of the A/D Converter clock f_{ANA} . After a reset operation, the A/D Converter is disabled and becomes enabled when registers ADC0_CLC and ADC0_FDR are written with appropriate values. Power-up calibration starts when the analog part of the A/D Converter is clocked after a reset operation.

The following example shows the setup for the fastest achievable (best case) power-up calibration time at $f_{SYS} = 75$ MHz. Note that the maximum frequency of f_{ANA} must not exceed 10 MHz. See also [Page 25-103](#) for further details on the ADC module clock generation.

Example ($f_{ADC} = 75$ MHz):

- $f_{SYS} = f_{CLC} = 75$ MHz
- $f_{ADC} = f_{CLC} / 1.875 = 40$ MHz (Setting FDR for divider ratio 1.875)
- $f_{BC} = f_{ADC} / 1 = 40$ MHz (CON.CTC = 00_H)
- $f_{ANA} = f_{BC} / 4 = 10$ MHz (fixed divider, f_{ANA} max. frequency)
- $t_{ANA} = 1 / f_{ANA} = 0.1$ μ s

These values result in a best case power-up calibration time of $3840 \times t_{ANA} = 384$ μ s.

The best case power-up calibration time as discussed above is of course increased by decreasing the f_{ANA} clock frequency. Note that the changing of clock related parameters is not recommended during a running power-up calibration.

25.1.5 Reference Voltages (V_{AREF} and V_{AGND})

The digital result of a conversion represents the analog input as a fraction of the reference ($V_{AREF} - V_{AGND}$) in steps of 2^{-n} by n-bit resolution:

$$\text{Result} = 2^n \times (V_{AIN} - V_{AGND}) / (V_{AREF} - V_{AGND}) \quad (25.5)$$

The ADC module offers the choice of four selectable reference voltages $V_{AREF}[0]$ to $V_{AREF}[3]$. The reference voltage can independently be selected for each analog channel via the respective bit field CHCONn.REF. $V_{AREF}[0]$ corresponds to the positive reference voltage V_{AREF} and is used for self calibration of the A/D Converter. Therefore, it must be stable during all conversions, even for those that use another reference voltage.

The reference voltages must fulfill the following specifications:

$$V_{AREF}[3:0] \leq V_{DDM} + 0.05 \text{ V}; V_{DDM} \leq 3.3 \text{ V} \quad (25.6)$$

A conversion with low reference voltage affects the accuracy of the A/D Converter. The TUE of an A/D Converter that is operated at a reduced positive reference voltage can be evaluated according to the following equations:

$$\text{TUE}|_A \rightarrow \text{TUE}|_B = K \times \text{TUE}|_A, (K \geq 1) \quad (25.7)$$

with factor K as:

$$V_{AREF}|_A \rightarrow V_{AREF}|_B = 1/K \times V_{AREF}|_A \quad (25.8)$$

where

- $V_{AREF}|_A$: minimum positive reference voltage range is specified for $0 \text{ V} \leq V_{AREF} \leq V_{DDM} + 0.05 \text{ V}$
- $V_{AREF}|_B$: positive reference voltage, which is below the specified range
- $\text{TUE}|_A$: total unadjusted error for reference voltages within the specified range
- $\text{TUE}|_B$: total unadjusted error for reference voltages below the specified range

Note: All unused analog input pins must be connected to a fixed potential either V_{AGND} or $V_{AREF}[0]$ to avoid disturbance of active analog inputs.

Note: It is not recommended in general to set V_{AREF} below 50% of V_{DDM} .

Note: The analog input voltages V_{AIN} must be in the range between V_{AGND} and the selected V_{AREF} .

25.1.6 Error through Overload Conditions

An additional error can occur when overloading an analog input (such as channel D). In this case, an additional leakage current exists between the analog input D and the adjacent analog inputs $D \pm 1$, affecting the conversion result of an analog input channel D by an additional error AEL based on the additional sampled voltage V_{AEL} (Analog Error Leakage).

$$V_{AEL} |_{D \pm 1} = R_{AIN} |_{D} \times |I_{OV} |_{D} \times k_A \quad (25.9)$$

The coupling factor k_A determines the physical relation of two adjacent analog inputs. The resulting error AEL is given by:

$$AEL |_{D \pm 1} = \frac{R_{AIN} |_{D} \times |I_{OV} |_{D} \times k_A}{V_{AREF}} \quad (25.10)$$

where

- V_{AREF} : reference voltage for conversion
- $R_{AIN} |_{D}$: resistance of the analog input channel D
- $I_{OV} |_{D}$: overload current of the analog input D
- AEL: additional error caused by a leakage current, related to V_{AREF}
- k_A : coupling factor for the analog input D

Note: If AEL is calculated in bit units, AEL must be multiplied by $2^n - 1$.

25.1.7 Limit Checking

Limit checking provides the means to check conversion results on exceeding or becoming lower than a defined limit. The checking parameters can be configured individually for each analog channel. Service requests can be generated for each analog channel on limit checking results such as on a limit violation or on successful limit checks.

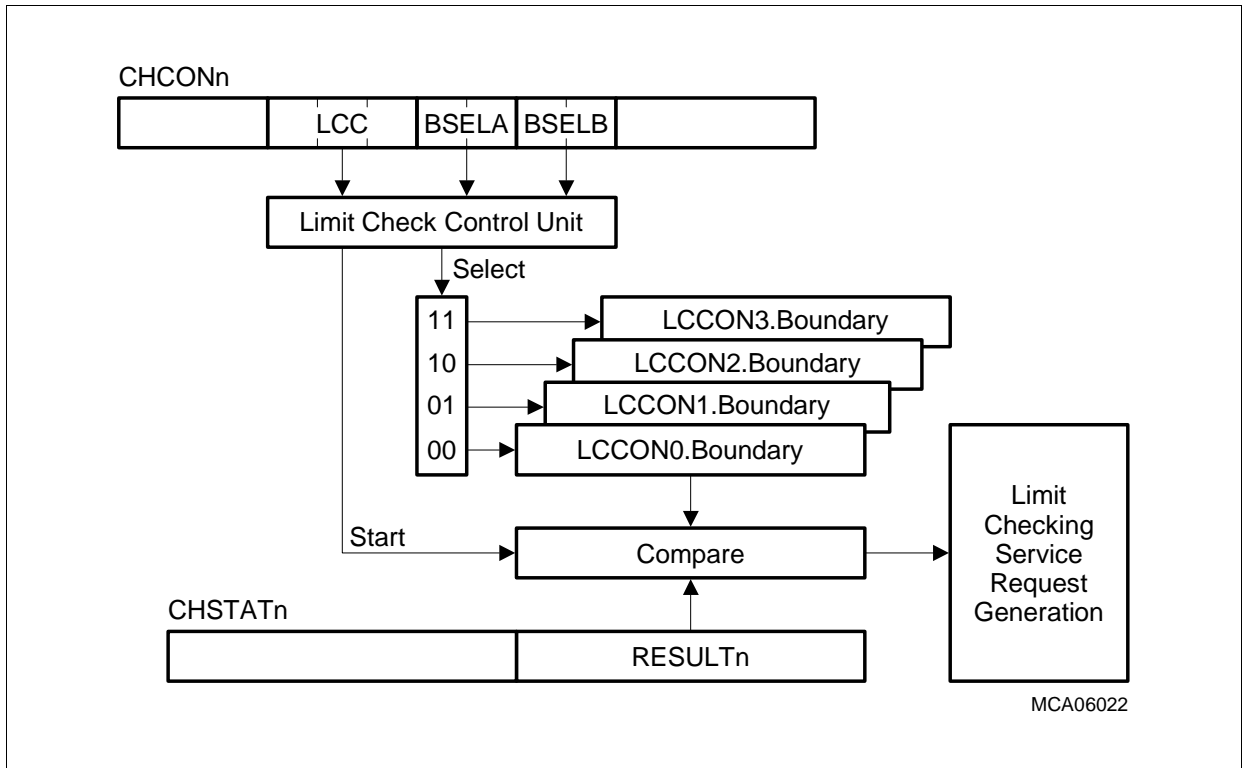


Figure 25-19 Limit Check Unit

As shown in [Figure 25-19](#), a limit check is performed for the conversion result stored in a specific channel status register. For limit checking, the A/D Converter’s measuring range is divided into three areas in order to check whether the conversion result meets the specified range. Two out of four boundaries can be selected and programmed per limit check. The boundaries are selected for each analog channel via the bit fields CHCONn.BSELA and CHCONn.BSELB, n = 0-15. Four boundaries can be set individually in the limit check control register LCCON0/1/2/3.

The limit check control bit field specifies if a limit check is performed for the current conversion result and which area must be met or avoided by the current conversion result (see [Figure 25-20](#)).

Depending on the selected limit check control parameter CHCONn.LCC, the service request flag is not set, is set if the selected area is hit, or is set if the selected area is missed for the related conversion result.

Figure 25-20 shows the selectable parameters for limit check.

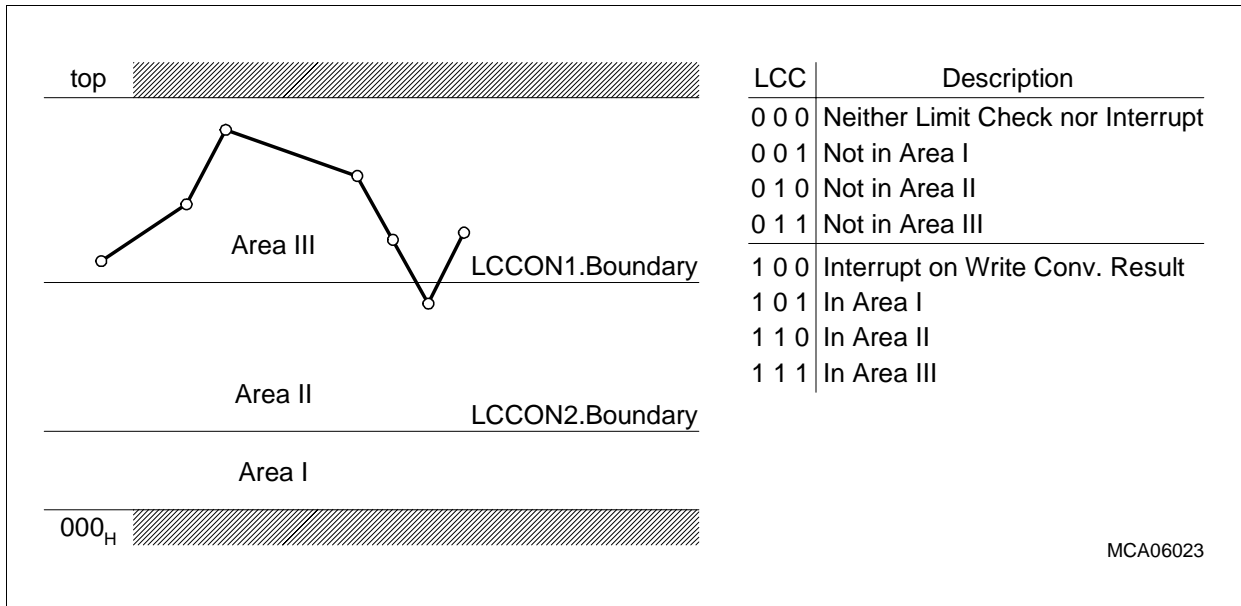


Figure 25-20 Limit Checking

The A/D Converter's measuring range is divided into the following three areas:

- Area I: From 000_H to (including) the lower boundary
- Area II: excluding the lower boundary to (including) the upper boundary
- Area III: excluding the upper boundary to top (top due to the selected resolution)

The value stored in LCCONn.BOUNDARY represents a boundary, that is selected by the channel-specific bit fields CHCONn.BSELA/B. Neither boundary A (selected by CHCONn.BSELA) nor boundary B (selected by CHCONn.BSELB) is fixed in its assignment as a lower or upper one. The boundary's value specifies whether it is assumed to be the upper or lower one.

In this example, channel number 5 is configured for limit checking. CHCON5.BSELA is set to 10_B and selects the boundary stored in LCCON2.BOUNDARY. CHCON5.BSELB is configured to 01_B and selects the boundary stored in LCCON1.BOUNDARY. Since the value of LCCON1.BOUNDARY is above than the value of LCCON2.BOUNDARY, it is assumed to be the upper one while the boundary stored in LCCON2.BOUNDARY is the lower one.

25.1.8 Expansion of Analog Channels

The number of analog inputs can be expanded in a very flexible and powerful way to satisfy the increased needs for analog inputs. In addition to the internal analog input channel group selection, external analog multiplexers can be connected to each analog channel if the following items are considered:

- Inverse current injection (overload) behavior
- ON resistance of the external multiplexer and load capacitance
- Timing of the external multiplexer
- Noise due to adjacent digital input pins

Note: The characteristics of the external multiplexers influence the accuracy of the A/D Converters. An accuracy of ± 2 LSB @ 10-bit resolution is no longer guaranteed.

Two control lines are provided to drive external multiplexers, as shown in **Figure 25-21**.

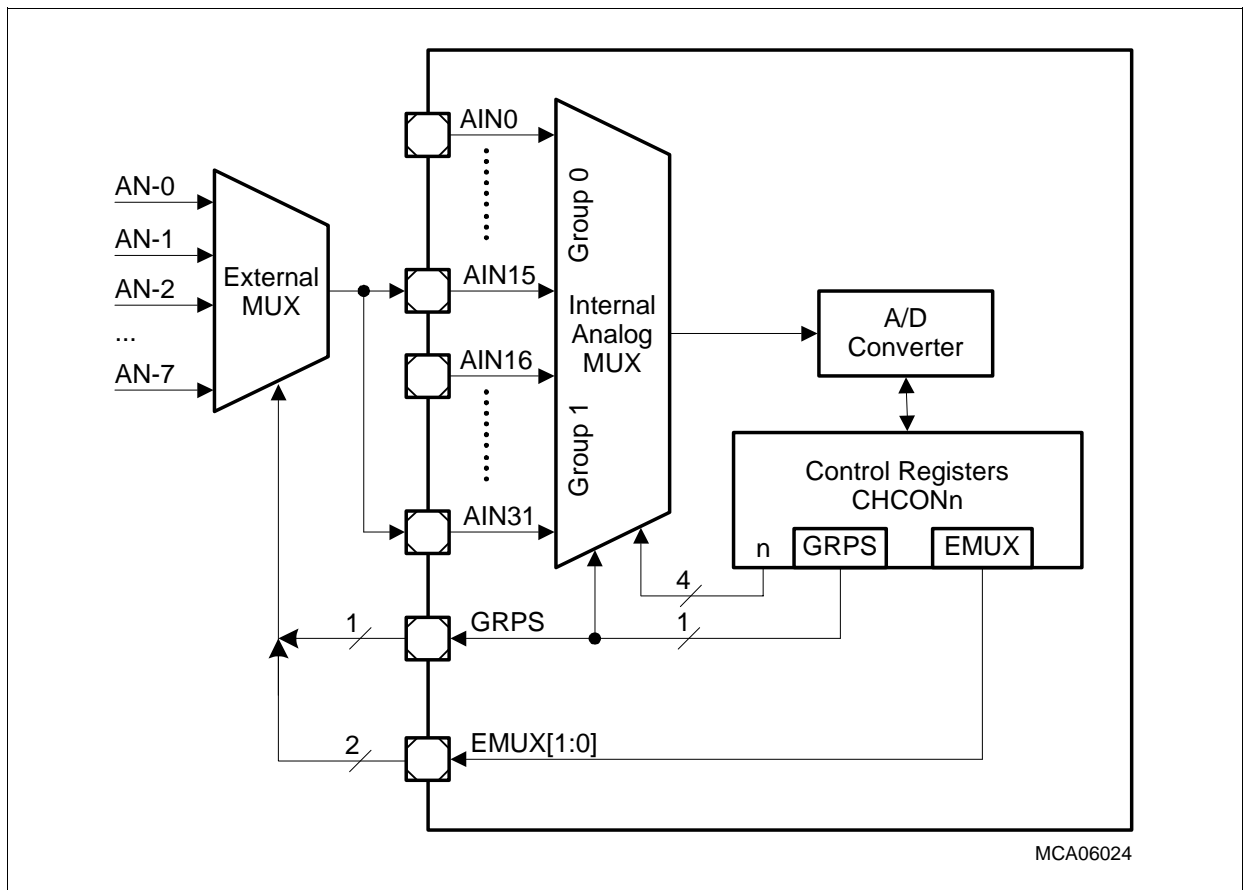


Figure 25-21 External Expansion of Analog Channels

External analog multiplexers receive the select information CHCONn.EMUX[1:0] (and optional GRPS, too) from the channel-specific control register individually for each analog channel. Sequential sources derive the external multiplexer control information from the conversion request control register (bit field CHIN.EMUX and QUEUE0.EMUX).

Analog-to-Digital Converter (ADC)

The external multiplexer controls (bit field CHCONn.EMUX) from the channel-specific control registers are not taken into account for sequential sources.

25.1.8.1 Inverse Current Injection (Overload) Behavior

An overload condition occurs when the analog input voltage is above or below the supply range. An overload condition at a channel connected to an external multiplexer (such as AN-0 in [Figure 25-21](#)) can affect the conversion of another channel connected to the same external multiplexer (such as AN-1 to AN-3), depending on the overload capability of the external multiplexer. In case of an overload condition at one channel while another channel of the same external multiplexer is sampled by the A/D Converter, an even higher conversion error must be expected.

Note: The overload behavior of every channel that is directly connected to the internal multiplexer or through another external multiplexer does not change.

25.1.8.2 On Resistance of the External Multiplexer

If an external multiplexer is connected to an analog input channel, a typical application might add an RC filter before the external multiplexer to each additional external analog inputs. For example, each of the external analog inputs AN-0 to AN-3 in [Figure 25-21](#) is adapted by an RC filter. In this case, the resistance of the external multiplexer reduces the efficiency of the external capacitors of the RC filter. An additional blocking capacitor between the external multiplexer and the analog input line could improve the noise suppression capability. However, in this case, the capacitance that must be charged, would be increased by the size of the blocking capacitor.

25.1.8.3 Timing of the External Multiplexer

An analog input channel of an external analog multiplexer is selected after the arbitration round is finished. Therefore, the information to drive an external multiplexer is available before the sample time begins.

25.1.8.4 Load Capacitance

Because each analog input of the external multiplexer might be applied by different analog voltages, the total input capacitance of the A/D Converter must be recharged within the sample time, each time an analog input channel of an external multiplexer is measured.

For analog input channels that are directly applied to the analog input pin of the A/D Converter, the input capacitance does not change. The analog voltage source of such channels must solely recharge the switched input capacitance of the A/D Converter.

25.1.9 Service Request Processing

A fully configurable and very flexible service request control structure is implemented in the A/D Converter Module (see [Figure 25-22](#)). The A/D Converter module provides a total of sixteen channels and four parallel/serial service request sources. Each service request source can be programmed to generate one of eight service request output signals SR[7:0]. The service request compressor logic allows more than one service request source to be assigned to one service request output.

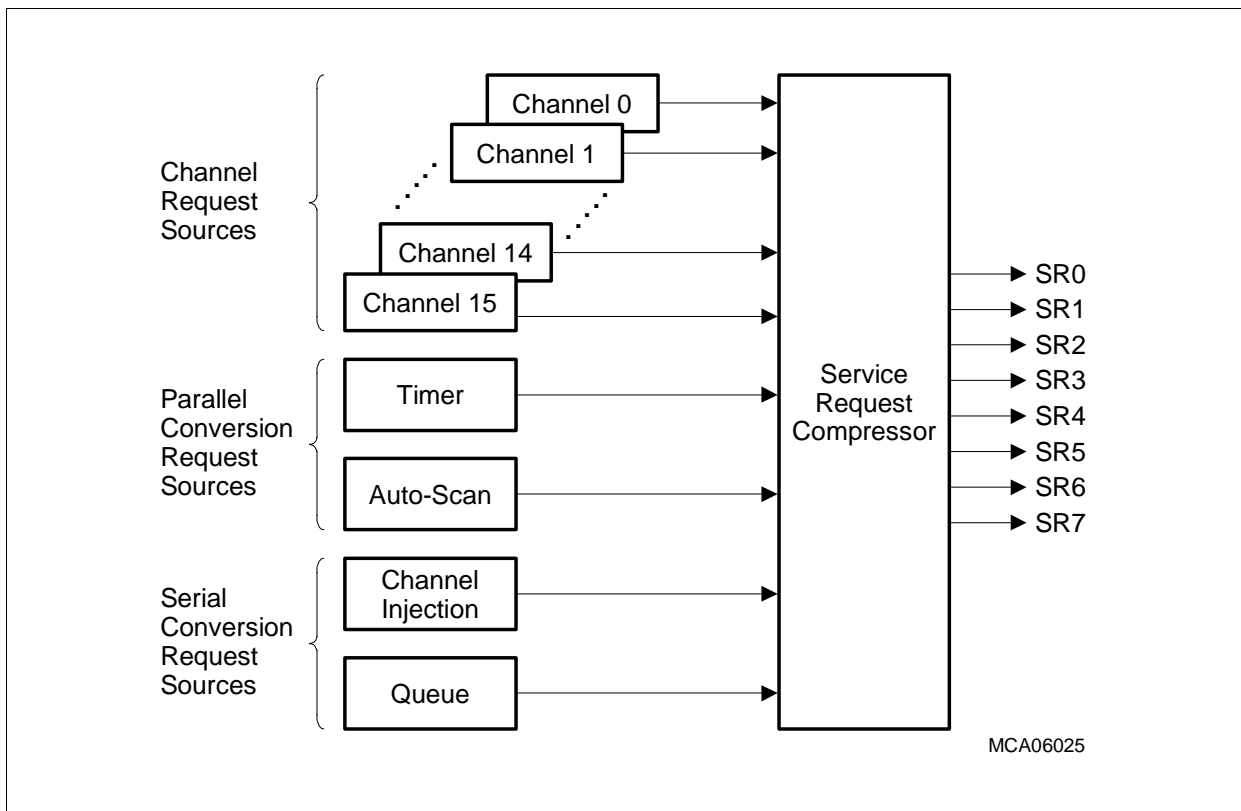


Figure 25-22 Service Request Configuration

Depending on the implementation of the A/D Converter Module in a specific microcontroller, the service request output signals SR[7:0] can be connected either to an interrupt node (service request control register) or can be used as DMA request input of a DMA controller unit. The TC1796 specific service request output configuration is described in [Figure 25-30](#) and on [Page 25-119](#).

The control logic for each of the service request sources provides the following functionality:

- Service request source selection (channel request sources only)
- Service request flag, that can be set/cleared by software, too
- Service request enable bit
- Service request node pointer

Analog-to-Digital Converter (ADC)

Table 25-9 lists the service request sources of the A/D Converter module and the related control and status flags/bits.

Table 25-9 Service Request Control/Status Bits/Flags

Service Request Source	Source Selection	Status Flag	Enable Bit	Service Request Node Pointer
Channel n ¹⁾	CHCONn.LCC ₁₎	MSS0.MSRCHn ₁₎	CHCONn.ENCH ₁₎	CHCONn.INP ₁₎
Timer	–	MSS1.MSRT	SRNP.ENPT	SRNP.PT
Queue		MSS1.MSRQR	SRNP.ENPQR	SRNP.PQR
Auto-scan		MSS1.MSRAS	SRNP.ENPAS	SRNP.PAS
Synchronous Channel Injection		MSS1.MRSY	SRNP.ENPSY	SRNP.PSY

1) n = 0-15

25.1.9.1 Channel Request Source Control

Figure 25-23 shows the request control logic of each conversion channel n. Bit field CHCONn.LCC determines which of the different request conditions is selected as channel request. These conditions are:

- Conversion result value is inside/outside a specified range (limit checking enabled)
- CHSTATn.RESULT is updated (limit checking disabled)
- No action (no channel request generated)

When a selected request condition becomes active, the channel n request status flag MSS0.MSRCHn is set and the channel n service request event is reported to the service request compressor logic (if enabled by CHCONn.ENCH). The service request flag MSS0.MSRCHn can only be cleared by software. For test purposes (CON.SRTEST = 1), the service request flag can be additionally set by hardware. Additional details on MSS0.MSRCHn software control are described on Page 25-48.

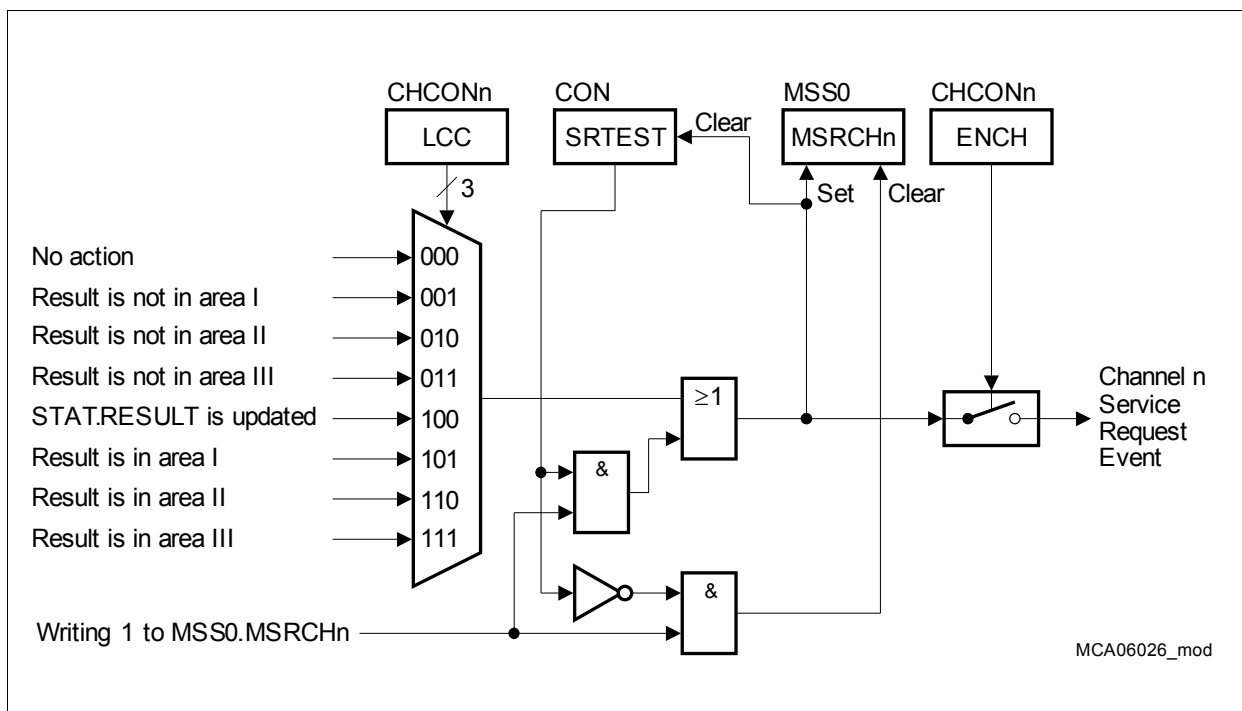


Figure 25-23 Channel n Request Source Logic

25.1.9.2 Parallel/Serial Request Source Control

The parallel/serial request control logic structure as shown in [Figure 25-24](#) is almost identical with the one shown in [Figure 25-23](#) except that the request input is controlled by one parallel/serial service request event. As defined in [Table 25-9](#), each of the four serial/parallel request source (Timer, Queue, Auto-Scan, Synchronized Injection) has its own request status flag and request enable bit. The service request flags that are located in register MSS1 can also be set or cleared by software.

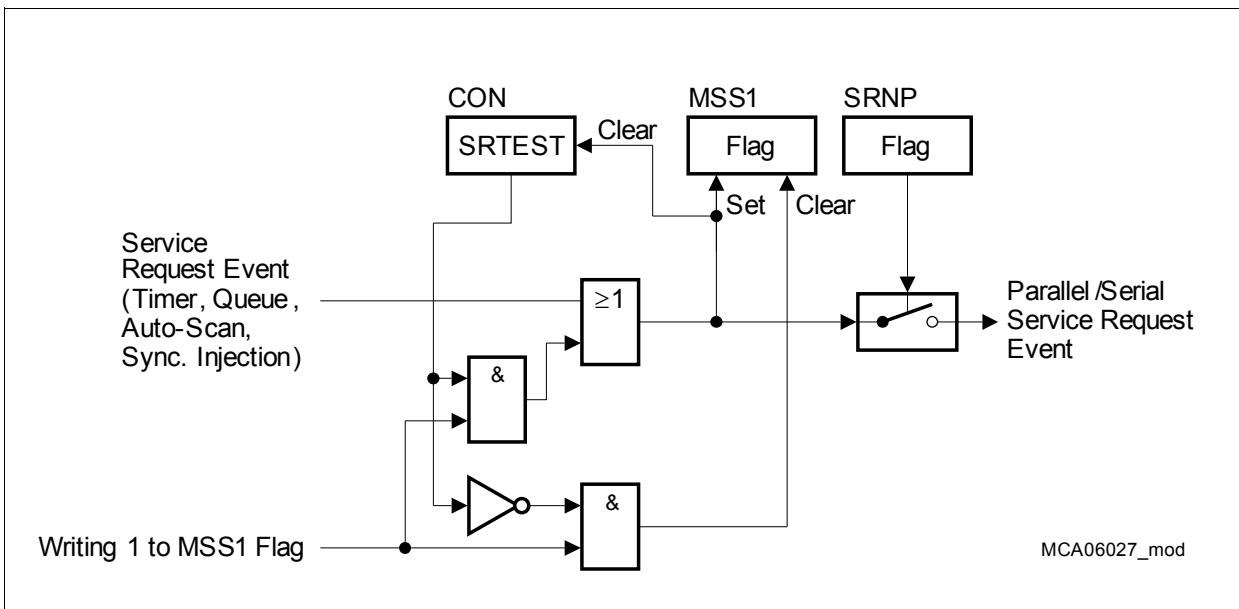


Figure 25-24 Parallel/Serial Request Source Logic

25.1.9.3 Service Request Compressor

As shown in **Figure 25-22**, the A/D Converter module is equipped with 20 service request sources and eight service request output lines SR[7:0]. In the service request compressor each of the 20 service request sources can be assigned to one of the eight service request output lines. One service request event can be only assigned to one service request output but one service request output can be used by multiple service request events.

Figure 25-25 illustrates the request compressor logic for two service request events. Each request event is controlled by one node pointer. This node pointer assigns a request event to one of the service request outputs. Channel n request events are controlled by a node pointer that is located in each channel n control register (CHCONn.INP). The node pointers for the four parallel/serial request events are located in register SRPN (see **Table 25-9**). The open inputs of the OR gates are connected to the remaining 18 service request event demultiplexers.

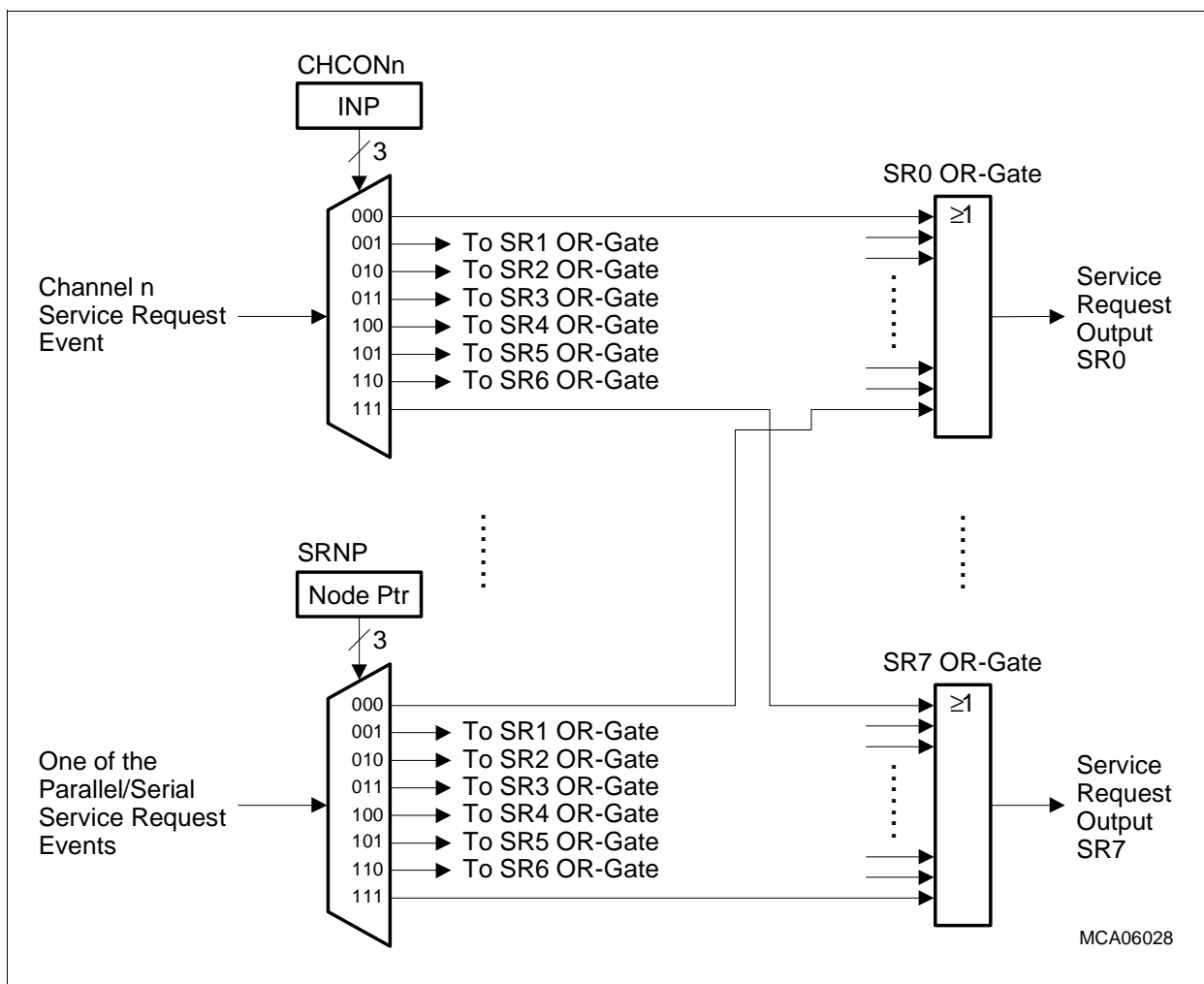


Figure 25-25 Service Request Compressor Logic

25.1.9.4 Service Request Flag Control

As shown in [Figure 25-23](#) and [Figure 25-24](#), the service request status flags are always set by hardware but can be cleared only via software (or a module reset operation). A service request status flag is cleared under software control by writing a 1 to the corresponding bit position in the MSS0/MSS1 register.

When bit CON.SRTEST is set, service request flags can also be set by software. A service request flag is set when CON.SRTEST = 1 is set and a 1 is written to the corresponding bit position in the MSS0/MSS1 register. After this MSS0/MSS1 write action is executed, bit CON.SRTEST becomes automatically cleared again.

[Table 25-10](#) summarizes the actions that are performed when a write action on a MSS0/MSS1 register service request status flag occurs.

Table 25-10 Service Request Status Flag Set/Clear Operations

SCON.SRTEST	MSS Flag Current Value	Value Written to MSS Flag	MSS Flag New Value After Write	Comment
0	0	0	0	No action
		1		No action
	1	0	1	No action
		1	0	MSS flag is cleared by software
1	0	0	0	No action
		1	1	MSS flag is set by software and a service request is generated
	1	0	1	No action
		1	1	MSS flag is set by software and a service request is generated

25.1.10 Synchronization of Two ADC Modules

In order to synchronize conversions in two ADC modules, a synchronization logic is implemented in the ADC module. A handshake mechanism guarantees the synchronization between two ADCs without additional CPU load. As shown in [Figure 25-26](#), two ADC modules are coupled via the synchronization bridge. Neither Module 0 nor Module 1 has a fixed assignment as master or slave. Because each module can request to be master, a synchronization and handshake mechanism guarantees an appropriate master-slave coordination.

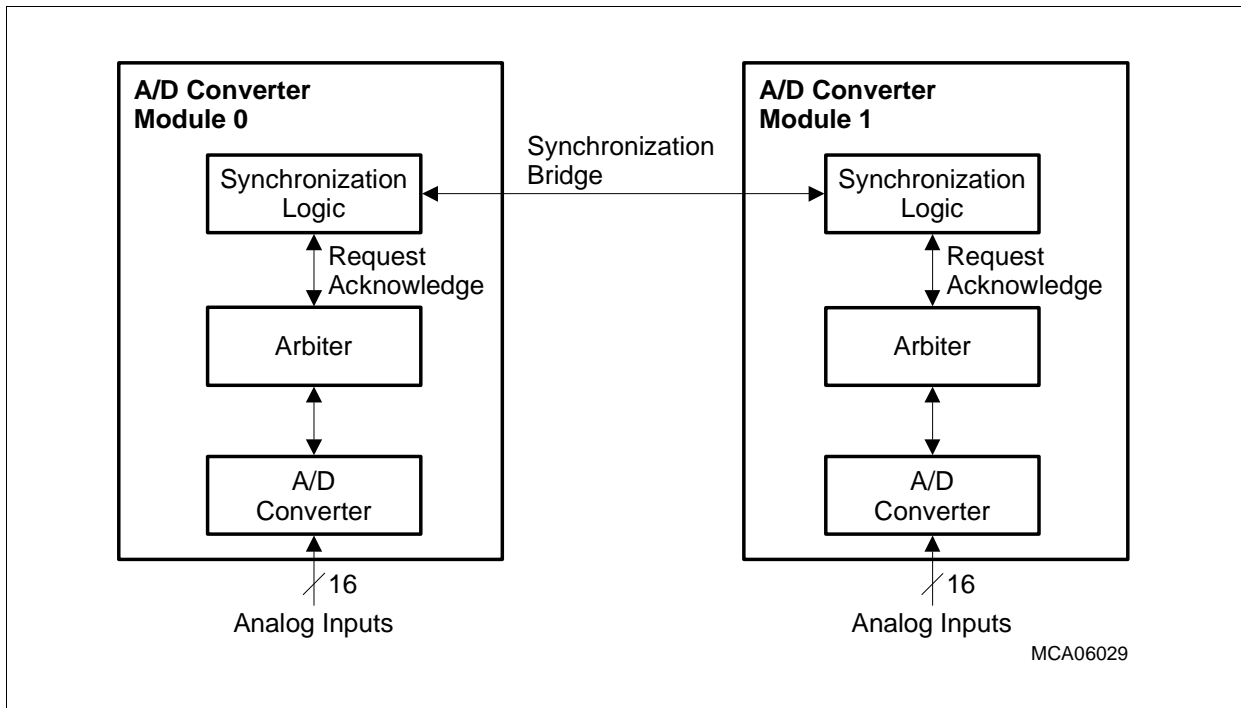


Figure 25-26 Synchronization of Two A/D-Converter

Each ADC module provides a synchronization status register SYSTAT. The conversion request and the control information for a synchronized conversion are always driven by the initiating ADC module, which is referred to as master. Because the master transfers all control information necessary for the synchronized conversion in the slave, the channel number, the A/D Converter resolution, the external multiplexer information and the cancel-synchronize-repeat information are identical in both modules. The timing information as well as the service request generation can be different in the two modules (for instance, a synchronized conversion is started in both ADC modules for channel number 5, 12-bit resolution, with identical external multiplexer information). Note that the cancel-synchronize-repeat information is needed only in the slave module in order to determine whether a currently running conversion will be cancelled.

The control bits of register SYSTAT retain their values if the synchronized conversion is started except that the request bit is automatically cleared on a start of the synchronized

Analog-to-Digital Converter (ADC)

conversion. The contents of register SYSTAT will be overwritten if a new synchronized conversion is requested. Status flags indicate the state (master and/or slave) of the ADC module during the synchronized conversion.

25.1.10.1 Synchronized Injection Mode

The Synchronized Injection Mode is controlled by bit field CHCONn.SYM in the ADC channel control register n. A synchronized conversion is always initiated by an analog channel operating in Synchronized Injection Mode instead of a normal request. The initiating ADC module is named as master, while the participating ADC module is named as slave.

To initiate a synchronized conversion in both ADC modules, the analog channel configured for Synchronized Injection Mode must be triggered by any source. If it wins the arbitration, a synchronized conversion is initiated in both ADC modules. This means that the control information needed for a synchronized conversion is transferred from the master to the slave (for instance, analog channel 2 of ADC module 0 is configured for Synchronized Injection Mode with sync-wait feature selected; then, each time this channel is triggered and wins the arbitration, a synchronized conversion is requested. Thus, ADC module 0 is assumed to be the master and the control information needed for a synchronized conversion is transferred to ADC Module 1 the slave).

Note: A Channel Injection request with an active cancel-inject-repeat feature that is requesting a Synchronized Injection does not cancel a running conversion in the master. A Channel Injection request with an active cancel-inject-repeat feature does not automatically set the cancel-sync-repeat mode in the Synchronized Injection.

The Synchronized Injection Mode provides two features:

- Sync-wait (CHCONn.SYM = 01_B):
The synchronized start of the conversion is delayed until the currently performed conversions in the partner (slave) and the initiating ADC module (master) are terminated.
- Cancel-sync-repeat (CHCONn.SYM = 10_B):
This feature provides the ability to cancel a conversion that is currently performed in the slave ADC module. Thus, the synchronized conversion is immediately started after a currently performed conversion in the initiating ADC module is terminated. Because a conversion is cancelled in the slave ADC module, the control information is restored and will participate in arbitration again.

25.1.10.2 Status Information During Synchronized Conversion

Each ADC module provides three status bits in register STAT that display the status of the ADC module during a Synchronized Injection.

- **Master Status:**
Bit STAT.REQSY is set in the initiating (master) ADC module during a synchronized conversion. It is set at the start of the synchronized conversion and is cleared after this synchronized conversion is finished.
- **Slave Status:**
Bit STAT.PARSY is set in the slave ADC module during a synchronized conversion. It is set at the start of the synchronized conversion and is cleared after this synchronized conversion is finished.
- **Master/Slave Status:**
Bit STAT.SYMS is set in **both** ADC modules, to indicate that both ADC modules requested a synchronized conversion at the same time with identical channel number. Bit STAT.SYMS is automatically cleared at the generation of the synchronized service request.

25.1.10.3 Master-Slave Functionality for Synchronized Injection

Each ADC module can operate either as master or slave or both. The ADC module operating functionality for Synchronized Injection (master, slave or master/slave functionality) is automatically detected. All associated controls for synchronized conversion are shown in [Table 25-11](#).

Table 25-11 Master-Slave Functionality and Control

Functionality during Sync. Conversion	Controls	Description
Master	CHCONn.SYM	Selects either sync-wait or cancel-sync-repeat feature
	STAT.REQSY	Status bit indicating master functionality
	STAT.IENREQ	Status bit is driven by master to indicate that the master finished its synchronized conversion
	STAT.IENPAR	Status bit is driven by slave to indicate that the slave finished its synchronized conversion
Slave	SYSTAT.SYREQ	Status bit is driven by master to request the slave for a synchronized conversion
	SYSTAT.CHNRSY	Status bit field is driven by master to indicate the channel to be converted for a synchronized conversion
	SYSTAT.RES	Status bit field is driven by master to indicate the resolution for a synchronized conversion
	SYSTAT.EMUX	Status bit field is driven by master to indicate the external multiplexer control info for a synchronized conversion
	SYSTAT.GRPS	Status bit field is driven by master to indicate the analog input multiplexer group select state for a synchronized conversion
	SYSTAT.CSREN	Status bit is driven by master to indicate whether sync-wait or cancel-sync-repeat feature was selected in the master
Master/Slave	STAT.SYMS	Status bit to indicate that both modules requested a synchronized conversion at the same time for the same channel

Master Functionality

After an arbitration winner is detected, the Synchronized Injection Mode bit field CHCONn.SYM is evaluated. If this bit field is configured either for sync-wait (CHCONn_SYM = 01_B) or cancel-sync-repeat (CHCONn.SYM = 10_B) functionality, a synchronized-request is generated for the slave ADC module. A synchronized request means setting bit SYSTAT.SYREQ in the slave's register.

Analog-to-Digital Converter (ADC)

Additionally to this synchronized-request, the channel number (SYSTAT.CHNRSY), the resolution (SYSTAT.RES), the external multiplexer information (SYSTAT.EMUX), the analog input multiplexer group select state (SYSTAT.GRPS), and the cancel-sync-repeat information (SYSTAT.CSREN) are transferred to the slave.

After the information is transferred to the slave, the master ADC module waits for the acknowledgement of the slave. This indicates that both ADC modules are ready to start their synchronized conversion. At reception of this acknowledge, the synchronized conversion is started and bit STAT.REQSY is set. Bit STAT.REQSY indicates that a synchronized conversion is currently performed and this ADC module provides master functionality. After the currently performed synchronized conversion is completely finished, bit STAT.REQSY is cleared and bit STAT.IENREQ is set.

Bits STAT.IENREQ and STAT.IENPAR are used for service request generation in the master ADC module. In order to generate a service request after both ADC modules have finished their synchronized conversion, the master checks bit STAT.IENPAR, which is driven by the slave. In case both ADC modules have finished their conversion, bit STAT.IENREQ and STAT.IENPAR are set. This generates a service request (bit MSS1.MSRSY is set). As well as setting bit MSS1.MSRSY, both bits STAT.IENREQ and STAT.IENPAR are automatically cleared.

Slave Functionality

On reception of the synchronized request (bit SYSTAT.SYREQ is set), the channel number (SYSTAT.CHNRSY), the resolution (SYSTAT.RES), the external multiplexer information (SYSTAT.EMUX), the analog input multiplexer group select state (SYSTAT.GRPS), as well as the cancel-sync-repeat information (SYSTAT.CSREN) are driven by the master. Beside this synchronized request derived from the master, **the evaluation of an arbitration result of the slave is disabled**. Thus, the slave itself cannot generate a request.

Then, the cancel-sync-repeat enable bit is evaluated. This bit specifies whether a conversion that is currently performed in the slave is cancelled (SYSTAT.CSREN = 1) or not (SYSTAT.CSREN = 0). Note that a synchronized conversion cannot be cancelled by another synchronized conversion. Bit STAT.REQSY is set to indicate that this module is the partner (slave) in a synchronized conversion.

The handshake guarantees that the master and the slave are ready to start a synchronized conversion if the synchronized request is still active (bit SYSTAT.SYREQ is set in the slave). In the case that bit SYSTAT.SYREQ is cleared in the meantime by the master, the ADC module continues with normal behavior.

Beside the start of conversion, the synchronized request (bit SYSTAT.SYREQ) is cleared, bit STAT.PARSY is set, and the write to the arbitration result is enabled anew. At the end of the synchronized conversion, the master's status bit STAT.IENPAR is controlled by the slave and the slave's status bit STAT.PARSY is cleared.

Master/Slave Functionality

The special master/slave mode is entered if both ADC modules requested to be master at the same time **and** both ADC modules requested a synchronized conversion for the same channel. In this case, each ADC module compares the received channel number from the synchronization bridge with the channel number stored in their arbitration result. Three cases must be treated:

- `SYSTAT.CHNRSY < channel number in arbitration result register`
ADC module behaves as master.
Clear the synchronized conversion request (bit `SYSTAT.SYREQ`) because this is the master (see description on master functionality).
- `SYSTAT.CHNRSY = channel number in arbitration result`
ADC module provides master/slave functionality.
- `SYSTAT.CHNRSY > channel number in arbitration result`
ADC module behaves as slave and bit `SYSTAT.SYREQ` remains set.
(see description on slave functionality)

In the case that this ADC module provides master/slave functionality, bit `STAT.SYMS` is set and any write action to the arbitration result is disabled. This means that the synchronized conversion is started next in the slave.

From this point, the behavior is similar to the one of a master until the synchronized conversion is finished. At the end of the synchronized conversion, bit `STAT.SYMS` is cleared and bit `MSS1.MSRSY` is set for each ADC module.

25.1.10.4 Conversion Timing during Synchronized Conversion

The settings for the conversion and sample timing can be selected individually for each ADC module. Thus, the conversions are started synchronously but the master can finish its synchronized conversion at a different time than the slave.

25.1.10.5 Service Request Generation in Synchronized Injection

The Synchronized Injection based service request is automatically generated either in the master ADC module or in each ADC module while each provides master/slave functionality.

In the case that both ADC modules have finished their conversion, bit `STAT.IENREQ` and `STAT.IENPAR` are set in the master ADC module. This sets bit `MSS1.MSRSY` and generates a service request if enabled and configured. Beside setting bit `MSS1.MSRSY`, bits `STAT.IENREQ` and `STAT.IENPAR` are automatically cleared.

A service request can be generated in both ADC modules for the converted channel if the channel-specific service request node pointer is configured and enabled.

25.1.10.6 Example for Synchronized Injection

Figure 25-27 shows the Synchronized Injection Mode with sync-wait functionality. The start of the synchronized conversion is always delayed until the currently performed conversion in the slave ADC module is finished.

In this example, channel 5 is the arbitration winner. Its CHCON5.SYM bit field is configured for Synchronized Injection with sync-wait functionality (CHCON5.SYM = 01_B). Thus, a synchronized request is transferred to the slave, causing the slave's bit SYSTAT.SYREQ to be set. This immediately locks the slaves' arbiter until the synchronized conversion is started. Any pending conversion requests in the slave (in this case the request by source "i") are served after the synchronized conversion is finished.

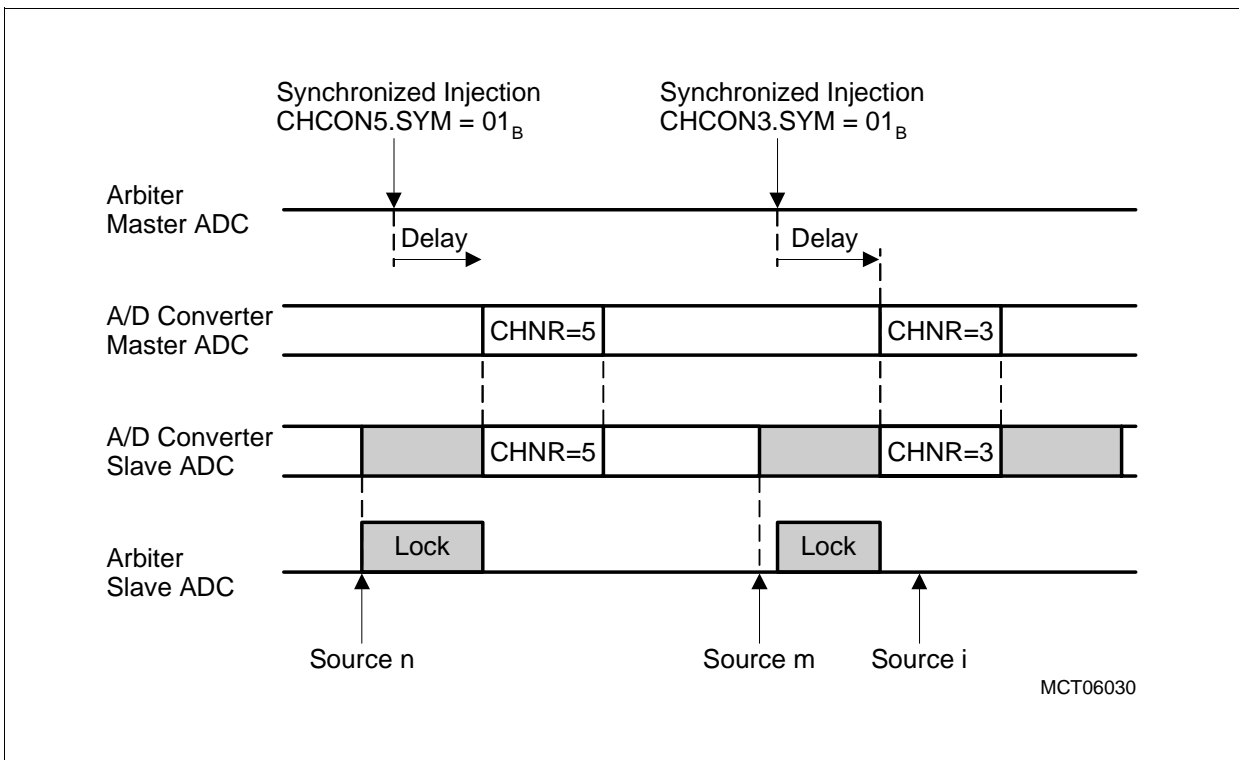


Figure 25-27 Synchronized Injection with Sync-Wait Functionality

Analog-to-Digital Converter (ADC)

Figure 25-28 shows the Synchronized Injection Mode with cancel-sync-repeat functionality. Currently performed conversions in the slave will always be cancelled, independent of their source arbitration levels. Note that a currently running synchronized conversion cannot be cancelled by any other source, not even by a new request for synchronized conversion. Thus, a request for a synchronized conversion will be delayed until the currently running synchronized conversion is finished.

In this example, channel 5 is the arbitration winner. Its CHCON5.SYM bit field is configured for Synchronized Injection with cancel-sync-repeat functionality (CHCON5.SYM = 10_B). Thus, a synchronized request is transferred to the slave and the currently performed conversion is immediately cancelled.

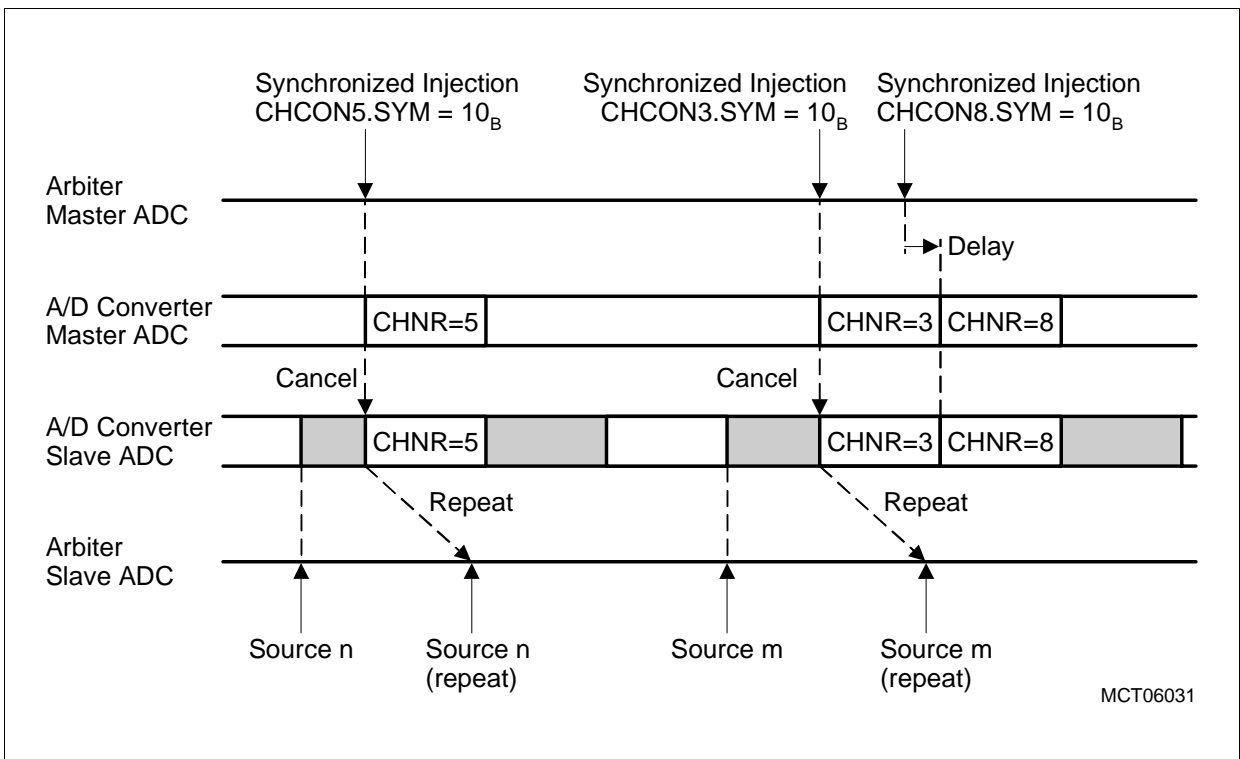


Figure 25-28 Synchronized Conversions with Cancel-Sync-Repeat Functionality

25.2 ADC0/ADC1 Kernel Registers

This section describes the kernel registers of the ADC module. All ADC kernel register names described in this section will be referenced in other parts of the TC1796 User's Manual by the module name prefix "ADC0_" for the ADC0 interface and "ADC1_" for the ADC1 module.

ADC Kernel Register Overview

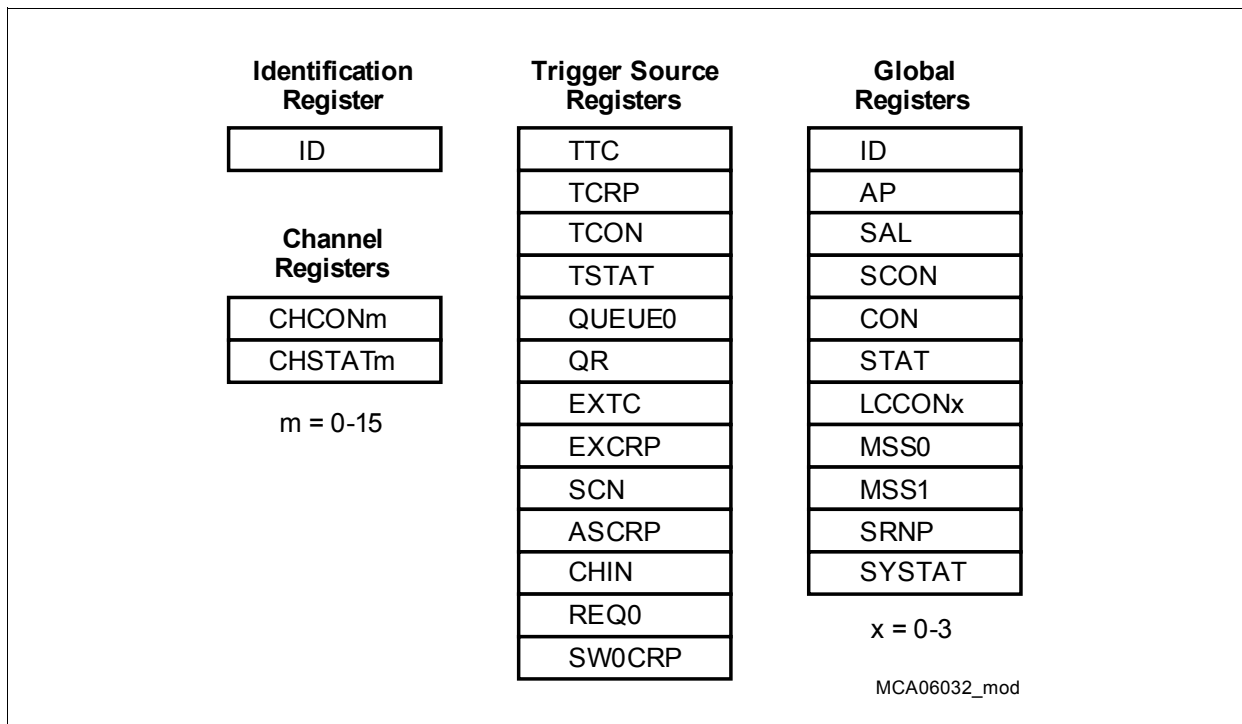


Figure 25-29 ADC Kernel Registers

The complete and detailed address map of the ADC0 module is described in [Table 18-29](#) on [Page 18-87](#) of the TC1796 User's Manual System Units part (Volume 1).

Note: For documentation automation purposes, the CHCON and CHSTAT registers use index "m" while LCCON registers use index "x" in the ADC module kernel register description. These indexes are represented by "n" for CHCON and CHSTAT registers and "m" for LCCON register in other sections of the chapter.

Table 25-12 Registers Address Space - ADC Kernel Registers

Module	Base Address	End Address	Note
ADC0	F010 0400 _H	F010 05FF _H	—
ADC1	F010 0600 _H	F010 07FF _H	—

Analog-to-Digital Converter (ADC)

Table 25-13 Registers Overview - ADC Kernel Registers

Register Short Name	Register Long Name	Offset Address ¹⁾	Description see
ID	Module Identification Register	008 _H	Page 25-59
CHCONm	Channel Control Register m (m = 0-15)	010 _H + n × 4 _H	Page 25-60
AP	Arbitration Participation Register	084 _H	Page 25-78
SAL	Source Arbitration Level Register	088 _H	Page 25-79
TTC	Timer Trigger Control Register	08C _H	Page 25-65
EXTC	External Trigger Control Register	090 _H	Page 25-73
SCON	Source Control Register	098 _H	Page 25-81
LCCONx	Limit Check Control Register x (x = 0-3)	100 _H + m × 4 _H	Page 25-80
TCON	Timer Control Register	114 _H	Page 25-66
CHIN	Channel Injection Control Register	118 _H	Page 25-90
QR	Queue Register	11C _H	Page 25-71
CON	Converter Control Register	120 _H	Page 25-82
SCN	Auto Scan Control Register	124 _H	Page 25-75
REQ0	Conversion Request Register SW0	128 _H	Page 25-92
CHSTATm	Channel Status Register m (m = 0-15)	130 _H + n × 4 _H	Page 25-63
QUEUE0	Queue Status Register	170 _H	Page 25-69
SW0CRP	Software SW0 Conv. Req. Pending Reg.	180 _H	Page 25-93
ASCRP	Auto Scan Conversion Req. Pending Reg.	188 _H	Page 25-77
SYSTAT	Synchronization Status Register	190 _H	Page 25-84
TSTAT	Timer Status Register	1B0 _H	Page 25-67
STAT	Converter Status Register	1B4 _H	Page 25-86
TCRP	Timer Conversion Req. Pending Register	1B8 _H	Page 25-68
EXCRP	External Conversion Req. Pending Register	1BC _H	Page 25-74
MSS0	Module Service Request Status Register 0	1D0 _H	Page 25-94
MSS1	Module Service Request Status Register 1	1D4 _H	Page 25-95
SRNP	Service Request Node Pointer Register	1DC _H	Page 25-97

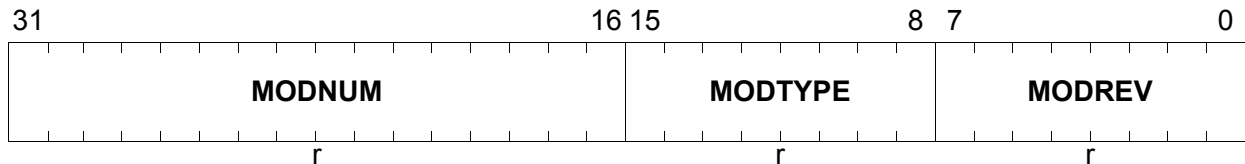
1) The absolute register address is calculated as follows:
Module Base Address ([Table 25-12](#)) + Offset Address (shown in this column)

25.2.1 Identification Register

ADC0_ID

ADC0 Module Identification Register (008_H)

Reset Value: 0030 C0XX_H



Field	Bits	Type	Description
MODREV	[7:0]	r	Module Revision Number MODREV defines the module revision number. The value of a module revision starts with 01 _H (first revision).
MODTYPE	[15:8]	r	Module Type This bit field defines the module as a 32-bit module: C0 _H
MODNUM	[31:16]	r	Module Number Value This bit field defines the module identification number for the ADC: 0030 _H

Note: In TC1796 only ADC0_ID is available. There is no ADC1_ID register available.

Analog-to-Digital Converter (ADC)

25.2.2 Channel Registers

CHCONm (m = 0-15)

Channel Control Register m (010_H+m*4) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16																
0		SYM		0	INP		EN CH		LCC		BSELA		BSELB																		
r	rw		r	rw		rw		rw		rw		rw		rw																	
15		14		13		12		11		10		9		8		7		6		5		4		3		2		1		0	
0		GRP S		EMUX		RES		REF		STC																					
r		rw		rw		rw		rw		rw																					

Field	Bits	Type	Description
STC	[7:0]	rw	<p>Sample Time Control</p> <p>STC determines the duration of the sample phase for channel n. Any modification of this bit field is taken into account after the currently running conversion is finished.</p>
REF	[9:8]	rw	<p>Analog Reference Voltage Control</p> <p>This bit determines the reference voltage for channel n.</p> <p>00_B V_{AREF}[0] is selected as reference voltage 01_B V_{AREF}[1] is selected as reference voltage 10_B V_{AREF}[2] is selected as reference voltage 11_B V_{AREF}[3] is selected as reference voltage</p> <p>See Page 25-113 for TC1796 specific implementation.</p>
RES	[11:10]	rw	<p>Conversion Resolution Control</p> <p>RES determines the resolution of the A/D Converter for the conversion of channel n. Any modification of this bit field is taken into account after the currently running conversion is finished.</p> <p>00_B 10-bit resolution selected 01_B 12-bit resolution selected 10_B 8-bit resolution selected 11_B Reserved</p>
EMUX	[13:12]	rw	<p>External Multiplexer Control</p> <p>EMUX drives the external multiplexer control lines. EMUX[0] is assigned to EMUX0 output. EMUX[1] is assigned to EMUX1 output.</p>

Analog-to-Digital Converter (ADC)

Field	Bits	Type	Description
GRPS	14	rw	<p>Analog Input Multiplexer Group Select This bit selects the analog input multiplexer group for analog input channel n.</p> <p>0_B Analog input multiplexer group 0 selected. Analog inputs AIN[15:0] are assigned to analog input channels [15:0].</p> <p>1_B Analog input multiplexer group 1 selected. Analog inputs AIN[31:16] are assigned to analog input channels [15:0].</p>
BSELA, BSELB	[17:16], [19:18]	rw	<p>Boundary Select Control This bit selects two limit check control registers for limit checking.</p> <p>00_B LCCON0 (BOUNDARY0) is selected 01_B LCCON1 (BOUNDARY1) is selected 10_B LCCON2 (BOUNDARY2) is selected 11_B LCCON3 (BOUNDARY3) is selected</p>
LCC	[22:20]	rw	<p>Limit Check Control</p> <p>000_B Neither limit check is performed nor a service request is generated on write of the conversion result to bit field STAT.RESULT.</p> <p>001_B Generate a service request if conversion result is in not area I.</p> <p>010_B Generate a service request if conversion result is not in area II.</p> <p>011_B Generate a service request if conversion result is not in area III.</p> <p>100_B Generate a service request on write of conversion result to bit field STAT.RESULT.</p> <p>101_B Generate a service request result if conversion result is in area I.</p> <p>110_B Generate a service request result if conversion result is in area II.</p> <p>111_B Generate a service request result if conversion result is in area III.</p>

Analog-to-Digital Converter (ADC)

Field	Bits	Type	Description
ENCH	23	rw	<p>Enable Channel Service Request Generation This bit enables the generation of a channel service request at the end of a channel conversion.</p> <p>0_B The channel service request generation is disabled.</p> <p>1_B The channel service request generation is enabled.</p>
INP	[26:24]	rw	<p>Interrupt Node Pointer This bit field selects which service request output line will be activated if the channel conversion is finished and the channel service request generation is enabled.</p> <p>000_B Service request line SR0 selected 001_B Service request line SR1 selected 010_B Service request line SR2 selected 011_B Service request line SR3 selected 100_B Service request line SR4 selected 101_B Service request line SR5 selected 110_B Service request line SR6 selected 111_B Service request line SR7 selected</p>
SYM	[29:28]	rw	<p>Synchronized Injection Mode This bit field determines whether channel n can trigger a synchronized conversion as master. If enabled, a synchronized conversion will be requested automatically when channel n is selected for a conversion.</p> <p>00_B Synchronized conversions are disabled for analog channel n. 01_B Synchronized conversions and sync-wait functionality is selected for channel n. 10_B Synchronized conversion and cancel-sync-repeat functionality is selected for channel n. 11_B Reserved</p>
0	15,27, [31:30]	r	<p>Reserved Read as 0; should be written with 0.</p>

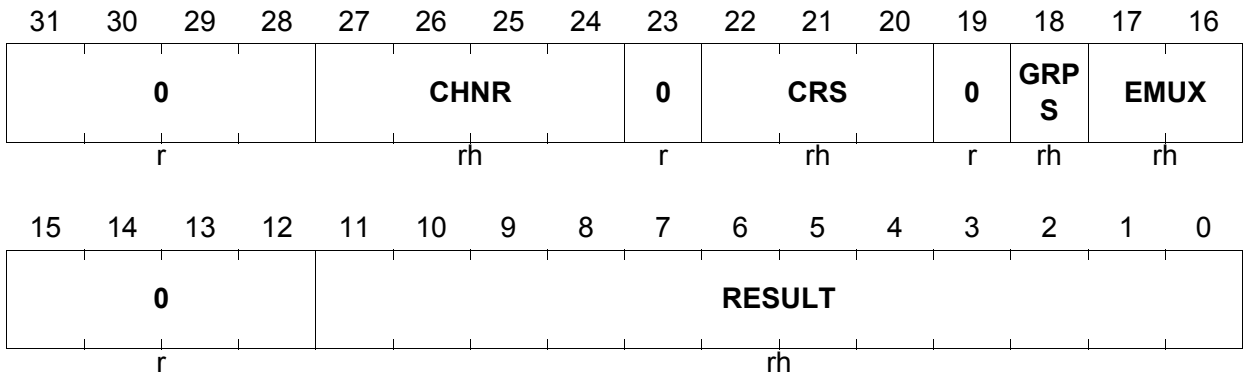
Analog-to-Digital Converter (ADC)

CHSTATm (m = 0-15)

Channel Status Register m

(130_H+m*4)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
RESULT	[11:0]	rh	Result of the Last Conversion This bit field contains the left aligned conversion result of the last conversion of channel n. 8-bit: Conversion result is stored in RESULT[11:4] 10-bit: Conversion result is stored in RESULT[11:2] 12-bit: Conversion result is stored in RESULT[11:0] Unused bits of RESULT are 0.
EMUX	[17:16]	rh	Status of External Multiplexer This bit field indicates the setting of the external multiplexer control of DMA channel n. This information is either derived from CHCONn.EMUX (parallel conversion request sources) or from CHIN.EMUX and QUEUE.EMUX (sequential conversion request sources).
GRPS	18	rh	Status of Analog Input Multiplexer Group Select This bit indicates the status of the analog input multiplexer group select bit for analog input channel n.

Analog-to-Digital Converter (ADC)

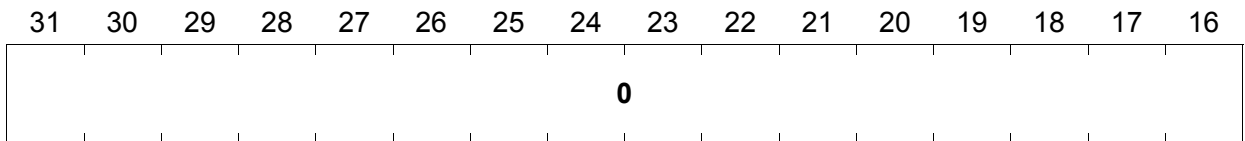
Field	Bits	Type	Description
CRS	[22:20]	rh	Conversion Request Source This bit field indicates for channel n the origin of the conversion result, stored in bit field RESULT. 000 _B Channel injection 001 _B Timer 010 _B Synchronized injection 011 _B External event 100 _B Software SW0 101 _B Reserved 110 _B Queue 111 _B Auto-Scan
CHNR	[27:24]	rh	Channel Number This bit field holds the channel number n.
0	[15:12], 19,23, [31:28]	r	Reserved Read as 0.

Note: The former contents of a CHSTATn register are overwritten with the new result for the same channel.

25.2.3 Timer Registers

TTC

Timer Trigger Control Register (08C_H) **Reset Value: 0000 0000_H**



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTC CH 15	TTC CH 14	TTC CH 13	TTC CH 12	TTC CH 11	TTC CH 10	TTC CH 9	TTC CH 8	TTC CH 7	TTC CH 6	TTC CH 5	TTC CH 4	TTC CH 3	TTC CH 2	TTC CH 1	TTC CH 0
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Field	Bits	Type	Description
TTCCHn (n = 0-15)	n	rW	Timer Trigger Control for Channel n TTCCHn determines whether a conversion request is triggered for channel n on timer underflow or not. 0 _B No conversion request is triggered for channel n. 1 _B A conversion request is triggered for channel n.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

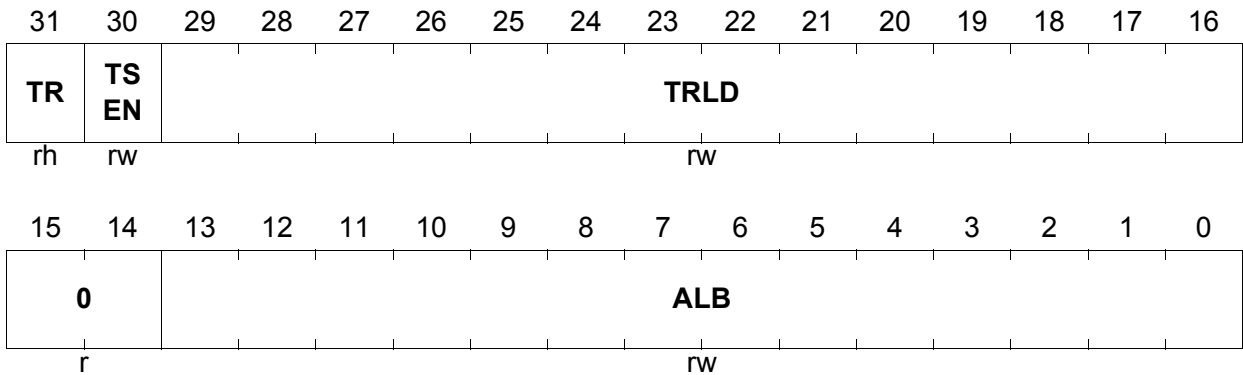
Analog-to-Digital Converter (ADC)

TCON

Timer Control Register

(114_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
ALB	[13:0]	rw	<p>Arbitration Lock Boundary</p> <p>The arbitration lock boundary is used to specify the arbitration lock time t_{LOCK}. Arbitration Lock Mode is automatically enabled if any value greater than zero is written to ALB.</p> <p><i>Note: The arbitration is locked if the value of ALB is above TRLD.</i></p>
TRLD	[29:16]	rw	<p>Timer Reload Value</p> <p>TRLD is loaded into the timer register TSTAT.TIMER when TSTAT.TIMER = 0 or each time when SCON.TRS is set.</p> <p><i>Note: If TRLD is zero, timer lock is always active and a service request can be generated for each timer clock.</i></p>
TSEN	30	rw	<p>Timer Stop Enable</p> <p>0_B TSTAT.TIMER = 0 has no effect on the timer run bit TCON.TR.</p> <p>1_B Timer run bit TCON.TR is cleared when TSTAT.TIMER = 0.</p>
TR	31	rh	<p>Timer Run Control</p> <p>0_B Timer TSTAT.TIMER is stopped.</p> <p>1_B Timer TSTAT.TIMER is running and is decremented with clock f_{TIMER}.</p> <p><i>Note: Clearing it TR causes the arbitration lock to be removed, if it is set.</i></p>

Analog-to-Digital Converter (ADC)

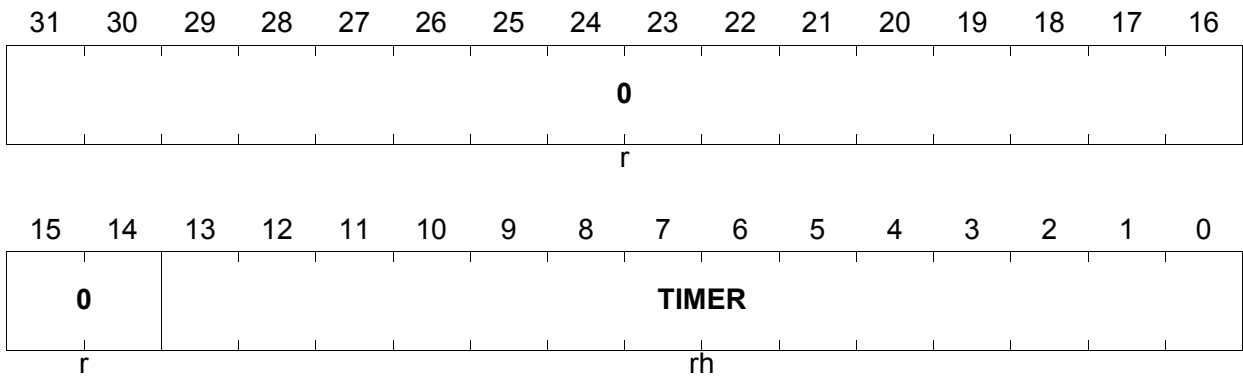
Field	Bits	Type	Description
0	[15:14]	r	Reserved Read as 0; should be written with 0.

TSTAT

Timer Status Register

(1B0_H)

Reset Value: 0000 0000_H



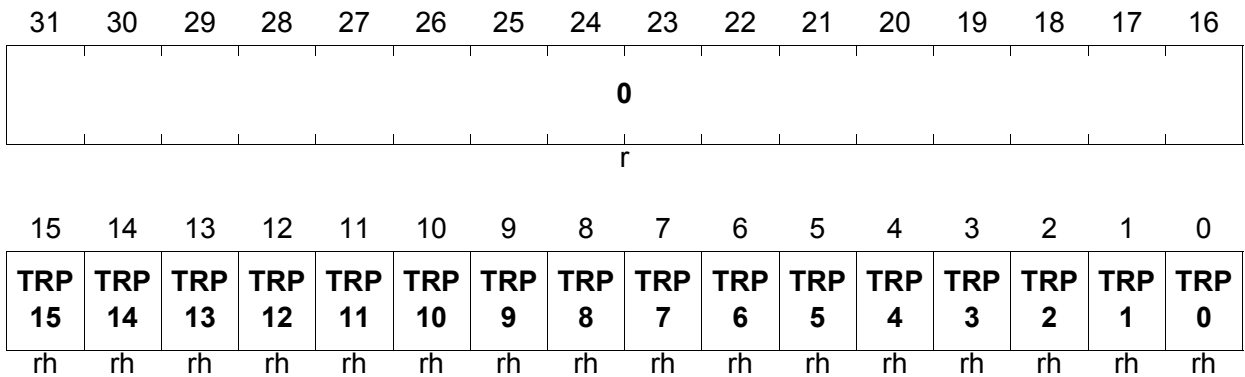
Field	Bits	Type	Description
TIMER	[13:0]	rh	Timer Register This bit field contains the current value of the timer.
0	[31:14]	r	Reserved Read as 0.

Analog-to-Digital Converter (ADC)

TCRP

**Timer Conversion Request Pending Register
(1B8_H)**

Reset Value: 0000 0000_H



Field	Bits	Type	Description
TRPn (n = 0-15)	n	rh	<p>Timer Conversion Request Pending Flag for Channel n</p> <p>A pending flag TRPn is set each time a conversion request is generated for channel n on timer underflow that could not be serviced immediately. A start of conversion of the pending request leads automatically to a cleared of the pending flag. All pending request flags can also be cleared under software control, if bit AP.TP is cleared.</p> <p>0_B No timer based conversion request is pending for channel n</p> <p>1_B A timer based conversion request is pending for channel n</p>
0	[31:16]	r	<p>Reserved</p> <p>Read as 0.</p>

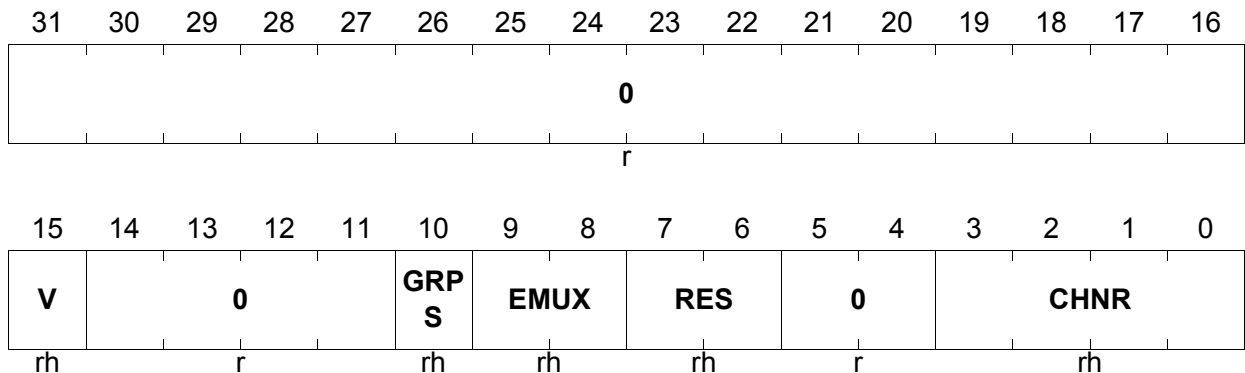
25.2.4 Queue Registers

QUEUE0

Queue Status Register

(170_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CHNR	[3:0]	rh	Channel to be Converted This bit field holds the channel number of the channel to be converted.
RES	[7:6]	rh	Conversion Resolution Status This bit field indicates the resolution of the A/D Converter for the conversion of the analog channel determined by CHNR. Any modification of this bit field is taken into account after the currently running conversion is finished. 00 _B 10-bit resolution 01 _B 12-bit resolution 10 _B 8-bit resolution 11 _B Reserved
EMUX	[9:8]	rh	External Multiplexer Control Line Status This bit field indicates the external multiplexer control line status of the analog channel determined by CHNR.
GRPS	10	rh	Analog Input Multiplexer Group Select Line Status This bit field indicates the status of the analog input multiplexer group select bit of the analog channel determined by CHNR. 0 _B Analog input multiplexer group 0 selected. 1 _B Analog input multiplexer group 1 selected.

Analog-to-Digital Converter (ADC)

Field	Bits	Type	Description
V	15	rh	<p>Valid Status This bit field indicates whether the information of register QR is valid or invalid.</p> <p>0_B CHNR, RES, EMUX and GRPS are invalid 1_B CHNR, RES, EMUX and GRPS are valid; a queue conversion request is pending.</p>
0	[5:4], [14:11], [31:16]	r	<p>Reserved Read as 0.</p>

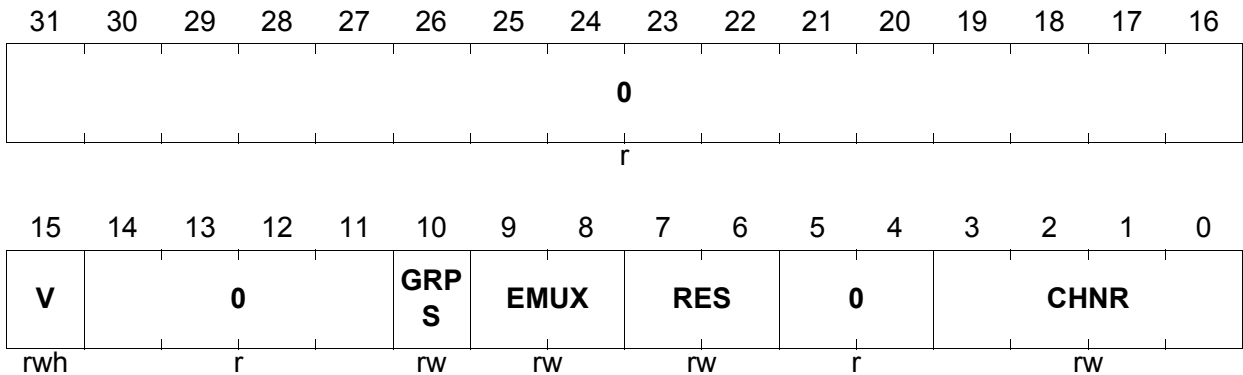
Analog-to-Digital Converter (ADC)

QR

Queue Register

(11C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CHNR	[3:0]	rw	Channel to be Converted
RES	[7:6]	rw	Conversion Resolution Control This bit field selects the resolution of the A/D Converter for the conversion of the analog channel as programmed for CHNR. Any modification of this bit field is taken into account after the currently running conversion is finished. 00 _B 10-bit resolution 01 _B 12-bit resolution 10 _B 8-bit resolution 11 _B Reserved
EMUX	[9:8]	rw	External Multiplexer Control This bit field determines the settings of the external multiplexer control lines for the conversion of the analog channel as programmed for CHNR.
GRPS	10	rw	Analog Input Multiplexer Group Select This bit selects the analog input multiplexer group for the conversion of the analog channel as programmed for CHNR. 0 _B Analog input multiplexer group 0 selected 1 _B Analog input multiplexer group 1 selected

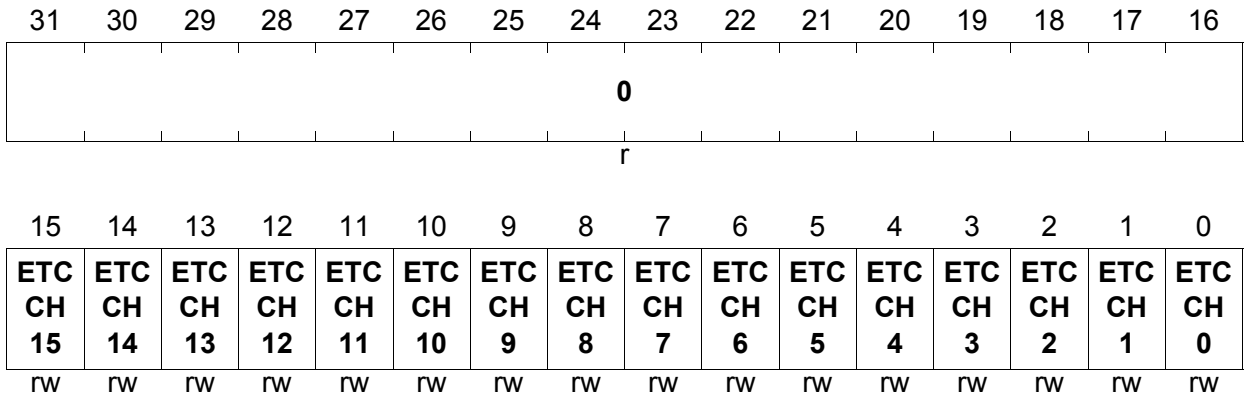
Analog-to-Digital Converter (ADC)

Field	Bits	Type	Description
V	15	rwh	<p>Valid Control</p> <p>This bit indicates whether the information of register QR is valid or invalid. Bit V is cleared by hardware when the QR contents are transferred to the queue.</p> <p>0_B CHNR, RES, EMUX and GRPS are invalid 1_B CHNR, RES, EMUX and GRPS are valid</p>
0	[5:4], [14:11], [31:16]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

25.2.5 External Trigger Registers

EXTC

External Trigger Control Register (090_H) **Reset Value: 0000 0000_H**



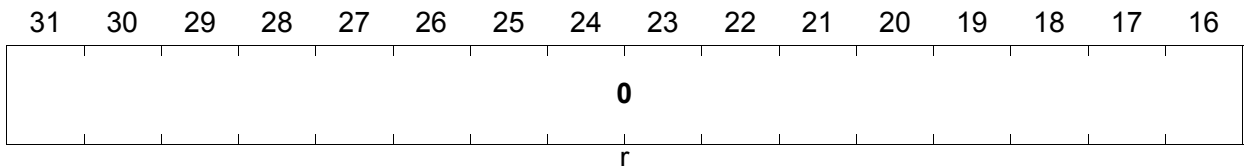
Field	Bits	Type	Description
ETCHn (n = 0-15)	n	rw	External Trigger Control for Channel n ETCHn specifies if a conversion request is triggered on an event on the selected input line (including gating) for channel n. 0 _B No conversion request is triggered for channel n. 1 _B A conversion request is triggered for channel n
0	[31:16]	r	Reserved Read as 0; should be written with 0.

Analog-to-Digital Converter (ADC)

EXCRP

**External Conversion Request Pending Register
(1BC_H)**

Reset Value: 0000 0000_H



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EX CRP	EX CRP	EX CRP	EX CRP	EX CRP	EX CRP	EX CRP	EX CRP	EX CRP	EX CRP	EX CRP	EX CRP	EX CRP	EX CRP	EX CRP	EX CRP
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
EXCRP_n (n = 0-15)	n	rh	<p>External Event Conversion Request Pending Flag for Channel n</p> <p>EXCRP_n is set each time a conversion request is generated for channel n by an external event that could not be serviced immediately. A start of conversion of the pending request leads automatically to a clear of the pending flag. All pending request flags can also be cleared under software control, if bit AP.EXP is cleared.</p> <p>0_B No external event based conversion request is pending for channel n</p> <p>1_B An external event based conversion request is pending for channel n</p>
0	[31:16]	r	<p>Reserved</p> <p>Read as 0.</p>

25.2.6 Auto-Scan Registers

SCN

Auto-Scan Control Register

(124_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GRPC		0													
rw		r													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRQ	SRQ	SRQ	SRQ	SRQ	SRQ	SRQ	SRQ	SRQ	SRQ	SRQ	SRQ	SRQ	SRQ	SRQ	SRQ
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
SRQ_n (n = 0-15)	n	rw	<p>Auto-Scan Request for Channel n</p> <p>SRQ_n determines whether or not channel n participates in an auto-scan sequence.</p> <p>0_B Channel n does not participate in an auto-scan sequence.</p> <p>1_B Channel n participates in an auto-scan sequence.</p> <p><i>Note: Bits SRQ_n maintain their values after auto-scan control bit field CON.SCNM is cleared.</i></p>

Analog-to-Digital Converter (ADC)

Field	Bits	Type	Description
GRPC	[31:30]	rw	<p>Group Control This bit field determines the behavior of the analog input group selection for the auto-scan sequence.</p> <p>00_B Group select bit CHCONn.GRPS is taken into account for the conversion of the corresponding channel if the conversion is triggered by auto-scan. Bit ASCRP.GRPS is set to 0.</p> <p>01_B Bit ASCRP.GRPS is set to 0 and it is taken into account for all conversions triggered by auto-scan (only group 0 is covered by the auto-scan).</p> <p>10_B Bit ASCRP.GRPS is set to 1 and it is taken into account for all conversions triggered by auto-scan (only group 1 is covered by the auto-scan).</p> <p>11_B Bit ASCRP.GRPS is toggled at the end of each auto-scan sequence. It is taken into account for all conversions triggered by auto-scan (group 0 is covered by an auto-scan sequence and group 1 by the following sequence, followed by group 0, etc.).</p>
0	[29:16]	r	<p>Reserved Read as 0; should be written with 0.</p>

Analog-to-Digital Converter (ADC)

ASCRP

Auto-Scan Conversion Request Pending Register

(188_H)

Reset Value: 0000 0000_H

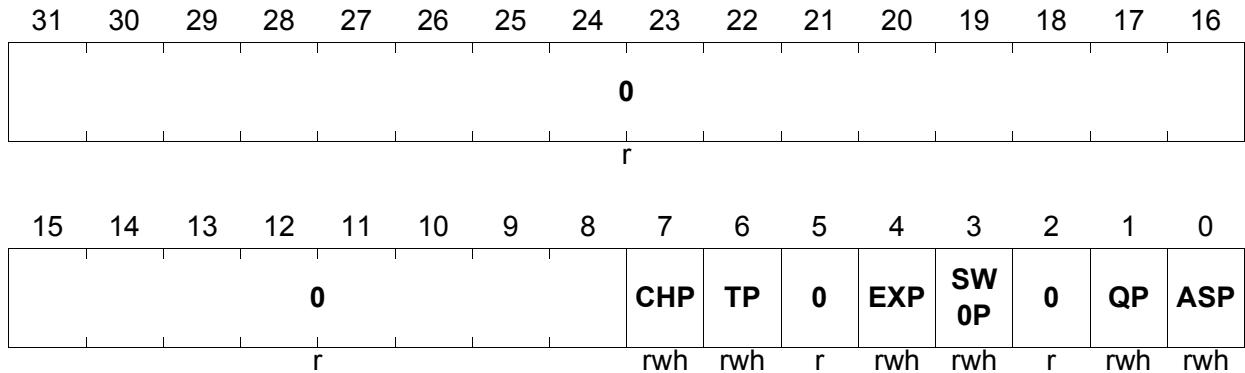
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GRPS		0													
rh								r							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ASCRP	ASCRP	ASCRP	ASCRP	ASCRP	ASCRP	ASCRP	ASCRP	ASCRP	ASCRP	ASCRP	ASCRP	ASCRP	ASCRP	ASCRP	ASCRP
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rh		rh		rh		rh		rh		rh		rh		rh	

Field	Bits	Type	Description
ASCRP_n , (n = 0-15)	n	rh	<p>Auto-Scan Conversion Request Pending Flag for Channel n</p> <p>The pending flag ASCRP_n is set each time a conversion request is generated for channel n by auto-scan that could not be serviced immediately. A start of conversion of the pending request leads automatically to a clear of the pending flag. All pending request flags can also be cleared under software control, if bit AP.ASP is cleared.</p> <p>0_B No auto-scan based conversion request is pending for channel n.</p> <p>1_B An auto-scan based conversion request is pending for channel n.</p>
GRPS	31	rh	<p>Group Select for Auto-Scan Conversions</p> <p>This flag indicates which analog input group is taken for conversions triggered by auto-scan. It is only taken into account if SCN.GRPC > 00_B.</p> <p>0_B The channels of group 0 are covered by the auto-scan sequence.</p> <p>1_B The channels of group 1 are covered by the auto-scan sequence.</p>
0	[30:16]	r	<p>Reserved</p> <p>Read as 0.</p>

25.2.7 Other Control/Status Registers

AP

Arbitration Participation Register (084_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
ASP	0	rwh	Auto-Scan Arbitration Participation 0 _B Source does not participate in arbitration 1 _B Source participates in arbitration
QP	1	rwh	Queue Arbitration Participation 0 _B Source does not participate in arbitration 1 _B Source participates in arbitration
SW0P	3	rwh	Software SW0 Arbitration Participation Flag 0 _B Source does not participate in arbitration 1 _B Source participates in arbitration
EXP	4	rwh	External Event Arbitration Participation Flag 0 _B Source does not participate in arbitration 1 _B Source participates in arbitration
TP	6	rwh	Timer Arbitration Participation Flag 0 _B Source does not participate in arbitration 1 _B Source participates in arbitration
CHP	7	rwh	Channel Injection Arbitration Participation Flag 0 _B Source does not participate in arbitration 1 _B Source participates in arbitration
0	2, 5, [31:8]	r	Reserved Read as 0; should be written with 0.

Analog-to-Digital Converter (ADC)

SAL

Source Arbitration Level Register

(088_H)

Reset Value: 0103 4067_H

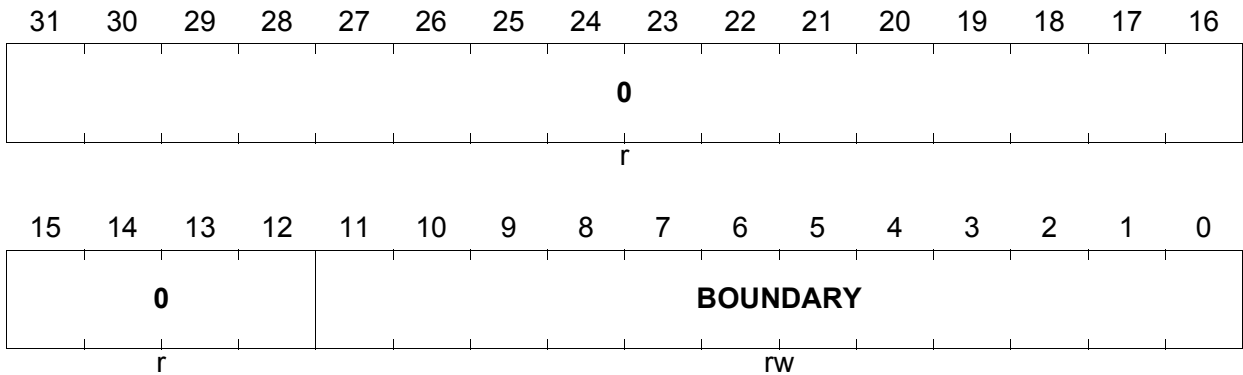
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		SALCHIN			0		SALT			0			SALEXT		
r		rw			r		rw			r			rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		SALSW0			0			SALQ			0		SALAS		
r		rw			r			rw			r		rw		

Field	Bits	Type	Description
SALAS	[2:0]	rw	Auto-Scan Source Arbitration Level 000 _B Highest priority for arbitration selected 111 _B Lowest priority for arbitration selected
SALQ	[6:4]	rw	Queue Source Arbitration Level 000 _B Highest priority for arbitration selected 111 _B Lowest priority for arbitration selected
SALSW0	[14:12]	rw	Software Source Arbitration Level 000 _B Highest priority for arbitration selected 111 _B Lowest priority for arbitration selected
SALEXT	[18:16]	rw	External Event Source Arbitration Level 000 _B Highest priority for arbitration selected 111 _B Lowest priority for arbitration selected
SALT	[26:24]	rw	Timer Source Arbitration Level 000 _B Highest priority for arbitration selected 111 _B Lowest priority for arbitration selected
SALCHIN	[30:28]	rw	Channel Injection Source Arbitration Level 000 _B Highest priority for arbitration selected 111 _B Lowest priority for arbitration selected
0	3, 15, 27,31, [11:7], [23:19]	r	Reserved Read as 0; should be written with 0.

Analog-to-Digital Converter (ADC)

LCCONx (x = 0-3)

Limit Check Control Register x (100_H+x*4) Reset Value: 0000 0000_H



Field	Bits	Type	Description
BOUNDARY	[11:0]	rw	Boundary for Limit Checks This bit field contains the boundary value used for limit checking. The relevant bits of this bit field for the different resolutions are: 8-bit: BOUNDARY[11:4] 10-bit: BOUNDARY[11:2] 12-bit: BOUNDARY[11:0] Unused bits should be written with 0.
0	[31:12]	r	Reserved Read as 0; should be written with 0.

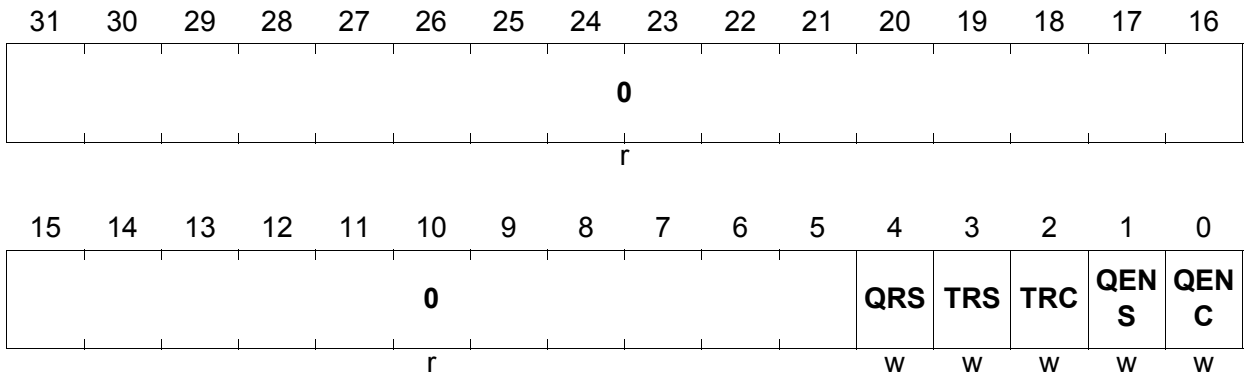
Analog-to-Digital Converter (ADC)

SCON

Source Control Register

(098_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
QENC	0	w	Queue Enable Clear Writing a 1 to this bit clears bit CON.QEN (also if QENS has been set simultaneously).
QENS	1	w	Queue Enable Set Writing a 1 to this bit and a 0 to QENC sets bit CON.QEN.
TRC	2	w	Timer Run Bit Clear Writing a 1 to this bit, clears bit TCON.TR (also if TRS has been set simultaneously).
TRS	3	w	Timer Run Bit Set Writing a 1 to this bit and a 0 to TRC sets bit TCON.TR.
QRS	4	w	Queue Reset Setting bit QRS tags all queue elements invalid (clears V bit of each queue element) and clears bits STAT.QF and STAT.QLP.
0	[31:5]	r	Reserved Read as 0; should be written with 0.

Note: All SCON bits are write-only bits. Reading SCON always delivers 0000 0000_H.

Analog-to-Digital Converter (ADC)

CON

Converter Control Register

(120_H)

Reset Value: 0000 0001_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SR TE ST	0	CPR	PC DIS					0						QWLP	
rw	r	rw	rw					r						rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QEN			0				SCNM							CTC	
rh			r				rw							rw	

Field	Bits	Type	Description
CTC	[7:0]	rw	<p>Conversion Time Control</p> <p>This bit determines the period of the ADC basic operating clock f_{BC}. Any modification of this bit field is taken into account after the currently performed conversion is finished.</p>
SCNM	[9:8]	rw	<p>Auto-Scan Mode</p> <p>This bit enables the auto-scan mode.</p> <p>00_B Auto-scan mode disabled 01_B Auto-scan single sequence mode enabled 10_B Auto-scan continuous sequence mode enabled 11_B Reserved</p>
QEN	15	rh	<p>Queue Enable</p> <p>This bit specifies if queue controlled conversions are enabled/disabled and queue based conversion requests are generated.</p> <p>0_B Queue is disabled 1_B Queue is enabled</p> <p><i>Note: The queue load is not affected by a queue disable condition.</i></p>
QWLP	[19:16]	rw	<p>Queue Warning Limit Pointer</p> <p>The value of the QWLP specifies the queue element to be watched.</p>

Analog-to-Digital Converter (ADC)

Field	Bits	Type	Description
PCDIS	27	rw	<p>Post Calibration Disable</p> <p>This bit disables the automatic post calibration. Post calibration takes place after each conversion. The next conversion is started after the end of the post calibration. If post calibration, the actual conversion length increases by two f_{ANA} analog clock cycles due to the post calibration).</p> <p>0_B The post calibration is enabled. 1_B The post calibration is disabled.</p>
CPR	28	rw	<p>Clear of Pending Conversion Requests in Parallel Sources by Arbiter</p> <p>Bit CPR determines whether or not all pending conversion requests for an A/D channel, indicated by STAT.CHNRCC, are cancelled by the arbiter, when the conversion for this channel has been started.</p> <p>0_B The individual clear by arbiter is enabled. Only the conversion request of channel n of the winning source is cleared when a conversion of channel n is started.</p> <p>1_B The global clear by arbiter is enabled. All conversion requests for channel n are cleared in parallel sources if a conversion of channel n is started.</p>
SRTEST	31	rw	<p>Service Request Test Mode</p> <p>This bit is used to set a source service request flag under software control.</p>
0	[14:10], [26:20], [30:29]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Analog-to-Digital Converter (ADC)

SYSTAT

Synchronization Status Register

(190_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SY REQ								0							
rh								r							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSR EN		0			GRPS		EMUX		RES		0		CHNRSY		
rh		r			rh		rh		rh		r		rh		

Field	Bits	Type	Description
CHNRSY	[3:0]	rh	<p>Channel to be Converted in Synchronized Conversion</p> <p>This bit field indicates the channel number of the analog channel which is converted in synchronized mode.</p>
RES	[7:6]	rh	<p>Conversion Resolution Status</p> <p>This bit field indicates the resolution of the A/D Converter for the conversion of the analog channel determined by CHNRSY.</p> <p>00_B 10-bit resolution 1_B 012-bit resolution 10_B 8-bit resolution 11_B Reserved</p>
EMUX	[9:8]	rh	<p>External Multiplexer Status</p> <p>This bit indicates the settings of the external multiplexer control lines that is used during an A/D conversion for the analog channel determined by CHNRSY.</p>
GRPS	10	rh	<p>Status of Analog Input Multiplexer Group Select</p> <p>This bit indicates the status of the analog input multiplexer group select of the analog channel determined by CHNRSY.</p> <p>0_B Analog input multiplexer group 0 selected 1_B Analog input multiplexer group 1 selected</p>

Analog-to-Digital Converter (ADC)

Field	Bits	Type	Description
CSREN	15	rh	<p>Cancel, Synchronize and Repeat State This bit indicates whether the Cancel, Synchronize and Repeat feature is enabled or disabled for the analog channel determined by CHNRSY.</p> <p>0_B Cancel, Synchronize and Repeat is disabled 1_B Cancel, Synchronize and Repeat is enabled</p>
SYREQ	31	rh	<p>Synchronized Injection Request State This bit indicates whether a synchronized conversion is requested for the analog channel determined by CHNRSY.</p> <p>0_B No synchronized conversion is requested 1_B A synchronized conversion is requested</p>
0	[5:4], [14:11], [30:16]	r	<p>Reserved Read as 0.</p>

Analog-to-Digital Converter (ADC)

STAT

Converter Status Register

(1B4_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			SY MS	IEN PAR	IEN REQ	PAR SY	REQ SY	0			QF	QLP			
r			rh	rh	rh	rh	rh	r			rh	rh			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BU SY	SM PL	CAL	AL	DA TA VAL	CHTSCC			0		CHNRCC					
rh	rh	rh	rh	rh	rh			r		rh					

Field	Bits	Type	Description
CHNRCC	[3:0]	rh	<p>Number of Channel Currently Converted</p> <p>This bit field indicates the number of the channel that is currently converted.</p> <p>0000_B Channel 0 is currently converted. 0001_B Channel 1 is currently converted. ..._B ... 1110_B Channel 14 is currently converted. 1111_B Channel 15 is currently converted.</p>
CHTSCC	[10:8]	rh	<p>Trigger Source of Channel Currently Converted</p> <p>This bit field indicates the origin of a conversion request that triggered the channel currently converted.</p> <p>000_B Channel Injection 001_B Timer 010_B Synchronization injection mode 011_B External events 100_B Software SW0 101_B Reserved 110_B Queue 111_B Auto-scan</p>
DATAVAL	11	rh	<p>Data Valid</p> <p>This bit indicates whether conversion data is valid. It is set when the conversion is finished and cleared one f_{ADC} clock cycle after the next conversion has been started.</p>

Analog-to-Digital Converter (ADC)

Field	Bits	Type	Description
AL	12	rh	<p>Arbitration Lock This bit is set, if the timer running in Arbitration Lock Mode meets the value specified in TCON.ALB. Bit is cleared on timer underflow.</p> <p>0_B Arbitration Lock Mode is inactive. 1_B Arbitration Lock Mode is active.</p>
CAL	13	rh	<p>Power-Up Calibration Status This bit indicates the status of the power-up calibration phase.</p> <p>0_B Power-up calibration is finished. 1_B The ADC is in power-up calibration phase.</p>
SMPL	14	rh	<p>Sample Phase Status This bit indicates whether the ADC is in a sample phase or not.</p> <p>0_B The ADC is currently not in the sample phase. 1_B The ADC currently samples the analog input voltage (sample phase).</p>
BUSY	15	rh	<p>Busy Status This bit indicates whether the ADC performs a conversion or not.</p> <p>0_B The ADC is currently idle. 1_B The ADC currently performs a conversion.</p>
QLP	[19:16]	rh	<p>Queue Level Pointer This bit field points to the empty queue element with the lowest queue element number. It is incremented on a queue load operation; it is decremented after a queue based conversion is started.</p>
QF	20	rh	<p>Queue Full Status This bit is set on a write action to the last empty queue element. It is cleared if at least one queue element is empty.</p> <p>0_B At least one queue element is empty. 1_B Queue is full.</p>

Analog-to-Digital Converter (ADC)

Field	Bits	Type	Description
REQSY	24	rh	<p>Requestor of Synchronized Conversion This bit is set during a synchronized conversion in the case that this ADC module is the master in the synchronized conversion.</p> <p>0_B No synchronized conversion is performed or this ADC module provides no master functionality in the synchronized conversion.</p> <p>1_B A synchronized conversion is performed and this ADC module provides master functionality.</p>
PARSY	25	rh	<p>Partner in Synchronized Conversion This bit is set during a synchronized conversion in the case that this ADC module is the slave in the synchronized conversion.</p> <p>0_B No synchronized conversion is performed or this ADC module provides no slave functionality in the synchronized conversion.</p> <p>1_B A synchronized conversion is performed and this ADC module provides slave functionality.</p>
IENREQ	26	rh	<p>Interrupt Enable by Requestor This bit is set in the master ADC module after the master finished its synchronized conversion.</p> <p>0_B The master does not finish the synchronized conversion, if any was requested.</p> <p>1_B The master finished its synchronized conversion.</p>
IENPAR	27	rh	<p>Interrupt Enable by Partner This bit is set in the master ADC module after the slave finished its synchronized conversion. In master/slave mode, bit IENPAR is driven by the opposite ADC module after the synchronized conversion is finished.</p> <p>0_B The slave does not finish the synchronized conversion, if any was requested.</p> <p>1_B The slave finished its synchronized conversion.</p>

Analog-to-Digital Converter (ADC)

Field	Bits	Type	Description
SYMS	28	rh	<p>Synchronized Master/Slave Functionality This bit is set if this ADC module enters the master/slave mode. It is cleared after the service request of synchronization mode is generated.</p> <p>0_B This synchronized conversion has not been triggered by both modules.</p> <p>1_B This synchronized conversion has been triggered by both modules at the same time.</p>
0	[7:4], [23:21], [31:29]	r	<p>Reserved Read as 0.</p>

25.2.8 Channel Inject Register

CHIN

Channel Injection Control Register (118_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CIN REQ								0							
rw								r							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CIR EN		0			GRPS	EMUX		RES		0	CHNRIN				
rw		r			rw	rw		rw		r	rw				

Field	Bits	Type	Description
CHNRIN	[3:0]	rw	Channel Number to be Injected This bit field indicates the channel number of the analog channel which is used for channel injection.
RES	[7:6]	rw	Conversion Resolution Control This bit field controls the resolution of the A/D Converter for the conversion of the analog channel determined by CHNRIN. Any modification of this bit field is taken into account after the currently running conversion is finished. 00 _B 10-bit resolution 01 _B 12-bit resolution 10 _B 8-bit resolution 11 _B Reserved
EMUX	[9:8]	rw	External Multiplexer Control This bit indicates the settings of the external multiplexer control lines that is used during an A/D conversion for the analog channel determined by CHNRIN.
GRPS	10	rw	Analog Input Multiplexer Group Select This bit selects the analog input multiplexer group for the conversion of the analog channel determined by CHNRIN. 0 _B Analog input multiplexer group 0 selected. 1 _B Analog input multiplexer group 1 selected.

Analog-to-Digital Converter (ADC)

Field	Bits	Type	Description
CIREN	15	rw	<p>Cancel, Inject and Repeat Enable This bit enables the Cancel, Inject and Repeat feature.</p> <p>0_B Cancel, Inject and Repeat feature is disabled. 1_B Cancel, Inject and Repeat feature is enabled.</p>
CINREQ	31	rw	<p>Channel Injection Request This bit is the request bit for Channel Injection. It is automatically cleared after the requested conversion is injected.</p> <p>0_B No Channel Injection requested 1_B Channel Injection requested</p> <p><i>Note: Clearing bit AP.CHP causes bit CHIN.CINREQ to be cleared.</i></p>
0	[5:4], [14:11], [30:16]	r	<p>Reserved Read as 0; should be written with 0.</p>

25.2.9 Software Request Registers

REQ0

Conversion Request Register SW0 (128_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REQ	REQ	REQ	REQ	REQ	REQ	REQ	REQ	REQ	REQ	REQ	REQ	REQ	REQ	REQ	REQ
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Field	Bits	Type	Description
REQ0n (n = 0-15)	n	rW	Software SW0 Conversion Request for Channel n 0 _B No conversion is requested for channel n. 1 _B A conversion is requested for channel n.
0	[31:16]	r	Reserved Read as 0.

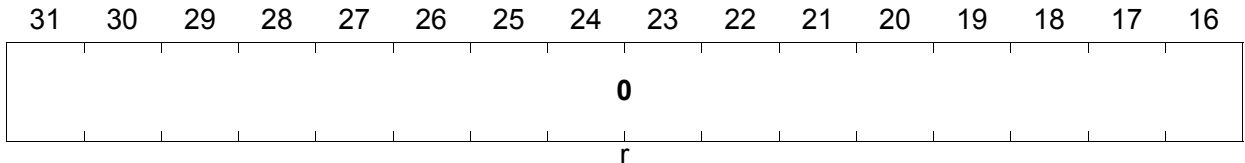
Analog-to-Digital Converter (ADC)

SW0CRP

Software SW0 Conversion Request Pending Register

(180_H)

Reset Value: 0000 0000_H



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SW0 CRP	SW0 CRP	SW0 CRP	SW0 CRP	SW0 CRP	SW0 CRP	SW0 CRP	SW0 CRP	SW0 CRP	SW0 CRP	SW0 CRP	SW0 CRP	SW0 CRP	SW0 CRP	SW0 CRP	SW0 CRP
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
SW0CRPn (n = 0-15)	n	rh	<p>Software SW0 Conversion Request Pending Flag for Channel n</p> <p>A SW0CRPn pending flag is set each time a conversion request is generated for channel n by SW0, that could not be serviced immediately. A start of the conversion for a pending request leads automatically to a clear of the pending flag. All pending request flags can also be cleared under software control, if bit AP.SW0P is cleared.</p> <p>0_B No SW0 based conversion request is pending for channel n.</p> <p>1_B A SW0 based conversion request is pending for channel n.</p>
0	[31:16]	r	<p>Reserved</p> <p>Read as 0.</p>

25.2.10 Service Request Registers

MSS0

Module Service Request Status Register 0

(1D0_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSR CH 15	MSR CH 14	MSR CH 13	MSR CH 12	MSR CH 11	MSR CH 10	MSR CH 9	MSR CH 8	MSR CH 7	MSR CH 6	MSR CH 5	MSR CH 4	MSR CH 3	MSR CH 2	MSR CH 1	MSR CH 0
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
MSRCH_n (n = 0-15)	n	rwh	<p>Module Service Request Status for Channel n</p> <p>A MSRCH_n pending flag determines if a channel service request has been generated by A/D Converter channel n.</p> <p>0_B No channel service request has been generated by channel n.</p> <p>1_B A channel service request has been generated by channel n.</p> <p>These bits are cleared by writing a 1 to the corresponding bit position.</p>
0	[31:16]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

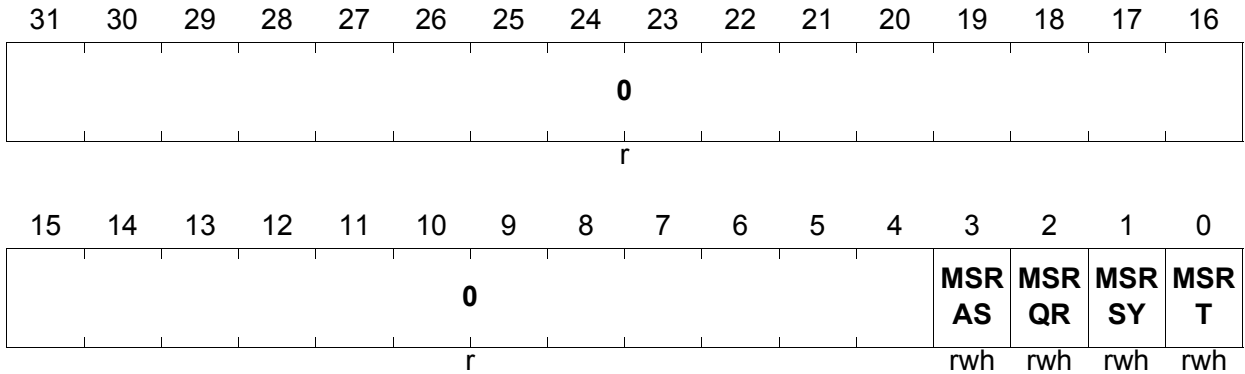
Analog-to-Digital Converter (ADC)

MSS1

Module Service Request Status Register 1

(1D4_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
MSRT	0	rwh	<p>Module Service Request Status for Source Timer This bit specifies if a timer source service request has been generated.</p> <p>0_B No timer source service request has been generated. 1_B A timer source service request has been generated.</p> <p>This bit is cleared by writing a 1 to its bit position.</p>
MSRSY	1	rwh	<p>Module Service Request Status for Source Synchronized Injection This bit specifies if a synchronized injection source service request has been generated.</p> <p>0_B No Synchronized Injection source service request has been generated. 1_B A Synchronized Injection source service request has been generated.</p> <p>This bit is cleared by writing a 1 to its bit position.</p>
MSRQR	2	rwh	<p>Module Service Request Status for Source Queue This bit specifies if a queue source service request has been generated.</p> <p>0_B No queue source service request has been generated. 1_B A queue source service request has been generated.</p> <p>This bit is cleared by writing a 1 to its bit position.</p>

Analog-to-Digital Converter (ADC)

Field	Bits	Type	Description
MSRAS	3	rwh	<p>Module Service Request Status for Source Auto-Scan</p> <p>This bit specifies if a auto-scan source service request has been generated.</p> <p>0_B No auto-scan source service request has been generated.</p> <p>1_B A auto-scan source service request has been generated.</p> <p>This bit is cleared by writing a 1 to its bit position.</p>
0	[31:4]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

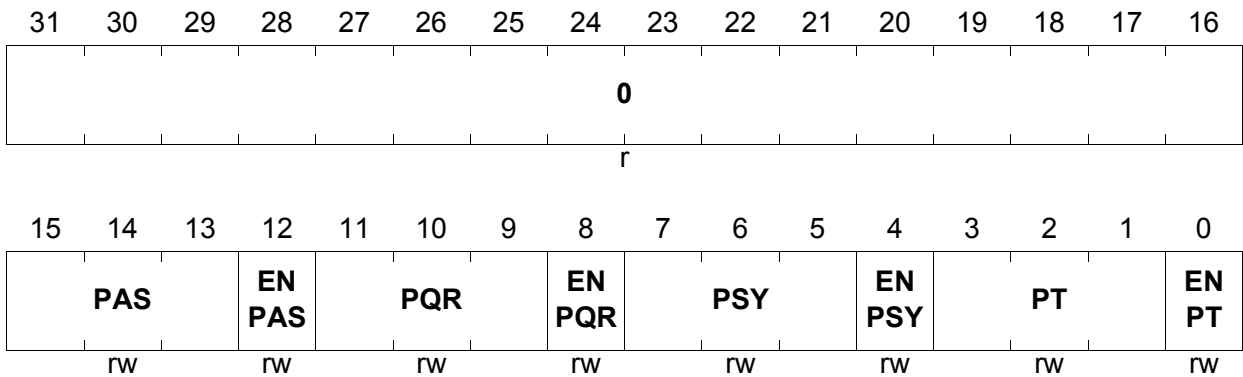
Analog-to-Digital Converter (ADC)

SRNP

Service Request Node Pointer Register

(1DC_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
ENPT	0	rw	<p>Timer Service Request Enable</p> <p>This bit enables the generation of a service request when a timer service request event occurs.</p> <p>0_B Timer service request generation is disabled. 1_B Timer service request generation is enabled.</p>
PT	[3:1]	rw	<p>Timer Service Request Node Pointer</p> <p>These bit fields selects which service request output line will be activated when a timer service request event occurs and ENPT = 1.</p> <p>000_B Service request output SR0 is selected. 001_B 0Service request output SR1 is selected. 010_B Service request output SR2 is selected. 011_B Service request output SR3 is selected. 100_B Service request output SR4 is selected. 101_B Service request output SR5 is selected. 110_B Service request output SR6 is selected. 111_B Service request output SR7 is selected.</p>
ENPSY	4	rw	<p>Synchronized Service Request Enable</p> <p>This bit enables the generation of a service request when a synchronized injection service request event occurs.</p> <p>0_B Synchronized injection service request generation is disabled. 1_B Synchronized injection generation is enabled.</p>

Analog-to-Digital Converter (ADC)

Field	Bits	Type	Description
PSY	[7:5]	rw	<p>Synchronized Service Request Node Pointer This bit fields selects which service request output line will be activated if the synchronized injection service request becomes active and ENPSY = 1.</p> <p>000_B Service request output SR0 is selected. 001_B Service request output SR1 is selected. 010_B Service request output SR2 is selected. 011_B Service request output SR3 is selected. 100_B Service request output SR4 is selected. 101_B Service request output SR5 is selected. 110_B Service request output SR6 is selected. 111_B Service request output SR7 is selected.</p>
ENPQR	8	rw	<p>Queue Service Request Enable This bit enables the generation of a service request when a queue service request event occurs.</p> <p>0_B Queue service request generation is disabled. 1_B Queue service request generation is enabled.</p>
PQR	[11:9]	rw	<p>Queue Service Request Node Pointer This bit fields selects which service request output line will be activated if the queue service request becomes active and ENPQR = 1.</p> <p>000_B Service request output SR0 is selected. 001_B Service request output SR1 is selected. 010_B Service request output SR2 is selected. 011_B Service request output SR3 is selected. 100_B Service request output SR4 is selected. 101_B Service request output SR5 is selected. 110_B Service request output SR6 is selected. 111_B Service request output SR7 is selected.</p>
ENPAS	12	rw	<p>Auto-Scan Service Request Enable This bit enables the generation of a service request when an auto-scan service request event occurs.</p> <p>0_B Auto-scan service request generation is disabled. 1_B Auto-scan service request is enabled.</p>

Analog-to-Digital Converter (ADC)

Field	Bits	Type	Description
PAS	[15:13]	rw	<p>Auto-Scan Service Request Node Pointer This bit fields selects which service request output line will be activated if the auto-scan service request becomes active and ENPAS = 1.</p> <p>000_B Service request output SR0 is selected. 001_B Service request output SR1 is selected. 010_B Service request output SR2 is selected. 011_B Service request output SR3 is selected. 100_B Service request output SR4 is selected. 101_B Service request output SR5 is selected. 110_B Service request output SR6 is selected. 111_B Service request output SR7 is selected.</p>
0	[31:16]	r	<p>Reserved Read as 0; should be written with 0.</p>

25.3 Implementation of ADC0/ADC1

This section describes the ADC0/ADC1 module related external functions such as port connections, interrupt and service request control, DMA and GPTA connections, address decoding, and clock control.

25.3.1 Interface Connections of the ADC Modules

Figure 25-30 shows the TC1796 specific implementation details and interconnections of the ADC0 and ADC1 modules.

Both ADC modules are supplied by a common separate clock control, address decoding, and interrupt control logic. Four of the eight module's service request outputs are connected to interrupt nodes and four are connected to the DMA controller.

The 44 analog inputs are wired to the analog inputs of the two ADC modules with a fixed scheme. One analog input of ADC1 is connected to an on-chip temperature measurement unit. Control outputs for external analog input extensions are available at Port 7 as alternate outputs. The external request unit that is located in the TC1796 System Control Unit (SCU) controls A/D conversion trigger and gating input signals that are generated from port lines, from MSC outputs, or from GPTA outputs.

Analog-to-Digital Converter (ADC)

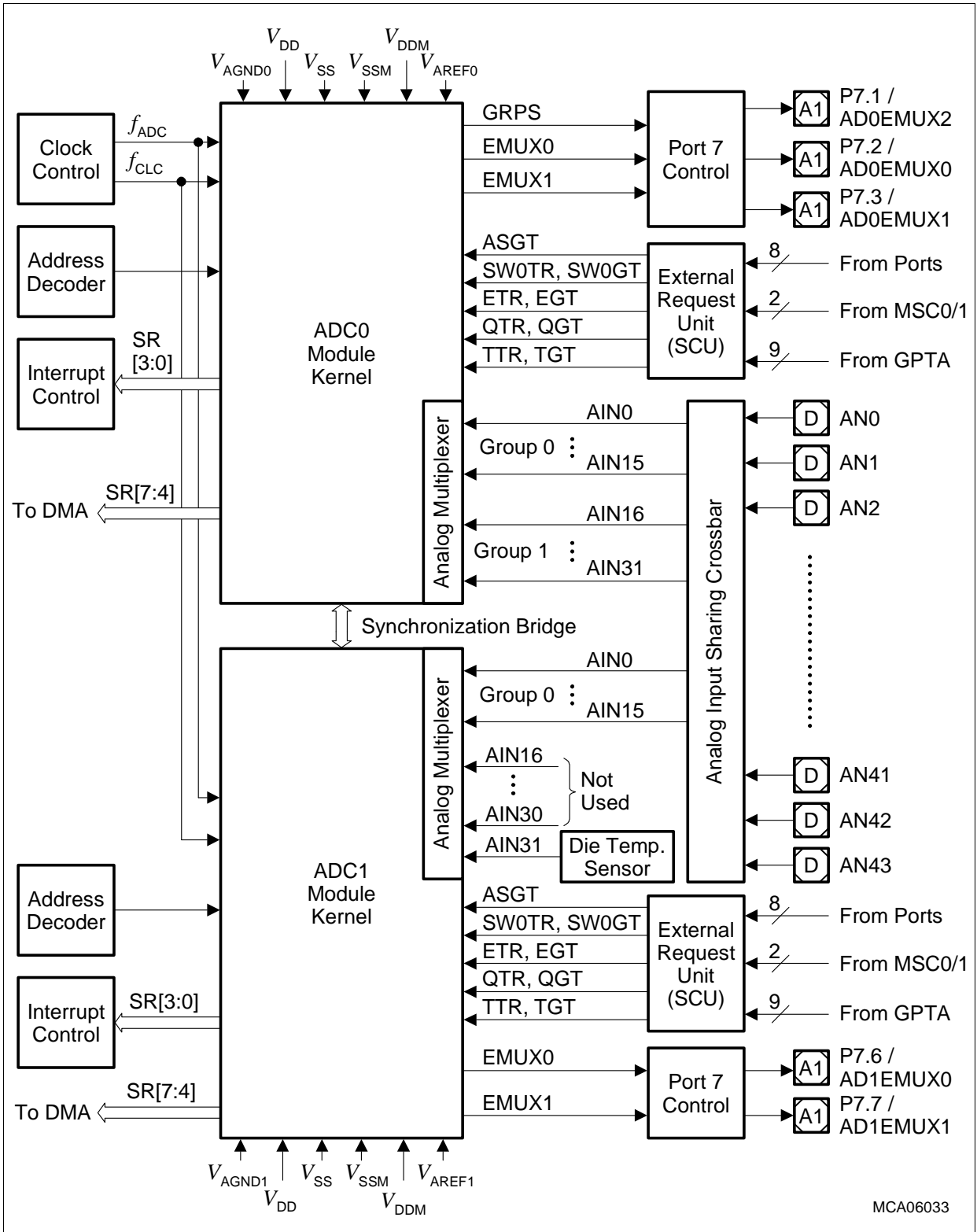


Figure 25-30 ADC0/ADC1 Modules Implementation and Interconnections

25.3.2 ADC0/ADC1 Module Related External Registers

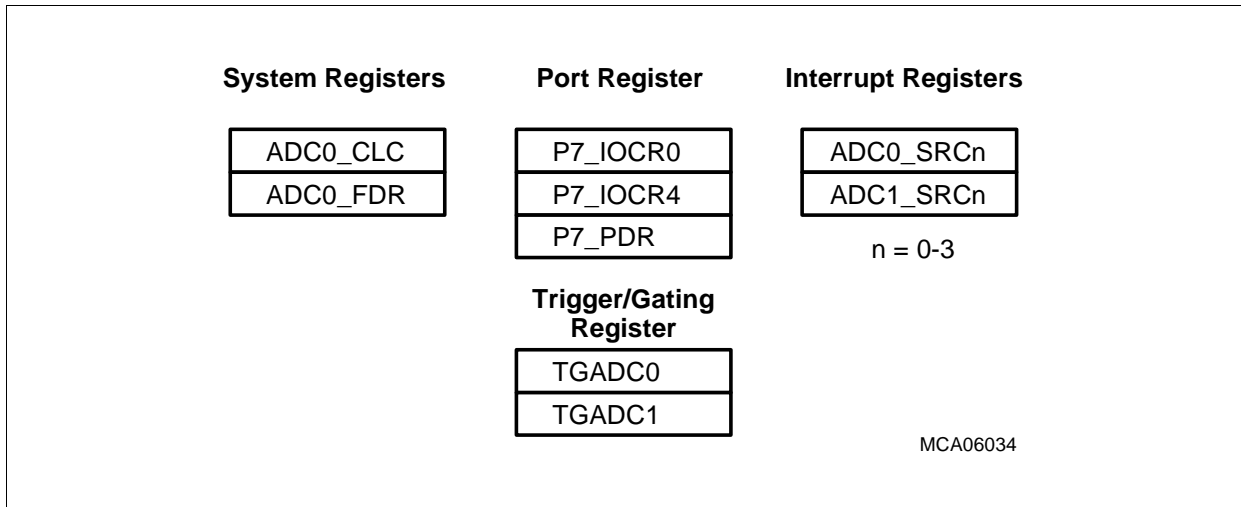


Figure 25-31 ADC0/ADC1 Implementation-Specific Special Function Registers

25.3.3 Clock Control

The ADC0/ADC1 modules are provided with common clock signals:

- f_{CLC}
This is the module control clock that is used inside the ADC0/ADC1 kernels for control purposes such as e.g. for clocking of control logic and register operations. The frequency of f_{CLC} is equal to the system clock frequency f_{SYS} . The clock control register ADC0_CLC makes it possible to enable/disable f_{CLC} under certain conditions.
- f_{ADC}
This clock is the module timing clock that is used in the ADC0/ADC1 kernels as basic and timing reference clock for the analog part. The fractional divider registers ADC0_FDR controls the frequency of f_{ADC} and makes it possible to enable/disable it independently of f_{CLC} . The fractional divider's external clock enable feature is not used.

Signal RST_EXT_DIV of the fractional divider which is controlled by bit fields SC and DM of the ADC0_FDR register makes it possible to put the analog parts of the ADCs in its reset state. When setting ADC0_FDR.SC = 11_B (and afterwards back to the previous state of ADC0_FDR.SC), the analog parts of ADC0 and ADC1 are reset. This feature makes it possible, for example, to start a power-up calibration without a reset operation of the complete ADCs.

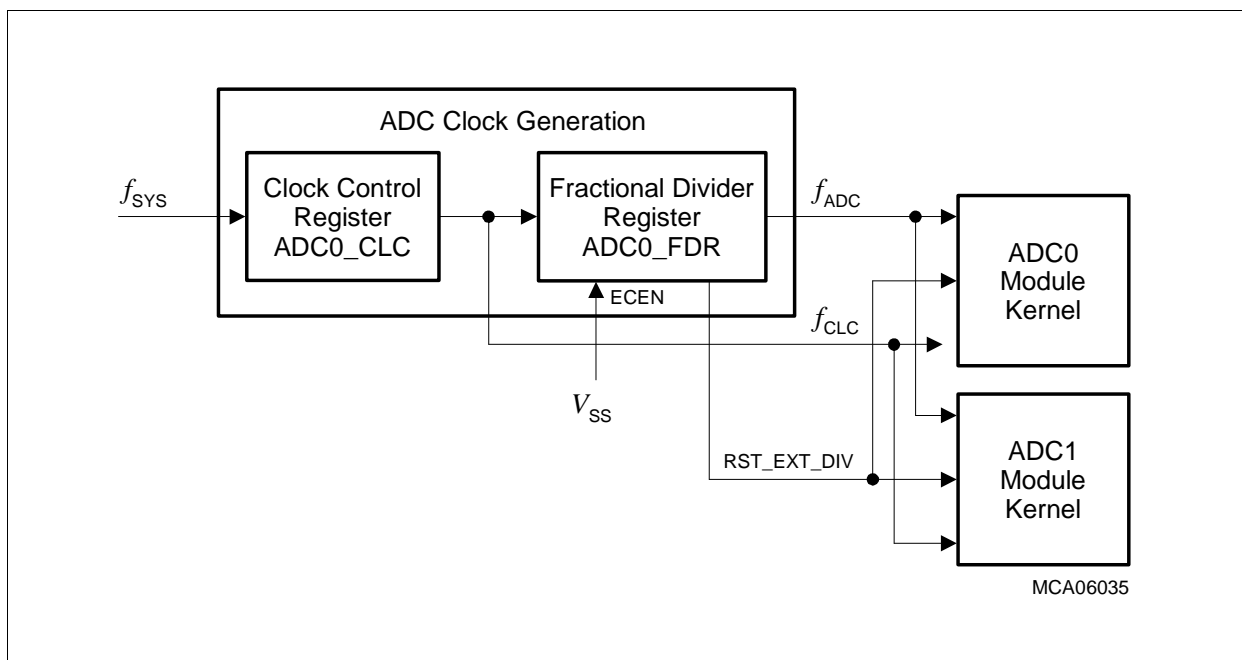


Figure 25-32 ADC0/ADC1 Clock Generation

The following formulas define the frequency of f_{ADC} :

$$f_{\text{ADC}} = f_{\text{SYS}} \times \frac{1}{n} \text{ with } n = 1024 - \text{FDR.STEP} \quad (25.11)$$

$$f_{\text{ADC}} = f_{\text{SYS}} \times \frac{n}{1024} \text{ with } n = 0-1023 \quad (25.12)$$

Equation (25.11) is valid for $\text{ADC0_FDR.DM} = 01_{\text{B}}$ (normal divider mode).

Equation (25.12) is valid for $\text{ADC0_FDR.DM} = 10_{\text{B}}$ (fractional divider mode).

Analog-to-Digital Converter (ADC)

25.3.3.1 ADC Clock Control Register

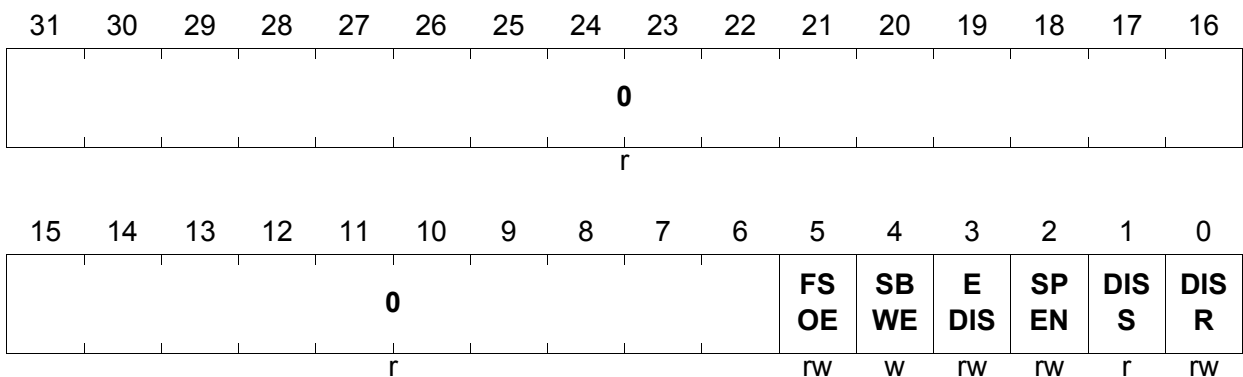
The clock control register makes it possible to control (enable/disable) the clock signal f_{CLC} under certain conditions. Both ADCs are controlled by one common clock control register ADC0_CLC. After a reset operation, the ADC0 and ADC1 modules are disabled and the module clock signals f_{CLC} and f_{ADC} are switched off.

ADC0_CLC

ADC Clock Control Register

(000_H)

Reset Value: 0000 0003_H



Field	Bits	Type	Description
DISR	0	rw	Module Disable Request Bit Used for enable/disable control of the module.
DISS	1	r	Module Disable Status Bit Bit indicates the current status of the module.
SPEN	2	rw	Module Suspend Enable for OCDS Used to enable the suspend mode.
EDIS	3	rw	External Request Disable Used to control the external clock disable request.
SBWE	4	w	Module Suspend Bit Write Enable for OCDS Determines whether SPEN and FSOE are write-protected.
FSOE	5	rw	Fast Switch Off Enable Used for fast clock switch off in suspend mode.
0	[31:6]	r	Reserved Read as 0; should be written with 0.

Note: Additional details on the clock control register functionality are described in section [“Clock Control Register CLC” on Page 3-24](#) of the TC1796 User’s Manual System Units part (Volume 1).

Analog-to-Digital Converter (ADC)

25.3.3.2 ADC Fractional Divider Register

The fractional divider register allows the programmer to control the clock rate of the module timing clock f_{ADC} . Both ADCs, ADC0 and ADC1, are controlled by one common clock control register ADC0_FDR.

ADC0_FDR

ADC Fractional Divider Register

(00C_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIS CLK	EN HW	SUS REQ	SUS ACK	0		RESULT									
rwh	rw	rh	rh	r		rh									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DM		SC		SM	0		STEP								
rw		rw		rw	r		rw								

Field	Bits	Type	Description
STEP	[9:0]	rw	Step Value Reload or addition value for RESULT.
SM	11	rw	Suspend Mode SM selects between granted or immediate suspend mode.
SC	[13:12]	rw	Suspend Control This bit field determines the behavior of the fractional divider in suspend mode.
DM	[15:14]	rw	Divider Mode This bit field selects normal divider mode, fractional divider mode, and off-state.
RESULT	[25:16]	rh	Result Value Bit field for the addition result.
SUSACK	28	rh	Suspend Mode Acknowledge Indicates state of <u>SPNDACK</u> signal.
SUSREQ	29	rh	Suspend Mode Request Indicates state of <u>SPND</u> signal.
ENHW	30	rw	Enable Hardware Clock Control Controls operation of ECEN input and DISCLK bit.

Analog-to-Digital Converter (ADC)

Field	Bits	Type	Description
DISCLK	31	rwh	Disable Clock Hardware controlled disable for f_{OUT} signal.
0	10, [27:26]	rw	Reserved Read as 0; should be written with 0.

*Note: Further details on the fractional divider register functionality are described in section **“Fractional Divider Operation”** on Page 3-29 of the TC1796 User’s Manual System Units part (Volume 1).*

Analog-to-Digital Converter (ADC)

25.3.4 Port Control

The external multiplexer control outputs (and the group select output GRPS for ADC0) of the ADC modules are controlled in the port logics. For these output lines the following port control operations must be executed:

- Input/output function selection (IOCR registers)
- Pad driver characteristics selection for outputs (PDR registers)

Input/Output Function Selection

The port input/output control registers contain the bit fields that select the ADC's digital I/O driver characteristics such as port direction (input/output), push-pull/open-drain capabilities, and alternate output selections. The output lines for the ADC modules are controlled by the port input/output control registers P7_IOCR0 and P7_IOCR4.

Table 25-14 shows how bits and bit fields must be programmed for the required I/O functionality of the ADC0/ADC1 I/O lines.

Table 25-14 ADC0/ADC1 I/O Control Selection and Setup

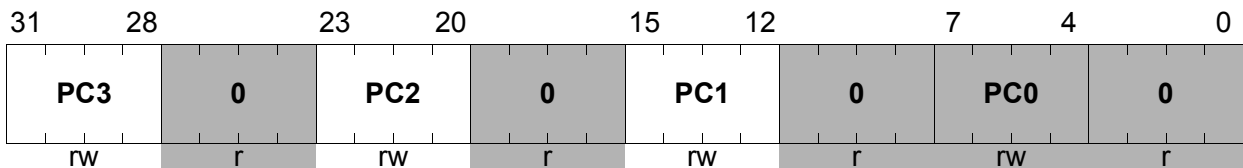
Module	Port Lines	Input/Output Control Register Bits	Output Characteristic	I/O
ADC0	P7.1/AD0EMUX2	P7_IOCR0.PC1 = 1001 _B	Push-pull	Output
		P7_IOCR0.PC1 = 1101 _B	Open-drain	
	P7.2/AD0EMUX0	P7_IOCR0.PC2 = 1001 _B	Push-pull	Output
		P7_IOCR0.PC2 = 1101 _B	Open-drain	
	P7.3/AD0EMUX1	P7_IOCR0.PC3 = 1001 _B	Push-pull	Output
		P7_IOCR0.PC3 = 1101 _B	Open-drain	
ADC1	P7.6/AD1EMUX0	P7_IOCR4.PC6 = 1001 _B	Push-pull	Output
		P7_IOCR4.PC6 = 1101 _B	Open-drain	
	P7.7/AD1EMUX1	P7_IOCR4.PC7 = 1001 _B	Push-pull	Output
		P7_IOCR4.PC7 = 1101 _B	Open-drain	

25.3.4.1 Input/Output Control Register

P7_IOCRO

Port 7 Input/Output Control Register 0 (10_H)

Reset Value: 2020 2020_H



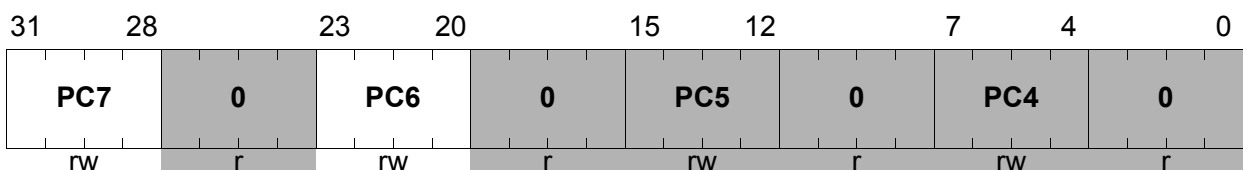
Field	Bits	Type	Description
PC1, PC2, PC3	[15:12], [23:20], [31:28]	rw	Port Output Control for Port 7 Pins 0-3¹⁾ These bit fields determine the output port functionality: Port input/output control for P7.1/AD0EMUX2 Port input/output control for P7.2/AD0EMUX0 Port input/output control for P7.3/AD0EMUX1

1) Coding of bit field see [Table 25-15](#). Shaded bits and bit fields are “don’t care” for ADC I/O port control.

P7_IOCRA

Port 7 Input/Output Control Register 4 (14_H)

Reset Value: 2020 2020_H



Field	Bits	Type	Description
PC6, PC7	[23:20], [31:28]	rw	Port Output Control for Port 7 Pins 4-7¹⁾ These bit fields determine the output port functionality: Port input/output control for P7.6/AD1EMUX0 Port input/output control for P7.7/AD1EMUX1

1) Coding of bit field see [Table 25-15](#). Shaded bits and bit fields are “don’t care” for ADC I/O port control.

Table 25-15 PCx Coding for ADC Outputs

PCx[3:0]	I/O	Output Characteristics	Selected Pull-up / Pull-down / Selected Output Function
1011 _B	Output	Push-pull	Output function ALT1
1111 _B		Open-drain	Output function ALT1

25.3.4.2 Pad Output Driver Characteristics Selection

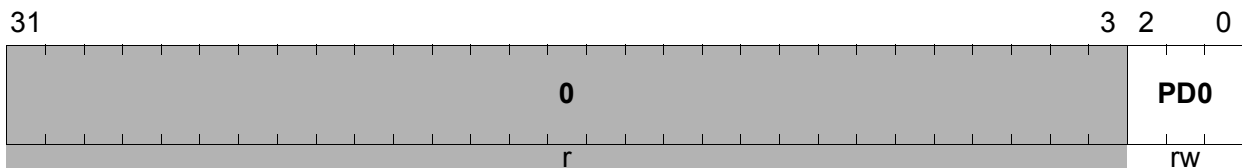
The Port 7 pad driver mode register contains bit fields that determine the output driver strength and the slew rate of ADC external multiplexer control outputs (class A1 outputs).

P7_PDR

Port 6 Pad Driver Mode Register

(40_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PD0	[2:0]	rw	Pad Driver Mode for P7.[7:0] XX0 _B Medium driver selected for P7.[7:0] XX1 _B Weak driver selected for P7.[7:0]
0	[31:3]	r	Reserved Read as 0; should be written with 0.

Analog-to-Digital Converter (ADC)

25.3.5 Analog Input Line to Analog Input Channel Connections

Table 25-16 defines the connections between the analog input lines AN[43:0] of the TC1796 and the analog inputs AIN[31:0] of the ADC0 and ADC1 modules. Note that for ADC1 only analog inputs 0 to 16 are connected. The bit GRPS (group select) located in several kernel registers determines the analog input multiplexer group number that is used for channel selection.

Table 25-16 AN[43:0] to ADC0/ADC1 Analog Input Connections

Analog Input Pin of TC1796	ADC0 Module		ADC1 Module	
	Connected to Analog Input of Module	GRPS	Connected to Analog Input of Module	GRPS
AN0	AIN0	0	–	–
AN1	AIN1			
AN2	AIN2			
AN3	AIN3			
AN4	AIN4		AIN0	0
AN5	AIN5		AIN1	
AN6	AIN6		AIN2	
AN7	AIN7		AIN3	
AN8	AIN8		–	–
AN9	AIN9			
AN10	AIN10			
AN11	AIN11			
AN12	AIN12			
AN13	AIN13			
AN14	AIN14			
AN15	AIN15			
AN16	AIN16	1		
AN17	AIN17			
AN18	AIN18			
AN19	AIN19			
AN20	AIN20			
AN21	AIN21			
AN22	AIN22			

Analog-to-Digital Converter (ADC)

Table 25-16 AN[43:0] to ADC0/ADC1 Analog Input Connections (cont'd)

Analog Input Pin of TC1796	ADC0 Module		ADC1 Module				
	Connected to Analog Input of Module	GRPS	Connected to Analog Input of Module	GRPS			
AN23	AIN23	1	–	–			
AN24 ¹⁾	AIN24						
AN25 ¹⁾	AIN25						
AN26 ¹⁾	AIN26						
AN27 ¹⁾	AIN27						
AN28 ¹⁾	AIN28						
AN29 ¹⁾	AIN29						
AN30 ¹⁾	AIN30						
AN31 ¹⁾	AIN31						
AN32	–	–	AIN4	0			
AN33			AIN5				
AN34			AIN6				
AN35			AIN7				
AN36			AIN8				
AN37			AIN9				
AN38			AIN10				
AN39			AIN11				
AN40			AIN12				
AN41			AIN13				
AN42			AIN14				
AN43			AIN15				
On-chip Die Temperature Sensor						AIN31	1 ²⁾

1) These analog input pins of the are also used as analog inputs of the Fast Analog/Digital Converter FADC.

2) Additionally, ADC1_CHCON15.EMUX must be set to 01_B and SCU_CON.DTSON must be set to 1.

25.3.6 On-Chip Connections

This section describes the on-chip interconnections of the ADC0/ADC1 modules.

25.3.6.1 Reference Voltage Selection

Generally, the ADC modules allow the analog reference voltage for each analog channel to be selected via the CHCONn.REF bit field. In the TC1796, only ADC0 is able to select among three analog reference voltages: V_{AREF0} , AIN0, and AIN1 (see [Figure 25-33](#)). In ADC1 always V_{AREF1} is selected as analog reference voltage source.

Note: The selections $ADC0_CHCONn.REF = 11_B$ must not be used. The settings of bit fields $ADC1_CHCONn.REF$ are “don’t care”.

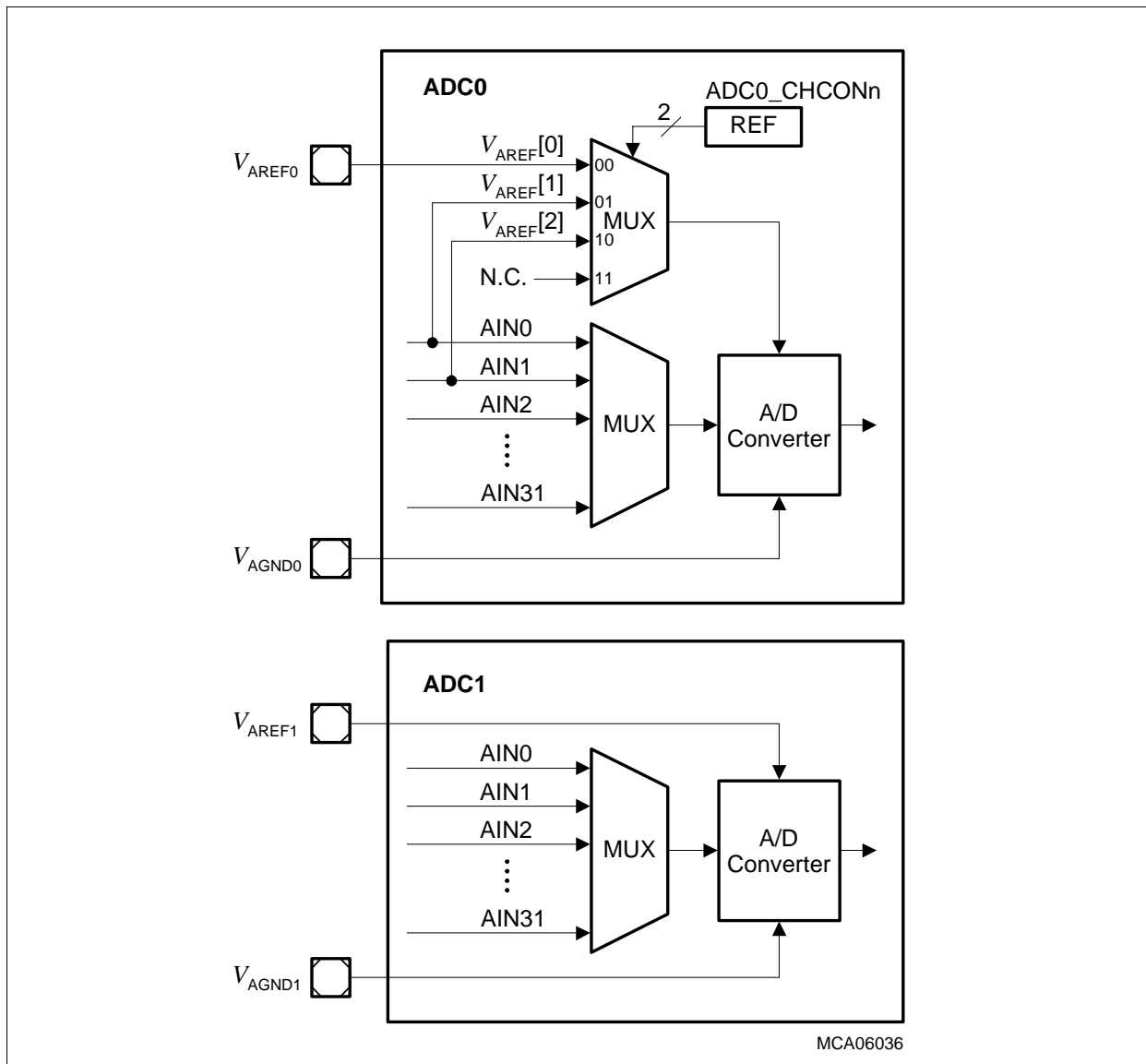


Figure 25-33 ADC0/ADC1 Reference Voltage Selections

Analog-to-Digital Converter (ADC)

25.3.6.2 Request/Gating Input Signal Connections

The on-chip interconnections of the A/D Converters request and trigger inputs are shown in **Figure 25-34**. Request and gating signals for ADC0 and ADC1, derived by pins, MSC outputs, or GPTA outputs are generated by the External Request Unit (ERU). The ERU is located in the System Control Unit (SCU) of the TC1796. Three output signals of GPTA0 are able to directly trigger A/D Converter channels.

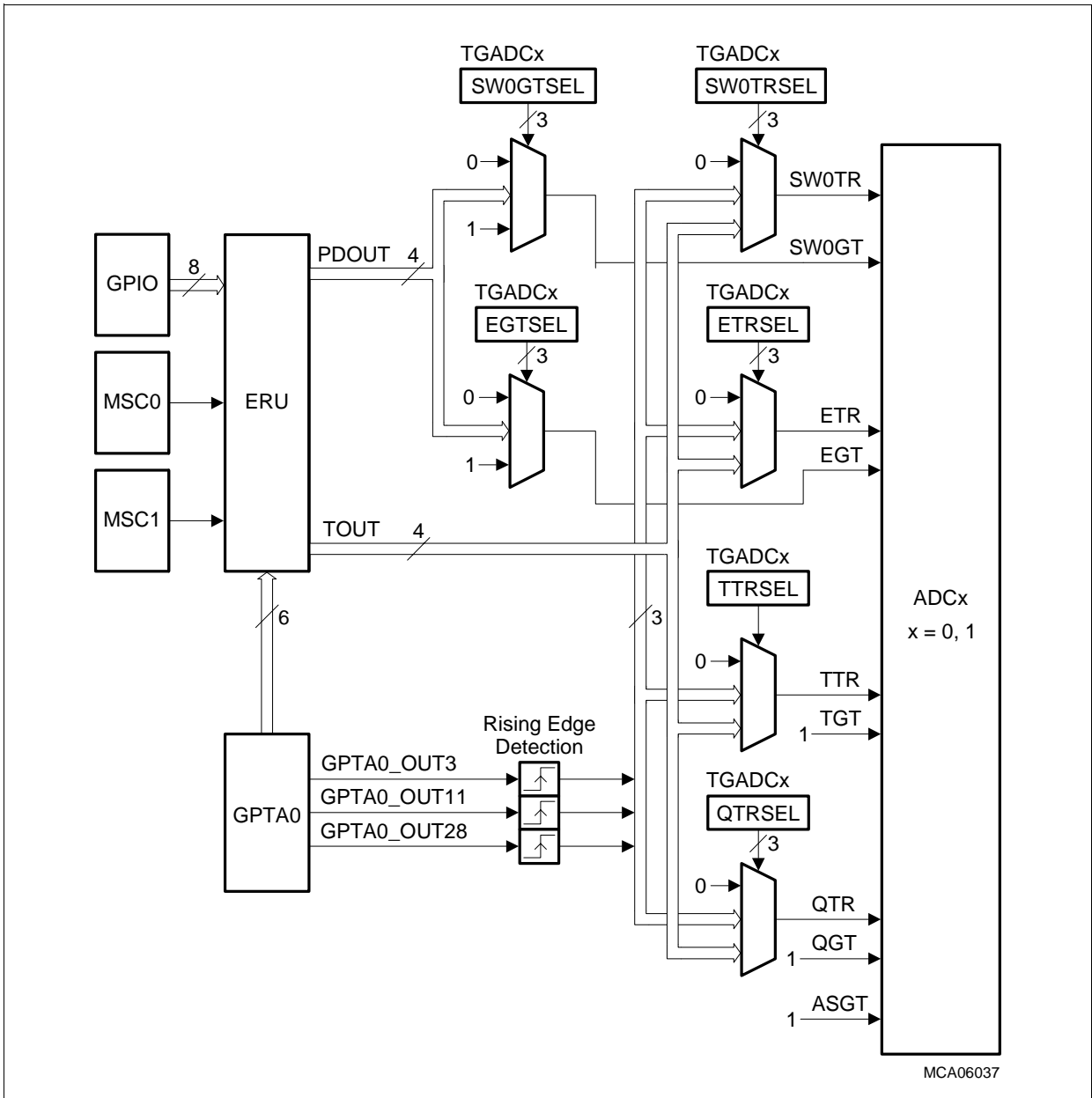


Figure 25-34 Request/Gating Input Signal Connections (ADC0/ADC1)

The signal source for trigger and gating signals of each A/D Converter, ADC0 and ADC1, is selected via register TGADC0 and TGADC1.

Analog-to-Digital Converter (ADC)

Trigger inputs can be completely disabled, connected to the TOUT signals of the ERU, or connected to one of three GPTA0 output signals. This third selection capability makes it possible to directly trigger A/D conversions with GPTA output signal transitions. When connected to the TOUT signals of the ERU, trigger signal source and behavior must be selected in the ERU. The functionality of the ERU is described in detail in [Section 5.3](#) on [Page 5-9](#) of the TC1796 System Units User's Manual.

Three gating input signals are connected to high level. This means, that its gating functionality is not used and trigger requests are always passed in the corresponding trigger control logic.

[Table 25-17](#) summarizes the trigger/gating capabilities for each trigger source and gives references to the figures in which the functionality of trigger/gating signals are shown.

Table 25-17 Trigger/Gating Source Input Selection

Trigger Source	Trigger/Gating Input	Selected Trigger/Gating Source
Timer (see Figure 25-5)	TTR	ERU trigger outputs TROUT[3:0] or three GPTA0 channels direct trigger outputs; selected by TGADCx.TTRSEL
	TGT	TGT = 1; timer gating always enabled;
External Event (see Figure 25-7)	ETR	ERU trigger outputs TROUT[3:0] or three GPTA0 channels direct trigger outputs; selected by TGADCx.ETRSEL
	EGT	non-inverted or inverted ERU pattern detection outputs PDOUT[3:1]; selected by TGADCx.EGTSEL
Software (see Figure 25-8)	SW0TR	ERU trigger outputs TROUT[3:0] or three GPTA0 channels direct trigger outputs; selected by TGADCx.SW0TRSEL
	SW0GT	non-inverted or inverted ERU pattern detection outputs PDOUT[0, 2, 3]; selected by TGADCx.SW0GTSEL
Auto-Scan (see Figure 25-9)	ASGT	ASGT = 1; auto-scan gating always enabled;
Queue (see Figure 25-15 and Figure 25-16)	QTR	ERU trigger outputs TROUT[3:0] or three GPTA0 channels direct trigger outputs; selected by TGADCx.QTRSEL
	QGT	QGT = 1; queue gating always enabled;

Analog-to-Digital Converter (ADC)

Field	Bits	Type	Description
SW0TRSEL	[6:4]	rw	<p>SW0 Trigger Request Selection</p> <p>This bit determines which trigger source will be used for the ADCx SW0 trigger request input.</p> <p>000_B SW0TR = 0; no trigger function of SW0TR. 001_B SW0TR is connected to GPTA0_OUT3. 010_B SW0TR is connected to GPTA0_OUT11. 011_B SW0TR is connected to GPTA0_OUT28. 100_B SW0TR is connected to TOUT0. 101_B SW0TR is connected to TOUT1. 110_B SW0TR is connected to TOUT2. 111_B SW0TR is connected to TOUT3.</p>
QTRSEL	[10:8]	rw	<p>Queue Trigger Request Selection</p> <p>This bit determines which trigger source will be used for the ADCx queue trigger request input QTR.</p> <p>000_B QTR = 0; no trigger function of QTR. 001_B QTR is connected to GPTA0_OUT3. 010_B QTR is connected to GPTA0_OUT11. 011_B QTR is connected to GPTA0_OUT28. 100_B QTR is connected to TOUT0. 101_B QTR is connected to TOUT1. 110_B QTR is connected to TOUT2. 111_B QTR is connected to TOUT3.</p>
TTRSEL	[14:12]	rw	<p>Timer Trigger Request Selection</p> <p>This bit determines which trigger source will be used for the ADCx timer trigger request input TTR.</p> <p>000_B TTR = 0; no trigger function of TTR. 001_B TTR is connected to GPTA0_OUT3. 010_B TTR is connected to GPTA0_OUT11. 011_B TTR is connected to GPTA0_OUT28. 100_B TTR is connected to TOUT0. 101_B TTR is connected to TOUT1. 110_B TTR is connected to TOUT2. 111_B TTR is connected to TOUT3.</p>

Analog-to-Digital Converter (ADC)

Field	Bits	Type	Description
EGTSEL	[18:16]	rw	<p>External Gating Selection</p> <p>This bit determines which trigger source will be used for the ADCx external gating input EGT.</p> <p>000_B EGT = 0; conversion request source “External Event” is permanently disabled.</p> <p>001_B EGT is connected to PDOOUT1.</p> <p>010_B EGT is connected to PDOOUT2.</p> <p>011_B EGT is connected to PDOOUT3.</p> <p>100_B EGT = 1; conversion request source “External Event” is permanently <u>enabled</u>.</p> <p>101_B EGT is connected to <u>PDOOUT1</u>.</p> <p>110_B EGT is connected to <u>PDOOUT2</u>.</p> <p>111_B EGT is connected to PDOOUT3.</p>
SW0GTSEL	[22:20]	rw	<p>SW0 Gating Selection</p> <p>This bit determines which trigger source will be used for the ADCx software gating input SW0GT.</p> <p>000_B SW0GT = 0; conversion request source “Software” is permanently disabled.</p> <p>001_B SW0GT is connected to PDOOUT0.</p> <p>010_B SW0GT is connected to PDOOUT2.</p> <p>011_B SW0GT is connected to PDOOUT3.</p> <p>100_B SW0GT = 1; conversion request source “Software” is permanently <u>enabled</u>.</p> <p>101_B SW0GT is connected to <u>PDOOUT0</u>.</p> <p>110_B SW0GT is connected to <u>PDOOUT2</u>.</p> <p>111_B SW0GT is connected to <u>PDOOUT3</u>.</p>
0	3, 7, 11, 15, 19, [31:23]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

25.3.6.3 Service Request Output Lines

Each ADC module has eight service request outputs. Four of these service request outputs (SR[3:0]) are connected to interrupt service request nodes while the remaining four service request outputs (SR[7:4]) are connected as DMA request line with the DMA controller of the TC1796 (see [Table 25-18](#)).

Table 25-18 Service Request Lines of ADC0/ADC1

Module	Service Request Line	Connected To	Description
ADC0	SR0	ADC0_SRC0	ADC0 Service Request Node 0
	SR1	ADC0_SRC1	ADC0 Service Request Node 1
	SR2	ADC0_SRC2	ADC0 Service Request Node 2
	SR3	ADC0_SRC3	ADC0 Service Request Node 3
	SR4	CH02_REQI2 CH12_REQI3	DMA Channel 02 Request Input 2 DMA Channel 12 Request Input 3
	SR5	CH03_REQI2 CH13_REQI3	DMA Channel 03 Request Input 2 DMA Channel 13 Request Input 3
	SR6	CH10_REQI3 CH14_REQI3	DMA Channel 10 Request Input 3 DMA Channel 14 Request Input 3
	SR7	CH11_REQI3 CH15_REQI3	DMA Channel 11 Request Input 3 DMA Channel 15 Request Input 3
ADC1	SR0	ADC1_SRC0	ADC1 Service Request Node 0
	SR1	ADC1_SRC1	ADC1 Service Request Node 1
	SR2	ADC1_SRC2	ADC1 Service Request Node 2
	SR3	ADC1_SRC3	ADC1 Service Request Node 3
	SR4	CH02_REQI3 CH12_REQI4	DMA Channel 02 Request Input 3 DMA Channel 12 Request Input 4
	SR5	CH03_REQI3 CH13_REQI4	DMA Channel 03 Request Input 3 DMA Channel 13 Request Input 4
	SR6	CH10_REQI4 CH14_REQI4	DMA Channel 10 Request Input 4 DMA Channel 14 Request Input 4
	SR7	CH11_REQI4 CH15_REQI4	DMA Channel 11 Request Input 4 DMA Channel 15 Request Input 4

25.3.6.4 Service Request Control Registers

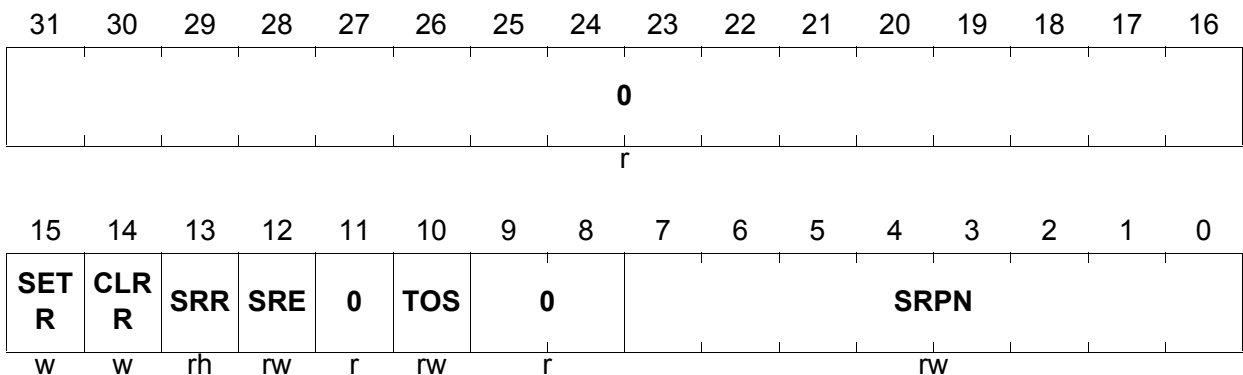
Each of the four interrupt request nodes of ADC0 and ADC1 is controlled by a service request control register.

SRC_m (m = 0-3)

Service Request Control Register m

(1FC_H-m*4)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SRPN	[7:0]	rw	Service Request Priority Number
TOS	10	rw	Type of Service Control
SRE	12	rw	Service Request Enable
SRR	13	rh	Service Request Flag
CLRR	14	w	Request Clear Bit
SETR	15	w	Request Set Bit
0	[9:8], 11, [31:16]	r	Reserved Read as 0; should be written with 0.

Note: Additional information about service request nodes and the service request control registers are described on [Page 14-3](#) of the TC1796 User's Manual System Units part (Volume 1).

25.3.6.5 Die Temperature Sensor

AIN31of ADC1 is connected to the output of an on-chip die temperature measurement sensor (see also [Page 25-101](#)). Detailed information about this die temperature sensor is provided in the section "[Die Temperature Sensor](#)" on [Page 5-48](#) of the TC1796 User's Manual System Unit (Volume 1).

26 Fast Analog-to-Digital Converter (FADC)

The TC1796 contains two medium speed Analog-to-Digital Converters (ADC0 and ADC1) with identical functionality and a third Fast Analog-to-Digital Converter (FADC).

ADC0 and ADC1 provide 2-3 μ s conversion time (10-bit) and are intended mainly for single-ended signals. The FADC offers conversion rates of less than 500 ns, thus allowing sampling of high frequency signals. For slow and mid-range frequency signals, heavy over-sampling can be performed to avoid the use of expensive filters.

ADC0 and ADC1 are described in [Chapter 25](#). This chapter describes in detail the FADC and contains the following sections:

- Functional description of the FADC Kernel (see [Page 26-2](#))
- FADC kernel registers description (see [Page 26-27](#))
- TC1796 implementation-specific details and registers of the FADC module, including on-chip interconnections, service request control, address decoding, and clock control (see [Page 26-56](#))

Note: The FADC Kernel register names described in [Section 26.2](#) will be referenced in the TC1796 User's Manual with the module name prefix "FADC_" for the ADC1 interface.

Fast Analog-to-Digital Converter (FADC)

26.1 FADC Kernel Description

The on-chip FADC module of the TC1796 is primarily a four-channel A/D converter with 10-bit resolution that operates by the method of successive approximation.

Features

- Extreme fast conversion, 21 cycles of f_{FADC} clock (280 ns @ $f_{FADC} = 75$ MHz)
- 10-bit A/D conversion
 - Higher resolution by averaging of consecutive conversions is supported
- Successive approximation conversion method
- Four differential input channels
- Offset and gain calibration support for each channel
- Differential input amplifier with programmable gain of 1, 2, 4 and 8 for each channel
- Free-running (Channel Timers) or triggered conversion modes
- Trigger and gating control for external signals
- Built-in Channel Timers for internal triggering
- Channel timer request periods independently selectable for each channel
- Selectable, programmable antialiasing and data reduction filter block

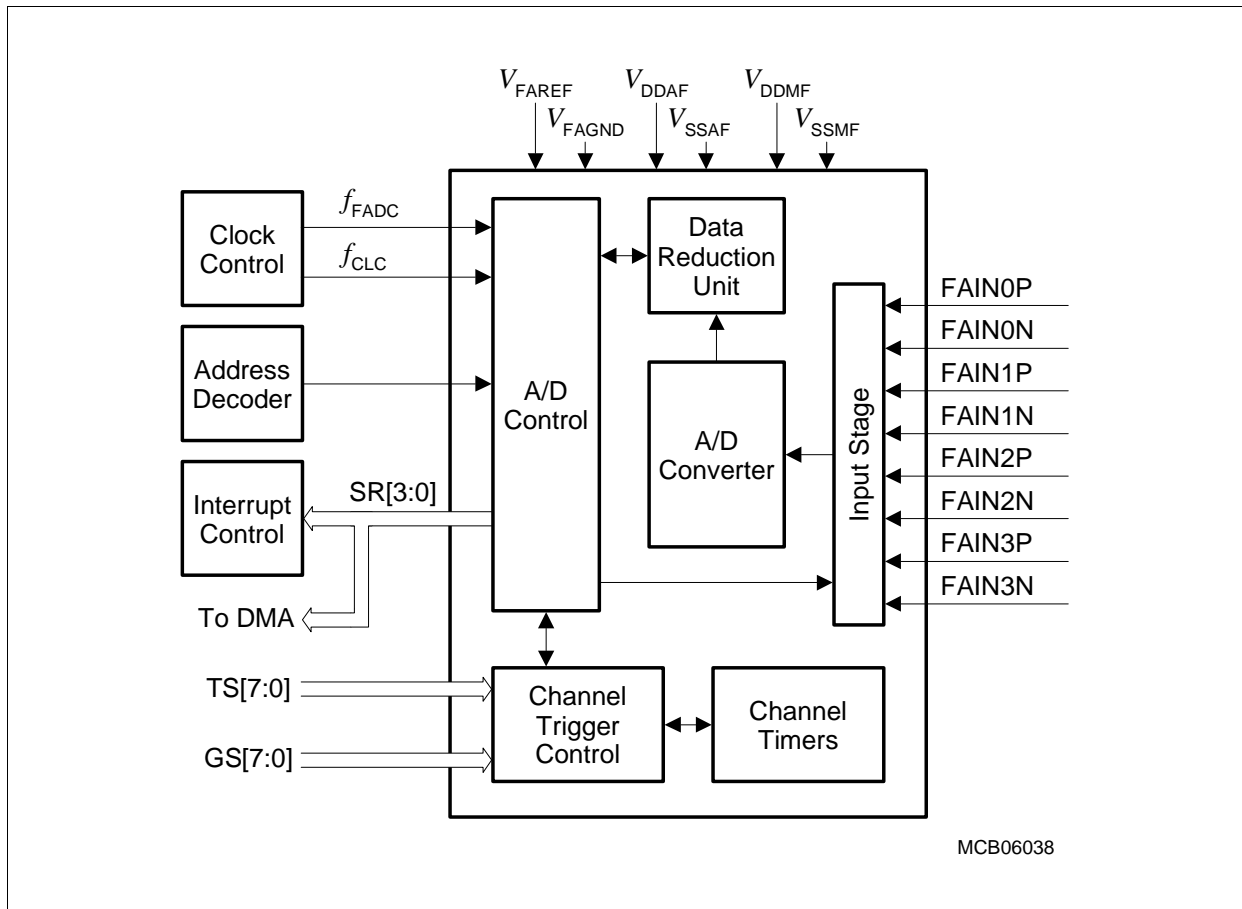


Figure 26-1 Block Diagram of the FADC Module

Fast Analog-to-Digital Converter (FADC)

As shown in **Figure 26-1**, the main FADC functional blocks are:

- The Input Stage – contains the differential inputs and the programmable amplifier
- The A/D Converter – is responsible for the analog-to-digital conversion
- The Data Reduction Unit – contains programmable antialiasing and data reduction filters
- The Channel Trigger Control block – determines the trigger and gating conditions for the four FADC channels
- The Channel Timers – can independently trigger the conversion of each FADC channel
- The A/D Control block is responsible for the overall FADC functionality.

The FADC module is supplied by the following power supply and reference voltage lines:

- V_{DDMF}/V_{DDMF} : FADC Analog Part Power Supply (3.3 V)
- V_{DDAF}/V_{DDAF} : FADC Analog Part Logic Power Supply (1.5 V)
- V_{FAREF}/V_{FAGND} : FADC Reference Voltage (3.3 V)/FADC Reference Ground

Fast Analog-to-Digital Converter (FADC)

26.1.1 Analog Inputs

Two analog inputs are assigned to each of the four FADC channels. The FADC analog input stage contains a differential channel amplifier for each channel and a common differential amplifier controlling the gain. The two input lines FAINxP/FAINxN (x = 0-3) of each channel analog input stage can be enabled independently in order to support single-ended measurements.

The gain of the common amplifier used during an A/D conversion is selected individually for each of the four channels depending on the currently active channel x.

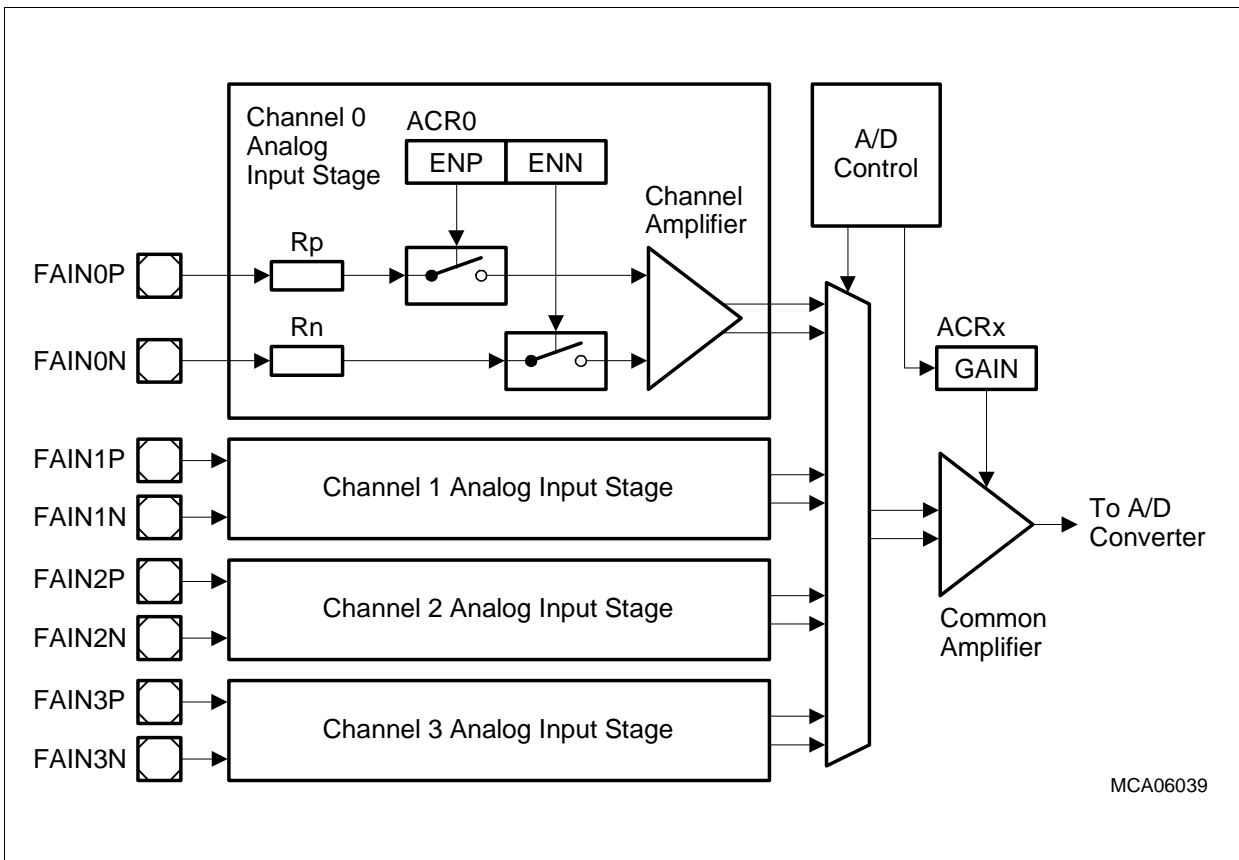


Figure 26-2 Analog Input Stage

In Multiplexer Test Mode (GCR.MUXTM = 1), the channel amplifiers are disconnected from the common amplifier. The measured conversion result in multiplexer test mode should be $10\ 0000\ 0000_B = (200_H)$ plus/minus the offset of the common amplifier.

26.1.1.1 Analog Input Stage Configurations

The analog input stage makes it possible to select different configurations to be selected independently for each FADC channel x. These combinations are shown in **Figure 26-3**.

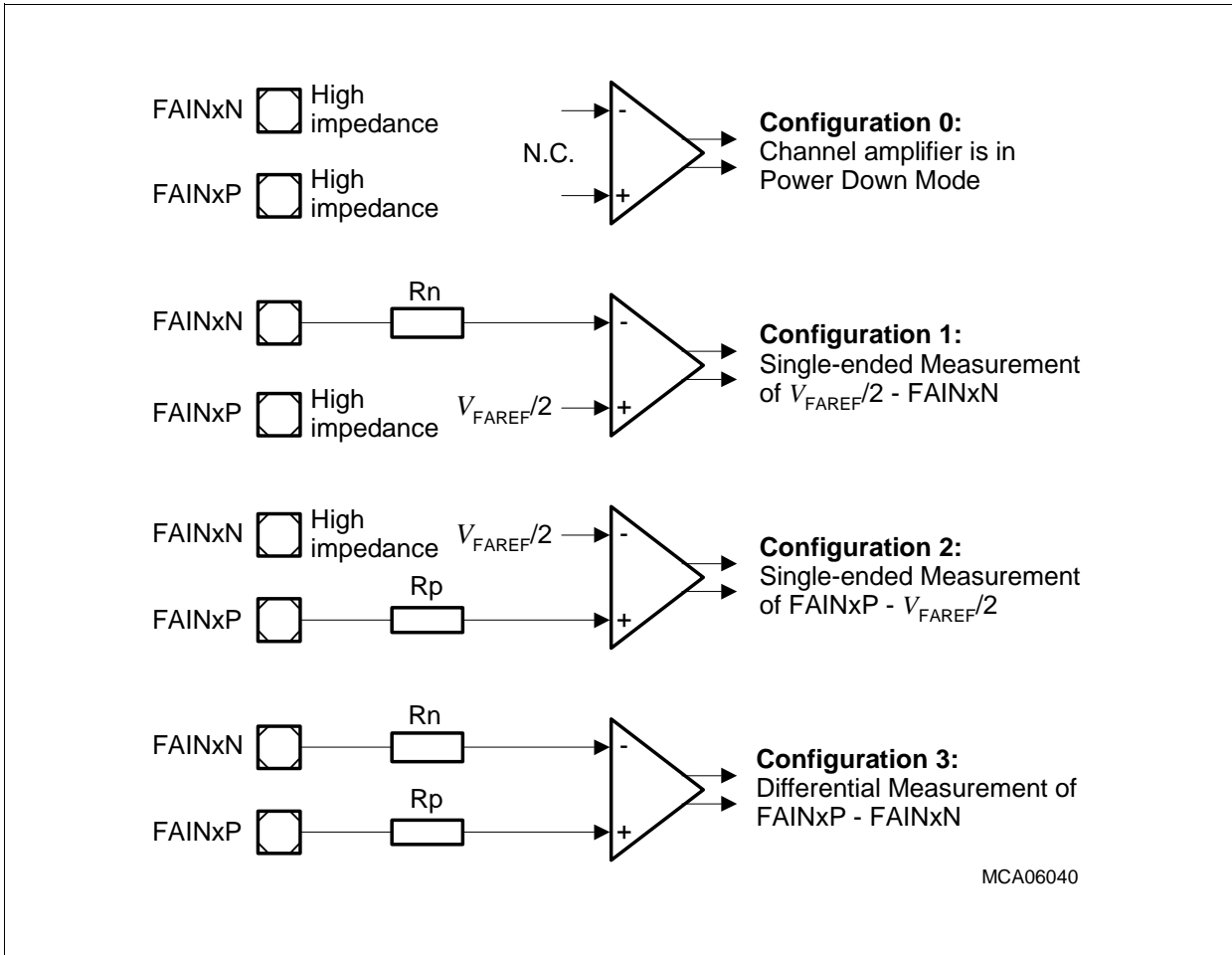


Figure 26-3 Analog Input Stage Configurations

Configuration 0 ($ACRx.ENP = ACRx.ENN = 0$)

FADC channel x inputs FAINxP and FAINxN are in a high impedance state and the channel amplifier of the analog input stage is in power-down mode.

Configuration 1 ($ACRx.ENP = 0$ and $ACRx.ENN = 1$)

This configuration enables the single-ended measurement mode for $V_{VAREF}/2 - FAINxN$: The positive analog input FAINxP is disconnected and is in a high impedance state. The negative analog input FAINxN is connected to the channel amplifier (input impedance is determined by R_n). The positive input of the channel amplifier is connected to $V_{FAREF}/2$ (1.65 V with $V_{FAREF} = 3.3$ V). If the voltage at the negative input FAINxN varies, the FADC will deliver conversion results as follows (gain = 1 selected by $ACRx.GAIN = 00_B$):

Fast Analog-to-Digital Converter (FADC)

- FAINxN = 0 V: FADC conversion result is 768
- FAINxN = 3.3 V: FADC conversion result is 256

To cover the full range of the measurement result in this single-ended measurement mode, a gain of 2 must be selected ($ACRx.GAIN = 01_B$). With gain = 2, the FADC will deliver conversion results as follows:

- FAINxN = 0 V: FADC conversion result is 1023
- FAINxN = 3.3 V: FADC conversion result is 0

The voltage at the disconnected positive analog input FAINxP has no influence on the conversion result.

Configuration 2 ($ACRx.ENP = 1$ and $ACRx.ENN = 0$)

This configuration enables the single-ended measurement mode for FAINxP - $V_{VAREF}/2$. The negative analog input FAINxN is disconnected and in a high impedance state. The positive analog input FAINxP is connected to the channel amplifier (input impedance is determined by Rp). The negative input of the channel amplifier is connected to $V_{FAREF}/2$ (1.65 V with $V_{FAREF} = 3.3$ V). If the voltage at the positive input FAINxP varies, the FADC will deliver conversion results as follows (gain = 1 selected by $ACRx.GAIN = 00_B$):

- FAINxP = 0 V: FADC conversion result is 256
- FAINxP = 3.3 V: FADC conversion result is 768

To cover the full range of the measurement result in this single-ended measurement mode, a gain of 2 must be selected ($ACRx.GAIN = 01_B$). With gain = 2, the FADC will deliver conversion results as follows:

- FAINxP = 0 V: FADC conversion result is 0
- FAINxP = 3.3 V: FADC conversion result is 1023

The voltage at the disconnected negative analog input FAINxN has no influence on the conversion result.

Configuration 3 ($ACRx.ENP = ACRx.ENN = 1$)

This configuration enables the differential measurement mode for FAINxP - FAINxN. Both analog inputs, FAINx.N and FAINxP, are connected to the channel amplifier inputs. Their impedances are determined by Rn and Rp. The full measurement range is available.

Table 26-1 provides a summary of the conversion results that are available at the different measurement modes. This table assumes a reference voltage of $V_{FAREF} = 3.3$ V.

Fast Analog-to-Digital Converter (FADC)

Table 26-1 Conversion Results in the Different Measurement Modes

Measurements Modes	ACRx. ENP	ACRx. ENN	FAINxP	FAINxN	ACRx. GAIN	Conv. Results
Single-ended Measurement Mode (Configuration 1)	0	1	don't care	0	00 _B	768
				3.3		256
				0	01 _B	1023
				3.3		0
Single-ended Measurement Mode (Configuration 2)	1	0	0 V	don't care	00 _B	256
			3.3			768
			0		01 _B	0
			3.3			1023
Differential Measurement Mode (Configuration 3)	1	1	0	1.65	00 _B	256
			0	3.3		0
			1.65	0		768
			3.3	0		1023
			3.3	1.65	01 _B	768
			0	1.65		0
			1.65	0		1023
			3.3	1.65		1023

Note: Due to the temperature characteristics of offset and gain of the internal amplifiers a TUE (total unadjusted error) cannot be specified. The input impedance for Rp and Rn is defined in the TC1796 Data Sheet.

Note: The analog input lines of the FADC can also be used as input lines for other ADCs. If both (regular ADC and FADC) are connected to the same pin at the same time, the input impedances of the analog inputs must be taken into account in order to minimize signal distortions and measurement errors.

Fast Analog-to-Digital Converter (FADC)

The conversion result for FADC channel x is given by the following equation:

$$\text{Conversion Result } V_{Mx} = \text{GAIN}_x ((V_{\text{FAINXP}} - V_{\text{FAREFM}}) + (V_{\text{FAREFM}} - V_{\text{FAINXN}})) \quad (26.1)$$

$$\text{with } V_{\text{FAREFM}} = V_{\text{FAGND}} + (V_{\text{FAREF}} - V_{\text{FAGND}})/2$$

The absolute value of the result V_{Mx} is limited to V_{FAREF} . The mapping of the conversion result V_{Mx} to the binary result RCHx.ADRES is as follows (see also [Table 26-1](#)):

$$V_{Mx} = -V_{\text{FAREF}} \quad \text{leads to RCHx.ADRES} = 00\ 0000\ 0000_{\text{B}}$$

$$V_{Mx} = 0 \quad \text{leads to RCHx.ADRES} = 10\ 0000\ 0000_{\text{B}}$$

$$V_{Mx} = +V_{\text{FAREF}} \quad \text{leads to RCHx.ADRES} = 11\ 1111\ 1111_{\text{B}}$$

For single-ended measurements, the following values are taken into account:

- if $\text{ENP}_x = 0$ then $V_{\text{FAINXP}} = V_{\text{FAREFM}}$
- if $\text{ENN}_x = 0$ then $V_{\text{FAINXN}} = V_{\text{FAREFM}}$

26.1.2 Conversion Timing

The conversion time of the FADC is determined by the frequency of clock f_{FADC} . The conversion time defined below includes sampling, converting, and storing of the conversion result.

$$\text{Conversion Time}_{\text{FADC}} = 21 \times \frac{1}{f_{\text{FADC}}} \quad (26.2)$$

Clock f_{FADC} is generated outside the FADC kernel in a product specific clock control unit (see [Page 26-58](#)).

26.1.3 Channel Triggers

As shown in **Figure 26-4**, the trigger behavior of an FADC channel x is determined by its channel x trigger control logic. An FADC channel x can be triggered by three trigger sources:

- External trigger source input signals TS[7:0]
- Internal channel x timer trigger signal
- Internal neighbor channel trigger signal

If one of these trigger sources is selected, becomes active, and the gating logic (controlling the external gating source inputs GS[7:0]) is programmed to enable trigger signals (signal ECHTIMx set), the conversion request flag CRFx becomes set indicating a valid request for FADC channel x.

The gating source input and gating mode selection logic generate an enable signal for channel x timer that determines whether any of the three conversion trigger signal sources is allowed to set the channel x conversion request flag CRFx.

This control logic does the following control tasks:

- Gating source input selection (CFGRx.GSEL)
- Gating mode selection (CFGRx.GM)
- Trigger source input selection (CFGRx.TSEL)
- Trigger mode selection (CFGRx.TM)
- Channel timer request generation
- Conversion request flag set/clear control

All these control tasks are executed independently in each of the four FADC channels.

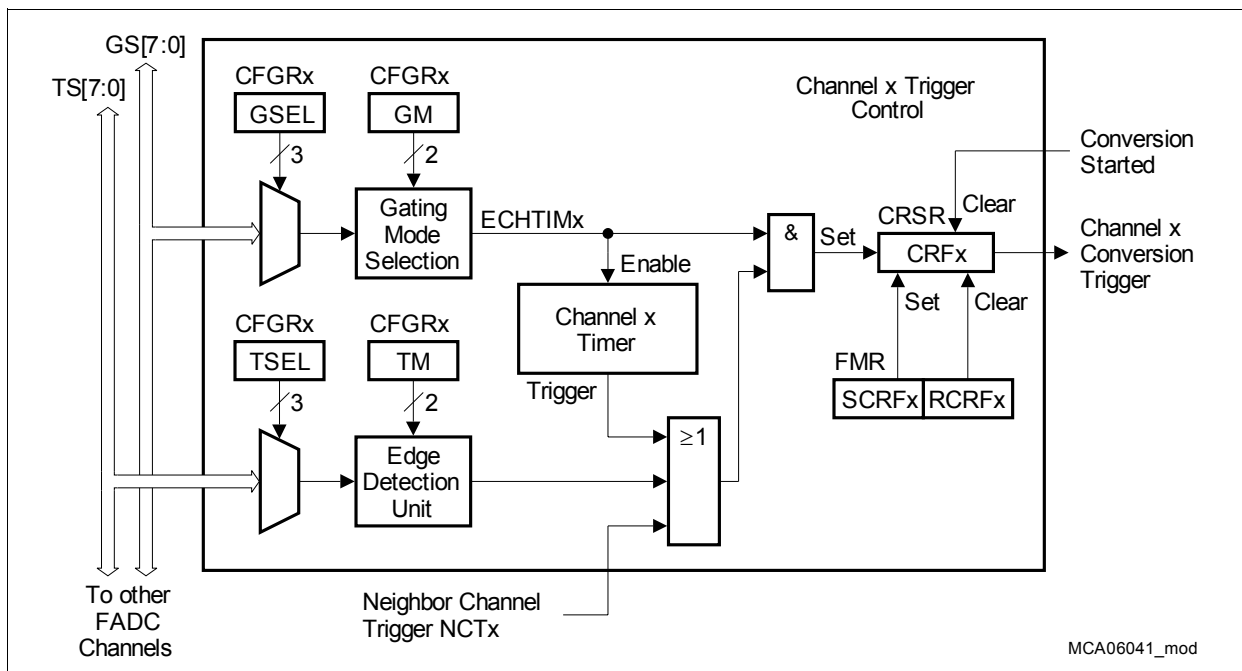


Figure 26-4 Channel Trigger Control Logic

Fast Analog-to-Digital Converter (FADC)

Table 26-2 describes the possible gating modes (enabled, disabled, active gating source input polarity) of an FADC channel.

Table 26-2 Gating Modes

CFGRx.GM	Gating Mode
00 _B	Conversion requests are disabled and Channel Timer is stopped. CRFx never becomes set (by hardware).
01 _B	Conversion requests and Channel Timer are always enabled. CRFx becomes set by hardware with each active trigger signal.
10 _B	When gating source input GS _n = 1 (as selected by CFGRx.GSEL), the Channel Timer is enabled and the conversion request flag CRFx becomes set by hardware with each active trigger signal.
11 _B	When gating source input GS _n = 0 (as selected by CFGRx.GSEL), the Channel Timer is enabled and the conversion request flag CRFx becomes set by hardware with each active trigger signal.

An edge detection unit determines which edge of the trigger source input signal (as selected by CFGRx.TSEL) is generating a conversion request trigger signal. Rising, falling or both edges can be selected for trigger signal generation.

Table 26-3 Trigger Modes

CFGRx.TM	Trigger Mode
00 _B	No trigger signal generated. Edge detection unit always delivers a 0.
01 _B	A conversion request trigger signal is generated on a rising edge of trigger source input TS _n (as selected by CFGRx.TSEL).
10 _B	A conversion request trigger signal is generated on a falling edge of trigger source input TS _n (as selected by CFGRx.TSEL).
11 _B	A conversion request trigger signal is generated on a rising or falling edge of trigger source input TS _n (as selected by CFGRx.TSEL).

The Conversion Request Flag CRFx is cleared by hardware when the conversion of channel x is started. CRFx can also be set or cleared by software via bits in the Flag Modification Register FMR. Writing a 1 to FMR.SCRF sets CRFx. Writing a 1 to FMR.RCRF clears CRFx (independently of FMR.SCRF).

Fast Analog-to-Digital Converter (FADC)

26.1.4 Channel Timer

Each of the FADC channels contains an 8-bit Channel Timer that can be used to generate periodic conversion requests. The Channel Timer is built up by a decrementing counter that is reloaded with a programmable value. When the Channel Timer reaches zero while running, a Channel Timer trigger event is generated and the Channel Timer is reloaded with the reload value $CFGRx.CTREL$ when the requested conversion is started. With the start of the A/D conversion, request flag $CRSR.CRFx$ is also cleared. Note that the request flag is set by a timer trigger event only if the gating condition is met (signal $ECHTIMx$ in **Figure 26-4** set).

A clock divider, fed by the module clock f_{FADC} and common for all Channel Timers, generates several clock signals with different periods. Bit field $CFGRx.CTF$ selects whether or not the channel x timer clock f_{CTx} is enabled and, if enabled, the frequency of channel x timer input clock f_{CTx} .

While the Channel Timer is disabled ($CFGRx.CTM = 00_B$) or if the gating condition is not met (gating line $ECHTIMx$ delivers 0), the channel x timer value is set to 04_H .

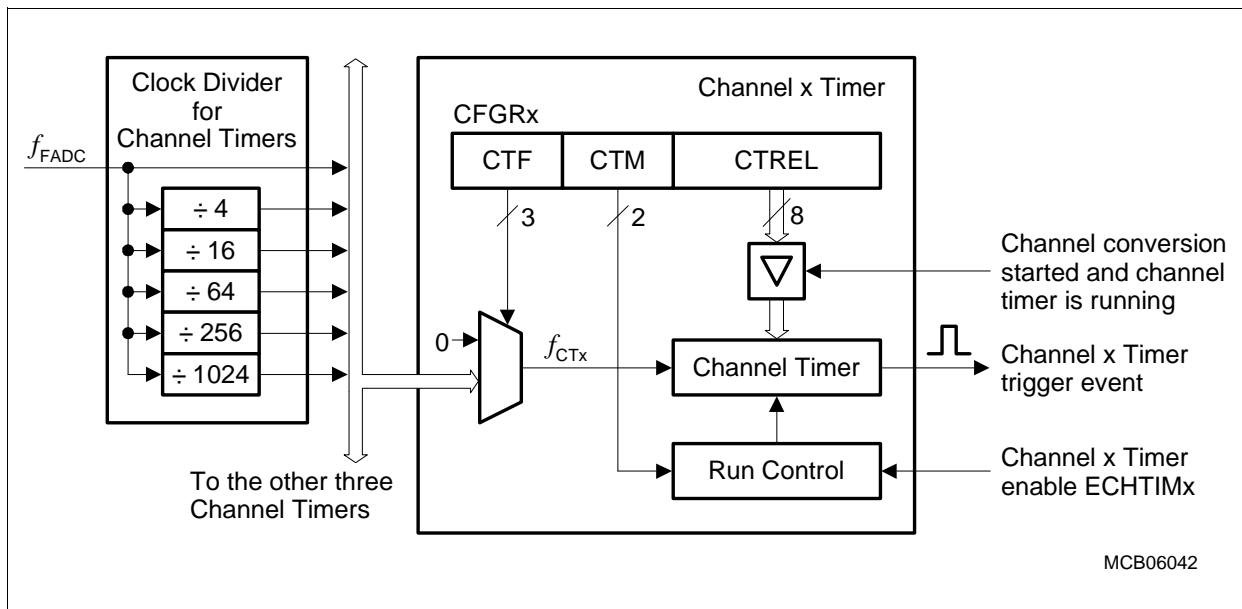


Figure 26-5 Channel Timer Block Diagram

Due to the common divider, the first event at the trigger output of $CHTIMx$ after the start has a maximum jitter of one clock cycle of the selected channel x timer clock f_{CTx} .

A Channel Timer x input clock pulse at f_{CTx} is ignored if it occurs in the f_{FADC} clock cycle directly after the Channel x Timer has reached 0. If there is at least one f_{FADC} clock cycle between two Channel x Timer input clock pulses, all Channel x Timer input clock pulses are taken into account. This leads to a Channel x Timer reload value ($CFGRx.CTREL$) whose definition depends on the ratio of f_{CLC} / f_{CTx} :

Fast Analog-to-Digital Converter (FADC)

- $f_{CLC} / f_{CTx} = 1$ Channel x timer divide factor = CTREL + 2
- $1 < f_{CLC} / f_{CTx} < 2$ Not recommended to be used!
- $2 \leq f_{CLC} / f_{CTx}$ Channel x timer divide factor = CTREL + 1

In case of a f_{CLC} / f_{CTx} ratio between 1 and 2, a mixture of both divide factor definitions occurs depending on the divider ratio as programmed by the fractional divider value. Therefore, it is recommended that this ratio should not be used.

26.1.5 Control Logic

26.1.5.1 Conversion Control

A conversion is started when at least one of the CRSR.CRFx bits is set. A running conversion cannot be aborted and is indicated by the busy flag CRSR.BSYx set. The corresponding bit CRSR.CRFx is cleared by hardware when the conversion starts.

26.1.5.2 Static Channel Priority

If more than one conversion request flag CRSR.CRFx (x = 0-3) is set, the channels are converted according to a priority scheme as defined by the bit field GCR.CRPRIO (without respecting the status of the current filter sequences).

Table 26-4 Static Channel Request Priority

Priority	GCR.CRPRIO			
	00 _B	01 _B	10 _B	11 _B
High	Channel 0	Channel 1	Channel 2	Channel 3
	Channel 1	Channel 2	Channel 3	Channel 0
	Channel 2	Channel 3	Channel 0	Channel 1
Low	Channel 3	Channel 0	Channel 1	Channel 2

26.1.5.3 Dynamic Priority Assignment

If dynamic priority assignment is enabled (GCR.DPAEN = 1), a channel that has the only active gate signal (signal ECHTIMx in [Figure 26-4](#)) among the four channels gets the highest priority (GCR.CRPRIO is set to the number of the channel). If more than one channel gating signal is active, GCR.CRPRIO is not changed automatically. In this case, it can be changed by software.

Fast Analog-to-Digital Converter (FADC)

The dynamic priority assignment feature ensures that two channels can be sampled in equidistant intervals.

26.1.5.4 Clock Generation

As shown in [Figure 26-1](#), the FADC module is provided with two clock signals: f_{CLC} and f_{FADC} . Clock f_{CLC} is used inside the FADC kernel for control purposes such as clocking of control logic, register operations, trigger detection, or filter calculation. The clock rate of f_{FADC} is programmable. Clock f_{FADC} is used inside the FADC kernel as the clock for the Channel Timer and the complete analog part.

The clock control as implemented in the TC1796 is described on [Page 26-58](#).

26.1.5.5 Suspend Mode Behavior

When a suspend/idle mode request is generated for the FADC module, a currently running conversion is completely finished (not aborted) and, if selected, a filter calculation still takes place. Thereafter, no new conversion will be started and the state of the FADC module is frozen until the suspend/idle mode request is released again.

26.1.6 Data Reduction Unit

If one or more channels of the FADC is operating in a fast continuous mode (for example, by using the Channel Timers as request sources with fast conversion data request rates), it can be sometimes difficult or even impossible for a CPU or another bus master to collect all conversion results without the risk of losing conversion data. Therefore, a Data Reduction Unit is implemented in the FADC that operates as a kind of antialiasing filter. This unit allows the number of conversion data requests that are issued to the CPU or other bus masters to be reduced by adding multiple conversion results according to a certain algorithm and presenting it to the CPU or other bus masters with a reduced conversion request rate.

The Data Reduction Unit contains two filter blocks. Each filter block allows selection of its input data source. The input data sources are the conversion result registers of the four A/D converter channel. Both filter blocks can also be concatenated. When the result of a filter operation is stored in one of the final result registers, a service request can be generated. Each filter block basically contains adder logic and intermediate storage registers that allow support for typical digital filter operations such as moving average calculations with intermediate results.

In order to achieve a higher resolution of the conversion result by averaging over several single conversions, the filter blocks only accumulates single conversion results.

Fast Analog-to-Digital Converter (FADC)

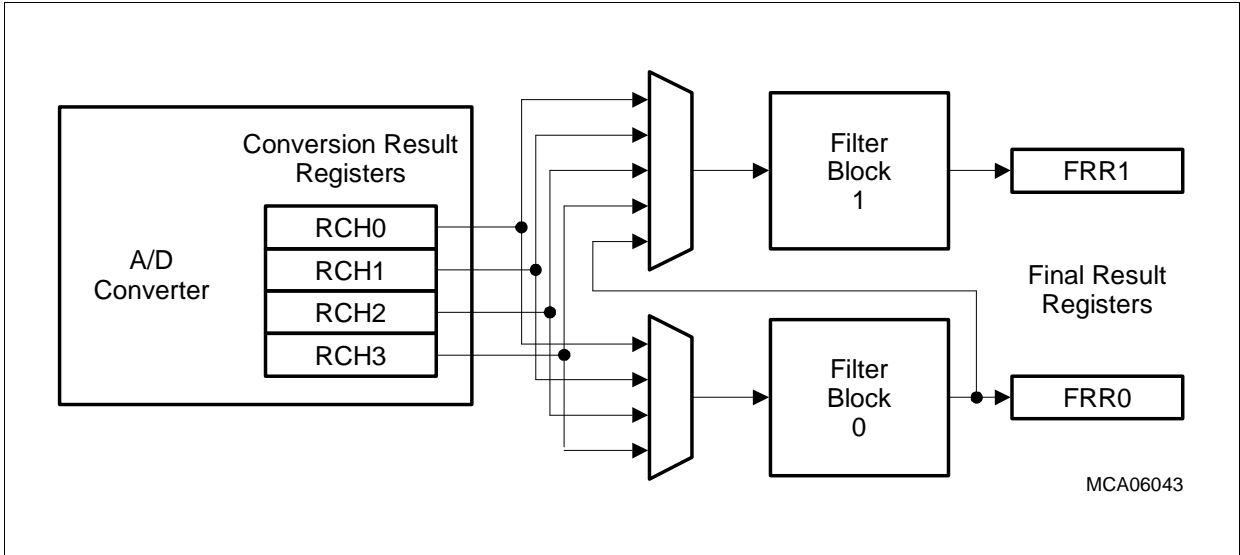


Figure 26-6 FADC Filter Blocks

26.1.6.1 Filter Block Structure

The filter block consists of an adder and several result registers for calculating filter output data from the filter input data. The Current Result Register CRRn is used for adding up conversion results. After a programmable number of conversion results have been added, the contents of CRRn are stored as intermediate result in the Intermediate Result Register IRR1n. The three intermediate result registers operate as a kind of pipeline. Before IRR1n is overwritten, IRR2n is transferred to IRR3n, and IRR1n is transferred to IRR2n. The Final Result Register FRRn stores the sum that is built by the contents of the current result register and the intermediate result registers.

All result registers of the filter block are fully transparent and can be read at any time. Please note that only one intermediate result register (IRR11) is available in filter block 1.

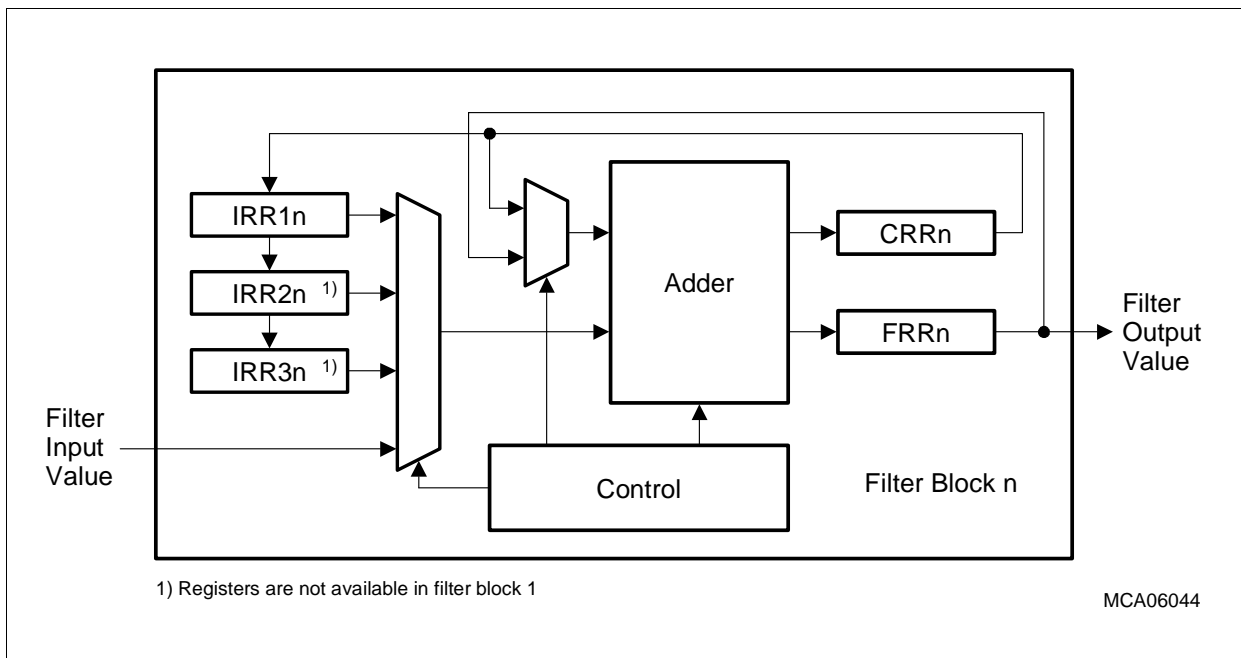


Figure 26-7 Filter Block Structure

Fast Analog-to-Digital Converter (FADC)

26.1.6.2 Filter Block Operation

Figure 26-8 illustrates how filter output values are calculated in a filter block. The following conditions are set for this example:

- A continuous A/D conversion is running on channel x.
- The filter input selection is set to channel x ($FCRn.INSEL = 1XX_B + x$).
- The addition length is 4 ($FCRn.ADDL = 011_B$). This means that four conversion results build one intermediate result.
- The final result is calculated by a moving average over the last four intermediate results ($FCRn.MAVL = 11_B$).
- The filter registers are at initial state (loaded with 0000_H).

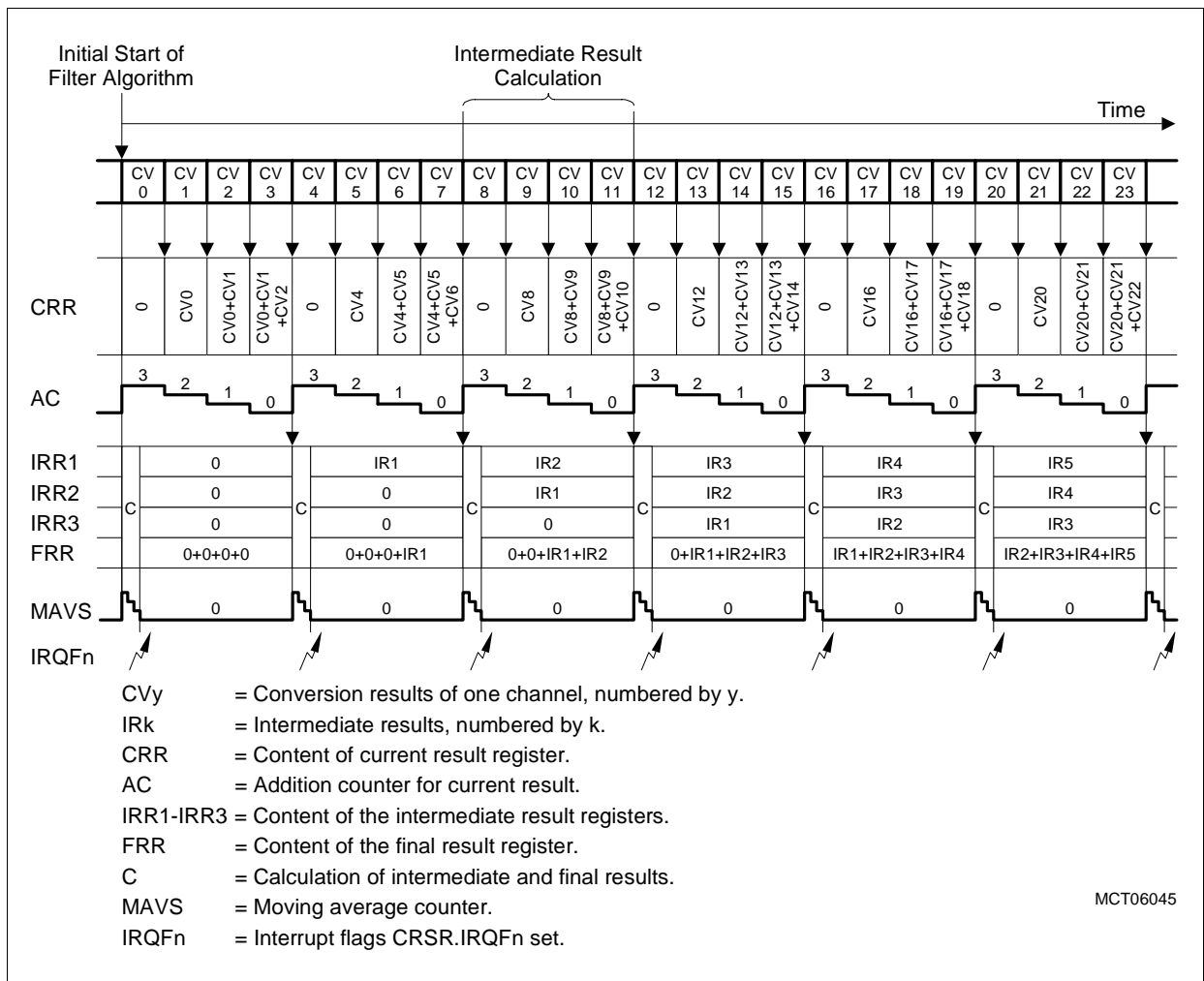


Figure 26-8 Filter Block Algorithm

Fast Analog-to-Digital Converter (FADC)

Intermediate Result Calculation

When the continuous conversion of channel x has been started, the Current Result Register CRR_n (n = filter block number) is set to 0. Each following conversion result is added to the content of CRR_n until a programmable number of conversion results have been summed up in CRR_n . At that point, the calculation of the next final result value is triggered. In the example of [Figure 26-8](#), one intermediate result is built by the sum of four conversion results ($FCR_n.ADDL = 011_B$). The state of the current result calculation cycle is indicated by the addition counter AC which is located in register CRR_n .

Final Result Calculation

The calculation of a final result is always started after an intermediate result calculation cycle has been finished. This new intermediate result (stored in CRR_n) plus the contents of the intermediate registers are added according to:

- Filter 0: $FRR_0 := CRR_0 + IRR_{10} + IRR_{20} + IRR_{30}$
- Filter 1: $FRR_1 := CRR_1 + IRR_{11}$

Therefore, FRR_n always contains the sum of maximum four (for filter block 0) or two (for filter block 1) intermediate results. Bit field $FCR_n.MAVL$ determines the number of intermediate results that are used for the final result calculation.

After a final result calculation, the old contents of IRR_{20} are transferred into IRR_{30} and IRR_{10} are transferred into IRR_{20} (for filter block 0). In filter block 1, the contents of CRR_1 are stored in IRR_{11} . The old contents of IRR_{30} or IRR_{11} are lost. Finally, the value of CRR_n is transferred as intermediate result into the intermediate result register IRR_{1n} . Thereafter, the CRR_n register is set to 0 for the next current result calculation cycle.

Each update of a result register FRR_n with a new final result value generates a filter block n service request. During a final result calculation phase (phase C in [Figure 26-8](#)), the contents of the FRR_n registers change. Therefore, it is recommended to read a final result from the FRR_n registers immediately (for example, by a DMA operation) after a corresponding interrupt request flag $CRSR.IRQFn$ has been set.

Filter Control Parameters

The Filter n Control Register FCR_n contains several parameters that determine the operation of filter n :

- **$FCR_n.INSEL$:**
This parameter selects the channel(s) whose conversion results are used as filter n input. A single channel, all four channels, or the final result of filter 0 (for filter 1 concatenation) can be selected. The filter n operation can also be stopped via $FCR_n.INSEL$.
- **$FCR_n.ADDL$:**
This parameter determines how many conversion results are added in filter n to calculate one intermediate result. $FCR_n.ADDL$ is loaded into $CRR_n.AC$ at the start of

Fast Analog-to-Digital Converter (FADC)

a filter sequence and after each intermediate result storage. CRRn.AC is decremented after each conversion result addition and indicates how many additions are still to be executed until the next intermediate result is stored. Values for one up to eight conversions can be selected for FCRn.ADDL.

- **FCRn.MAVL:**

This parameter determines the number of intermediate result registers that are used for a final result calculation. FCRn.MAVL is loaded into CRRn.MAVS at the start of a filter sequence. Values for none up to three intermediate result registers can be selected for FCRn.MAVL.

Initial State

In order to start a filter algorithm for filter block n, the following actions must be executed:

1. Program bit field FCRn.ADDL with FCRn.INSEL = 000_B (filter disabled)
2. Select filter n input source and operation by writing FCRn.INSEL with the appropriate value
3. Reset filter block n by writing GCR.RSTFn = 1
4. Start a continuous conversion

Fast Analog-to-Digital Converter (FADC)

26.1.6.3 Filter Concatenation

As shown in Figure 26-6, it is possible to concatenate the two filters by using the result value of filter 0 as input value for filter 1. Figure 26-9 shows an example for the filter concatenation.

Filter 0 operates with following parameters:

- Intermediate results are calculated from four filter input values (conversion results)
- A final result is calculated by four intermediate results

Filter 1 operates with following parameters:

- Concatenation enabled (FCR1.INSEL = 010_B)
- Intermediate results are calculated from three filter 1 input values
- A final result is calculated by two intermediate results

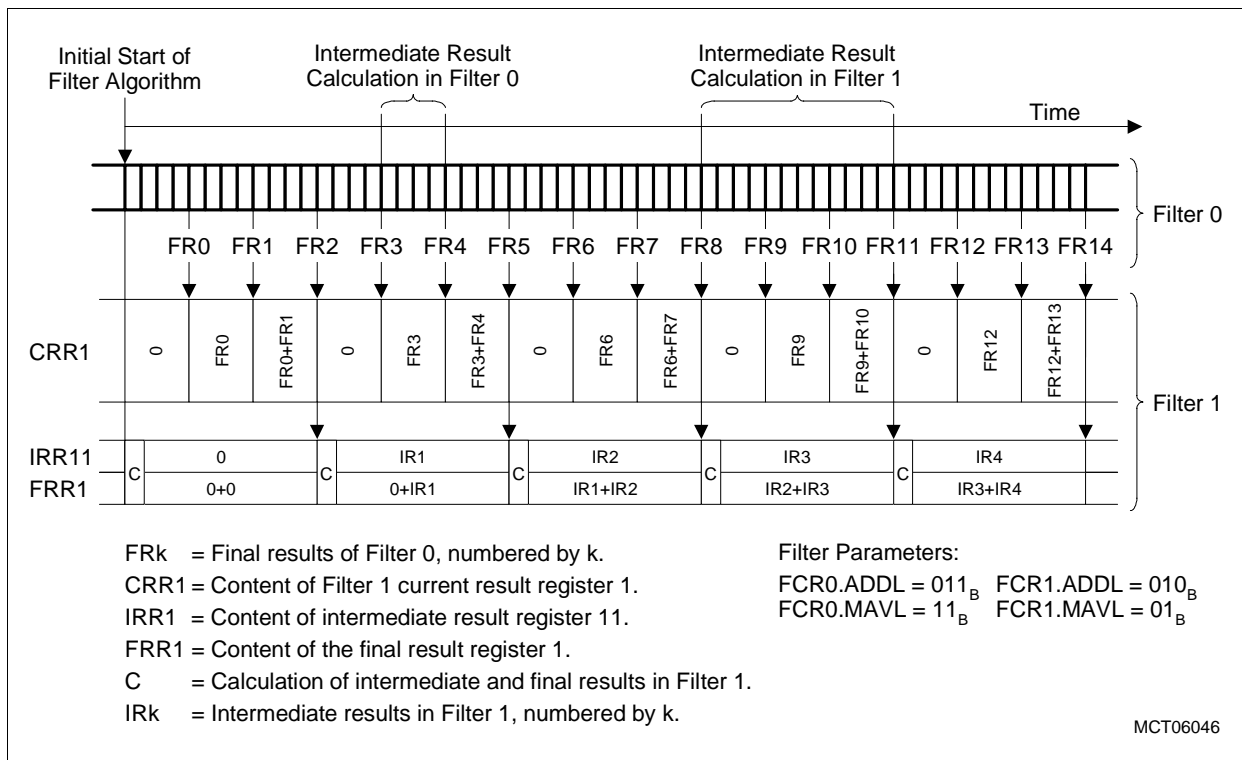


Figure 26-9 Filter Concatenation Example

Filter 0 operates with the same parameters as in the example shown in Figure 26-8. A Filter 1 input value is given by each final result of filter 0. Filter 1 can store only one intermediate result in its IRR11 register.

Fast Analog-to-Digital Converter (FADC)

26.1.6.4 Width of Result Registers

The additions executed in filter 0 and filter 1 together with the possible maximum values of the filter parameters determine the width of the current, intermediate, and final result registers.

An FADC conversion result always has a width of ten bits. A maximum of eight conversion results can be added in the current result registers for an intermediate result. This results in $10 + 3 = 13$ bit width for the current result register and for the intermediate result registers. The final result is built by the addition of at maximum four current/intermediate result registers. Therefore, the width of the final result register is $13 + 2 = 15$. These values for the result widths are valid for filter 0.

When both filters are concatenated, the width of FRR0 (15-bit) must be taken into account when the filter 1 result register width is calculated. Filter 1 also allows eight input values (filter 0 final results) to be added in its current result register for an intermediate result. This results in $15 + 3 = 18$ bits width for the current result register and for the intermediate result registers of filter 1. The final result is built in filter 1 by the addition of maximum two current/intermediate result values. Therefore, the width of the filter 1 final result register is $18 + 1 = 19$. Caused by the design, the two missing intermediate result registers of filter 1 must be considered and the width of the filter 1 final result register is additionally increased to $19 + 1 = 20$.

Table 26-5 Data Width of Result Registers

Register Long Name	Register Short Name	Result Width
Filter 0 Current Result Register	CRR0	13-bit
Filter 0 Intermediate Result Register 1	IRR10	
Filter 0 Intermediate Result Register 2	IRR20	
Filter 0 Intermediate Result Register 3	IRR30	
Filter 0 Final Result Register	FRR0	15-bit
Filter 1 Current Result Register	CRR1	18-bit
Filter 1 Intermediate Result Register 1	IRR11	
Filter 1 Final Result Register	FRR1	20-bit

26.1.7 Neighbor Channel Trigger

The neighbor channel trigger feature allows the concatenation of channel conversions. This means that the start of a conversion for one channel can generate multiple channel trigger requests for the other three channels. A channel conversion request flag of a neighbor channel becomes only set by a neighbor channel trigger request if the gating condition (gating mode selection output at high level in [Figure 26-4](#)) in the corresponding neighbor channel is valid.

All neighbor channel trigger enable bits EN_{xy} are all located in the Neighbor Channel Trigger Register NCTR. Index “x” indicates the number of the channel that starts a neighbor channel trigger. Index “y” is the number of the neighbor channel to be triggered.

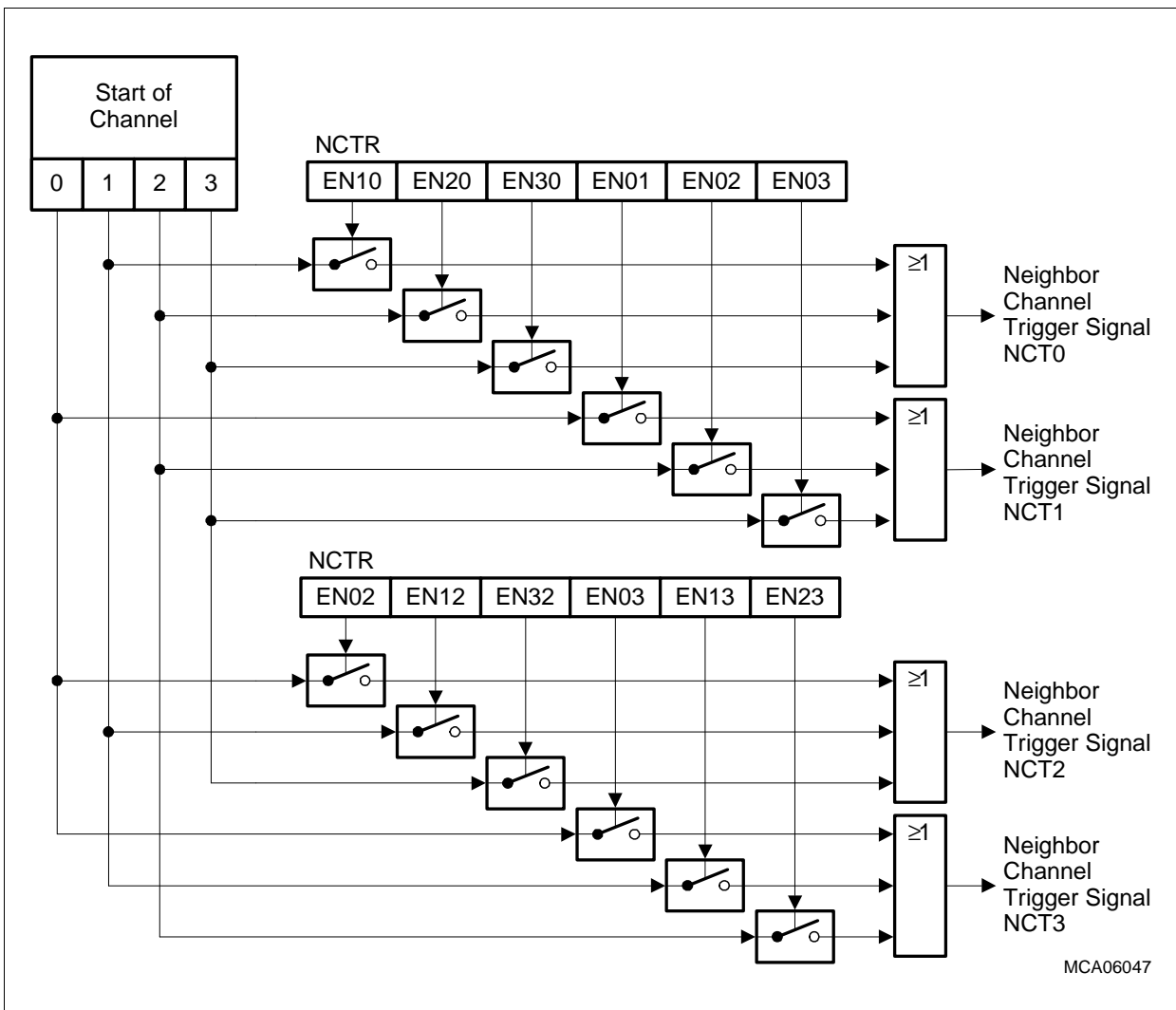


Figure 26-10 Neighbor Channel Trigger

Fast Analog-to-Digital Converter (FADC)

26.1.8 Offset and Gain Calibration

The offset and gain calibration is used to minimize the error of the FADC conversion results independently for each channel. The output of each channel amplifier can be adjusted to deliver a minimum offset value for zero input voltage difference. The channel which is calibrated is selected by GCR.CALCH. The calibration mode is selected by GCR.CALMODE. During the calibration process of a channel, the other channels can be used in normal conversion mode without restrictions.

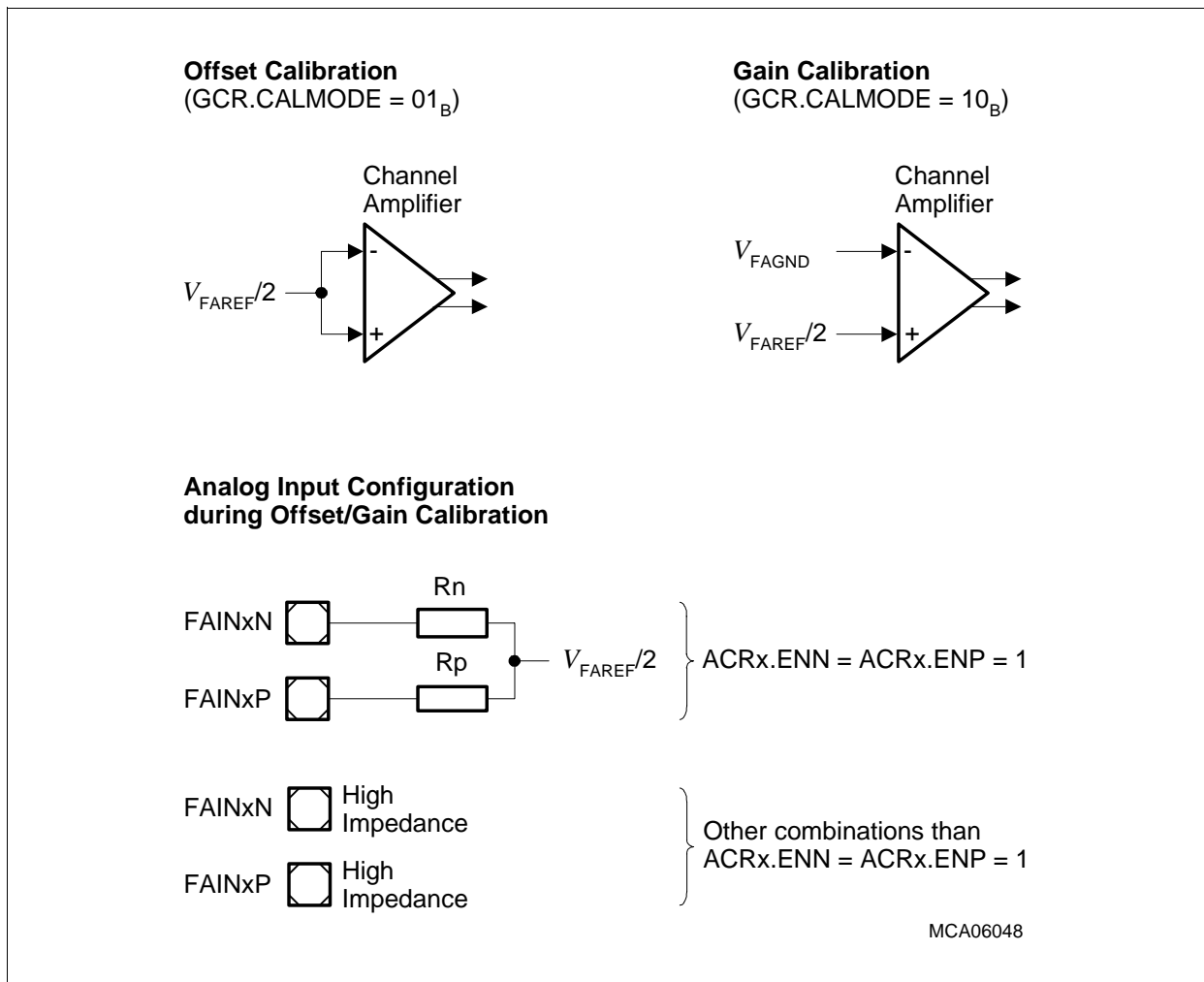


Figure 26-11 Analog Input and Channel Amplifier Configuration at Calibration

Figure 26-11 shows the channel amplifier configuration as well as the analog input pin configurations that are selected during offset and gain calibration. Note that in the calibration modes, the impedance of the analog inputs depends on the settings of the ENN and ENP bits of the corresponding Channel x Analog Control Register ACRx.

26.1.8.1 Offset Calibration

When offset calibration is selected ($GCR.CALMODE = 01_B$), the channel amplifier inputs of the selected channel are both connected to $V_{FAREF}/2$. After enabling a channel amplifier for offset calibration, a delay of minimum $5 \mu s$ must be respected before starting a conversion for this channel. The conversion result must be compared by software with the tolerated offset value (a conversion result with zero offset is equal to 512). If the conversion result exceeds the tolerated range, bit field $ACRx.CALOFF$ (with x specifying the calibrated channel) can be adjusted and a new conversion can be started. The calibration process is finished when the conversion result is in the tolerated offset range. After switching back to normal mode for channel x , a delay of minimum $5 \mu s$ must be respected before starting a new conversion for this channel.

After an offset calibration has been finished for a channel, its gain can also be calibrated.

26.1.8.2 Gain Calibration

When gain calibration is selected ($GCR.CALMODE = 10_B$), the gain can be adjusted by software to cover the complete input voltage range. The procedure is similar to the one for the offset calibration.

Fast Analog-to-Digital Converter (FADC)

26.1.9 Interrupt Generation

A flexible service request control structure is implemented in the FADC. The FADC provides four channel conversion request sources and two filter block request sources, that can be programmed to generate one of four service request output signals SR[3:0]. The service request compressor also makes it possible to assign more than one service request source to one service request output.

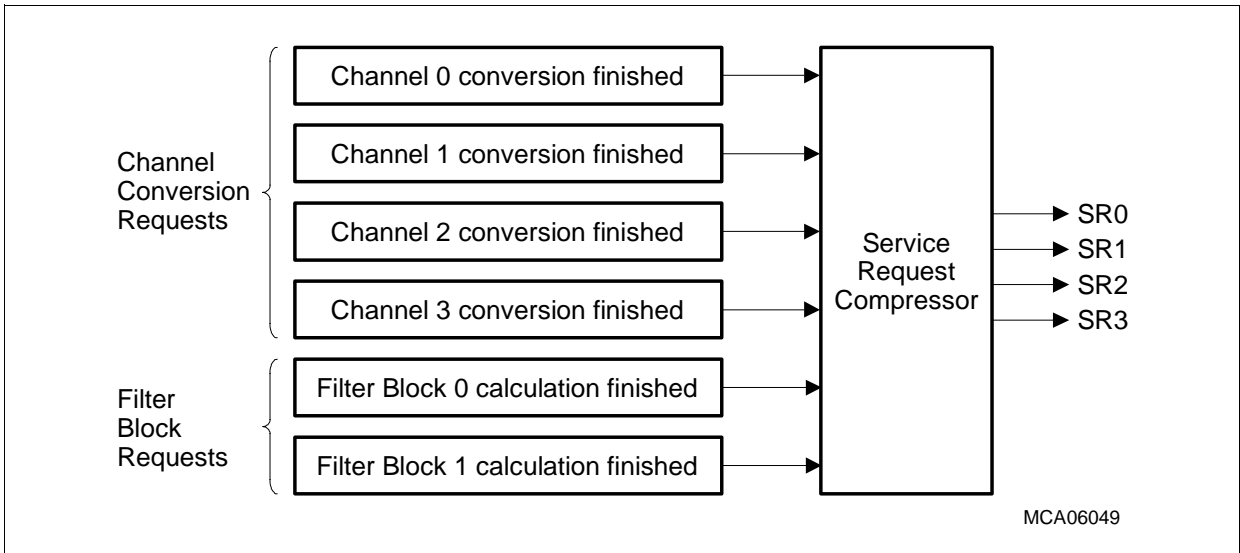


Figure 26-12 Service Request Configuration

All six service requests are controlled by an identical control logic. This control logic as shown in Figure 26-13 provides the following functionality:

- Service Request Flag
- Set/Clear Request Flag Control Bits
- Service Request Enable Bit
- Service Request Node Pointer

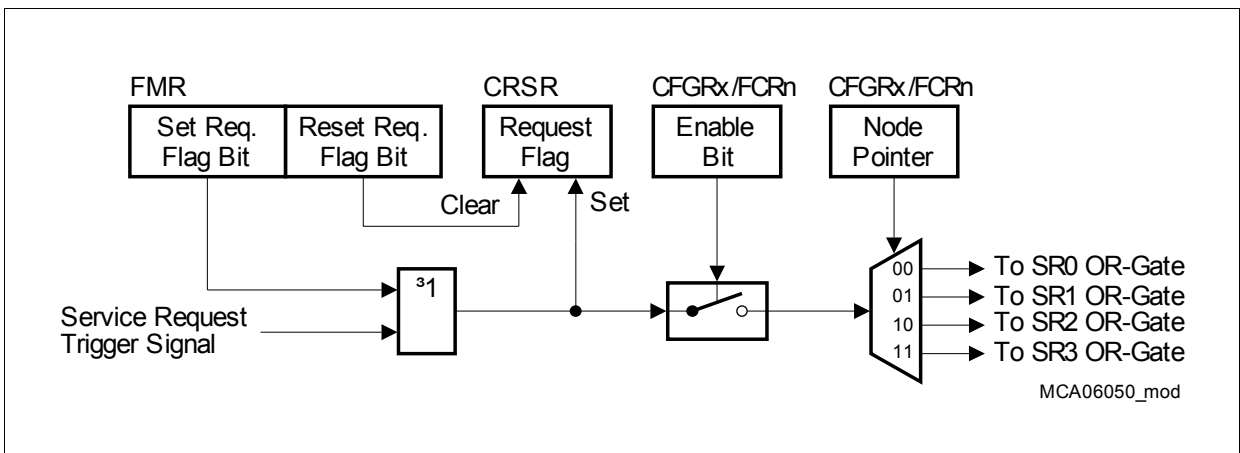


Figure 26-13 Service Request Control Logic

Fast Analog-to-Digital Converter (FADC)

The request flag is always set by hardware when the corresponding request event occurs. It can also be set or cleared by software when writing a 1 to the corresponding set/clear request flag bit in the Flag Modification Register FMR. Finally, a service request event is directed to one of the four service request output lines SR[3:0] when the corresponding service request enable bit IEN is set. The service request node pointer determines which of the service request output lines SR[3:0] becomes activated.

Table 26-6 lists the six service request sources of the FADC Module with its related control and status flags/bits.

Table 26-6 Service Request Control/Status Bits/Flags

Service Request Source	Request Flag	Enable Bit	Set Request Bit / Clear Request Bit	Service Request Node Pointer
Channel x Conversion Request (x = 0-3)	CRSR.IRQx	CFGRx.IEN	FMR.SIRQx / FMR.RIRQx	CFGRx.INP
Filter Block n Request (n = 0, 1)	CRSR.IRQFn	FCRn.IEN	FMR.SIRQFn / FMR.RIRQFn	FCRn.INP

In the service request compressor logic shown in **Figure 26-14**, the inputs of one SRx OR-gate are connected to all demultiplexer outputs with identical INP node pointer value. Therefore, one service request event can only be assigned to one of the four service request outputs but one service request output can be used by multiple service request events.

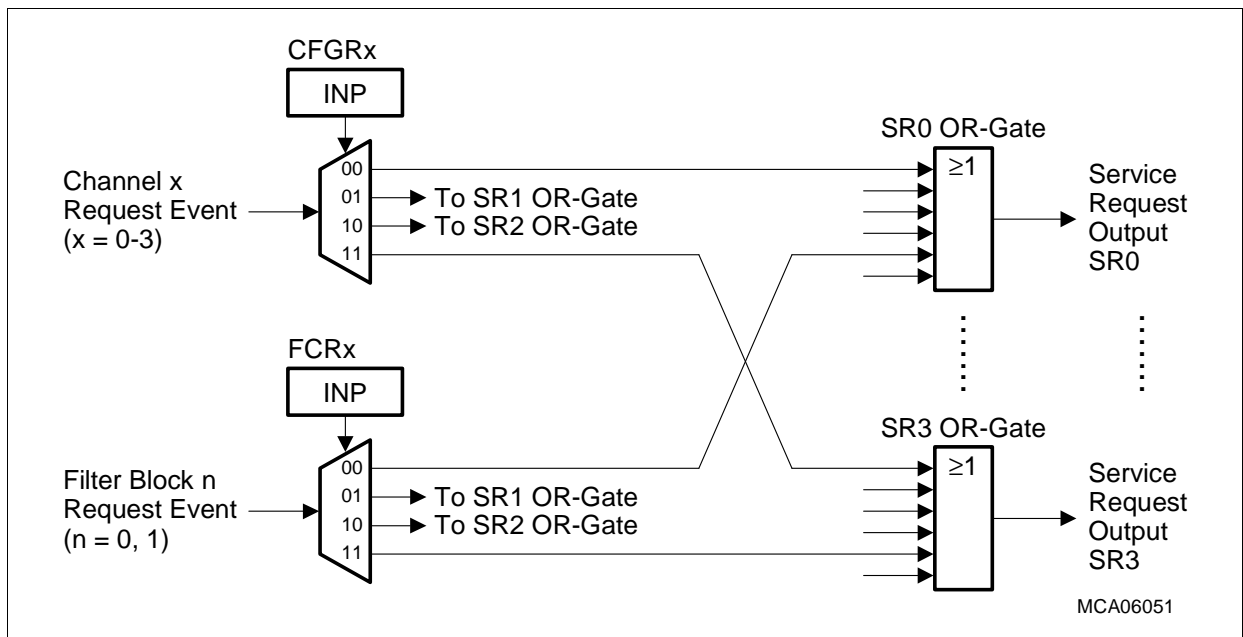


Figure 26-14 Service Request Compressor Logic

Fast Analog-to-Digital Converter (FADC)

Depending on the implementation of the FADC Module in a specific microcontroller, the service request output signals SR[3:0] can either be connected to an interrupt node (controlled by a service request control register) or can be used as DMA request input of a DMA controller unit. The TC1796 specific request output connections are described in [Figure 26-16](#) and on [Page 26-65](#).

Fast Analog-to-Digital Converter (FADC)

26.2 FADC Kernel Registers

This section describes the kernel registers of the FADC module. The complete and detailed address maps of the FADC module is described in [Table 18-28](#) on [Page 18-84](#) of the TC1796 User's Manual System Units part (Volume 1).

All FADC kernel register names described in this section are referenced in other parts of the TC1796 User's Manual by the module name prefix "FADC_".

FADC Kernel Register Overview

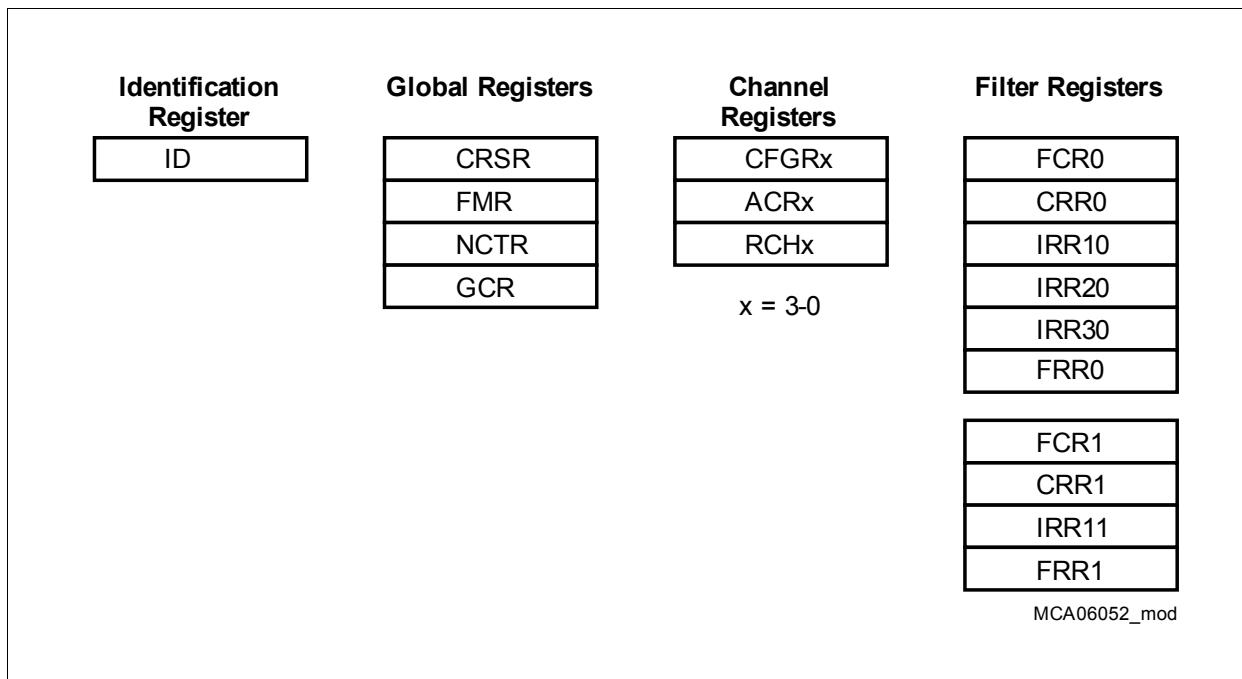


Figure 26-15 FADC Kernel Registers

In the TC1796, the registers of the FADC module are located in the following address range.

Table 26-7 Registers Address Space - FADC Module

Module	Base Address	End Address	Note
FADC	F010 0300 _H	F010 03FF _H	—

Fast Analog-to-Digital Converter (FADC)

Table 26-8 Registers Overview - FADC Kernel Registers

Register Short Name	Register Long Name	Offset Address	Description see
Global Registers			
ID	Module Identification Register	08 _H	Page 26-29
CRSR	Conversion Request Status Register	10 _H	Page 26-30
FMR	Flag Modification Register	14 _H	Page 26-32
NCTR	Neighbor Channel Trigger Register	18 _H	Page 26-34
GCR	Global Control Register	1C _H	Page 26-37
Channel Registers			
CFGR _x	Channel x Configuration Register (x = 0-1)	20 _H + (x × 4)	Page 26-41
ACR _x	Channel x Analog Control Register (x = 0-1)	30 _H + (x × 4)	Page 26-45
RCH _x	Channel x Conversion Result Register (x = 0-1)	40 _H + (x × 4)	Page 26-47
Filter Registers			
FCR _n	Filter n Control Register (n = 0-1)	60 _H + (n × 20 _H)	Page 26-48
CRR _n	Filter n Current Result Register (n = 0-1)	64 _H + (n × 20 _H)	Page 26-51 Page 26-52
IRR1 _n	Filter n Intermediate Result Register 1 (n = 0-1)	68 _H + (n × 20 _H)	Page 26-54
IRR2 _n	Filter n Intermediate Result Register 2 (n = 0)	6C _H + (n × 20 _H)	
IRR3 _n	Filter n Intermediate Result Register 3 (n = 0)	70 _H + (n × 20 _H)	
FRR _n	Filter n Final Result Register (n = 0-1)	74 _H + (n × 20 _H)	Page 26-55

Fast Analog-to-Digital Converter (FADC)

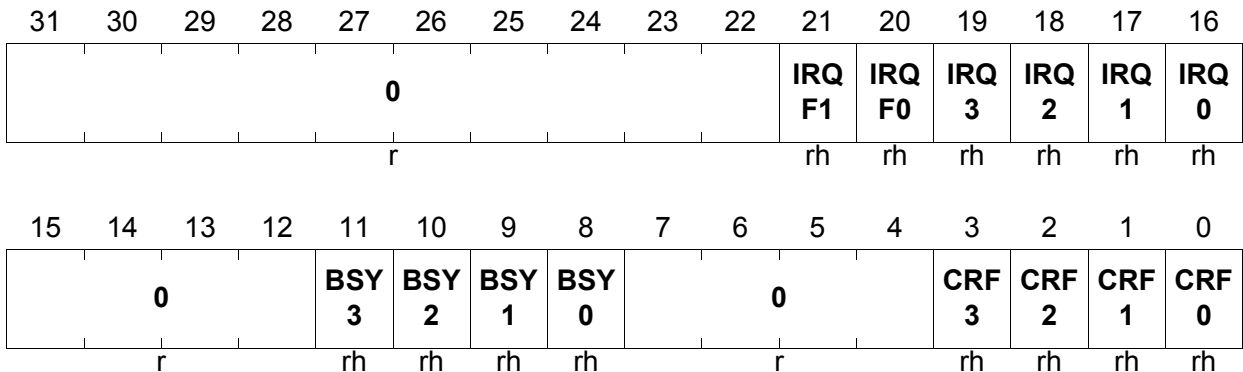
26.2.2 Global Registers

The Conversion Request Status Register CRSR contains the flags for monitoring the state of pending conversions and the interrupt request flags.

CRSR

Conversion Request Status Register (10_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CRFx (x = 0-3)	x	rh	<p>Conversion Request Flag</p> <p>This bit monitors whether a conversion request is pending for channel x. CRFx is set by hardware when a trigger event is detected while the gating condition delivers 1. CRFx is automatically cleared by hardware when a conversion of the channel x is started.</p> <p>0_B A conversion of channel x has not been requested.</p> <p>1_B A conversion of channel x has been requested.</p> <p>Bits CRFx can be set/cleared by software via bits FMR.RCRFx and FMR.SCRFx (see Page 26-32).</p> <p>If a set and a clear condition for CRFx occur simultaneously (generated by hardware and/or software), the clear condition always wins.</p>
BSYx (x = 0-3)	8 + x	rh	<p>Busy Flag</p> <p>This bit indicates if a conversion is currently running for channel x.</p> <p>0_B A conversion is not running.</p> <p>1_B A conversion is running.</p>

Fast Analog-to-Digital Converter (FADC)

Field	Bits	Type	Description
IRQx (x = 0-3)	16 + x	rh	<p>Interrupt Request Flag</p> <p>This bit indicates that a conversion of channel x has been finished since it has been cleared by software. Interrupt requests can also be generated while IRQx is still set. An interrupt can only be generated when $CFGRx.IEN = 1$.</p> <p>0_B A conversion has not been finished. 1_B A conversion has been finished.</p> <p>Bits IRQx can be set/cleared by software via bits FMR.SIRQx and FMR.RIRQx (see Page 26-32).</p>
IRQFn (n = 0-1)	20 + n	rh	<p>Interrupt Request Flag for Filter n</p> <p>This bit indicates that a filter sequence of filter n has been finished (new final result is available) since it has been cleared by software. Interrupt requests can also be generated while IRQ is still set. An interrupt can only be generated when $FCRn.IEN = 1$.</p> <p>0_B A filter sequence has not been finished. 1_B A filter sequence has been finished.</p> <p>Bits IRQFn can be set/cleared by software via bits FMR.SIRQFn and FMR.RIRQFn (see Page 26-32).</p>
0	[7:4], [15:12], [31:22]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Fast Analog-to-Digital Converter (FADC)

The bits of the Flag Modification Register FMR allow the flags of the conversion request status register to be set/cleared by software.

FMR

Flag Modification Register

(14_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	S	S	S	S	S	S				R	R	R	R	R	R	
	IRQ	IRQ	IRQ	IRQ	IRQ	IRQ		0		IRQ	IRQ	IRQ	IRQ	IRQ	IRQ	
	F1	F0	3	2	1	0				F1	F0	3	2	1	0	
	r	w	w	w	w	w	w	r		w	w	w	w	w	w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
				S	S	S	S						R	R	R	R
				CRF	CRF	CRF	CRF		0				CRF	CRF	CRF	CRF
				3	2	1	0						3	2	1	0
	r			w	w	w	w		r				w	w	w	w

Field	Bits	Type	Description
RCRFx (x = 0-3)	x	w	Reset Conversion Request Flag This bit allows bit CRSR.CRFx to be cleared by software. 0 _B No operation 1 _B Bit CRSR.CRFx is cleared (also if bit SCRFx is written simultaneously with 1)
SCRFx (x = 0-3)	8 + x	w	Set Conversion Request Flag This bit allows bit CRSR.CRFx to be set by software. 0 _B No operation 1 _B Bit CRSR.CRFx is set
RIRQx (x = 0-3)	16 + x	w	Reset Interrupt Request Flag This bit allows bit CRSR.IRQx to be cleared by software. 0 _B No operation 1 _B Bit CRSR.IRQx is cleared
RIRQFn (n = 0-1)	20 + n	w	Reset Interrupt Request Flag for Filter n This bit allows bit CRSR.IRQFn to be cleared by software. 0 _B No operation 1 _B Bit CRSR.IRQFn is cleared

Fast Analog-to-Digital Converter (FADC)

Field	Bits	Type	Description
SIRQx (x = 0-3)	24 + x	w	Set Interrupt Request Flag This bit allows bit CRSR.IRQx to be set by software (if bit RIRQx is written simultaneously with 0). 0 _B No operation 1 _B Bit CRSR.IRQx is set. An interrupt is generated if CFGRx.IEN = 1.
SIRQFn (n = 0-1)	28 + n	w	Set Interrupt Request Flag for Filter n This bit allows bit CRSR.IRQFn to be set by software (if bit RIRQFn is written simultaneously with 0). 0 _B No operation 1 _B Bit CRSR.IRQFn is set. An interrupt is generated if FCRn.IEN = 1.
0	[7:4], [15:12], [23:22], [31:30]	r	Reserved Read as 0; should be written with 0.

Fast Analog-to-Digital Converter (FADC)

The Neighbor Channel Trigger Register NCTR contains the enable bits for the neighbor channel trigger signal (NCTx) generation (see [Page 26-21](#)).

NCTR

Neighbor Channel Trigger Register (18_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				EN 32	EN 31	EN 30	0				EN 23	0	EN 21	EN 20	
r				rw	rw	rw	r				rw	r	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				EN 13	EN 12	0	EN 10	0				EN 03	EN 02	EN 01	0
r				rw	rw	r	rw	r				rw	rw	rw	r

Field	Bits	Type	Description
EN01	1	rw	Enable Neighbor Channel Trigger 01 This bit enables the neighbor channel trigger for channel 1 when a conversion of channel 0 is started. 0 _B No action. 1 _B A trigger will be generated.
EN02	2	rw	Enable Neighbor Channel Trigger 02 This bit enables the neighbor channel trigger for channel 2 when a conversion of channel 0 is started. 0 _B No action. 1 _B A trigger will be generated.
EN03	3	rw	Enable Neighbor Channel Trigger 03 This bit enables the neighbor channel trigger for channel 3 when a conversion of channel 0 is started. 0 _B No action. 1 _B A trigger will be generated.
EN10	8	rw	Enable Neighbor Channel Trigger 10 This bit enables the neighbor channel trigger for channel 0 when a conversion of channel 1 is started. 0 _B No action. 1 _B A trigger will be generated.

Fast Analog-to-Digital Converter (FADC)

Field	Bits	Type	Description
EN12	10	rw	<p>Enable Neighbor Channel Trigger 12 This bit enables the neighbor channel trigger for channel 2 when a conversion of channel 1 is started.</p> <p>0_B No action. 1_B A trigger will be generated.</p>
EN13	11	rw	<p>Enable Neighbor Channel Trigger 13 This bit enables the neighbor channel trigger for channel 3 when a conversion of channel 1 is started.</p> <p>0_B No action. 1_B A trigger will be generated.</p>
EN20	16	rw	<p>Enable Neighbor Channel Trigger 20 This bit enables the neighbor channel trigger for channel 0 when a conversion of channel 2 is started.</p> <p>0_B No action. 1_B A trigger will be generated.</p>
EN21	17	rw	<p>Enable Neighbor Channel Trigger 21 This bit enables the neighbor channel trigger for channel 1 when a conversion of channel 2 is started.</p> <p>0_B No action. 1_B A trigger will be generated.</p>
EN23	19	rw	<p>Enable Neighbor Channel Trigger 23 This bit enables the neighbor channel trigger for channel 3 when a conversion of channel 2 is started.</p> <p>0_B No action. 1_B A trigger will be generated.</p>
EN30	24	rw	<p>Enable Neighbor Channel Trigger 30 This bit enables the neighbor channel trigger for channel 0 when a conversion of channel 3 is started.</p> <p>0_B No action. 1_B A trigger will be generated.</p>
EN31	25	rw	<p>Enable Neighbor Channel Trigger 31 This bit enables the neighbor channel trigger for channel 1 when a conversion of channel 3 is started.</p> <p>0_B No action. 1_B A trigger will be generated.</p>

Fast Analog-to-Digital Converter (FADC)

Field	Bits	Type	Description
EN32	26	rw	Enable Neighbor Channel Trigger 32 This bit enables the neighbor channel trigger for channel 2 when a conversion of channel 3 is started. 0 _B No action. 1 _B A trigger will be generated.
0	0, [7:4], 9, [15:12], 18, [23:20], [31:27]	r	Reserved Read as 0; should be written with 0.

Note: The hardware does not check whether the enable bits are set in such a way as to describe a loop of conversion requests (e.g. 0 triggers 2, 2 triggers 3 and 3 triggers 0, etc.). It is in the responsibility of the user to set these bits in an appropriate way.

Fast Analog-to-Digital Converter (FADC)

The Global Control Register GCR contains bits used to clear the Channel Timers, the filters and to control global FADC settings.

GCR

Global Control Register

(1C_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
0				CALCH	CALMODE	0			AN ON	MUX TM	RES WEN	DPA EN	CRPRIO			
r				rw		rw			r	rw	rw	rw	rw	rwh		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0				RST F1	RST F0	RCD	0				RCT 3	RCT 2	RCT 1	RCT 0		
r				w	w	w	r				w	w	w	w		

Field	Bits	Type	Description
RCT_x (x = 0-3)	x	w	Reload Channel Timer 0 _B Channel x Timer will not be changed. 1 _B Channel x Timer will be loaded with its reload value.
RCD	8	w	Reset Common Divider 0 _B The common divider will not be changed. 1 _B The common divider will be cleared.
RSTF_n (n = 0-1)	9 + n	w	Reset Filter n 0 _B The contents of filter n will not be changed. 1 _B The contents of filter n will be cleared. The values of the bits in the filter registers will be cleared, except bit field CRR _n .AC which is loaded with the value of FCR _n .ADDL.

Fast Analog-to-Digital Converter (FADC)

Field	Bits	Type	Description
CRPRIO	[17:16]	rwh	<p>Conversion Request Priority</p> <p>This bit field determines the priority of the conversion requests if more than one channel is requested. If the dynamic priority assignment is enabled, the priority is automatically changed as a function of the gating inputs. The priority of the channels is:</p> <p>00_B Channel 0 before channel 1 before channel 2 before channel 3</p> <p>01_B Channel 1 before channel 2 before channel 3 before channel 0</p> <p>10_B Channel 2 before channel 3 before channel 0 before channel 1</p> <p>11_B Channel 3 before channel 0 before channel 1 before channel 2</p>
DPAEN	18	rw	<p>Dynamic Priority Assignment Enable</p> <p>If the dynamic priority assignment is enabled, the priority bit field CRPRIO is automatically changed as a function of the gating input signals. In this case, the channel that is active while the other three channels are not active gets the highest priority.</p> <p>0_B The dynamic priority assignment is disabled.</p> <p>1_B The dynamic priority assignment is enabled.</p>
RESWEN	19	rw	<p>Result Write Enable</p> <p>This bit enables a write action to the result registers RCHx (x = 0-3) of the FADC.</p> <p>0_B Write accesses to the result registers are not taken into account. The written data is discarded.</p> <p>1_B Write accesses to the result registers are taken into account. The former value of the written result register is overwritten by the write data. If a filter is sensitive to the written result register, the written value is taken as new filter input value.</p>

Fast Analog-to-Digital Converter (FADC)

Field	Bits	Type	Description
MUXTM	20	rw	<p>Multiplexer Test Mode The input multiplexer to select a channel for the conversion can be tested by opening all multiplexer inputs. In multiplexer test mode, the channel amplifiers are not connected to the common amplifier.</p> <p>0_B The multiplexer test mode is disabled. 1_B The multiplexer test mode is enabled.</p>
ANON	21	rw	<p>Analog Part ON This bit enables the analog part of the FADC. This bit must be set to convert the analog input signal to a digital value.</p> <p>0_B The complete analog part is in power-down mode, the amplifiers and comparators are switched off. Conversions are not possible. (default) 1_B The analog part is enabled.</p>
CALMODE	[25:24]	rw	<p>Calibration Mode This bit field enables the calibration for offset and gain for the channel selected by CALCH.</p> <p>00_B No calibration process is running. All channels are in normal mode (default after reset). 01_B The analog channel selected by CALCH is in offset calibration mode. The other channels are in normal mode. 10_B The analog channel selected by CALCH is in gain calibration mode. The other channels are in normal mode. 11_B Reserved</p>

Fast Analog-to-Digital Converter (FADC)

Field	Bits	Type	Description
CALCH	[27:26]	rw	<p>Calibration Channel</p> <p>This bit field selects the channel for the calibration process determined by CALMODE. The setting of CALCH is only taken into account while a calibration process is running.</p> <p>00_B The analog input channel 0 is selected for a calibration process.</p> <p>01_B The analog input channel 1 is selected for a calibration process.</p> <p>10_B The analog input channel 2 is selected for a calibration process.</p> <p>11_B The analog input channel 3 is selected for a calibration process.</p>
0	[7:4], [15:11], [23:22], [31:28]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Fast Analog-to-Digital Converter (FADC)

26.2.3 Channel Registers

The Channel x Configuration Register CFGRx contains the bits for the selection of the trigger source, the gating source, and other channel settings of channel x.

CFGRx (x = 0-3)

Channel x Configuration Register ($20_H + x \cdot 4_H$)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IEN		0		INP		0		CTREL							
rw		r		rw		r		rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		CTF		CTM		TM		GM		TSEL		GSEL			
r		rw		rw		rw		rw		rw		rw			

Field	Bits	Type	Description
GSEL	[2:0]	rw	Gating Selection This bit field selects the gating source input signal for channel x. 000 _B Gating source input signal GS0 selected 001 _B Gating source input signal GS1 selected 010 _B Gating source input signal GS2 selected 011 _B Gating source input signal GS3 selected 100 _B Gating source input signal GS4 selected 101 _B Gating source input signal GS5 selected 110 _B Gating source input signal GS6 selected 111 _B Gating source input signal GS7 selected
TSEL	[5:3]	rw	Trigger Selection This bit field selects the trigger source input signal for channel x. 000 _B Trigger source input signal TS0 selected 001 _B Trigger source input signal TS1 selected 010 _B Trigger source input signal TS2 selected 011 _B Trigger source input signal TS3 selected 100 _B Trigger source input signal TS4 selected 101 _B Trigger source input signal TS5 selected 110 _B Trigger source input signal TS6 selected 111 _B Trigger source input signal TS7 selected

Fast Analog-to-Digital Converter (FADC)

Field	Bits	Type	Description
GM	[7:6]	rw	<p>Gating Mode</p> <p>This bit field determines the functionality of the gating (enable) signal. It determines whether and under which condition the generation of conversion requests by trigger signals is possible.</p> <p>00_B Conversion requests are disabled and the Channel Timer is stopped. CRFx never becomes set (by hardware).</p> <p>01_B Conversion requests and the Channel Timer are always enabled. CRFx becomes set by hardware with each active trigger signal.</p> <p>10_B Conversion requests and the Channel Timer are enabled only if the gating source input (as selected by CFGRx.GSEL) is at high level.</p> <p>11_B Conversion requests and the Channel Timer are enabled only if the gating source input (as selected by CFGRx.GSEL) is at low level.</p>
TM	[9:8]	rw	<p>Trigger Mode</p> <p>This bit field enables the triggering and determines the edge of the trigger source input signal that generates a conversion trigger signal.</p> <p>00_B No conversion trigger signals are generated. Edge detection unit is switched off.</p> <p>01_B A conversion request is generated (if gating enabled) on a rising edge of a trigger source input (as selected by CFGRx.TSEL).</p> <p>10_B A conversion request is generated (if gating enabled) on a falling edge of a trigger source input (as selected by CFGRx.TSEL).</p> <p>11_B A conversion request is generated (if gating enabled) on both, rising and falling, edges of a trigger source input (as selected by CFGRx.TSEL).</p>

Fast Analog-to-Digital Converter (FADC)

Field	Bits	Type	Description
CTM	[11:10]	rw	<p>Channel Timer Mode</p> <p>This bit determines the operating mode of channel x timer.</p> <p>00_B Channel x timer is switched off. 01_B Channel timer is permanently running. 10_B Channel timer is running only if ECHTIMx = 1. 11_B Reserved</p> <p>A Channel Timer trigger event is generated each time the channel x timer value reaches 00_H. While the Channel Timer is not running (CTM = 00_B or signal ECHTIMx = 0), the Channel Timer is loaded with 04_H.</p>
CTF	[14:12]	rw	<p>Channel Timer Frequency</p> <p>This bit field controls the channel x timer input clock f_{CT} (enable control and frequency selection).</p> <p>000_B f_{CTx} is disabled. 001_B f_{CTx} is enabled with frequency f_{FADC}. 010_B f_{CTx} is enabled with frequency $f_{FADC} / 4$. 011_B f_{CTx} is enabled with frequency $f_{FADC} / 16$. 100_B f_{CTx} is enabled with frequency $f_{FADC} / 64$. 101_B f_{CTx} is enabled with frequency $f_{FADC} / 256$. 110_B f_{CTx} is enabled with frequency $f_{FADC} / 1024$. 111_B Reserved; do not use this combination.</p>
CTREL	[23:16]	rw	<p>Channel Timer Reload Value</p> <p>This bit field determines the reload value of the Channel Timer CHTIMx. The divider factor for the channel x timer is given by:</p> <p>CTREL + 1, if $2 \leq f_{CLC} / f_{CTx}$ CTREL + 2, if $f_{CLC} / f_{CTx} = 1$</p> <p>If CTREL = 0, no trigger event is generated. See also description on Page 26-11.</p>
INP	[29:28]	rw	<p>Interrupt Node Pointer</p> <p>This bit field selects which service request output line will be activated when a conversion of channel x is finished while CFGRx.IEN is set.</p> <p>00_B Service request output SR0 is selected. 01_B Service request output SR1 is selected. 10_B Service request output SR2 is selected. 11_B Service request output SR3 is selected.</p>

Fast Analog-to-Digital Converter (FADC)

Field	Bits	Type	Description
IEN	31	rw	<p>Interrupt Enable</p> <p>This bit enables the generation of a service request when a conversion of channel x is finished.</p> <p>0_B Channel x conversion service request generation is disabled.</p> <p>1_B Channel x conversion service request generation is enabled.</p>
0	15, [27:24], 30	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Fast Analog-to-Digital Converter (FADC)

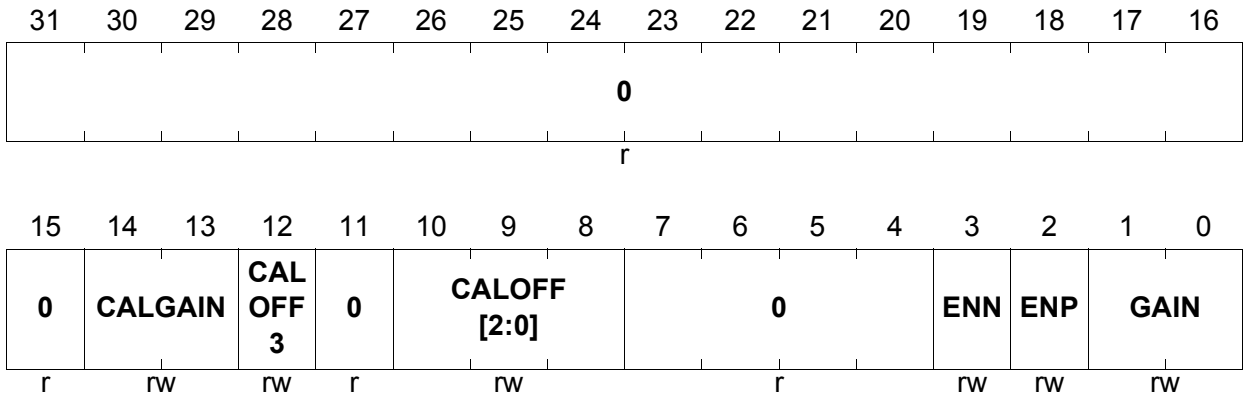
The Channel x Analog Control Register ACRx contains the bits that control the analog input stage.

ACRx (x = 0-3)

Channel x Analog Control Register

(30_H+x*4_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
GAIN	[1:0]	rw	<p>Amplifier Gain</p> <p>This bit field determines the amplifier gain for channel x.</p> <p>00_B The selected amplifier gain is 1. 01_B The selected amplifier gain is 2. 10_B The selected amplifier gain is 4. 11_B The selected amplifier gain is 8.</p>
ENP	2	rw	<p>Enable Positive Input</p> <p>This bit enables the voltage measurement on the FAINxP analog input.</p> <p>0_B Analog input FAINxP is high-impedance. The upper half of the measuring range is not available. 1_B Analog input FAINxP line is connected to the channel amplifier. The upper half of the measuring range is available.</p>

Fast Analog-to-Digital Converter (FADC)

Field	Bits	Type	Description
ENN	3	rw	<p>Enable Negative Input This bit enables the voltage measurement on the FAINxN analog input.</p> <p>0_B Analog input FAINxN is high-impedance. The lower half of the measuring range is not available.</p> <p>1_B Analog input FAINxN line is connected to the channel amplifier. The lower half of the measuring range is available.</p>
CALOFF[2:0]	[10:8]	rw	<p>Calibrate Offset This bit field determines the value applied for the offset calibration for channel x. The calibrate offset value is composed by the most significant bit CALOFF3 and bit field CALOFF[2:0], resulting in a 4-bit bit field CALOFF[3:0].</p>
CALOFF3	12	rw	
CALGAIN	[14:13]	rw	<p>Calibrate Gain This bit field determines the value applied for the gain calibration for channel x.</p>
0	[7:4], 11, [31:15]	r	<p>Reserved Read as 0; should be written with 0.</p>

Note: If ENN = 0 and ENP = 0, the channel amplifier is in power-down mode. The conversion result is not valid in this case.

Fast Analog-to-Digital Converter (FADC)

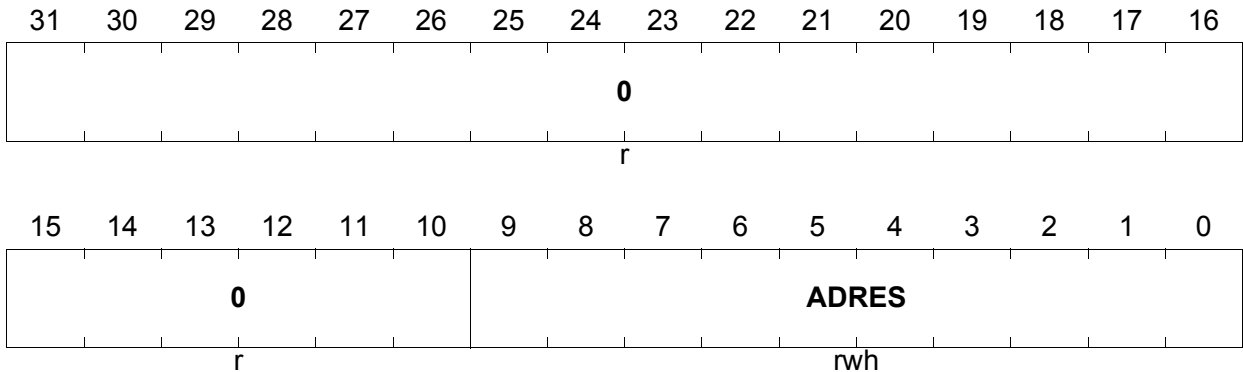
The Channel x Conversion Result Register RCHx contains the conversion result ADRES of channel x.

RCHx (x = 0-3)

Channel x Conversion Result Register

($40_H + x * 4_H$)

Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
ADRES	[9:0]	rwh	AD Conversion Result This bit field contains the conversion result of channel x. ADRES can only be overwritten by software if GCR.RESWEN = 1 (see also register GCR description on Page 26-38).
0	[31:10]	r	Reserved Read as 0; should be written with 0.

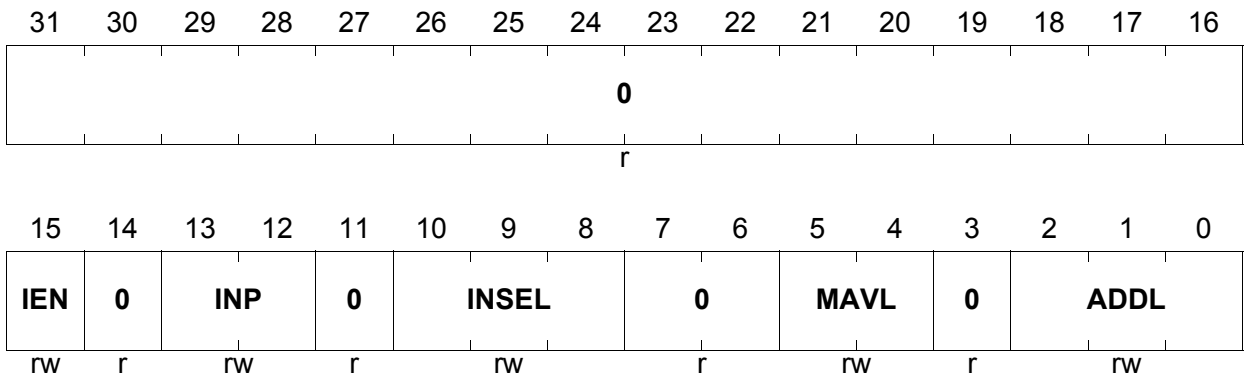
Fast Analog-to-Digital Converter (FADC)

26.2.4 Filter Registers

Filter blocks 0 and 1 are controlled by bits in the Filter n Control Registers FCRn.

FCRn (n = 0-1)

Filter n Control Register (60_H+n*20_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
ADDL	[2:0]	rw	<p>Addition Length</p> <p>This bit field determines the number of filter input values that are added to obtain one intermediate result.</p> <p>000_B Each filter input value is considered as intermediate result.</p> <p>001_B 2 filter input values are added up.</p> <p>010_B 3 filter input values are added up.</p> <p>011_B 4 filter input values are added up.</p> <p>100_B 5 filter input values are added up.</p> <p>101_B 6 filter input values are added up.</p> <p>110_B 7 filter input values are added up.</p> <p>111_B 8 filter input values are added up.</p>

Fast Analog-to-Digital Converter (FADC)

Field	Bits	Type	Description
MAVL	[5:4]	rw	<p>Moving Average Length</p> <p>This bit field determines the number of intermediate results that are added up for a final result.</p> <p>00_B No moving average is selected. Each intermediate result is considered as final result value: $FRRn.FR = CRRn.CR$</p> <p>01_B A moving average of 2 values is selected. The final result is calculated by 2 values: $FRRn.FR = CRRn.CR + IRR1n.IR$</p> <p>10_B A moving average of 3 values is selected. The final result is calculated by 3 values: $FRRn.FR = CRRn.CR + IRR1n.IR + IRR2n.IR$</p> <p>11_B A moving average of 4 values is selected. The final result is calculated by 4 values: $FRRn.FR = CRRn.CR + IRR1n.IR + IRR2n.IR + IRR3n.IR$</p> <p>Bit combinations 10_B and 11_B are not available in filter block 1.</p>
INSEL	[10:8]	rw	<p>Input Selection</p> <p>This bit field enables the filter block and determines which input value is taken for filter block n.</p> <p>000_B The filter block is disabled. Intermediate and final sum calculations are not executed. The filter register values are not changed (except by a filter block reset).</p> <p>001_B Any conversion result of each channel is taken as new filter input value.</p> <p>010_B Filter block 0: filter is stopped (as 000_B). Filter block 1: filter input value is the output value (final result) of filter block 0.</p> <p>011_B Reserved</p> <p>100_B Channel 0 conversion result is taken as filter input value.</p> <p>101_B Channel 1 conversion result is taken as filter input value.</p> <p>110_B Channel 2 conversion result is taken as filter input value.</p> <p>111_B Channel 3 conversion result is taken as filter input value.</p>

Fast Analog-to-Digital Converter (FADC)

Field	Bits	Type	Description
INP	[13:12]	rw	Interrupt Node Pointer This bit field selects which service request output line will be activated when a final result of filter block n is available while bit IEN is set. 00 _B Service request output SR0 selected 01 _B Service request output SR1 selected 10 _B Service request output SR2 selected 11 _B Service request output SR3 selected
IEN	15	rw	Interrupt Enable This bit enables the generation of a new final result service request of filter block n. 0 _B Service request generation disabled 1 _B Service request generation enabled
0	3, [7:6], 11, 14, [31:16]	r	Reserved Read as 0.

Note: If the length of the moving average is reduced, the execution time (number of clock cycles) is also reduced. In any case, the filter calculation and execution time is much smaller than a conversion time.

Fast Analog-to-Digital Converter (FADC)

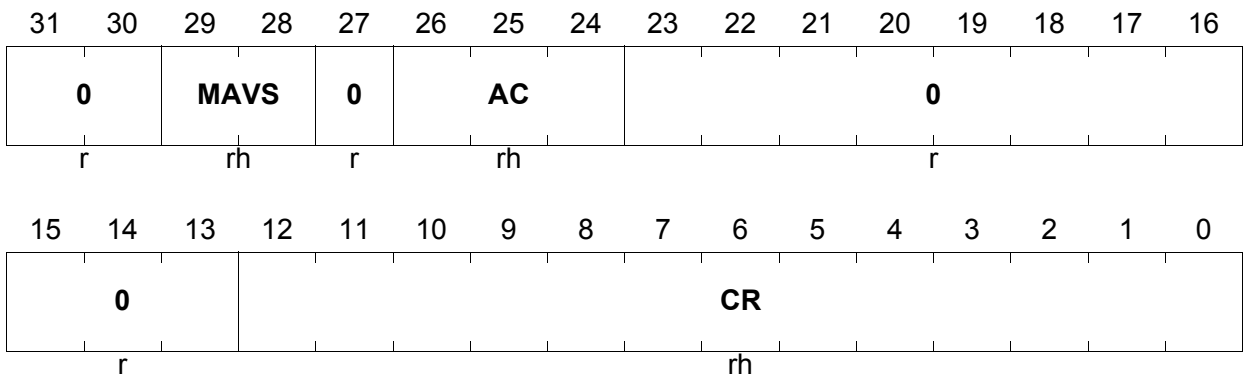
The Current Result Registers CRRn store the current result of filter n. Further, status information of filter block n can be read from CRRn.

CRR0

Filter 0 Current Result Register

(64_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CR	[12:0]	rh	Current Result This bit field ([12:0] for filter 0, [17:0] for filter 1) contains the right-aligned current result value of filter 0. CR is cleared when writing GCR.RSTFn = 1.
AC	[26:24]	rh	Addition Count This bit field indicates the number of additions of filter input values with remain to be executed before the next intermediate result register transfer occurs. AC is loaded with the value of FCRn.ADDL for a new addition sequence. CR is cleared when writing GCR.RSTFn = 1.

Fast Analog-to-Digital Converter (FADC)

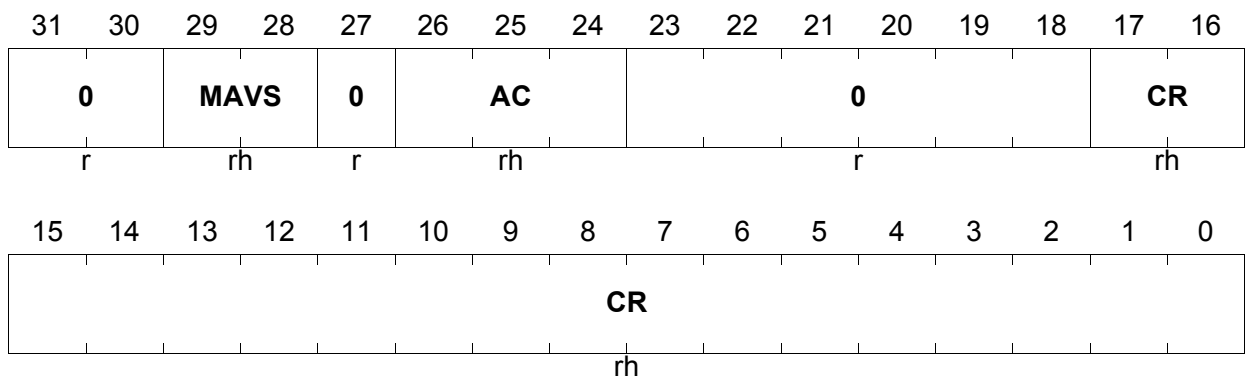
Field	Bits	Type	Description
MAVS	[29:28]	rh	<p>Moving Average State</p> <p>This bit field indicates how many intermediate register transfers remain to be executed for the generation of the next final result.</p> <p>MAVS = 0 indicates the end of a filter calculation operation. Since the filter calculation is executed very fast in comparison to a conversion, MAVS > 0 can be interpreted only as a kind of calculation busy flag. Therefore, it is recommended to read a valid filter result from register FRRn only when the corresponding interrupt request flag CRSR.IRQFn is set.</p> <p>MAVS is cleared when writing GCR.RSTFn = 1.</p>
0	[23:13], 27, [31:30]	r	<p>Reserved</p> <p>Read as 0.</p>

CRR1

Filter 1 Current Result Register

(84_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CR	[17:0]	rh	<p>Current Result</p> <p>This bit field ([12:0] for filter 0, [17:0] for filter 1) contains the right-aligned current result value of filter 0. CR is cleared when writing GCR.RSTFn = 1.</p>

Fast Analog-to-Digital Converter (FADC)

Field	Bits	Type	Description
AC	[26:24]	rh	<p>Addition Count</p> <p>This bit field indicates the number of additions of filter input values with remain to be executed before the next intermediate result register transfer occurs. AC is loaded with the value of FCRn.ADDL for a new addition sequence.</p> <p>CR is cleared when writing GCR.RSTFn = 1.</p>
MAVS	[29:28]	rh	<p>Moving Average State</p> <p>This bit field indicates how many intermediate register transfers remain to be executed for the generation of the next final result.</p> <p>MAVS = 0 indicates the end of a filter calculation operation. Since the filter calculation is executed very fast in comparison to a conversion, MAVS > 0 can be interpreted only as a kind of calculation busy flag. Therefore, it is recommended to read a valid filter result from register FRRn only when the corresponding interrupt request flag CRSR.IRQFn is set.</p> <p>MAVS is cleared when writing GCR.RSTFn = 1.</p>
0	[23:18], 27, [31:30]	r	<p>Reserved</p> <p>Read as 0.</p>

Fast Analog-to-Digital Converter (FADC)

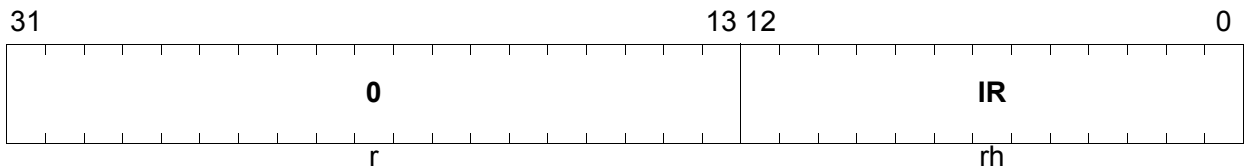
The Intermediate Result Registers IRR_{xn} hold the intermediate results *x* of filter *n*.

IRR_{x0} (x = 1-3)

Filter 0 Intermediate Result Register *x*

(64_H+*x**4_H)

Reset Value: 0000 0000_H



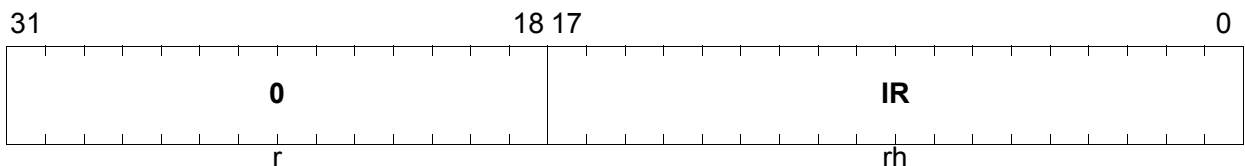
Field	Bits	Type	Description
IR	[12:0]	rh	Intermediate Result This bit field contains the right-aligned intermediate result of filter 0. IR is cleared when writing GCR.RSTFn = 1.
0	[31:13]	r	Reserved Read as 0.

IRR11

Filter 1 Intermediate Result Register 1

(88_H)

Reset Value: 0000 0000_H



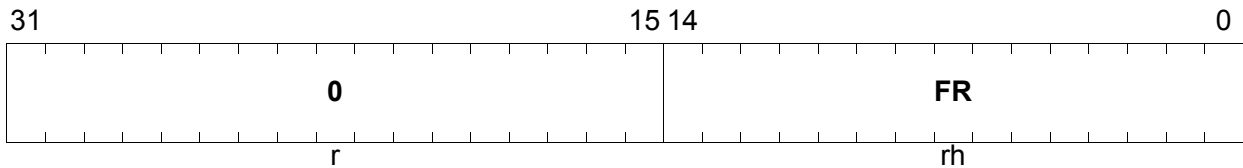
Field	Bits	Type	Description
IR	[17:0]	rh	Intermediate Result This bit field contains the right-aligned intermediate result of filter 1. IR is cleared when writing GCR.RSTFn = 1.
0	[31:18]	r	Reserved Read as 0.

Fast Analog-to-Digital Converter (FADC)

The Final Result Registers for filters 0 and 1 hold the final result of the filter operations.

FRR0

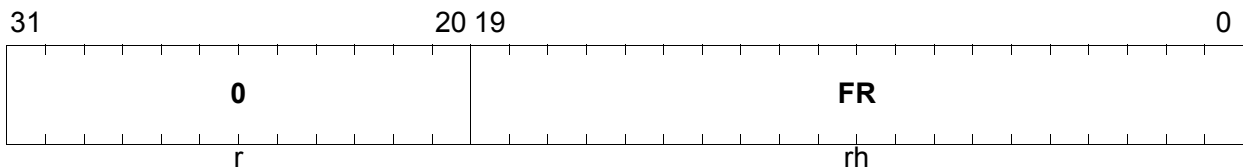
Filter 0 Final Result Register (74_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
FR	[14:0]	rh	Final Result This bit field contains the right-aligned 15-bit final result of filter 0. FR is cleared when writing GCR.RSTFn = 1.
0	[31:15]	r	Reserved Read as 0.

FRR1

Filter 1 Final Result Register (94_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
FR	[19:0]	rh	Final Result This bit field contains the right-aligned 20-bit final result of filter 1. FR is cleared when writing GCR.RSTFn = 1.
0	[31:20]	r	Reserved Read as 0.

Note: During a final result calculation phase (phase C in [Figure 26-8](#) on [Page 26-16](#)), the contents of the FRRn registers change. Therefore, it is recommended to read a final result from the FRRn registers immediately (for example, by a DMA operation) after a corresponding interrupt request flag CRSR.IRQFn has been set.

Fast Analog-to-Digital Converter (FADC)

26.3 Implementation of FADC

This section describes the FADC module related external functions, such as port connections, interrupt and service request control, DMA connections, address decoding and clock control.

26.3.1 Interfaces of the FADC Module

Figure 26-16 shows the TC1796 specific implementation details and interconnections of the FADC module.

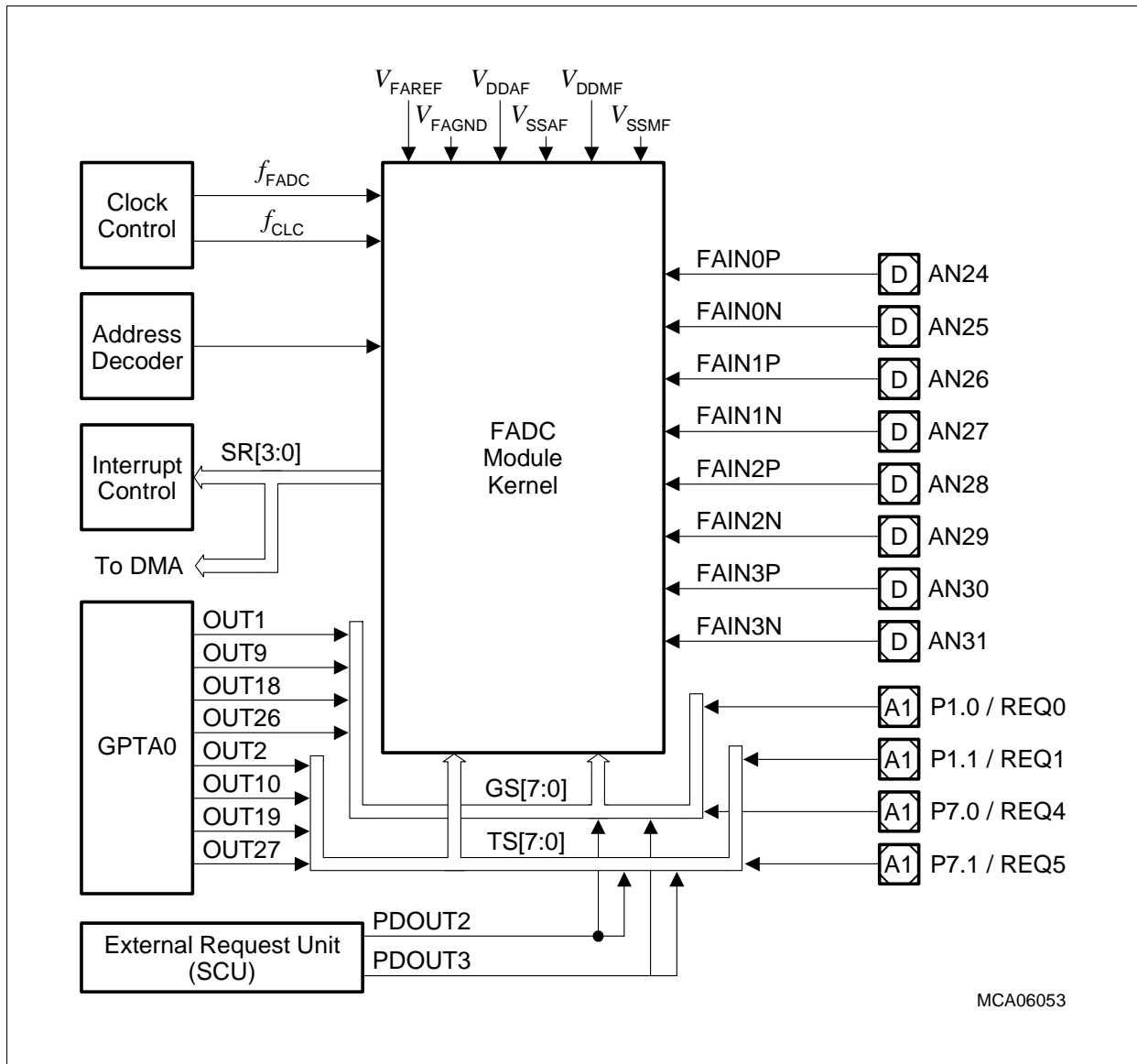


Figure 26-16 FADC Module Implementation and Interconnections

Fast Analog-to-Digital Converter (FADC)

The FADC module is supplied with a clock control, address decoding, and interrupt control logic. Each of the four service request outputs is connected to an interrupt node. In parallel, the four service request outputs are wired to request inputs of the DMA controller. The 2×8 gating and trigger source input lines are connected to four request input pins, eight outputs of the GPTA0 module, and two outputs of the external request unit. The eight analog inputs are wired to analog inputs of the TC1796.

The analog input lines of the FADC can also be used as input lines for ADC0 and ADC1. If both, ADC0/ADC1 and FADC, are connected to the same analog input pin at the same time, the input impedances of both analog inputs must be taken into account in order to minimize signal distortions and measurement errors.

26.3.2 FADC Module Related External Registers

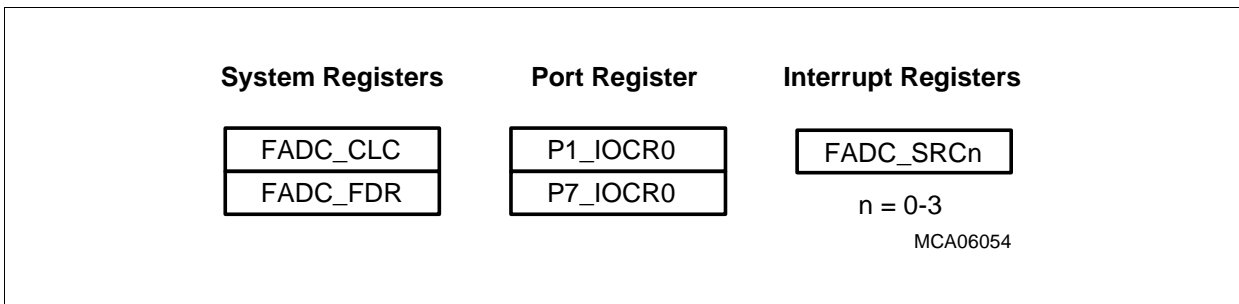


Figure 26-17 FADC Implementation-specific Special Function Registers

Fast Analog-to-Digital Converter (FADC)

26.3.3 Clock Control

The FADC module is provided with two clock signals:

- f_{CLC}
This is the module clock that is used inside the FADC kernel for control purposes such as clocking of control logic and register operations. The frequency of f_{CLC} is always identical to the system clock frequency f_{SYS} . The clock control register FADC_CLC makes it possible to enable/disable f_{CLC} under certain conditions.
- f_{FADC}
This clock is the module clock that is used in the FADC as the clock for the Channel Timer and the analog part. It also controls the conversion timing (see [Page 26-8](#)). The fractional divider registers FADC_FDR controls the frequency of f_{FADC} and allows it to be enabled/disabled independently of f_{CLC} .

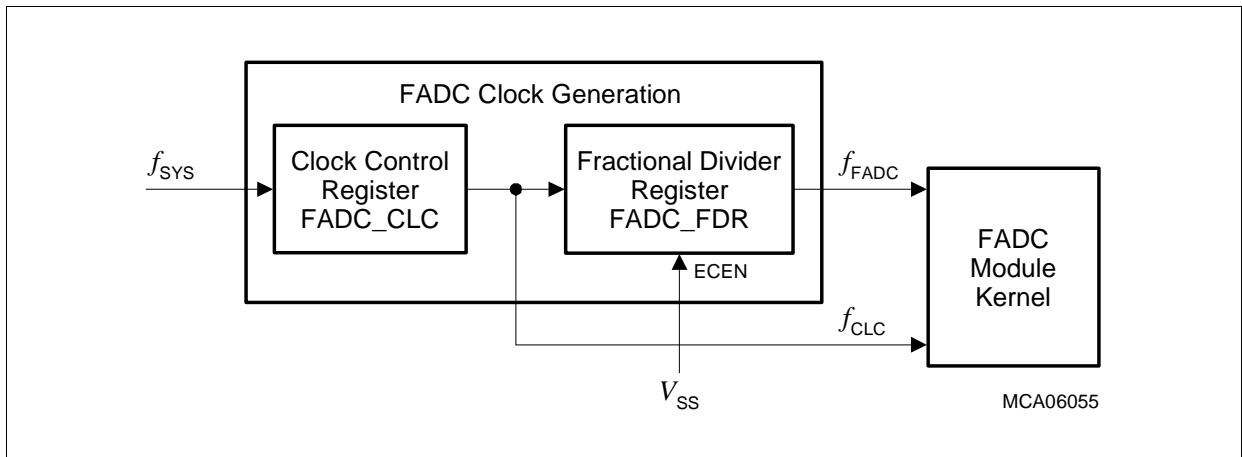


Figure 26-18 FADC Clock Generation

The following formulas define the frequency of f_{FADC} :

$$f_{FADC} = f_{SYS} \times \frac{1}{n} \text{ with } n = 1024 - \text{FDR.STEP} \quad (26.3)$$

$$f_{FADC} = f_{SYS} \times \frac{n}{1024} \text{ with } n = 0-1023 \quad (26.4)$$

Equation (26.3) is valid for FADC_FDR.DM = 01_B (normal divider mode).

Equation (26.4) is valid for FADC_FDR.DM = 10_B (fractional divider mode).

Fast Analog-to-Digital Converter (FADC)

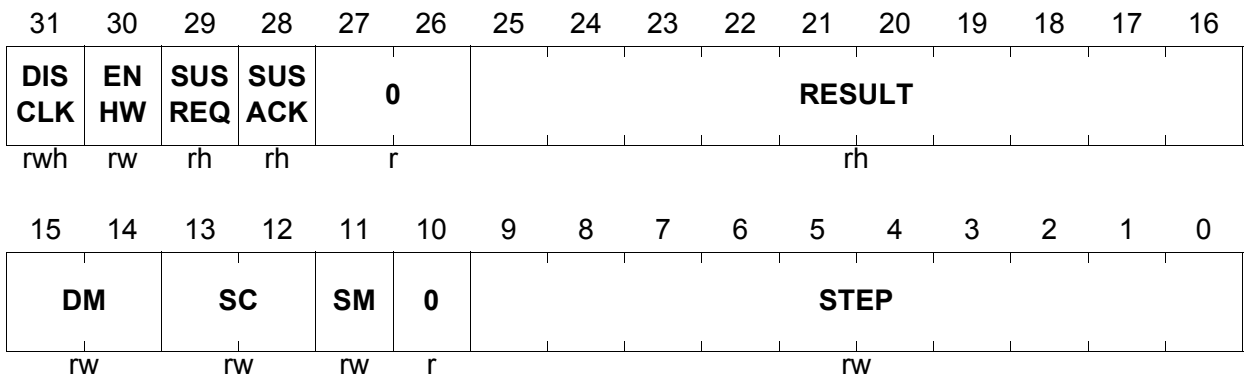
The Fractional Divider Register allows the programmer to control the clock rate of the module clock f_{FADC} .

FDR

Fractional Divider Register

(0C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
STEP	[9:0]	rw	Step Value Reload or addition value for RESULT.
SM	11	rw	Suspend Mode SM selects between granted or immediate suspend mode.
SC	[13:12]	rw	Suspend Control This bit field determines the behavior of the fractional divider in suspend mode.
DM	[15:14]	rw	Divider Mode This bit field selects normal divider mode, fractional divider mode, and off-state.
RESULT	[25:16]	rh	Result Value Bit field for the addition result.
SUSACK	28	rh	Suspend Mode Acknowledge Indicates state of SPNDACK signal.
SUSREQ	29	rh	Suspend Mode Request Indicates state of SPND signal.
ENHW	30	rw	Enable Hardware Clock Control Controls operation of ECEN input and DISCLK bit. Should be always written with 0.

Fast Analog-to-Digital Converter (FADC)

Field	Bits	Type	Description
DISCLK	31	rwh	Disable Clock Hardware controlled disable for f_{FADC} signal.
0	10, [27:26]	rw	Reserved Read as 0; should be written with 0.

Note: Additional details on the fractional divider register functionality are described in section “[Fractional Divider Operation](#)” on Page 3-29 of the TC1796 User’s Manual System Units part (Volume 1).

26.3.4 Port Control

The external request inputs REQ0, REQ1, REQ4, and REQ5 used by the FADC module are controlled in the port logic. If these input lines are used as FADC input, the input function selection (IOCR registers) must be checked.

Input Function Selection

The port input/output control registers contain the bit fields that select the FADC’s digital input driver characteristics such as pull-up/pull-down device connection capabilities. The request input lines used by the FADC module are controlled by the port input/output control registers P1_IOCR0 and P7_IOCR0.

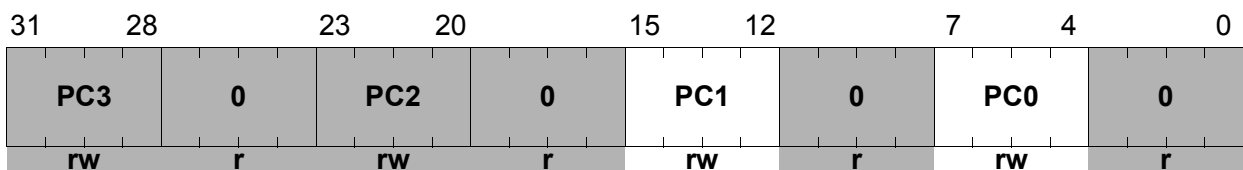
After reset, for the external request inputs the input function with a pull-up device is selected. For reference, the P1_IOCR0 and P7_IOCR0 functionality is shown on the next page in respect to the FADC external request inputs.

P1_IOCR0

Port 7 Input/Output Control Register 0

(10_H)

Reset Value: 2020 2020_H



Field	Bits	Type	Description
PC0, PC1	[7:4], [15:12]	rw	Port Output Control for Port 1 Pins 0-3¹⁾ These bit fields determine the output port functionality: Port input/output control for P1.0/REQ0 Port input/output control for P1.1/REQ1

Fast Analog-to-Digital Converter (FADC)

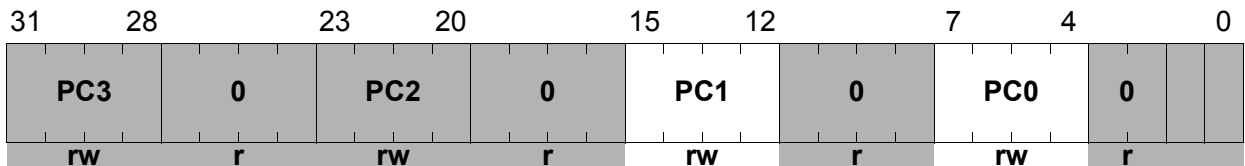
1) Coding of bit field see [Table 26-8](#). Shaded bits and bit fields are “don’t care” for FADC I/O port control.

P7_IOCRO

Port 7 Input/Output Control Register 0

(10_H)

Reset Value: 2020 2020_H



Field	Bits	Type	Description
PC0, PC1	[7:4],	rw	Port Output Control for Port 7 Pins 0-3¹⁾ These bit fields determine the output port functionality: Port input/output control for P7.0/REQ4 Port input/output control for P7.1/REQ5
	[15:12]		

1) Coding of bit field see [Table 26-8](#). Shaded bits and bit fields are “don’t care” for FADC I/O port control.

Table 26-9 PCx Coding for FADC Inputs

PCx[3:0]	I/O	Selected Input Function
0X00 _B	Input	No pull device connected
0X01 _B		Pull-down device connected
0X10 _B ¹⁾		Pull-up device connected
0X11 _B		No pull device connected

1) Default after reset.

Fast Analog-to-Digital Converter (FADC)

26.3.5 Interrupt Control

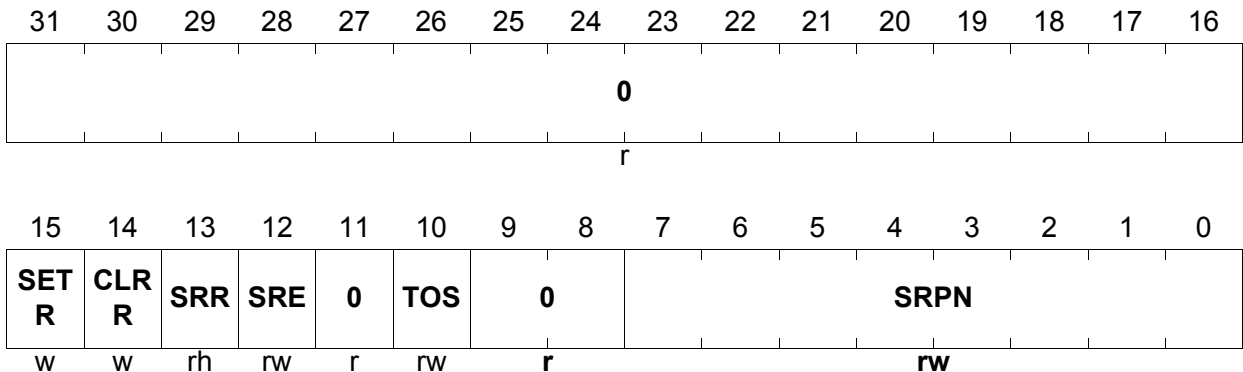
Each of the four FADC interrupts is controlled by a service request control register.

FADC_SRC_m (m = 0-3)

FADC Service Request Control Register m

(FC_H-m*4_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SRPN	[7:0]	rw	Service Request Priority Number
TOS	10	rw	Type of Service Control
SRE	12	rw	Service Request Enable
SRR	13	rh	Service Request Flag
CLRR	14	w	Request Clear Bit
SETR	15	w	Request Set Bit
0	[9:8], 11, [31:16]	r	Reserved Read as 0; should be written with 0.

Note: Additional details on service request nodes and the service request control registers are described on [Page 14-3](#) of the TC1796 User's Manual System Units part (Volume 1).

26.3.6 On-Chip Connections

This section describes the on-chip connections of the FADC module.

26.3.6.1 Analog Input Lines

The analog inputs FAINxP/FAINxN (x = 0-3) are connected to the TC1796 analog input pins AN[31:24] as shown on [Page 26-56](#).

26.3.6.2 Trigger/Gating Source Input Connections

Gating source inputs GS[7:0] and trigger source inputs TS[7:0] of the FADC module are connected to GPTA0 module outputs, of GPIO port lines, and external request unit outputs according to [Table 26-10](#).

Table 26-10 FADC Trigger/Gating Source Input Connections

Inputs	Signal	From Module
Gating Source Inputs		
GS0	P1.0 / REQ0	Port 1
GS1	P7.0 / REQ4	Port 7
GS2	PDOUT2	External Request Unit ERU (located in the SCU)
GS3	PDOUT3	
GS4	OUT1	GPTA0
GS5	OUT9	
GS6	OUT18	
GS7	OUT26	
Trigger Source Inputs		
TS0	P1.1 / REQ1	Port 1
TS1	P7.1 / REQ5	Port 7
TS2	PDOUT2	External Request Unit ERU (located in the SCU)
TS3	PDOUT3	
TS4	OUT2	GPTA0
TS5	OUT10	
TS6	OUT19	
TS7	OUT27	

26.3.6.3 Service Request Lines

The FADC module has four service request output lines. These service request outputs are connected to interrupt service request nodes with its control register (see [Page 26-63](#)) and in parallel with several request inputs of the TC1796's DMA controller. [Table 26-11](#) shows the FADC service request lines interconnections.

Table 26-11 Service Request Lines Interconnections

Module	Service Request Line	Connected To	Description
FADC	SR0	FADC_SRC0 CH10_REQI2 CH14_REQI2	FADC Service Request Node 0 DMA Channel 10 Request Input 2 DMA Channel 14 Request Input 2
	SR1	FADC_SRC1 CH11_REQI2 CH15_REQI2	FADC Service Request Node 1 DMA Channel 11 Request Input 2 DMA Channel 15 Request Input 2
	SR2	FADC_SRC2 CH12_REQI2 CH16_REQI2	FADC Service Request Node 2 DMA Channel 12 Request Input 2 DMA Channel 16 Request Input 2
	SR3	FADC_SRC3 CH13_REQI2 CH17_REQI2	FADC Service Request Node 3 DMA Channel 13 Request Input 2 DMA Channel 17 Request Input 2

Keyword Index

This section lists a number of keywords which refer to specific details of the TC1796 in terms of its architecture, its functional units, or functions. Bold page number entries identify the main definition material for a topic. The “Keyword Index” refers to page numbers in both parts of the TC1796 User’s Manual, the “System Units” (volume 1 with marking “[1]”) and the “Peripheral Units” (volume 2 with marking “[2]”) parts.

A

- Abbreviations 1-5 [1]
- Access mode definitions 1-4 [1]
- ADC
 - Analog input AN7 testmode 5-60 [1]
 - Arbitration 25-29 [2]
 - Block diagram 25-2 [2]
 - Clocking 25-33 [2]
 - Conversion request sources 25-5 [2]
 - Auto-scan 25-15 [2]
 - Channel injection 25-21 [2]
 - External event 25-11 [2]
 - Queue 25-25 [2]
 - Software 25-14 [2]
 - Timer 25-8 [2]
 - Conversions times 25-33 [2]
 - Expansion of analog channels 25-41 [2]
 - Features 25-1 [2]
 - Functional units 25-3 [2]
 - Input multiplexer 25-4 [2]
 - Limit checking 25-39 [2]
 - Module implementation
 - Input/output function selection 25-108 [2]
 - Module clock control 25-103 [2]
 - Pad output driver characteristics selection 25-110 [2]
 - Overload condition error 25-38 [2]
 - Power-up calibration 25-36 [2]
 - Reference voltages 25-37 [2]
 - Registers 25-57 [2]
 - AP **25-78 [2]**
 - ASCRP **25-77 [2]**
 - CHCONm **25-60 [2]**
 - CHIN **25-90 [2]**
 - CHSTATm **25-63 [2]**
 - CON **25-82 [2]**
 - EXCRP **25-74 [2]**
 - EXTC **25-73 [2]**
 - ID **25-59 [2]**
 - LCCONx **25-80 [2]**
 - MSS0 **25-94 [2]**
 - MSS1 **25-95 [2]**
 - Offset addresses 25-58 [2]
 - QR **25-71 [2]**
 - QUEUE0 **25-69 [2]**
 - REQ0 **25-92 [2]**
 - SAL **25-79 [2]**
 - SCN **25-75 [2]**
 - SCON **25-81 [2]**
 - SRNP **25-97 [2]**
 - STAT **25-86 [2]**
 - SW0CRP **25-93 [2]**
 - SYSTAT **25-84 [2]**
 - TCON **25-66 [2]**
 - TCRP **25-68 [2]**
 - TSTAT **25-67 [2]**
 - TTC **25-65 [2]**
 - Service request processing 25-43 [2]
 - Synchronization of two ADCs 25-49 [2]
 - Timing control 25-34 [2]
- Address map of segment 15 18-2 [1]
- Alternate Boot Mode 4-28 [1]
- ASC
 - Asynchronous mode 19-4 [2]–19-8 [2]
 - Data frames 19-5 [2]–19-6 [2]

Baud rate generation 19-12 [2]–
19-16 [2]
 Asynchronous modes 19-13 [2]
 Synchronous mode 19-16 [2]
Block diagram
 Asynchronous modes 19-4 [2]
 Synchronous mode 19-9 [2]
DMA request outputs 19-42 [2]
Error detection 19-17 [2]
Features 19-2 [2]
Interrupt generation 19-17 [2]
Module implementation 19-30 [2]–??
 DMA request outputs 19-42 [2]
 Input/output function selection
 19-36 [2]
 Pad output driver characteristics
 selection 19-39 [2]
Registers 19-19 [2]–19-29 [2]
 BG **19-27 [2]**
 CON **19-22 [2]**
 FDV **19-27 [2]**
 ID **19-20 [2]**
 Offset addresses 19-19 [2]
 Overview **19-19 [2]**
 PISEL **19-21 [2]**
 RBUF **19-29 [2]**
 TBUF **19-28 [2]**
 WHBCON **19-25 [2]**
Synchronous mode 19-9 [2]–19-11 [2]
 Timings 19-11 [2]

B

BCU

LBCU

 Offset addresses 6-7 [1]

SBCU

 Offset addresses 6-34 [1]

BCUs

DBCUs and PBCUs 6-5 [1]

 Bus agents and priorities 6-5 [1]

 Error handling 6-6 [1]

 Operation 6-5 [1]

 Registers 6-7 [1]

SBCU and RBCU 6-24 [1]

 Bus agents and priorities 6-24 [1]

 Bus arbitration 6-24 [1]

 Bus error handling 6-25 [1]

 Registers 6-34 [1]

 Starvation prevention 6-25 [1]

Boot 4-12 [1]

 Normal boot 4-14 [1]

 Scheme 4-12 [1]

 Selection table 4-12 [1]

Boot ROM 4-16 [1]

 Alternate Boot Mode 4-28 [1]

 Program Structure 4-16 [1]

C

CAN

 Address map 22-214 [2]

 Basics 22-2 [2]

 Addressing and arbitration 22-2 [2]

 Basic-/Full-CAN 22-10 [2]

 Error detection and handling
 22-9 [2]

 Frame formats 22-3 [2]

 Nominal bit time 22-8 [2]

 Block diagram 22-14 [2]

 DMA requests 22-209 [2]

 Module implementation 22-199 [2]–
 22-212 [2]

 External registers 22-200 [2]

 I/O line control 22-205 [2]

 Input/output function selection
 22-205 [2]

 Interfaces 22-199 [2]

 Interrupt control 22-210 [2]

 Module clock generation 22-201 [2]

MultiCAN

 Bit timing 22-24 [2]

 Block diagram 22-18 [2]

 Clock generation 22-21 [2]

 Interrupt structure 22-20 [2]

 Message acceptance filtering
 22-37 [2]

 Message object data handling

- 22-44 [2]
- Message object FIFO 22-51 [2]
- Message object functionality 22-50 [2]
- Message object interrupts 22-40 [2]
- Message object lists 22-29 [2]
- Node control 22-23 [2]
- Node interrupts 22-27 [2]
- Suspend mode 22-22 [2]
- MultiCAN module features 22-15 [2]
- MultiCAN registers
 - Address map 22-60 [2]
 - ID **22-61 [2]**
 - LISTi **22-68 [2]**
 - MCR **22-66 [2]**
 - MITR **22-67 [2]**
 - MOAMRn **22-107 [2]**
 - MOARn **22-109 [2]**
 - MOCTRn **22-91 [2]**
 - MODATAHn **22-113 [2]**
 - MODATALn **22-112 [2]**
 - MOFCRn **22-101 [2]**
 - MOFGPRn **22-106 [2]**
 - MOIPRn **22-99 [2]**
 - MOSTATn **22-94 [2]**
 - MSIDk **22-71 [2]**
 - MSIMASK **22-72 [2]**
 - MSPNDk **22-70 [2]**
 - NBTRx **22-84 [2]**
 - NCRx **22-73 [2]**
 - NECNTx **22-86 [2]**
 - NFCRx **22-87 [2]**
 - NIPRx **22-81 [2]**
 - NPCRx **22-83 [2]**
 - NSRx **22-77 [2]**
 - Offset addresses 22-59 [2]
 - Overview 22-58 [2]
 - PANCTR **22-62 [2]**
- TTCAN 22-114 [2]
 - Application watchdog 22-150 [2]
 - Automatic TUR adjust 22-115 [2]
 - Configuration 22-148 [2]
 - Cycle time 22-115 [2]
 - Error handling 22-150 [2]
 - Interrupt control 22-152 [2]
 - Local offset and global time 22-118 [2]
 - Local time generation 22-114 [2]
 - Master reference mark 22-117 [2]
 - MSC handling 22-151 [2]
 - Reference message 22-120 [2]
 - Scheduler 22-123 [2]
 - Scheduler entry setup 22-142 [2]
 - Scheduler entry types 22-125 [2]
 - ARBE 22-131 [2]
 - BCE 22-140 [2]
 - EOS 22-141 [2]
 - ICE 22-129 [2]
 - RCE 22-136 [2]
 - RME 22-138 [2]
 - TCE 22-133 [2]
 - TME 22-126 [2]
 - Scheduler instruction sequence 22-145 [2]
 - Scheduler memory 22-124 [2]
 - Transmit enable window 22-117 [2]
 - Transmit trigger 22-119 [2]
 - TTCAN extension features 22-17 [2]
- TTCAN Basics 22-11 [2]
 - Basic cycle 22-11 [2]
 - Global system time 22-12 [2]
 - Network time unit 22-13 [2]
 - System matrix 22-12 [2]
 - Time reference 22-11 [2]
- TTCAN extension registers 22-154 [2]
 - Address map 22-156 [2]
 - AWDR **22-168 [2]**
 - CYCTMR **22-163 [2]**
 - GMR **22-166 [2]**
 - LGMR **22-167 [2]**
 - LOR **22-164 [2]**
 - LREFMR **22-162 [2]**
 - LTR **22-159 [2]**
 - Offset addresses 22-154 [2]
 - Overview 22-154 [2]

- REFMR **22-161 [2]**
 - SISR **22-195 [2]**
 - STPTR0 **22-192 [2]**
 - STSRH **22-194 [2]**
 - STSRL **22-193 [2]**
 - SYNMR **22-160 [2]**
 - TTCFGR **22-173 [2]**
 - TTCR **22-169 [2]**
 - TTFMR **22-180 [2]**
 - TTIER **22-186 [2]**
 - TTINPR **22-190 [2]**
 - TTIRR **22-182 [2]**
 - TTSR **22-175 [2]**
 - TURR **22-157 [2]**
 - Chip select outputs 13-28 [1]
 - CLC register 3-24 [1]
 - Implementation in the modules 3-39 [1]
 - Clock system
 - CGU block diagram 3-3 [1]
 - Clock source 3-10 [1]
 - Clock tree diagram 3-2 [1]
 - Features 3-1 [1]
 - Gain control 3-7 [1]
 - Main oscillator 3-4 [1]
 - Main oscillator circuits 3-5 [1]
 - Module clock generation 3-23 [1]
 - CLC register 3-24 [1]
 - Fractional divider 3-29 [1]
 - OSC_CON register 3-8 [1]
 - Oscillator run detection 3-6 [1]
 - Overview 3-1 [1]
 - Configuration inputs 5-8 [1]
 - Register 10-24 [1]
 - CPS
 - Registers
 - CPU_SBSRC **2-19 [1]**
 - CPU_SRCn **2-14 [1]**
 - CPU
 - Block diagram 2-3 [1]
 - Execution unit 2-5 [1]
 - Features 2-2 [1]
 - General purpose register file 2-6 [1]
 - Implementation-specific features
 - 2-2 [1]
 - Instruction fetch unit 2-4 [1]
 - Registers 2-9 [1]
 - Core debug registers 2-17 [1]
 - CPS registers 2-13 [1]
 - CPU_SRCn 2-14 [1]
 - CSFRs 2-10 [1]
 - GPRs 2-15 [1]
 - Memory protection registers 2-20 [1]
 - MMU_CON 2-12 [1]
 - PSW 2-11 [1]
 - SBSRC0 2-19 [1]
 - TRnEVT 2-18 [1]
 - CPU, see also Processor subsystem
 - CSCOMB control 13-28 [1]
 - CSFR registers 2-10 [1]
- D**
- Dedicated peripheral I/O lines 10-83 [1]
 - for MSC0 and MSC1 10-85 [1]
 - for SSC0 and SSC1 10-83 [1]
 - Die temperature sensor 5-48 [1], 25-120 [2]
 - DMA 12-2 [1]
 - Access protection 12-40 [1]
 - Access protection assignment 12-98 [1]
 - Block diagram 12-2 [1]
 - Bus bandwidth allocation 12-22 [1]
 - Bus switch 12-21 [1]
 - Channel operation 12-6 [1]
 - Channel operation modes 12-11 [1]
 - Channel request control 12-10 [1]
 - Channel reset operation 12-16 [1]
 - Circular buffer 12-19 [1]
 - Debug capabilities 12-23 [1]
 - Definition of terms 12-4 [1]
 - Error conditions 12-15 [1]
 - Features 12-3 [1]
 - Implementation diagram 12-91 [1]
 - Interrupts 12-27 [1]
 - Channel interrupts 12-27 [1]

- Move engine interrupts 12-30 [1]
- Transaction lost interrupts 12-29 [1]
- Wrap buffer interrupts 12-32 [1]
- Pattern detection 12-34 [1]
- Principle 12-5 [1]
- Registers
 - ADRCRmx **12-83 [1]**
 - Block address map 12-109 [1]
 - CHCRmx **12-76 [1]**
 - CHICRmx **12-81 [1]**
 - CHRSTR **12-51 [1]**
 - CHSRmx **12-80 [1]**
 - CLRE **12-62 [1]**
 - DADRmx **12-89 [1]**
 - EER **12-56 [1]**
 - ERRSR **12-59 [1]**
 - GINTR **12-50 [1]**
 - HTREQ **12-55 [1]**
 - ID **12-45 [1]**
 - INTCR **12-67 [1]**
 - INTSR **12-64 [1]**
 - MEmAENR **12-73 [1]**
 - MEMARR **12-74 [1]**
 - MEMPR **12-72 [1]**
 - MEMR **12-71 [1]**
 - MESR **12-69 [1]**
 - OCDSR **12-46 [1]**
 - Offset addresses 12-43 [1]
 - Overview **12-42 [1]**
 - SADRmx **12-88 [1]**
 - SHADRmx **12-90 [1]**
 - STREQ **12-54 [1]**
 - SUSPMR **12-48 [1]**
 - TRSR **12-52 [1]**
 - WRPSR **12-66 [1]**
- Request wiring matrix 12-91 [1]
- Transaction control 12-20 [1]
- DMI
 - Dual-ported RAM 2-32 [1]
 - Features 2-31 [1]
 - Registers
 - DMI_ATR **2-39 [1]**
 - DMI_CON **2-36 [1]**
 - DMI_CON1 **2-37 [1]**
 - DMI_STR **2-38 [1]**
- DMU
 - Access performance 8-7 [1]
 - Block diagram 8-1 [1]
 - Data access overlay operation 8-3 [1]
 - Data access redirection 8-4 [1]
 - Emulation memory overlay 8-7 [1]
 - Registers 8-10 [1]
 - ID **8-11 [1]**
 - OMASKx **8-16 [1]**
 - OTARx **8-15 [1]**
 - RABRx **8-13 [1]**
 - SBRCTR **8-12 [1]**
 - SBRAM 8-2 [1]
 - SRAM 8-2 [1]
- Document
 - Structure 1-1 [1]
 - Terminology and abbreviations 1-3 [1]
 - Textual conventions 1-1 [1]
- E**
- EBU 13-1 [1]
 - Access parameter selection 13-35 [1]
 - Access phases 13-41 [1]
 - Address phase 13-41 [1]
 - Burst phase 13-46 [1]
 - Command delay phase 13-43 [1]
 - Command phase 13-43 [1]
 - Data hold phase 13-45 [1]
 - Recovery phase 13-47 [1]
 - Address region selection 13-31 [1]
 - Asynchronous accesses
 - Demultiplexed device configurations 13-51 [1]
 - Demultiplexed read cycles 13-55 [1]
 - Demultiplexed write cycles 13-56 [1]
 - Signals 13-50 [1]
 - Wait control 13-57 [1]
 - Boot operation 13-20 [1]
 - Boot read access cycle 13-22 [1]

- Configuration word 13-23 [1]
- Emulation mode 13-20 [1]
- External boot mode 13-21 [1]
- Start-up modes 13-20 [1]
- Burst Flash
 - Application configurations 13-66 [1], 13-67 [1]
 - Burst read timing 13-74 [1], 13-75 [1]
 - Clock input BFCLKI 13-72 [1]
 - Clock output BFCLKO 13-71 [1]
 - Cycle termination 13-78 [1]
 - Signals 13-65 [1]
 - Wait control 13-75 [1]
- Chip select override mode 13-34 [1]
- Data re-alignment 13-37 [1]
- External bus arbitration 13-7 [1]
 - Arbitration modes 13-10 [1]
 - Arbitration signals 13-7 [1]
 - Arbitration timing 13-11 [1]
 - EBU as participant 13-14 [1]
 - EBU as sole master 13-10 [1]
 - Locking external bus 13-17 [1]
 - Modes 13-7 [1]
 - PLMB access to external bus 13-18 [1]
- Interfacing INTEL-style devices 13-61 [1]
- Interfacing MOTOROLA-style devices 13-63 [1]
- Memory regions 13-26 [1]
- Registers
 - ADDRSELx **13-90 [1]**
 - BFCON **13-86 [1]**
 - BUSAPx **13-97 [1]**
 - BUSCONx **13-92 [1]**
 - CLC **13-82 [1]**
 - CON **13-83 [1]**
 - EMUAS **13-101 [1]**
 - EMUBAP **13-106 [1]**
 - EMUBC **13-102 [1]**
 - EMUOVL **13-110 [1]**
 - ID **13-81 [1]**
 - Overview 13-79 [1]
 - USERCON **13-111 [1]**
- EBU, see “External bus interface unit”
- EEPROM emulation 7-26 [1]
- Emergency stop output control 5-57 [1]
 - for GPTA 5-58 [1]
 - Register SCU_EMSSR **5-59 [1]**
- Entering Sleep Mode 5-6 [1]
- Error correction in Flash 7-34 [1]
- External bus interface unit, see “EBU”
- External request unit 5-9 [1]
 - Block diagram 5-9 [1]
 - Features 5-9 [1]
 - Implementation 5-14 [1]
 - Interrupt gating logic 5-12 [1]
 - Registers 5-16 [1]
 - EICR0 **5-17 [1]**
 - EICR1 **5-20 [1]**
 - EIFR **5-23 [1]**
 - FMR **5-24 [1]**
 - IGCR0 **5-26 [1]**
 - IGCR1 **5-29 [1]**
 - Overview 5-16 [1]
 - PDDR **5-25 [1]**
 - TGADC0 **5-32 [1]**
 - TGADC1 **5-32 [1]**
 - TGADCx **25-116 [2]**
 - Request select logic 5-10 [1]

F

FADC

- Analog inputs 26-4 [2]
- Calibration 26-22 [2]
- Channel timers 26-11 [2]
- Channel triggers 26-9 [2]
- Clock generation 26-13 [2]
- Conversion priority 26-12 [2]
- Data reduction filter 26-13 [2]
- Features 26-2 [2]
- Filter block structure 26-15 [2]
- Filter concatenation 26-19 [2]
- Gating modes 26-10 [2]
- Interrupts 26-24 [2]

- Module implementation
 - Input function selection 26-61 [2]
 - Module clock control 26-58 [2]
- Neighbor channel triggers 26-21 [2]
- Registers
 - ACRx **26-45 [2]**
 - CFGRx **26-41 [2]**
 - CRRn **26-51 [2]**
 - CRSR **26-30 [2]**
 - FCRn **26-48 [2]**
 - FMR **26-32 [2]**
 - FRRn **26-55 [2]**
 - GCR **26-37 [2]**
 - ID **26-29 [2]**
 - IRRmn **26-54 [2]**
 - NCTR **26-34 [2]**
 - Offset addresses 26-28 [2]
 - Overview **26-27 [2]**
 - RCHx **26-47 [2]**
- Trigger modes 26-10 [2]
- FDR register 3-35 [1]
- Features
 - Development support 1-13 [1]
 - External bus interface 1-10 [1]
 - Instruction set 1-10 [1]
 - Interrupt system 1-11 [1]
 - On-chip memory 1-10 [1]
- Flash
 - Block diagram 7-3 [1]
 - Command sequences overview 7-14 [1]
 - Commands
 - Clear Status 7-25 [1]
 - Disable Read Protection 7-24 [1]
 - Disable Write Protection 7-23 [1]
 - Enter Page Mode 7-15 [1]
 - Erase Sector 7-20 [1]
 - Erase User Configuration Block 7-22 [1]
 - Load Page Buffer 7-16 [1]
 - Reset-to-Read 7-15 [1]
 - Resume Protection 7-24 [1]
 - Write Page 7-18 [1]
 - Write User Configuration Page 7-19 [1]
- Data Flash
 - Bank, sector, and page definitions 7-8 [1]
 - Features 7-8 [1]
 - Overview 7-7 [1]
 - Structure 7-7 [1]
- EEPROM emulation 7-26 [1]
- Error correction 7-34 [1]
- Interrupt 5-36 [1]
- Interrupt generation and control 7-36 [1]
- Margin control 7-35 [1]
- Operating modes 7-11 [1]
 - Command mode 7-12 [1]
 - Page mode 7-13 [1]
 - Read mode 7-11 [1]
- OTP protection 7-30 [1]
- Overview 7-2 [1]
- Password check control 7-33 [1]
- Power saving modes 7-38 [1]
- Power supply 7-38 [1]
- Program Flash
 - Bank, sector, and page definitions 7-6 [1]
 - Features 7-5 [1]
 - Overview 7-4 [1]
 - Structure 7-4 [1]
- Protection configuration 7-59 [1]
- Read protection 7-27 [1], 7-32 [1]
- Register
 - Offset address 7-40 [1]
- Registers
 - FCON **7-53 [1]**
 - FSR **7-43 [1]**
 - ID **7-41 [1]**
 - MARD **7-52 [1]**
 - MARP **7-51 [1]**
 - Overview 7-40 [1]
 - PROCON0 **7-59 [1]**
 - PROCON1 **7-60 [1]**
 - PROCON2 **7-61 [1]**

- Reset control 7-39 [1]
- User configuration blocks
 - Content definitions 7-27 [1]
 - Definitions 7-10 [1]
 - Overview 7-9 [1]
 - Structure 7-9 [1]
- Write protection 7-27 [1], 7-30 [1]
- FPI Buses 6-19 [1]
 - Basic operations 6-22 [1]
 - Block diagram 6-20 [1]
 - Overview 6-19 [1]
 - Transaction types 6-21 [1]
- FPU
 - Interrupt 5-35 [1]
- Fractional divider 3-29 [1]
 - Block diagram 3-30 [1]
 - FDR register 3-35 [1]
 - Function table 3-39 [1]
 - Implementation in the modules 3-40 [1]
 - Operating modes 3-32 [1]
 - Suspend mode 3-34 [1]
- G**
- GPTA
 - Address ranges 24-274 [2]
 - Block diagram 24-6 [2]
 - Clock generation unit (CGU) 24-8 [2]
 - Clock distribution module 24-35 [2]
 - Digital phase locked loop cell 24-28 [2]
 - Duty cycle measurement unit 24-24 [2]
 - Filter and prescaler cell 24-10 [2]
 - Phase discrimination logic 24-19 [2]
 - Emergency stop output control 5-58 [1]
 - Features of GPTA0/GPTA1 24-3 [2]
 - Input IN1 control 5-49 [1]
 - Interrupt processing and control 24-112 [2], 24-231 [2]
 - Module implementations 24-247 [2]
 - Block diagram 24-248 [2]
 - Cascading limits 24-272 [2]
 - External registers 24-249 [2]
 - Input/output function selection 24-251 [2]
 - Module clock generation 24-265 [2]
 - On-chip connections 24-256 [2]
 - Port control and connections 24-250 [2]
 - Overview 24-2 [2]
 - Programming hints 24-148 [2]
 - Pseudo-code description 24-115 [2]
 - Registers
 - Address ranges 24-274 [2]
 - CKBCTR **24-173 [2]**
 - DCMCAV_k **24-163 [2]**
 - DCMCOV_k **24-164 [2]**
 - DCMCTR_k **24-161 [2]**
 - DCMTIM_k **24-163 [2]**
 - FPCCTR_k **24-156 [2]**
 - FPCSTAT **24-155 [2]**
 - FPCTIM_k **24-158 [2]**
 - GIMCRHg **24-200 [2]**
 - GIMCRLg **24-198 [2]**
 - GTCCTR_k **24-175 [2]**, **24-177 [2]**
 - GTCTR_k **24-170 [2]**
 - GTCXR_k **24-179 [2]**
 - GTREV_k **24-172 [2]**
 - GTTIM_k **24-171 [2]**
 - ID **24-154 [2]**
 - LIMCRHg **24-204 [2]**
 - LIMCRLg **24-202 [2]**
 - LTCCTR63 **24-188 [2]**
 - LTCCTR_k **24-180 [2]**
 - LTCXR63 **24-190 [2]**
 - LTCXR_k **24-190 [2]**
 - MRACTL **24-191 [2]**
 - MRADIN **24-193 [2]**
 - MRADOUT **24-193 [2]**
 - Offset addresses 11-53 [1], 24-151 [2]
 - OMCRHg **24-196 [2]**
 - OMCRLg **24-194 [2]**
 - Overview 24-150 [2]
 - PDLCTR **24-159 [2]**

PLLCNT **24-167 [2]**
 PLLCTR **24-165 [2]**
 PLLDTR **24-168 [2]**
 PLLMTI **24-166 [2]**
 PLLREV **24-168 [2]**
 PLLSTP **24-167 [2]**
 SRNR **24-215 [2]**
 SRSC0 **24-206 [2]**
 SRSC1 **24-209 [2]**
 SRSC2 **24-211 [2]**
 SRSC3 **24-213 [2]**
 SRSS0 **24-208 [2]**
 SRSS1 **24-210 [2]**
 SRSS2 **24-212 [2]**
 SRSS3 **24-214 [2]**
 Signal generation unit (SGU)
 Global timer cell **24-55 [2]**
 Global timers **24-37 [2]**
 Local timer cell **24-67 [2]**
 GPTA0
 Registers
 GTTIMk **24-171 [2]**

I

Idle mode **5-5 [1]**
 Instruction timing **2-40 [1]–??**
 Interrupt system
 Arbitration cycles **14-12 [1]**
 Arbitration process **14-12 [1]**
 Block diagram **14-2 [1]**
 Control register ICR **14-8 [1]**
 External interrupts **14-22 [1]**
 Hints for applications **14-18 [1]–14-22 [1]**
 Interrupt control unit **14-8 [1]**
 Interrupt vector table **14-15 [1]**
 Overview **14-1 [1]**
 Priorities **14-19 [1]**
 Service request control register **14-3 [1], 14-4 [1]**
 Service request node table **14-23 [1]**
 Service request nodes **14-3 [1]**
 Service routine entering **14-13 [1]**

 Service routine exiting **14-14 [1]**
 Software initiated interrupts **14-22 [1]**
 Interrupts
 Special system interrupts **5-35 [1]**
 External interrupts **5-36 [1]**
 Flash interrupt **5-36 [1]**
 FPU interrupt **5-35 [1]**

L

LFI bridge **6-15 [1]**
 Address translation **6-15 [1]**
 Register
 CON **6-17 [1]**
 Offset address **6-16 [1]**
 Overview **6-16 [1]**
 LMB **6-2 [1]**
 Basic operation **6-4 [1]**
 Default master **6-6 [1]**
 Features **6-2 [1]**
 Terms **6-2 [1]**
 Transaction types **6-3 [1]**
 LMI **8-8 [1]**
 Data read buffer **8-8 [1]**
 Local memory bus, see “LMB”
 LTCA2
 Registers
 ID **24-235 [2]**
 LIMCRHg **24-245 [2]**
 LIMCRLg **24-243 [2]**
 LTCCTR63 **24-188 [2]**
 LTCCTRk **24-180 [2]**
 LTCXR63 **24-190 [2]**
 LTCXRk **24-190 [2]**
 MRACTL **24-236 [2]**
 MRADIN **24-238 [2]**
 MRADOUT **24-238 [2]**
 Offset addresses **24-233 [2]**
 OMCRHg **24-241 [2]**
 OMCRLg **24-239 [2]**
 Overview **24-233 [2]**
 SRSC2 **24-211 [2]**
 SRSC3 **24-213 [2]**
 SRSS2 **24-212 [2]**

- SRSS3 **24-214** [2]
- LVDS outputs 10-85 [1]
- M**
- Memory checker 12-110 [1]
 - Functionality 12-110 [1]
 - Registers 12-111 [1]
 - ID **12-112** [1]
 - IR **12-113** [1]
 - Offset addresses 12-111 [1]
 - Overview **12-111** [1]
 - RR **12-113** [1]
 - WR **12-114** [1]
- Memory maps 9-1 [1]
 - Access restrictions 9-22 [1]
 - from DLMB 9-19 [1]
 - from PLMB 9-15 [1]
 - Segment 0 to 14 from SPB and RPB 9-6 [1]
 - Segment 15 from SPB and RPB 9-10 [1]
 - Segment contents 9-4 [1]
- Memory protection system
 - Registers
 - Offset addresses 2-21 [1]
- MLI
 - Access protection 23-47 [2]
 - Block diagram 1-25 [1], 23-49 [2]
 - Communication principles 23-6 [2]
 - Frames 23-10 [2]
 - Answer frame 23-18 [2]
 - Command frame 23-17 [2]
 - Copy base address frame 23-12 [2]
 - Layout 23-11 [2]
 - Optimized read frame 23-16 [2]
 - Optimized write frame 23-14 [2]
 - Write offset and data frame 23-13 [2]
 - Handshake signalling 23-19 [2]
 - Interrupts 23-55 [2]
 - Receiver interrupts 23-60 [2]
 - Transmitter interrupts 23-57 [2]
 - Kernel registers 23-75 [2]
- Module implementation 23-124 [2]
 - Access protection 23-141 [2]
 - Clock generation 23-128 [2]
 - Input/output function selection 23-130 [2]
 - MLIO block diagram 23-125 [2], 23-126 [2]
 - On-chip connections 23-139 [2]
 - Transfer window map 23-146 [2]
- Naming conventions 23-3 [2]
- Receiver
 - I/O line control 23-51 [2]
- Registers
 - AER **23-88** [2]
 - ARR **23-89** [2]
 - FDR **23-79** [2]
 - Offset addresses 23-76 [2]
 - OICR **23-84** [2]
 - Overview 23-75 [2]
 - RDARR **23-116** [2]
 - RDATAR **23-114** [2]
 - RIER **23-117** [2]
 - RINPR **23-122** [2]
 - RISR **23-120** [2]
 - RPxBAR **23-115** [2]
 - RPxSTATR **23-113** [2]
 - SCR **23-78** [2], **23-81** [2], **23-83** [2]
 - TCBAR **23-104** [2], **23-110** [2]
 - TCMDR **23-97** [2]
 - TCR **23-90** [2]
 - TDRAR **23-101** [2]
 - TIER **23-105** [2]
 - TINPR **23-108** [2]
 - TISR **23-107** [2]
 - TPxAOFR **23-103** [2]
 - TPxBAR **23-102** [2]
 - TPxDATAR **23-101** [2]
 - TPxSTATR **23-95** [2]
 - TRSTATR **23-99** [2]
 - TSTATR **23-93** [2]
- Transaction flow diagrams
 - Command frame 23-39 [2]
 - Copy base address 23-26 [2]

- Read frame 23-33 [2]
- Write frame 23-28 [2]
- Transmitter
 - Description of frame transmission 23-26 [2]
 - I/O line control 23-51 [2]
 - Typical application 23-2 [2]
- Module clock generation 3-23 [1]
- MSC
 - Applications 21-2 [2]
 - Downstream channel 21-5 [2]
 - Baud rate 21-20 [2]
 - Block diagram 21-5 [2]
 - Command frames 21-7 [2]
 - Data frames 21-9 [2]
 - Data repetition mode 21-15 [2]
 - Downstream counter 21-19 [2]
 - Frame formats 21-6 [2]
 - Output control 21-27 [2]
 - Passive time frames 21-11 [2]
 - Shift register operation 21-12 [2]
 - Transmission mode 21-14 [2]
 - Triggered mode 21-14 [2]
 - Features 21-4 [2]
 - I/O control 21-27 [2]
 - Interrupts 21-31 [2]
 - Command frame interrupt 21-32 [2]
 - Data frame interrupt 21-32 [2]
 - Interrupt request compressor 21-35 [2]
 - Receive data interrupt 21-34 [2]
 - Time frame finished interrupt 21-33 [2]
 - Kernel block diagram 1-19 [1], 21-3 [2]
 - Module implementation
 - Input/output function selection 21-70 [2]
 - Module clock control 21-65 [2]
 - Pad output driver characteristics selection 21-74 [2]
 - Overview 21-3 [2]
 - Registers
 - DC **21-59 [2]**
 - DD **21-59 [2]**
 - DSC **21-41 [2]**
 - DSDSH **21-47 [2]**
 - DSDSL **21-46 [2]**
 - DSS **21-44 [2]**
 - ESR **21-48 [2]**
 - ICR **21-49 [2]**
 - ID **21-38 [2]**
 - ISC **21-54 [2]**
 - ISR **21-52 [2]**
 - OCR **21-56 [2]**
 - Offset addresses 21-36 [2]
 - Overview 21-36 [2]
 - UDx **21-60 [2]**
 - USR **21-39 [2]**
 - Upstream channel 21-21 [2]
 - Baud rate 21-25 [2]
 - Block diagram 21-21 [2]
 - Data frame protocol 21-22 [2]
 - Data reception 21-23 [2], 21-24 [2]
 - Input control 21-30 [2]
 - Parity checking 21-22 [2]
 - Sampling 21-26 [2]
- N**
 - NMI 14-25 [1]
 - NMI input 14-25 [1]
 - PLL NMI 14-25 [1]
 - SRAM parity error control 5-37 [1]
 - Status register NMISR 14-27 [1]
 - Watchdog timer NMI 14-25 [1]
- O**
 - OCDS
 - Cerberus 17-12 [1]
 - Communication mode 17-13 [1]
 - Features 17-12 [1]
 - Multi-core break switch 17-13 [1]
 - Registers 17-16 [1]
 - RW mode 17-12 [1]
 - Triggered transfers 17-13 [1]
 - Components 17-2 [1]
 - JTAG interface 17-15 [1]

- Registers 17-16 [1]
- OCDS level 1 17-4 [1]
 - Debug actions 17-7 [1]
 - Debug event generation 17-6 [1]
 - of BCU 17-9 [1]
 - of CPU 17-4 [1]
 - of DMA 17-9 [1]
 - of PCP 17-8 [1]
 - Registers 17-8 [1]
- OCDS level 2 17-10 [1]
 - Concurrent debugging 17-11 [1]
 - CPU trace 17-10 [1]
 - DMA trace 17-10 [1]
- OCDS level 3 17-1 [1]
- Overview 17-1 [1]
- System block diagram 17-2 [1]

- On-chip debug support
 - Registers
 - TR0EVT **2-18 [1]**
 - TR1EVT **2-18 [1]**

P

- Pad test mode 5-50 [1]
 - Block diagram 5-50 [1]
 - Enable control 5-51 [1]
 - Registers 5-51 [1]
 - SCU_PTCON **5-52 [1]**
 - SCU_PTDAT0 **5-53 [1]**
 - SCU_PTDAT1 **5-54 [1]**
 - SCU_PTDAT2 **5-55 [1]**
 - SCU_PTDAT3 **5-56 [1]**
- Parity protection
 - General control 5-37 [1]
 - in CAN memories 22-213 [2]
 - in DMI memories 2-33 [1]
 - in DMU memories 8-3 [1]
 - in PCP memories 11-125 [1]
 - in PMI memories 2-25 [1]

- PCP 11-1 [1]
 - Accessing from FPI bus 11-48 [1]
 - Architecture 11-2 [1]
 - Channel programs 11-25 [1]
 - Context models 11-11 [1]

- Control and interrupt registers 11-52 [1]
- Debugging 11-50 [1]
- Error handling 11-38 [1]
- General purpose registers 11-6 [1]
- Implementation in TC1796 11-125 [1]
- Instruction set details 11-72 [1]
- Instruction set overview 11-40 [1]
- Instruction timing 11-108 [1]
- Interrupt operation 11-32 [1]
- Overview 11-1 [1]
- Programming 11-112 [1]
- Programming model 11-6 [1]
- Programming tips 11-118 [1]
- Registers

- ID **11-54 [1]**
- Overview 11-52 [1]
- PCP_CLC **11-55 [1]**
- PCP_CS **11-56 [1]**
- PCP_ES **11-59 [1]**
- PCP_ICON **11-64 [1]**
- PCP_ICR **11-61 [1]**
- PCP_ITR **11-63 [1]**
- PCP_SRC0 **11-67 [1]**
- PCP_SRC1 **11-67 [1]**
- PCP_SRC10 **11-70 [1]**
- PCP_SRC11 **11-70 [1]**
- PCP_SRC2 **11-68 [1]**
- PCP_SRC3 **11-68 [1]**
- PCP_SRC4 **11-69 [1]**
- PCP_SRC5 **11-69 [1]**
- PCP_SRC6 **11-69 [1]**
- PCP_SRC7 **11-69 [1]**
- PCP_SRC8 **11-69 [1]**
- PCP_SRC9 **11-70 [1]**
- PCP_SSR **11-66 [1]**

- Peripheral control processor, see PCP
- Pin configuration 1-35 [1]
- Pin definitions and functions 1-34 [1]–??
- PLL 3-10 [1]
 - Lock detection 3-19 [1]
 - Loss-of-lock recovery 3-19 [1]
 - Parameters 3-12 [1]

- PLL_CLC register 3-16 [1]
- Setup after reset 3-18 [1]
- Startup 3-21 [1]
- Switching parameters 3-18 [1]
- PMI
 - Block diagram 2-24 [1]
 - Features 2-24 [1]
 - Registers 2-26 [1]
 - PMI_CON0 **2-28 [1]**
 - PMI_CON1 **2-29 [1]**
 - PMI_CON2 **2-30 [1]**
- PMU
 - Block diagram 7-1 [1]
 - Boot ROM 7-1 [1]
 - Emulation interface 7-61 [1]
 - Overlay Memory Control Registers
 - Offset addresses 8-10 [1], 13-79 [1]
 - Overview 8-10 [1], 13-79 [1]
 - Program and data Flash 7-2 [1]–7-61 [1]
 - Register
 - ID **7-42 [1]**
- Ports
 - Basic operation 10-2 [1]
 - Emergency stop control 5-57 [1]
 - General port structure 10-2 [1]
 - Overview diagram 10-1 [1]
 - Pad driver control 10-10 [1]
 - Pad driver mode selection 10-11 [1]
 - Pad test mode 5-50 [1]
 - Port 0 10-18 [1]
 - Configuration diagram 10-18 [1]
 - Function table 10-19 [1]
 - Register overview 10-22 [1]
 - Software configuration 10-24 [1]
 - Port 1 10-26 [1]
 - Configuration diagram 10-26 [1]
 - Function table 10-27 [1]
 - Register overview 10-30 [1]
 - Port 10 10-81 [1]
 - Configuration diagram 10-81 [1]
 - Register overview 10-82 [1]
 - Port 2 10-32 [1]
 - Configuration diagram 10-32 [1]
 - Function table 10-33 [1]
 - Register overview 10-37 [1]
 - Port 3 10-40 [1]
 - Configuration diagram 10-40 [1]
 - Function table 10-41 [1]
 - Register overview 10-45 [1]
 - Port 4 10-47 [1]
 - Configuration diagram 10-47 [1]
 - Function table 10-48 [1]
 - Register overview 10-51 [1]
 - Port 5 10-54 [1]
 - Configuration diagram 10-54 [1]
 - Function table 10-55 [1]
 - Register overview 10-57 [1]
 - Port 6 10-59 [1]
 - Configuration diagram 10-59 [1]
 - Function table 10-60 [1]
 - Register overview 10-63 [1]
 - Port 7 10-65 [1]
 - Configuration diagram 10-65 [1]
 - Function table 10-66 [1]
 - Register overview 10-68 [1]
 - Port 8 10-70 [1]
 - Configuration diagram 10-70 [1]
 - Function table 10-71 [1]
 - Register overview 10-73 [1]
 - Port 9 10-75 [1]
 - Configuration diagram 10-75 [1]
 - Function table 10-76 [1]
 - Register overview 10-78 [1]
 - Port control coding 10-10 [1]
 - Registers 10-5 [1]
 - Emergency stop register 10-16 [1]
 - Input register 10-17 [1]
 - Input/output control registers 10-7 [1]
 - Output modification register 10-14 [1]
 - Output register 10-13 [1]
 - Pad driver mode register 10-10 [1]
 - Temperature compensation 5-41 [1]
- Ports and I/O lines overview 10-1 [1]

Power management 5-2 [1]
 Idle mode 5-5 [1]
 Mode definitions 5-2 [1]
 Register PMG_CSR **5-3 [1]**
 Sleep mode 5-6 [1]
 Summary 5-7 [1]
 Processor subsystem
 Core SFRs 2-10 [1]
 Implementation-specific features 2-7 [1]
 Interrupt system 2-8 [1]
 Subsystem block diagram 2-1 [1]

R

Register overview and address map 18-1 [1]
 Remote peripheral bus 6-19 [1]
 Reset
 Debug system reset 4-10 [1]
 External hardware reset 4-7 [1]
 Module behavior 4-10 [1]
 Overview 4-1 [1]
 Power-on reset 4-7 [1]
 Registers
 RST_REQ **4-5 [1]**
 RST_SR **4-3 [1]**
 Software reset 4-8 [1]
 WDT reset 4-8 [1]

S

SCU
 Miscellaneous registers
 CHIPID **5-70 [1]**
 MANID **5-69 [1]**
 RTID **5-71 [1]**
 SCU_CON **5-64 [1]**
 SCU_STAT **5-67 [1]**
 Registers
 Offset addresses 5-61 [1]
 Overview 5-61 [1]
 Sleep mode 5-6 [1]
 Software configuration
 in TC1796 devices 10-25 [1]

via SWOPT bits 10-24 [1]
 SSC
 Baud rate generation 20-18 [2], 20-48 [2]
 Baud rate generation formulas 20-18 [2], 20-49 [2]
 Block diagram 20-4 [2]
 Chip select generation 20-21 [2]
 DMA request outputs 20-60 [2]
 Error detection 20-24 [2]
 FIFO operation
 Receive FIFO 20-14 [2]
 Transmit FIFO 20-12 [2]
 Transparent Mode 20-16 [2]
 Full-duplex operation 20-6 [2]
 Half-duplex operation 20-9 [2]
 Interrupts 20-24 [2]
 Module implementation 20-46 [2]–??
 DMA request outputs 20-60 [2]
 Interrupt registers 20-59 [2]
 Module clock control 20-48 [2]
 Port control 20-52 [2]
 Registers 20-27 [2]–20-45 [2]
 BR **20-39 [2]**
 CON **20-32 [2]**
 EFM **20-35 [2]**
 FSTAT **20-44 [2]**
 ID **20-29 [2]**
 Offset addresses 20-27 [2]
 Overview 20-27 [2]
 PISEL **20-30 [2]**
 RB **20-45 [2]**
 RXFCON **20-40 [2]**
 SSOC **20-37 [2]**
 SSOTC **20-38 [2]**
 STAT **20-34 [2]**
 TB **20-45 [2]**
 TXFCON **20-42 [2]**
 Slave select input operation 20-20 [2]
 Slave select output operation 20-21 [2]
 STM, see “System timer” 15-1 [1]
 System clock output 3-41 [1]
 System control unit, see “SCU”

System peripheral bus 6-19 [1]
 System timer
 Block diagram 15-3 [1]
 Compare register operation 15-5 [1]
 Interrupt control 15-6 [1]
 Operation 15-1 [1]
 Overview 15-1 [1]
 Registers
 Offset addresses 15-8 [1]
 Overview 15-7 [1]
 STM_CAP **15-14 [1]**
 STM_CLC **15-9 [1]**
 STM_CMCON **15-16 [1]**
 STM_CMPx **15-15 [1]**
 STM_ICR **15-18 [1]**
 STM_ID **15-10 [1]**
 STM_ISRR **15-20 [1]**
 STM_SRCx **15-21 [1]**
 STM_TIM0 **15-11 [1]**
 STM_TIM1 **15-11 [1]**
 STM_TIM2 **15-12 [1]**
 STM_TIM3 **15-12 [1]**
 STM_TIM4 **15-12 [1]**
 STM_TIM5 **15-13 [1]**
 STM_TIM6 **15-13 [1]**
 Resolutions and ranges 15-4 [1]

 in OCDS suspend mode 16-17 [1]
 Modify access to WDT_CON0
 16-11 [1]
 Monitoring diagram 16-26 [1]
 Operating modes 16-7 [1]
 Disable mode 16-8 [1], 16-15 [1]
 Normal mode 16-8 [1], 16-14 [1]
 Prewarning mode 16-9 [1],
 16-16 [1]
 Time-out mode 16-8 [1], 16-13 [1]
 Operation sequence example 16-5 [1]
 Overview 16-1 [1]
 Period calculation 16-18 [1]
 Period in power-saving modes
 16-21 [1]
 Registers 16-28 [1]
 Offset addresses 16-28 [1]
 WDT_CON0 **16-29 [1]**
 WDT_CON1 **16-31 [1]**
 WDT_SR **16-32 [1]**
 Service sequence diagram 16-25 [1]
 Servicing 16-23 [1]
 System initialization 16-22 [1]
 Time-out period 16-18 [1]
 WDT, see "Watchdog timer"

T

Temperature compensation 5-41 [1]
 Block diagram 5-42 [1]
 Registers
 SCU_TCCLR1 **5-47 [1]**
 SCU_TCCON **5-44 [1]**
 SCU_TCLR0 **5-46 [1]**
 Switching thresholds 5-43 [1]
 TTCAN, see "CAN"

W

Watchdog timer 16-1 [1]–??
 During power-saving modes 16-17 [1]
 Endinit function 16-3 [1]
 Features 16-2 [1]
 Functional description 16-5 [1]

Register Index

This section lists the references to the Special Function Registers of the TC1796. It refers to page numbers in both parts of the TC1796 User's Manual, the "System Units" (volume 1 with marking "[1]") and the "Peripheral Units" (volume 2 with marking "[2]") parts.

A

- A0 18-110 [1]
- A1 18-110 [1]
- A10 18-110 [1]
- A11 18-111 [1]
- A12 18-111 [1]
- A13 18-111 [1]
- A14 18-111 [1]
- A15 18-111 [1]
- A2 18-110 [1]
- A3 18-110 [1]
- A4 18-110 [1]
- A5 18-110 [1]
- A6 18-110 [1]
- A7 18-110 [1]
- A8 18-110 [1]
- A9 18-110 [1]
- ADC0 register address map 18-87 [1]
- ADC0_AP 18-88 [1], 25-78 [2]
- ADC0_ASCRP 18-90 [1], 25-77 [2]
- ADC0_CHCON0 18-87 [1]
- ADC0_CHCON1 18-87 [1]
- ADC0_CHCON10 18-87 [1]
- ADC0_CHCON11 18-87 [1]
- ADC0_CHCON12 18-88 [1]
- ADC0_CHCON13 18-88 [1]
- ADC0_CHCON14 18-88 [1]
- ADC0_CHCON15 18-88 [1]
- ADC0_CHCON2 18-87 [1]
- ADC0_CHCON3 18-87 [1]
- ADC0_CHCON4 18-87 [1]
- ADC0_CHCON5 18-87 [1]
- ADC0_CHCON6 18-87 [1]
- ADC0_CHCON7 18-87 [1]
- ADC0_CHCON8 18-87 [1]
- ADC0_CHCON9 18-87 [1]
- ADC0_CHCONm 25-60 [2]
- ADC0_CHCONn 25-60 [2]
- ADC0_CHIN 18-89 [1], 25-90 [2]
- ADC0_CHSTAT0 18-89 [1]
- ADC0_CHSTAT1 18-89 [1]
- ADC0_CHSTAT10 18-90 [1]
- ADC0_CHSTAT11 18-90 [1]
- ADC0_CHSTAT12 18-90 [1]
- ADC0_CHSTAT13 18-90 [1]
- ADC0_CHSTAT14 18-90 [1]
- ADC0_CHSTAT15 18-90 [1]
- ADC0_CHSTAT2 18-89 [1]
- ADC0_CHSTAT3 18-89 [1]
- ADC0_CHSTAT4 18-89 [1]
- ADC0_CHSTAT5 18-89 [1]
- ADC0_CHSTAT6 18-89 [1]
- ADC0_CHSTAT7 18-89 [1]
- ADC0_CHSTAT8 18-89 [1]
- ADC0_CHSTAT9 18-90 [1]
- ADC0_CHSTATm 25-63 [2]
- ADC0_CLC 18-87 [1], 25-105 [2]
- ADC0_CON 18-89 [1], 25-82 [2]
- ADC0_EXCRP 18-91 [1], 25-74 [2]
- ADC0_EXTC 18-88 [1], 25-73 [2]
- ADC0_FDR 18-87 [1], 25-106 [2]
- ADC0_ID 18-87 [1], 25-59 [2]
- ADC0_LCCON0 18-88 [1]
- ADC0_LCCON1 18-88 [1]
- ADC0_LCCON2 18-88 [1]
- ADC0_LCCON3 18-89 [1]
- ADC0_LCCONk 25-80 [2]
- ADC0_MSS0 18-91 [1], 25-94 [2]

ADC0_MSS1 18-91 [1], 25-95 [2]	ADC1_CHSTAT1 18-94 [1]
ADC0_QR 18-89 [1], 25-71 [2]	ADC1_CHSTAT10 18-94 [1]
ADC0_QUEUE0 18-90 [1], 25-69 [2]	ADC1_CHSTAT11 18-95 [1]
ADC0_REQ0 18-89 [1], 25-92 [2]	ADC1_CHSTAT12 18-95 [1]
ADC0_SAL 18-88 [1], 25-79 [2]	ADC1_CHSTAT13 18-95 [1]
ADC0_SCN 18-89 [1], 25-75 [2]	ADC1_CHSTAT14 18-95 [1]
ADC0_SCON 18-88 [1], 25-81 [2]	ADC1_CHSTAT15 18-95 [1]
ADC0_SRC0 18-91 [1]	ADC1_CHSTAT2 18-94 [1]
ADC0_SRC1 18-91 [1]	ADC1_CHSTAT3 18-94 [1]
ADC0_SRC2 18-91 [1]	ADC1_CHSTAT4 18-94 [1]
ADC0_SRC3 18-91 [1]	ADC1_CHSTAT5 18-94 [1]
ADC0_SRCn 25-120 [2]	ADC1_CHSTAT6 18-94 [1]
ADC0_SRNP 18-91 [1], 25-97 [2]	ADC1_CHSTAT7 18-94 [1]
ADC0_STAT 18-91 [1], 25-86 [2]	ADC1_CHSTAT8 18-94 [1]
ADC0_SW0CRP 18-90 [1], 25-93 [2]	ADC1_CHSTAT9 18-94 [1]
ADC0_SYSTAT 18-90 [1], 25-84 [2]	ADC1_CHSTATm 25-63 [2]
ADC0_TCON 18-89 [1], 25-66 [2]	ADC1_CON 18-94 [1], 25-82 [2]
ADC0_TCRP 18-91 [1], 25-68 [2]	ADC1_EXCRP 18-96 [1], 25-74 [2]
ADC0_TSTAT 18-91 [1], 25-67 [2]	ADC1_EXTC 18-93 [1], 25-73 [2]
ADC0_TTC 18-88 [1], 25-65 [2]	ADC1_LCCON0 18-93 [1]
ADC1 register address map 18-92 [1]	ADC1_LCCON1 18-93 [1]
ADC1_AP 18-93 [1], 25-78 [2]	ADC1_LCCON2 18-93 [1]
ADC1_ASCRP 18-95 [1], 25-77 [2]	ADC1_LCCON3 18-93 [1]
ADC1_CHCON0 18-92 [1]	ADC1_LCCONk 25-80 [2]
ADC1_CHCON1 18-92 [1]	ADC1_MSS0 18-96 [1], 25-94 [2]
ADC1_CHCON10 18-92 [1]	ADC1_MSS1 18-96 [1], 25-95 [2]
ADC1_CHCON11 18-92 [1]	ADC1_QR 18-94 [1], 25-71 [2]
ADC1_CHCON12 18-92 [1]	ADC1_QUEUE0 18-95 [1], 25-69 [2]
ADC1_CHCON13 18-92 [1]	ADC1_REQ0 18-94 [1], 25-92 [2]
ADC1_CHCON14 18-93 [1]	ADC1_SAL 18-93 [1], 25-79 [2]
ADC1_CHCON15 18-93 [1]	ADC1_SCN 18-94 [1], 25-75 [2]
ADC1_CHCON2 18-92 [1]	ADC1_SCON 18-93 [1], 25-81 [2]
ADC1_CHCON3 18-92 [1]	ADC1_SRC0 18-96 [1]
ADC1_CHCON4 18-92 [1]	ADC1_SRC1 18-96 [1]
ADC1_CHCON5 18-92 [1]	ADC1_SRC2 18-96 [1]
ADC1_CHCON6 18-92 [1]	ADC1_SRC3 18-96 [1]
ADC1_CHCON7 18-92 [1]	ADC1_SRCn 25-120 [2]
ADC1_CHCON8 18-92 [1]	ADC1_SRNP 18-96 [1], 25-97 [2]
ADC1_CHCON9 18-92 [1]	ADC1_STAT 18-96 [1], 25-86 [2]
ADC1_CHCONm 25-60 [2]	ADC1_SW0CRP 18-95 [1], 25-93 [2]
ADC1_CHCONn 25-60 [2]	ADC1_SYSTAT 18-95 [1], 25-84 [2]
ADC1_CHIN 18-94 [1], 25-90 [2]	ADC1_TCON 18-93 [1], 25-66 [2]
ADC1_CHSTAT0 18-94 [1]	ADC1_TCRP 18-96 [1], 25-68 [2]

- ADC1_TSTAT 18-95 [1], 25-67 [2]
 - ADC1_TTC 18-93 [1], 25-65 [2]
 - ASC module kernel registers 19-19 [2]
 - ASC0 register address map 18-20 [1]
 - ASC0_BG 18-20 [1], 19-27 [2]
 - ASC0_CLC 18-20 [1], 19-32 [2]
 - ASC0_CON 18-20 [1], 19-22 [2]
 - ASC0_ESRC 18-20 [1], 19-41 [2]
 - ASC0_FDVB 18-20 [1], 19-27 [2]
 - ASC0_ID 18-20 [1], 19-20 [2]
 - ASC0_PISEL 18-20 [1], 19-35 [2]
 - ASC0_RBUF 18-20 [1], 19-29 [2]
 - ASC0_RSRC 18-20 [1], 19-41 [2]
 - ASC0_TB SRC 18-20 [1], 19-41 [2]
 - ASC0_TBUF 18-20 [1], 19-28 [2]
 - ASC0_TSRC 18-20 [1], 19-41 [2]
 - ASC0_WHBCON 18-20 [1], 19-25 [2]
 - ASC1 register address map 18-21 [1]
 - ASC1_BG 18-21 [1], 19-27 [2]
 - ASC1_CON 18-21 [1], 19-22 [2]
 - ASC1_ESRC 18-21 [1], 19-41 [2]
 - ASC1_FDVB 18-21 [1], 19-27 [2]
 - ASC1_ID 18-21 [1], 19-20 [2]
 - ASC1_PISEL 18-21 [1], 19-35 [2]
 - ASC1_RBUF 18-21 [1], 19-29 [2]
 - ASC1_RSRC 18-21 [1], 19-41 [2]
 - ASC1_TB SRC 18-21 [1], 19-41 [2]
 - ASC1_TBUF 18-21 [1], 19-28 [2]
 - ASC1_TSRC 18-21 [1], 19-41 [2]
 - ASC1_WHBCON 18-21 [1], 19-25 [2]
- B**
- BCU module registers 6-34 [1]
 - BIV 2-10 [1], 18-109 [1]
 - BTV 2-10 [1], 18-109 [1]
- C**
- CAN module registers 22-59 [2]
 - CAN register address map 18-65 [1]
 - CAN_AWDR 18-69 [1], 22-168 [2]
 - CAN_CLC 18-65 [1], 22-202 [2]
 - CAN_CYCTMR 18-69 [1], 22-163 [2]
 - CAN_FDR 18-65 [1], 22-203 [2]
 - CAN_GMR 18-69 [1], 22-166 [2]
 - CAN_ID 18-65 [1], 22-61 [2]
 - CAN_LGMR 18-69 [1], 22-167 [2]
 - CAN_LIST0 18-66 [1]
 - CAN_LIST1 18-66 [1]
 - CAN_LIST2 18-66 [1]
 - CAN_LIST3 18-66 [1]
 - CAN_LIST4 18-66 [1]
 - CAN_LIST5 18-66 [1]
 - CAN_LIST6 18-66 [1]
 - CAN_LIST7 18-66 [1]
 - CAN_LISTi 22-68 [2]
 - CAN_LOR 18-69 [1], 22-164 [2]
 - CAN_LREFMR 18-68 [1], 22-162 [2]
 - CAN_LTR 18-68 [1], 22-159 [2]
 - CAN_MCR 18-68 [1], 22-66 [2]
 - CAN_MITR 18-68 [1], 22-67 [2]
 - CAN_MOAMR0 18-72 [1]
 - CAN_MOAMR1 18-72 [1]
 - CAN_MOAMR126 18-74 [1]
 - CAN_MOAMR127 18-75 [1]
 - CAN_MOAMR2 18-73 [1]
 - CAN_MOAMRn 18-74 [1], 22-107 [2]
 - CAN_MOAR0 18-72 [1]
 - CAN_MOAR1 18-72 [1]
 - CAN_MOAR126 18-75 [1]
 - CAN_MOAR127 18-75 [1]
 - CAN_MOAR2 18-73 [1]
 - CAN_MOARn 18-74 [1], 22-109 [2]
 - CAN_MOCTR0 18-72 [1]
 - CAN_MOCTR1 18-73 [1]
 - CAN_MOCTR126 18-75 [1]
 - CAN_MOCTR127 18-76 [1]
 - CAN_MOCTR2 18-73 [1]
 - CAN_MOCTRn 18-74 [1], 22-91 [2]
 - CAN_MODATA00 18-72 [1]
 - CAN_MODATA04 18-72 [1]
 - CAN_MODATA10 18-72 [1]
 - CAN_MODATA1260 18-74 [1]
 - CAN_MODATA1264 18-75 [1]
 - CAN_MODATA1270 18-75 [1]
 - CAN_MODATA1274 18-75 [1]
 - CAN_MODATA14 18-72 [1]

CAN_MODATA20	18-73 [1]	CAN_MSPND3	18-67 [1]
CAN_MODATA24	18-73 [1]	CAN_MSPND4	18-67 [1]
CAN_MODATAHn	22-113 [2]	CAN_MSPND5	18-67 [1]
CAN_MODALn	22-112 [2]	CAN_MSPND6	18-67 [1]
CAN_MODALn0	18-74 [1]	CAN_MSPND7	18-67 [1]
CAN_MODALn4	18-74 [1]	CAN_MSPNDk	22-70 [2]
CAN_MOFCR0	18-71 [1]	CAN_NBTR0	18-68 [1]
CAN_MOFCR1	18-72 [1]	CAN_NBTR1	18-70 [1]
CAN_MOFCR126	18-74 [1]	CAN_NBTR2	18-70 [1]
CAN_MOFCR127	18-75 [1]	CAN_NBTR3	18-71 [1]
CAN_MOFCR2	18-73 [1]	CAN_NBTRx	22-84 [2]
CAN_MOFCRn	18-73 [1], 22-101 [2]	CAN_NCR0	18-68 [1]
CAN_MOFGPR0	18-71 [1]	CAN_NCR1	18-70 [1]
CAN_MOFGPR1	18-72 [1]	CAN_NCR2	18-70 [1]
CAN_MOFGPR126	18-74 [1]	CAN_NCR3	18-71 [1]
CAN_MOFGPR127	18-75 [1]	CAN_NCRx	22-73 [2]
CAN_MOFGPR2	18-73 [1]	CAN_NECNT0	18-68 [1]
CAN_MOFGPRn	18-73 [1], 22-106 [2]	CAN_NECNT1	18-70 [1]
CAN_MOIPR0	18-71 [1]	CAN_NECNT2	18-71 [1]
CAN_MOIPR1	18-72 [1]	CAN_NECNT3	18-71 [1]
CAN_MOIPR126	18-74 [1]	CAN_NECNTx	22-86 [2]
CAN_MOIPR127	18-75 [1]	CAN_NFCR0	18-68 [1]
CAN_MOIPR2	18-73 [1]	CAN_NFCR1	18-70 [1]
CAN_MOIPRn	18-74 [1], 22-99 [2]	CAN_NFCR2	18-71 [1]
CAN_MOSTAT0	18-72 [1]	CAN_NFCR3	18-71 [1]
CAN_MOSTAT1	18-73 [1]	CAN_NFCRx	22-87 [2]
CAN_MOSTAT126	18-75 [1]	CAN_NIPR0	18-68 [1]
CAN_MOSTAT127	18-76 [1]	CAN_NIPR1	18-70 [1]
CAN_MOSTAT2	18-73 [1]	CAN_NIPR2	18-70 [1]
CAN_MOSTATn	18-74 [1], 22-94 [2]	CAN_NIPR3	18-71 [1]
CAN_MSID0	18-67 [1]	CAN_NIPRx	22-81 [2]
CAN_MSID1	18-67 [1]	CAN_NPCR0	18-68 [1]
CAN_MSID2	18-67 [1]	CAN_NPCR1	18-70 [1]
CAN_MSID3	18-67 [1]	CAN_NPCR2	18-70 [1]
CAN_MSID4	18-67 [1]	CAN_NPCR3	18-71 [1]
CAN_MSID5	18-67 [1]	CAN_NPCRx	22-83 [2]
CAN_MSID6	18-67 [1]	CAN_NSR0	18-68 [1]
CAN_MSID7	18-67 [1]	CAN_NSR1	18-70 [1]
CAN_MSIDk	22-71 [2]	CAN_NSR2	18-70 [1]
CAN_MSIMASK	18-67 [1], 22-72 [2]	CAN_NSR3	18-71 [1]
CAN_MSPND0	18-66 [1]	CAN_NSRx	22-77 [2]
CAN_MSPND1	18-66 [1]	CAN_PANCTR	18-67 [1], 22-62 [2]
CAN_MSPND2	18-66 [1]	CAN_REFMR	18-68 [1], 22-161 [2]

CAN_SISR 18-70 [1], 22-195 [2]
 CAN_SRC0 18-66 [1]
 CAN_SRC1 18-66 [1]
 CAN_SRC10 18-65 [1]
 CAN_SRC11 18-65 [1]
 CAN_SRC12 18-65 [1]
 CAN_SRC13 18-65 [1]
 CAN_SRC14 18-65 [1]
 CAN_SRC15 18-65 [1]
 CAN_SRC2 18-66 [1]
 CAN_SRC3 18-66 [1]
 CAN_SRC4 18-66 [1]
 CAN_SRC5 18-65 [1]
 CAN_SRC6 18-65 [1]
 CAN_SRC7 18-65 [1]
 CAN_SRC8 18-65 [1]
 CAN_SRC9 18-65 [1]
 CAN_SRCm 22-212 [2]
 CAN_STPTR0 22-192 [2]
 CAN_STSRH 18-70 [1], 22-194 [2]
 CAN_STSRL 18-69 [1], 22-193 [2]
 CAN_SYNMR 18-68 [1], 22-160 [2]
 CAN_TTCFGR 18-69 [1], 22-173 [2]
 CAN_TTCR 18-69 [1], 22-169 [2]
 CAN_TTFMR 18-69 [1], 22-180 [2]
 CAN_TTIER 18-69 [1], 22-186 [2]
 CAN_TTINPR 18-69 [1], 22-190 [2]
 CAN_TTIRR 18-69 [1], 22-182 [2]
 CAN_TTSR 18-69 [1], 22-175 [2]
 CAN_TURR 18-69 [1], 22-157 [2]
 CHIPID 5-70 [1], 18-8 [1]
 Core debug register address map
 18-108 [1]
 CPM0 2-22 [1], 2-23 [1], 18-107 [1]
 CPM1 2-22 [1], 2-23 [1], 18-108 [1]
 CPMx 2-23 [1]
 CPR0_0L 2-22 [1], 18-106 [1]
 CPR0_0U 2-22 [1], 18-106 [1]
 CPR0_1L 2-22 [1], 18-107 [1]
 CPR0_1U 2-22 [1], 18-107 [1]
 CPR1_0L 2-22 [1], 18-107 [1]
 CPR1_0U 2-22 [1], 18-107 [1]
 CPR1_1L 2-22 [1], 18-107 [1]

CPR1_1U 2-22 [1], 18-107 [1]
 CPS register address map 18-104 [1]
 CPS_ID 2-13 [1], 18-104 [1]
 CPU_ID 2-10 [1], 18-109 [1]
 CPU_SBSRC 2-13 [1]
 CPU_SBSRC0 2-19 [1], 18-104 [1]
 CPU_SRC0 18-104 [1]
 CPU_SRC1 18-104 [1]
 CPU_SRC2 18-104 [1]
 CPU_SRC3 18-104 [1]
 CPU_SRCn 2-13 [1], 2-14 [1]
 CREVT 18-108 [1]
 CSFR register address map 18-108 [1]

D

D0 18-109 [1]
 D1 18-109 [1]
 D10 18-110 [1]
 D11 18-110 [1]
 D12 18-110 [1]
 D13 18-110 [1]
 D14 18-110 [1]
 D15 18-110 [1]
 D2 18-109 [1]
 D3 18-110 [1]
 D4 18-110 [1]
 D5 18-110 [1]
 D6 18-110 [1]
 D7 18-110 [1]
 D8 18-110 [1]
 D9 18-110 [1]
 DBCU register address map 18-122 [1]
 DBCU_ID 6-8 [1], 18-122 [1]
 DBCU_LEADDR 6-12 [1], 18-122 [1]
 DBCU_LEATT 6-9 [1], 18-122 [1]
 DBCU_LEDATH 6-13 [1], 18-122 [1]
 DBCU_LEDATL 6-12 [1], 18-122 [1]
 DBCU_SRC 6-13 [1], 18-122 [1]
 DBGSR 18-108 [1]
 DCX 18-108 [1]
 DMA module registers 12-43 [1]
 DMA register address map 18-53 [1]
 DMA_ADRCR00 18-54 [1]

Register Index

DMA_ADRCR01	18-55 [1]	DMA_CHICR0x	12-81 [1]
DMA_ADRCR02	18-55 [1]	DMA_CHICR10	18-58 [1]
DMA_ADRCR03	18-56 [1]	DMA_CHICR11	18-59 [1]
DMA_ADRCR04	18-56 [1]	DMA_CHICR12	18-59 [1]
DMA_ADRCR05	18-57 [1]	DMA_CHICR13	18-60 [1]
DMA_ADRCR06	18-57 [1]	DMA_CHICR14	18-60 [1]
DMA_ADRCR07	18-58 [1]	DMA_CHICR15	18-61 [1]
DMA_ADRCR0x	12-83 [1]	DMA_CHICR16	18-61 [1]
DMA_ADRCR10	18-58 [1]	DMA_CHICR17	18-62 [1]
DMA_ADRCR11	18-59 [1]	DMA_CHICR1x	12-81 [1]
DMA_ADRCR12	18-59 [1]	DMA_CHRSTR	12-51 [1], 18-53 [1]
DMA_ADRCR13	18-60 [1]	DMA_CHSR00	18-54 [1]
DMA_ADRCR14	18-60 [1]	DMA_CHSR01	18-55 [1]
DMA_ADRCR15	18-61 [1]	DMA_CHSR02	18-55 [1]
DMA_ADRCR16	18-61 [1]	DMA_CHSR03	18-56 [1]
DMA_ADRCR17	18-62 [1]	DMA_CHSR04	18-56 [1]
DMA_ADRCR1x	12-83 [1]	DMA_CHSR05	18-57 [1]
DMA_CHCR00	18-54 [1]	DMA_CHSR06	18-57 [1]
DMA_CHCR01	18-55 [1]	DMA_CHSR07	18-58 [1]
DMA_CHCR02	18-55 [1]	DMA_CHSR0x	12-80 [1]
DMA_CHCR03	18-56 [1]	DMA_CHSR10	18-58 [1]
DMA_CHCR04	18-56 [1]	DMA_CHSR11	18-59 [1]
DMA_CHCR05	18-57 [1]	DMA_CHSR12	18-59 [1]
DMA_CHCR06	18-57 [1]	DMA_CHSR13	18-60 [1]
DMA_CHCR07	18-58 [1]	DMA_CHSR14	18-60 [1]
DMA_CHCR0x	12-76 [1]	DMA_CHSR15	18-61 [1]
DMA_CHCR10	18-58 [1]	DMA_CHSR16	18-61 [1]
DMA_CHCR11	18-59 [1]	DMA_CHSR17	18-62 [1]
DMA_CHCR12	18-59 [1]	DMA_CHSR1x	12-80 [1]
DMA_CHCR13	18-60 [1]	DMA_CLC	12-105 [1], 18-53 [1]
DMA_CHCR14	18-60 [1]	DMA_CLRE	12-62 [1], 18-53 [1]
DMA_CHCR15	18-61 [1]	DMA_DADR00	18-55 [1]
DMA_CHCR16	18-61 [1]	DMA_DADR01	18-55 [1]
DMA_CHCR17	18-62 [1]	DMA_DADR02	18-56 [1]
DMA_CHCR1x	12-76 [1]	DMA_DADR03	18-56 [1]
DMA_CHICR00	18-54 [1]	DMA_DADR04	18-57 [1]
DMA_CHICR01	18-55 [1]	DMA_DADR05	18-57 [1]
DMA_CHICR02	18-55 [1]	DMA_DADR06	18-58 [1]
DMA_CHICR03	18-56 [1]	DMA_DADR07	18-58 [1]
DMA_CHICR04	18-56 [1]	DMA_DADR0x	12-89 [1]
DMA_CHICR05	18-57 [1]	DMA_DADR10	18-59 [1]
DMA_CHICR06	18-57 [1]	DMA_DADR11	18-59 [1]
DMA_CHICR07	18-58 [1]	DMA_DADR12	18-60 [1]

DMA_DADR13	18-60 [1]	DMA_SADR15	18-61 [1]
DMA_DADR14	18-61 [1]	DMA_SADR16	18-61 [1]
DMA_DADR15	18-61 [1]	DMA_SADR17	18-62 [1]
DMA_DADR16	18-62 [1]	DMA_SADR1x	12-88 [1]
DMA_DADR17	18-62 [1]	DMA_SHADR00	18-55 [1]
DMA_DADR1x	12-89 [1]	DMA_SHADR01	18-55 [1]
DMA_EER	12-56 [1], 18-53 [1]	DMA_SHADR02	18-56 [1]
DMA_ERRSR	12-59 [1], 18-53 [1]	DMA_SHADR03	18-56 [1]
DMA_GINTR	12-50 [1], 18-53 [1]	DMA_SHADR04	18-57 [1]
DMA_HTREQ	12-55 [1], 18-53 [1]	DMA_SHADR05	18-57 [1]
DMA_ID	12-45 [1], 18-53 [1]	DMA_SHADR06	18-58 [1]
DMA_INTCR	12-67 [1], 18-54 [1]	DMA_SHADR07	18-58 [1]
DMA_INTSR	12-64 [1], 18-54 [1]	DMA_SHADR0x	12-90 [1]
DMA_ME0AENR	12-73 [1], 18-54 [1]	DMA_SHADR10	18-59 [1]
DMA_ME0ARR	12-74 [1], 18-54 [1]	DMA_SHADR11	18-59 [1]
DMA_ME0PR	12-72 [1], 18-53 [1]	DMA_SHADR12	18-60 [1]
DMA_ME0R	12-71 [1], 18-53 [1]	DMA_SHADR13	18-60 [1]
DMA_ME1AENR	12-73 [1], 18-54 [1]	DMA_SHADR14	18-61 [1]
DMA_ME1ARR	12-74 [1], 18-54 [1]	DMA_SHADR15	18-61 [1]
DMA_ME1PR	12-72 [1], 18-54 [1]	DMA_SHADR16	18-62 [1]
DMA_ME1R	12-71 [1], 18-53 [1]	DMA_SHADR17	18-62 [1]
DMA_MESR	12-69 [1], 18-53 [1]	DMA_SHADR1x	12-90 [1]
DMA_MLI0SRC0	12-107 [1], 18-63 [1]	DMA_SRC0	12-106 [1], 18-63 [1]
DMA_MLI0SRC1	18-63 [1]	DMA_SRC1	12-106 [1], 18-63 [1]
DMA_MLI0SRC2	18-63 [1]	DMA_SRC2	12-106 [1], 18-63 [1]
DMA_MLI0SRC3	18-63 [1]	DMA_SRC3	12-106 [1], 18-63 [1]
DMA_MLI1SRC0	12-107 [1], 18-63 [1]	DMA_SRC4	12-106 [1], 18-63 [1]
DMA_MLI1SRC1	18-63 [1]	DMA_SRC5	12-106 [1], 18-63 [1]
DMA_OCDSR	12-46 [1], 18-54 [1]	DMA_SRC6	12-106 [1], 18-64 [1]
DMA_SADR00	18-54 [1]	DMA_SRC7	12-106 [1], 18-64 [1]
DMA_SADR01	18-55 [1]	DMA_STREQ	12-54 [1], 18-53 [1]
DMA_SADR02	18-55 [1]	DMA_SUSPMR	12-48 [1], 18-54 [1]
DMA_SADR03	18-56 [1]	DMA_SYSSRC0	12-108 [1], 18-63 [1]
DMA_SADR04	18-56 [1]	DMA_SYSSRC1	18-63 [1]
DMA_SADR05	18-57 [1]	DMA_SYSSRC2	18-62 [1]
DMA_SADR06	18-57 [1]	DMA_SYSSRC3	18-62 [1]
DMA_SADR07	18-58 [1]	DMA_SYSSRC4	18-62 [1]
DMA_SADR0x	12-88 [1]	DMA_TRSR	12-52 [1], 18-53 [1]
DMA_SADR10	18-58 [1]	DMA_WRPSR	12-66 [1], 18-54 [1]
DMA_SADR11	18-59 [1]	DMI register address map	18-123 [1]
DMA_SADR12	18-59 [1]	DMI_ATR	2-39 [1], 18-123 [1]
DMA_SADR13	18-60 [1]	DMI_CON	2-36 [1], 18-123 [1]
DMA_SADR14	18-60 [1]	DMI_CON1	2-37 [1], 18-123 [1]

DMI_ID 2-35 [1], 18-123 [1]
 DMI_STR 2-38 [1], 18-123 [1]
 DMS 18-108 [1]
 DMU register address map 18-118 [1]
 DMU_ID 8-11 [1], 18-118 [1]
 DMU_OMASK0 18-118 [1]
 DMU_OMASK1 18-118 [1]
 DMU_OMASK10 18-120 [1]
 DMU_OMASK11 18-120 [1]
 DMU_OMASK12 18-120 [1]
 DMU_OMASK13 18-120 [1]
 DMU_OMASK14 18-121 [1]
 DMU_OMASK15 18-121 [1]
 DMU_OMASK2 18-118 [1]
 DMU_OMASK3 18-119 [1]
 DMU_OMASK4 18-119 [1]
 DMU_OMASK5 18-119 [1]
 DMU_OMASK6 18-119 [1]
 DMU_OMASK7 18-119 [1]
 DMU_OMASK8 18-120 [1]
 DMU_OMASK9 18-120 [1]
 DMU_OMASKx 8-16 [1]
 DMU_OTAR0 18-118 [1]
 DMU_OTAR1 18-118 [1]
 DMU_OTAR10 18-120 [1]
 DMU_OTAR11 18-120 [1]
 DMU_OTAR12 18-120 [1]
 DMU_OTAR13 18-120 [1]
 DMU_OTAR14 18-121 [1]
 DMU_OTAR15 18-121 [1]
 DMU_OTAR2 18-118 [1]
 DMU_OTAR3 18-119 [1]
 DMU_OTAR4 18-119 [1]
 DMU_OTAR5 18-119 [1]
 DMU_OTAR6 18-119 [1]
 DMU_OTAR7 18-119 [1]
 DMU_OTAR8 18-119 [1]
 DMU_OTAR9 18-120 [1]
 DMU_OTARx 8-15 [1]
 DMU_RABR0 18-118 [1]
 DMU_RABR1 18-118 [1]
 DMU_RABR10 18-120 [1]
 DMU_RABR11 18-120 [1]
 DMU_RABR12 18-120 [1]
 DMU_RABR13 18-120 [1]
 DMU_RABR14 18-121 [1]
 DMU_RABR15 18-121 [1]
 DMU_RABR2 18-118 [1]
 DMU_RABR3 18-118 [1]
 DMU_RABR4 18-119 [1]
 DMU_RABR5 18-119 [1]
 DMU_RABR6 18-119 [1]
 DMU_RABR7 18-119 [1]
 DMU_RABR8 18-119 [1]
 DMU_RABR9 18-120 [1]
 DMU_RABRx 8-13 [1]
 DMU_SBRCTR 8-12 [1], 18-121 [1]
 DPM0 2-22 [1], 18-107 [1]
 DPM1 2-22 [1], 18-107 [1]
 DPR0_0L 2-21 [1], 18-105 [1]
 DPR0_0U 2-21 [1], 18-105 [1]
 DPR0_1L 2-21 [1], 18-105 [1]
 DPR0_1U 2-21 [1], 18-105 [1]
 DPR0_2L 2-21 [1], 18-105 [1]
 DPR0_2U 2-21 [1], 18-105 [1]
 DPR0_3L 2-21 [1], 18-105 [1]
 DPR0_3U 2-21 [1], 18-105 [1]
 DPR1_0L 2-21 [1], 18-106 [1]
 DPR1_0U 2-21 [1], 18-106 [1]
 DPR1_1L 2-21 [1], 18-106 [1]
 DPR1_1U 2-21 [1], 18-106 [1]
 DPR1_2L 2-21 [1], 18-106 [1]
 DPR1_2U 2-21 [1], 18-106 [1]
 DPR1_3L 2-21 [1], 18-106 [1]
 DPR1_3U 2-21 [1], 18-106 [1]

E

EBU register address map 18-112 [1]
 EBU_ADDRSEL0 13-90 [1], 18-113 [1]
 EBU_ADDRSEL1 13-90 [1], 18-113 [1]
 EBU_ADDRSEL2 13-90 [1], 18-113 [1]
 EBU_ADDRSEL3 13-90 [1], 18-113 [1]
 EBU_BFCON 13-86 [1], 18-112 [1]
 EBU_BUSAP0 13-97 [1], 18-114 [1]
 EBU_BUSAP1 13-97 [1], 18-114 [1]
 EBU_BUSAP2 13-97 [1], 18-114 [1]

EBU_BUSAP3 13-97 [1], 18-114 [1]
 EBU_BUSCON0 13-92 [1], 18-113 [1]
 EBU_BUSCON1 13-92 [1], 18-114 [1]
 EBU_BUSCON2 13-92 [1], 18-114 [1]
 EBU_BUSCON3 13-92 [1], 18-114 [1]
 EBU_CLC 13-82 [1], 18-112 [1]
 EBU_CON 13-83 [1], 18-112 [1]
 EBU_EMUAS 13-101 [1], 18-115 [1]
 EBU_EMUBAP 13-106 [1], 18-115 [1]
 EBU_EMUBC 13-102 [1], 18-115 [1]
 EBU_EMUOVL 13-110 [1], 18-115 [1]
 EBU_ID 13-81 [1], 18-112 [1]
 EBU_USERCON 13-111 [1], 18-115 [1]
 EICR0 5-17 [1], 18-8 [1]
 EICR1 5-20 [1], 18-9 [1]
 EIFR 5-23 [1], 18-9 [1]
 EXEVT 18-108 [1]

F

FADC register address map 18-84 [1]
 FADC_ACR0 18-84 [1], 26-45 [2]
 FADC_ACR1 18-84 [1], 26-45 [2]
 FADC_ACR2 18-84 [1], 26-45 [2]
 FADC_ACR3 18-84 [1], 26-45 [2]
 FADC_CFGR0 18-84 [1], 26-41 [2]
 FADC_CFGR1 18-84 [1], 26-41 [2]
 FADC_CFGR2 18-84 [1], 26-41 [2]
 FADC_CFGR3 18-84 [1], 26-41 [2]
 FADC_CLC 18-84 [1], 26-59 [2]
 FADC_CRR0 18-85 [1], 26-51 [2]
 FADC_CRR1 18-85 [1], 26-52 [2]
 FADC_CRSR 18-84 [1], 26-30 [2]
 FADC_FCR0 18-85 [1], 26-48 [2]
 FADC_FCR1 18-85 [1], 26-48 [2]
 FADC_FDR 18-84 [1], 26-60 [2]
 FADC_FMR 18-84 [1], 26-32 [2]
 FADC_FRR0 18-85 [1], 26-55 [2]
 FADC_FRR1 18-86 [1], 26-55 [2]
 FADC_GCR 18-84 [1], 26-37 [2]
 FADC_ID 18-84 [1], 26-29 [2]
 FADC_IRR10 18-85 [1], 26-54 [2]
 FADC_IRR11 18-85 [1], 26-54 [2]
 FADC_IRR20 18-85 [1], 26-54 [2]

FADC_IRR30 18-85 [1], 26-54 [2]
 FADC_NCTR 18-84 [1], 26-34 [2]
 FADC_RCH0 18-85 [1], 26-47 [2]
 FADC_RCH1 18-85 [1], 26-47 [2]
 FADC_RCH2 18-85 [1], 26-47 [2]
 FADC_RCH3 18-85 [1], 26-47 [2]
 FADC_SRC0 18-86 [1]
 FADC_SRC1 18-86 [1]
 FADC_SRC2 18-86 [1]
 FADC_SRC3 18-86 [1]
 FADC_SRCm 26-63 [2]
 FCX 2-10 [1], 18-109 [1]
 FLASH register address map 18-117 [1]
 FLASH_FCON 7-53 [1], 18-117 [1]
 FLASH_FSR 7-43 [1], 18-117 [1]
 FLASH_ID 7-41 [1], 18-117 [1]
 FLASH_MARD 7-52 [1], 18-117 [1]
 FLASH_MARP 7-51 [1], 18-117 [1]
 FLASH_PROCON0 7-59 [1], 18-117 [1]
 FLASH_PROCON1 7-60 [1], 18-117 [1]
 FLASH_PROCON2 7-61 [1], 18-117 [1]
 FMR 5-24 [1], 18-9 [1]

G

GPR register address map 18-109 [1]
 GPTA module registers 11-53 [1],
 25-58 [2]
 GPTA_EDCTR 18-38 [1]
 GPTA0 register address map 18-33 [1]
 GPTA0_CKBCTR 18-37 [1], 24-173 [2]
 GPTA0_CLC 18-33 [1], 24-267 [2]
 GPTA0_DBGCTR 18-33 [1], 24-271 [2]
 GPTA0_DCMCAV0 18-35 [1]
 GPTA0_DCMCAV1 18-36 [1]
 GPTA0_DCMCAV2 18-36 [1]
 GPTA0_DCMCAV3 18-36 [1]
 GPTA0_DCMCAVk 24-163 [2]
 GPTA0_DCMCOV0 18-35 [1]
 GPTA0_DCMCOV1 18-36 [1]
 GPTA0_DCMCOV2 18-36 [1]
 GPTA0_DCMCOV3 18-36 [1]
 GPTA0_DCMCOVk 24-164 [2]
 GPTA0_DCMCTR0 18-35 [1]

GPTA0_DCMCTR1	18-35 [1]	GPTA0_LTCXRk	24-190 [2]
GPTA0_DCMCTR2	18-36 [1]	GPTA0_LTCXRn	18-38 [1]
GPTA0_DCMCTR3	18-36 [1]	GPTA0_MMXCTR00	18-38 [1], 24-260 [2]
GPTA0_DCMCTRk	24-161 [2]	GPTA0_MMXCTR01	18-38 [1], 24-260 [2]
GPTA0_DCMTIM0	18-35 [1]	GPTA0_MMXCTR10	18-38 [1], 24-261 [2]
GPTA0_DCMTIM1	18-35 [1]	GPTA0_MMXCTR11	18-38 [1], 24-261 [2]
GPTA0_DCMTIM2	18-36 [1]	GPTA0_MRACTL	18-33 [1], 24-191 [2]
GPTA0_DCMTIM3	18-36 [1]	GPTA0_MRADIN	18-34 [1], 24-193 [2]
GPTA0_DCMTIMk	24-163 [2]	GPTA0_MRADOUT	18-34 [1], 24-193 [2]
GPTA0_EDCTR	24-270 [2]	GPTA0_PDLCTR	18-35 [1], 24-159 [2]
GPTA0_FDR	18-33 [1], 24-268 [2]	GPTA0_PLLCNT	18-37 [1], 24-167 [2]
GPTA0_FPCCTR0	18-34 [1]	GPTA0_PLLCTR	18-36 [1], 24-165 [2]
GPTA0_FPCCTR1	18-34 [1]	GPTA0_PLLDTR	18-37 [1], 24-168 [2]
GPTA0_FPCCTR2	18-34 [1]	GPTA0_PLLMTI	18-36 [1], 24-166 [2]
GPTA0_FPCCTR3	18-34 [1]	GPTA0_PLLREV	18-37 [1], 24-168 [2]
GPTA0_FPCCTR4	18-35 [1]	GPTA0_PLLSTP	18-37 [1], 24-167 [2]
GPTA0_FPCCTR5	18-35 [1]	GPTA0_SRC00	18-41 [1]
GPTA0_FPCCTRk	24-156 [2]	GPTA0_SRC01	18-41 [1]
GPTA0_FPCSTAT	18-34 [1], 24-155 [2]	GPTA0_SRC02	18-40 [1]
GPTA0_FPCTIM0	18-34 [1]	GPTA0_SRC03	18-40 [1]
GPTA0_FPCTIM1	18-34 [1]	GPTA0_SRC04	18-40 [1]
GPTA0_FPCTIM2	18-34 [1]	GPTA0_SRC05	18-40 [1]
GPTA0_FPCTIM3	18-34 [1]	GPTA0_SRC06	18-40 [1]
GPTA0_FPCTIM4	18-35 [1]	GPTA0_SRC07	18-40 [1]
GPTA0_FPCTIM5	18-35 [1]	GPTA0_SRC08	18-40 [1]
GPTA0_FPCTIMk	24-158 [2]	GPTA0_SRC09	18-40 [1]
GPTA0_GTCCTRk	24-175 [2], 24-177 [2]	GPTA0_SRC10	18-40 [1]
GPTA0_GTCCTRn	18-37 [1]	GPTA0_SRC11	18-40 [1]
GPTA0_GTCTR0	18-37 [1]	GPTA0_SRC12	18-40 [1]
GPTA0_GTCTR1	18-37 [1]	GPTA0_SRC13	18-40 [1]
GPTA0_GTCTRk	24-170 [2]	GPTA0_SRC14	18-40 [1]
GPTA0_GTCXRk	24-179 [2]	GPTA0_SRC15	18-40 [1]
GPTA0_GTCXRn	18-37 [1]	GPTA0_SRC16	18-40 [1]
GPTA0_GTREV0	18-37 [1]	GPTA0_SRC17	18-39 [1]
GPTA0_GTREV1	18-37 [1]	GPTA0_SRC18	18-39 [1]
GPTA0_GTREVk	24-172 [2]	GPTA0_SRC19	18-39 [1]
GPTA0_GTTIM0	18-37 [1]	GPTA0_SRC20	18-39 [1]
GPTA0_GTTIM1	18-37 [1]	GPTA0_SRC21	18-39 [1]
GPTA0_ID	18-33 [1], 24-154 [2]	GPTA0_SRC22	18-39 [1]
GPTA0_LTCCTR63	24-188 [2]	GPTA0_SRC23	18-39 [1]
GPTA0_LTCCTRk	24-180 [2]	GPTA0_SRC24	18-39 [1]
GPTA0_LTCCTRn	18-38 [1]	GPTA0_SRC25	18-39 [1]
GPTA0_LTCXR63	24-190 [2]	GPTA0_SRC26	18-39 [1]

GPTA0_SRC27	18-39 [1]	GPTA1_FPCCTR0	18-43 [1]
GPTA0_SRC28	18-39 [1]	GPTA1_FPCCTR1	18-43 [1]
GPTA0_SRC29	18-39 [1]	GPTA1_FPCCTR2	18-43 [1]
GPTA0_SRC30	18-39 [1]	GPTA1_FPCCTR3	18-43 [1]
GPTA0_SRC31	18-39 [1]	GPTA1_FPCCTR4	18-43 [1]
GPTA0_SRC32	18-39 [1]	GPTA1_FPCCTR5	18-44 [1]
GPTA0_SRC33	18-38 [1]	GPTA1_FPCCTRk	24-156 [2]
GPTA0_SRC34	18-38 [1]	GPTA1_FPCSTAT	18-43 [1], 24-155 [2]
GPTA0_SRC35	18-38 [1]	GPTA1_FPCTIM0	18-43 [1]
GPTA0_SRC36	18-38 [1]	GPTA1_FPCTIM1	18-43 [1]
GPTA0_SRC37	18-38 [1]	GPTA1_FPCTIM2	18-43 [1]
GPTA0_SRCK	24-273 [2]	GPTA1_FPCTIM3	18-43 [1]
GPTA0_SRNR	18-33 [1], 24-215 [2]	GPTA1_FPCTIM4	18-44 [1]
GPTA0_SRSC0	18-33 [1], 24-206 [2]	GPTA1_FPCTIM5	18-44 [1]
GPTA0_SRSC1	18-33 [1], 24-209 [2]	GPTA1_FPCTIMk	24-158 [2]
GPTA0_SRSC2	18-33 [1], 24-211 [2]	GPTA1_GTCCTRk	24-175 [2], 24-177 [2]
GPTA0_SRSC3	18-33 [1], 24-213 [2]	GPTA1_GTCCTRn	18-46 [1]
GPTA0_SRSS0	18-33 [1], 24-208 [2]	GPTA1_GTCTR0	18-46 [1]
GPTA0_SRSS1	18-33 [1], 24-210 [2]	GPTA1_GTCTR1	18-46 [1]
GPTA0_SRSS2	18-33 [1], 24-212 [2]	GPTA1_GTCTRk	24-170 [2]
GPTA0_SRSS3	18-33 [1], 24-214 [2]	GPTA1_GTCXRk	24-179 [2]
GPTA1 register address map	18-42 [1]	GPTA1_GTCXRn	18-46 [1]
GPTA1_CKBCTR	18-46 [1], 24-173 [2]	GPTA1_GTREV0	18-46 [1]
GPTA1_DCMCAV0	18-44 [1]	GPTA1_GTREV1	18-46 [1]
GPTA1_DCMCAV1	18-44 [1]	GPTA1_GTREVK	24-172 [2]
GPTA1_DCMCAV2	18-45 [1]	GPTA1_GTTIM0	18-46 [1]
GPTA1_DCMCAV3	18-45 [1]	GPTA1_GTTIM1	18-46 [1]
GPTA1_DCMCAVk	24-163 [2]	GPTA1_GTTIMk	24-171 [2]
GPTA1_DCMCOV0	18-44 [1]	GPTA1_ID	18-42 [1], 24-154 [2]
GPTA1_DCMCOV1	18-45 [1]	GPTA1_LTCCTR63	24-188 [2]
GPTA1_DCMCOV2	18-45 [1]	GPTA1_LTCCTRk	24-180 [2]
GPTA1_DCMCOV3	18-45 [1]	GPTA1_LTCCTRn	18-46 [1]
GPTA1_DCMCOVk	24-164 [2]	GPTA1_LTCXR63	24-190 [2]
GPTA1_DCMCTR0	18-44 [1]	GPTA1_LTCXRk	24-190 [2]
GPTA1_DCMCTR1	18-44 [1]	GPTA1_LTCXRn	18-46 [1]
GPTA1_DCMCTR2	18-45 [1]	GPTA1_MRACTL	18-42 [1], 24-191 [2]
GPTA1_DCMCTR3	18-45 [1]	GPTA1_MRADIN	18-42 [1], 24-193 [2]
GPTA1_DCMCTRk	24-161 [2]	GPTA1_MRADOUT	18-43 [1], 24-193 [2]
GPTA1_DCMTIM0	18-44 [1]	GPTA1_PDLCTR	18-44 [1], 24-159 [2]
GPTA1_DCMTIM1	18-44 [1]	GPTA1_PLLCNT	18-45 [1], 24-167 [2]
GPTA1_DCMTIM2	18-45 [1]	GPTA1_PLLCTR	18-45 [1], 24-165 [2]
GPTA1_DCMTIM3	18-45 [1]	GPTA1_PLLDTR	18-46 [1], 24-168 [2]
GPTA1_DCMTIMk	24-163 [2]	GPTA1_PLLMTI	18-45 [1], 24-166 [2]

GPTA1_PLLREV 18-46 [1], 24-168 [2]
 GPTA1_PLLSTP 18-45 [1], 24-167 [2]
 GPTA1_SRC00 18-49 [1]
 GPTA1_SRC01 18-49 [1]
 GPTA1_SRC02 18-49 [1]
 GPTA1_SRC03 18-49 [1]
 GPTA1_SRC04 18-49 [1]
 GPTA1_SRC05 18-49 [1]
 GPTA1_SRC06 18-49 [1]
 GPTA1_SRC07 18-48 [1]
 GPTA1_SRC08 18-48 [1]
 GPTA1_SRC09 18-48 [1]
 GPTA1_SRC10 18-48 [1]
 GPTA1_SRC11 18-48 [1]
 GPTA1_SRC12 18-48 [1]
 GPTA1_SRC13 18-48 [1]
 GPTA1_SRC14 18-48 [1]
 GPTA1_SRC15 18-48 [1]
 GPTA1_SRC16 18-48 [1]
 GPTA1_SRC17 18-48 [1]
 GPTA1_SRC18 18-48 [1]
 GPTA1_SRC19 18-48 [1]
 GPTA1_SRC20 18-48 [1]
 GPTA1_SRC21 18-48 [1]
 GPTA1_SRC22 18-48 [1]
 GPTA1_SRC23 18-47 [1]
 GPTA1_SRC24 18-47 [1]
 GPTA1_SRC25 18-47 [1]
 GPTA1_SRC26 18-47 [1]
 GPTA1_SRC27 18-47 [1]
 GPTA1_SRC28 18-47 [1]
 GPTA1_SRC29 18-47 [1]
 GPTA1_SRC30 18-47 [1]
 GPTA1_SRC31 18-47 [1]
 GPTA1_SRC32 18-47 [1]
 GPTA1_SRC33 18-47 [1]
 GPTA1_SRC34 18-47 [1]
 GPTA1_SRC35 18-47 [1]
 GPTA1_SRC36 18-47 [1]
 GPTA1_SRC37 18-47 [1]
 GPTA1_SRCk 24-273 [2]
 GPTA1_SRNR 18-42 [1], 24-215 [2]
 GPTA1_SRSC0 18-42 [1], 24-206 [2]

GPTA1_SRSC1 18-42 [1], 24-209 [2]
 GPTA1_SRSC2 18-42 [1], 24-211 [2]
 GPTA1_SRSC3 18-42 [1], 24-213 [2]
 GPTA1_SRSS0 18-42 [1], 24-208 [2]
 GPTA1_SRSS1 18-42 [1], 24-210 [2]
 GPTA1_SRSS2 18-42 [1], 24-212 [2]
 GPTA1_SRSS3 18-42 [1], 24-214 [2]

I

ICR 2-10 [1], 14-8 [1], 18-109 [1]
 IGCR0 5-26 [1], 18-9 [1]
 IGCR1 5-29 [1], 18-9 [1]
 ISP 2-10 [1], 18-109 [1]

L

LCX 2-10 [1], 18-109 [1]
 LFI_CON 6-17 [1], 18-126 [1]
 LFI_ID 6-17 [1], 18-126 [1]
 LFI-bridge register address map
 18-126 [1]
 LTCA2 register address map 18-50 [1]
 LTCA2_ID 18-50 [1], 24-235 [2]
 LTCA2_LIMCRHg 24-245 [2]
 LTCA2_LIMCRLg 24-243 [2]
 LTCA2_LTCCTR63 24-188 [2]
 LTCA2_LTCCTRk 24-180 [2]
 LTCA2_LTCCTRn 18-51 [1]
 LTCA2_LTCXR63 24-190 [2]
 LTCA2_LTCXRk 24-190 [2]
 LTCA2_LTCXRn 18-51 [1]
 LTCA2_MRACTL 18-50 [1], 24-236 [2]
 LTCA2_MRADIN 18-50 [1], 24-238 [2]
 LTCA2_MRADOUT 18-50 [1], 24-238 [2]
 LTCA2_OMCRHg 24-241 [2]
 LTCA2_OMCRLg 24-239 [2]
 LTCA2_SRC00 18-52 [1]
 LTCA2_SRC01 18-52 [1]
 LTCA2_SRC02 18-52 [1]
 LTCA2_SRC03 18-52 [1]
 LTCA2_SRC04 18-52 [1]
 LTCA2_SRC05 18-52 [1]
 LTCA2_SRC06 18-52 [1]
 LTCA2_SRC07 18-52 [1]

LTCA2_SRC08 18-51 [1]
 LTCA2_SRC09 18-51 [1]
 LTCA2_SRC10 18-51 [1]
 LTCA2_SRC11 18-51 [1]
 LTCA2_SRC12 18-51 [1]
 LTCA2_SRC13 18-51 [1]
 LTCA2_SRC14 18-51 [1]
 LTCA2_SRC15 18-51 [1]
 LTCA2_SRCK 24-273 [2]
 LTCA2_SRSC2 18-50 [1], 24-211 [2]
 LTCA2_SRSC3 18-50 [1], 24-213 [2]
 LTCA2_SRSS2 18-50 [1], 24-212 [2]
 LTCA2_SRSS3 18-50 [1], 24-214 [2]

M

MANID 5-69 [1], 18-8 [1]
 MCHK_ID 12-112 [1], 18-103 [1]
 MCHK_IR 12-113 [1], 18-103 [1]
 MCHK_RR 12-113 [1], 18-103 [1]
 MCHK_WR 12-114 [1], 18-103 [1]
 Memory checker module registers
 12-111 [1]
 MLI module registers 23-76 [2]
 MLI0 register address map 18-97 [1]
 MLI0_AER 18-99 [1], 23-88 [2]
 MLI0_ARR 18-99 [1], 23-89 [2]
 MLI0_FDR 18-97 [1], 23-79 [2]
 MLI0_GINTR 18-99 [1], 23-83 [2]
 MLI0_ID 18-97 [1], 23-78 [2]
 MLI0_OICR 18-99 [1], 23-84 [2]
 MLI0_RADRR 18-99 [1], 23-116 [2]
 MLI0_RCR 18-98 [1], 23-110 [2]
 MLI0_RDATAR 18-99 [1], 23-114 [2]
 MLI0_RIER 18-99 [1], 23-117 [2]
 MLI0_RINPR 18-99 [1], 23-122 [2]
 MLI0_RISR 18-99 [1], 23-120 [2]
 MLI0_RP0BAR 18-98 [1], 23-115 [2]
 MLI0_RP0STATR 18-98 [1], 23-113 [2]
 MLI0_RP1BAR 18-98 [1], 23-115 [2]
 MLI0_RP1STATR 18-99 [1], 23-113 [2]
 MLI0_RP2BAR 18-98 [1], 23-115 [2]
 MLI0_RP2STATR 18-99 [1], 23-113 [2]
 MLI0_RP3BAR 18-98 [1], 23-115 [2]
 MLI0_RP3STATR 18-99 [1], 23-113 [2]
 MLI0_SCR 18-99 [1], 23-81 [2]
 MLI0_TCBAR 23-104 [2]
 MLI0_TCMDR 18-97 [1], 23-97 [2]
 MLI0_TCR 18-97 [1], 23-90 [2]
 MLI0_TDRAR 18-98 [1], 23-101 [2]
 MLI0_TIER 18-99 [1], 23-105 [2]
 MLI0_TINPR 18-99 [1], 23-108 [2]
 MLI0_TISR 18-99 [1], 23-107 [2]
 MLI0_TP0AOFR 18-97 [1], 23-103 [2]
 MLI0_TP0BAR 18-98 [1], 23-102 [2]
 MLI0_TP0DATAR 18-98 [1], 23-101 [2]
 MLI0_TP0STATR 18-97 [1], 23-95 [2]
 MLI0_TP1AOFR 18-97 [1], 23-103 [2]
 MLI0_TP1BAR 18-98 [1], 23-102 [2]
 MLI0_TP1DATAR 18-98 [1], 23-101 [2]
 MLI0_TP1STATR 18-97 [1], 23-95 [2]
 MLI0_TP2AOFR 18-97 [1], 23-103 [2]
 MLI0_TP2BAR 18-98 [1], 23-102 [2]
 MLI0_TP2DATAR 18-98 [1], 23-101 [2]
 MLI0_TP2STATR 18-97 [1], 23-95 [2]
 MLI0_TP3AOFR 18-97 [1], 23-103 [2]
 MLI0_TP3BAR 18-98 [1], 23-102 [2]
 MLI0_TP3DATAR 18-98 [1], 23-101 [2]
 MLI0_TP3STATR 18-97 [1], 23-95 [2]
 MLI0_TRSTATR 18-97 [1], 23-99 [2]
 MLI0_TSTATR 18-97 [1], 23-93 [2]
 MLI1 register address map 18-100 [1]
 MLI1_AER 18-102 [1], 23-88 [2]
 MLI1_ARR 18-102 [1], 23-89 [2]
 MLI1_FDR 18-100 [1], 23-79 [2]
 MLI1_GINTR 18-102 [1], 23-83 [2]
 MLI1_ID 18-100 [1], 23-78 [2]
 MLI1_OICR 18-102 [1], 23-84 [2]
 MLI1_RADRR 18-102 [1], 23-116 [2]
 MLI1_RCR 18-101 [1], 23-110 [2]
 MLI1_RDATAR 18-102 [1], 23-114 [2]
 MLI1_RIER 18-102 [1], 23-117 [2]
 MLI1_RINPR 18-102 [1], 23-122 [2]
 MLI1_RISR 18-102 [1], 23-120 [2]
 MLI1_RP0BAR 18-101 [1], 23-115 [2]
 MLI1_RP0STATR 18-101 [1], 23-113 [2]
 MLI1_RP1BAR 18-101 [1], 23-115 [2]

MLI1_RP1STATR 18-101 [1], 23-113 [2]
 MLI1_RP2BAR 18-101 [1], 23-115 [2]
 MLI1_RP2STATR 18-102 [1], 23-113 [2]
 MLI1_RP3BAR 18-101 [1], 23-115 [2]
 MLI1_RP3STATR 18-102 [1], 23-113 [2]
 MLI1_SCR 18-102 [1], 23-81 [2]
 MLI1_TCBAR 23-104 [2]
 MLI1_TCMDR 18-100 [1], 23-97 [2]
 MLI1_TCR 18-100 [1], 23-90 [2]
 MLI1_TDRAR 18-101 [1], 23-101 [2]
 MLI1_TIER 18-102 [1], 23-105 [2]
 MLI1_TINPR 18-102 [1], 23-108 [2]
 MLI1_TISR 18-102 [1], 23-107 [2]
 MLI1_TP0AOFR 18-100 [1], 23-103 [2]
 MLI1_TP0BAR 18-101 [1], 23-102 [2]
 MLI1_TP0DATAR 18-100 [1], 23-101 [2]
 MLI1_TP0STATR 18-100 [1], 23-95 [2]
 MLI1_TP1AOFR 18-100 [1], 23-103 [2]
 MLI1_TP1BAR 18-101 [1], 23-102 [2]
 MLI1_TP1DATAR 18-101 [1], 23-101 [2]
 MLI1_TP1STATR 18-100 [1], 23-95 [2]
 MLI1_TP2AOFR 18-100 [1], 23-103 [2]
 MLI1_TP2BAR 18-101 [1], 23-102 [2]
 MLI1_TP2DATAR 18-101 [1], 23-101 [2]
 MLI1_TP2STATR 18-100 [1], 23-95 [2]
 MLI1_TP3AOFR 18-100 [1], 23-103 [2]
 MLI1_TP3BAR 18-101 [1], 23-102 [2]
 MLI1_TP3DATAR 18-101 [1], 23-101 [2]
 MLI1_TP3STATR 18-100 [1], 23-95 [2]
 MLI1_TRSTATR 18-100 [1], 23-99 [2]
 MLI1_TSTATR 18-100 [1], 23-93 [2]
 MMU register address map 18-105 [1]
 MMU_CON 2-10 [1], 2-12 [1], 18-105 [1]
 MPR register address map 18-105 [1]
 MSC module registers 21-36 [2]
 MSC0 register address map 18-16 [1]
 MSC0_CLC 18-16 [1], 21-67 [2]
 MSC0_DC 18-16 [1], 21-59 [2]
 MSC0_DD 18-16 [1], 21-59 [2]
 MSC0_DSC 18-16 [1], 21-41 [2]
 MSC0_DSDSH 18-16 [1], 21-47 [2]
 MSC0_DSDSL 18-16 [1], 21-46 [2]
 MSC0_DSS 18-16 [1], 21-44 [2]
 MSC0_ESR 18-16 [1], 21-48 [2]
 MSC0_FDR 18-16 [1], 21-68 [2]
 MSC0_ICR 18-17 [1], 21-49 [2]
 MSC0_ID 18-16 [1], 21-38 [2]
 MSC0_ISC 18-17 [1], 21-54 [2]
 MSC0_ISR 18-17 [1], 21-52 [2]
 MSC0_OCR 18-17 [1], 21-56 [2]
 MSC0_SRC0 18-17 [1], 21-78 [2]
 MSC0_SRC1 18-17 [1], 21-78 [2]
 MSC0_UD0 18-16 [1], 21-60 [2]
 MSC0_UD1 18-16 [1], 21-60 [2]
 MSC0_UD2 18-16 [1], 21-60 [2]
 MSC0_UD3 18-17 [1], 21-60 [2]
 MSC0_USR 18-16 [1], 21-39 [2]
 MSC1 register address map 18-17 [1]
 MSC1_CLC 18-17 [1], 21-67 [2]
 MSC1_DC 18-18 [1], 21-59 [2]
 MSC1_DD 18-18 [1], 21-59 [2]
 MSC1_DSC 18-17 [1], 21-41 [2]
 MSC1_DSDSH 18-18 [1], 21-47 [2]
 MSC1_DSDSL 18-18 [1], 21-46 [2]
 MSC1_DSS 18-17 [1], 21-44 [2]
 MSC1_ESR 18-18 [1], 21-48 [2]
 MSC1_FDR 18-17 [1], 21-68 [2]
 MSC1_ICR 18-18 [1], 21-49 [2]
 MSC1_ID 18-17 [1], 21-38 [2]
 MSC1_ISC 18-18 [1], 21-54 [2]
 MSC1_ISR 18-18 [1], 21-52 [2]
 MSC1_OCR 18-18 [1], 21-56 [2]
 MSC1_SRC0 18-19 [1], 21-78 [2]
 MSC1_SRC1 18-19 [1], 21-78 [2]
 MSC1_UD0 18-18 [1], 21-60 [2]
 MSC1_UD1 18-18 [1], 21-60 [2]
 MSC1_UD2 18-18 [1], 21-60 [2]
 MSC1_UD3 18-18 [1], 21-60 [2]
 MSC1_USR 18-17 [1], 21-39 [2]

N

NMISR 14-27 [1], 18-7 [1]

O

OCDS register address map 18-14 [1]
 OSC_CON 3-8 [1], 18-7 [1]

Overlay memory control registers 8-10 [1],
13-79 [1]

P

P0_IN 10-17 [1], 18-22 [1]
 P0_IOCR0 10-7 [1], 18-22 [1]
 P0_IOCR12 10-9 [1], 18-22 [1]
 P0_IOCR4 10-8 [1], 18-22 [1]
 P0_IOCR8 10-8 [1], 18-22 [1]
 P0_OMR 10-14 [1], 18-22 [1]
 P0_OUT 10-13 [1], 18-22 [1]
 P0_PDR 10-12 [1], 10-23 [1], 18-22 [1]
 P1_IN 10-17 [1], 18-23 [1]
 P1_IOCR0 10-7 [1], 18-23 [1], 23-133 [2],
26-61 [2]
 P1_IOCR12 10-9 [1], 18-23 [1], 23-135 [2]
 P1_IOCR4 10-8 [1], 18-23 [1], 23-134 [2]
 P1_IOCR8 10-8 [1], 18-23 [1], 23-134 [2]
 P1_OMR 10-14 [1], 18-23 [1]
 P1_OUT 10-13 [1], 18-23 [1]
 P1_PDR 10-31 [1], 18-23 [1], 23-137 [2]
 P10_IN 10-82 [1], 18-32 [1]
 P2_ESR 10-38 [1], 18-24 [1]
 P2_IN 10-38 [1], 18-24 [1]
 P2_IOCR0 10-38 [1], 18-24 [1], 20-55 [2]
 P2_IOCR12 10-9 [1], 18-24 [1]
 P2_IOCR4 10-8 [1], 18-24 [1], 20-55 [2]
 P2_IOCR8 10-8 [1], 18-24 [1]
 P2_OMR 10-37 [1], 18-24 [1]
 P2_OUT 10-37 [1], 18-24 [1]
 P2_PDR 10-39 [1], 18-24 [1], 20-57 [2]
 P3_ESR 10-16 [1], 18-25 [1]
 P3_IN 10-17 [1], 18-25 [1]
 P3_IOCR0 10-7 [1], 18-25 [1]
 P3_IOCR12 10-9 [1], 18-25 [1]
 P3_IOCR4 10-8 [1], 18-25 [1]
 P3_IOCR8 10-8 [1], 18-25 [1]
 P3_OMR 10-14 [1], 18-25 [1]
 P3_OUT 10-13 [1], 18-25 [1]
 P3_PDR 10-46 [1], 18-25 [1]
 P4_ESR 10-16 [1], 18-26 [1]
 P4_IN 10-17 [1], 18-26 [1]
 P4_IOCR0 10-7 [1], 18-26 [1]
 P4_IOCR12 10-9 [1], 18-26 [1]
 P4_IOCR4 10-8 [1], 18-26 [1]
 P4_IOCR8 10-8 [1], 18-26 [1]
 P4_OMR 10-14 [1], 18-26 [1]
 P4_OUT 10-13 [1], 18-26 [1]
 P4_PDR 10-53 [1], 18-26 [1]
 P5_IN 10-57 [1], 18-27 [1]
 P5_IOCR0 10-7 [1], 18-27 [1], 19-37 [2]
 P5_IOCR4 10-8 [1], 18-27 [1], 21-71 [2],
23-135 [2]
 P5_OMR 10-57 [1], 18-27 [1]
 P5_OUT 10-57 [1], 18-27 [1]
 P5_PDR 10-58 [1], 18-27 [1], 19-39 [2],
21-74 [2], 23-137 [2]
 P6_IN 10-63 [1], 18-28 [1]
 P6_IOCR12 10-9 [1], 18-28 [1], 22-206 [2]
 P6_IOCR4 10-8 [1], 18-28 [1], 20-56 [2]
 P6_IOCR8 10-8 [1], 18-28 [1], 19-38 [2],
22-206 [2]
 P6_OMR 10-63 [1], 18-28 [1]
 P6_OUT 10-63 [1], 18-28 [1]
 P6_PDR 10-64 [1], 18-28 [1], 19-39 [2],
20-57 [2], 22-208 [2]
 P7_IN 10-68 [1], 18-29 [1]
 P7_IOCR0 10-7 [1], 18-29 [1], 25-109 [2],
26-62 [2]
 P7_IOCR4 10-8 [1], 18-29 [1], 25-109 [2]
 P7_OMR 10-68 [1], 18-29 [1]
 P7_OUT 10-68 [1], 18-29 [1]
 P7_PDR 10-69 [1], 18-29 [1], 25-110 [2]
 P8_ESR 10-73 [1], 18-30 [1]
 P8_IN 10-73 [1], 18-30 [1]
 P8_IOCR0 10-7 [1], 18-30 [1], 23-136 [2]
 P8_IOCR4 10-8 [1], 18-30 [1], 23-136 [2]
 P8_OMR 10-73 [1], 18-30 [1]
 P8_OUT 10-73 [1], 18-30 [1]
 P8_PDR 10-74 [1], 18-30 [1], 23-138 [2]
 P9_ESR 10-79 [1], 18-31 [1]
 P9_IN 10-79 [1], 18-31 [1]
 P9_IOCR0 10-7 [1], 18-31 [1], 21-71 [2]
 P9_IOCR4 10-8 [1], 18-31 [1], 21-72 [2]
 P9_IOCR8 10-79 [1], 18-31 [1], 21-72 [2]
 P9_OMR 10-78 [1], 18-31 [1]

P9_OUT 10-78 [1], 18-31 [1]
P9_PDR 10-80 [1], 18-31 [1], 21-74 [2]
PBCU register address map 18-125 [1]
PBCU_ID 6-8 [1], 18-125 [1]
PBCU_LEADDR 6-12 [1], 18-125 [1]
PBCU_LEATT 6-9 [1], 18-125 [1]
PBCU_LEDATH 6-13 [1], 18-125 [1]
PBCU_LEDATL 6-12 [1], 18-125 [1]
PBCU_SRC 6-13 [1], 18-125 [1]
PC 2-10 [1], 18-109 [1]
PCP register address map 18-77 [1]
PCP_CLC 11-55 [1], 18-77 [1]
PCP_CS 11-56 [1], 18-77 [1]
PCP_ES 11-59 [1], 18-77 [1]
PCP_ICON 11-64 [1], 18-77 [1]
PCP_ICR 11-61 [1], 18-77 [1]
PCP_ID 11-54 [1], 18-77 [1]
PCP_ITR 11-63 [1], 18-77 [1]
PCP_SRC0 11-67 [1], 18-78 [1]
PCP_SRC1 11-67 [1], 18-78 [1]
PCP_SRC10 11-70 [1], 18-77 [1]
PCP_SRC11 11-70 [1], 18-77 [1]
PCP_SRC2 11-68 [1], 18-78 [1]
PCP_SRC3 11-68 [1], 18-78 [1]
PCP_SRC4 11-69 [1], 18-78 [1]
PCP_SRC5 11-69 [1], 18-78 [1]
PCP_SRC6 11-69 [1], 18-78 [1]
PCP_SRC7 11-69 [1], 18-78 [1]
PCP_SRC8 11-69 [1], 18-77 [1]
PCP_SRC9 11-70 [1], 18-77 [1]
PCP_SSR 11-66 [1], 18-77 [1]
PCXI 2-10 [1], 18-108 [1]
PDRR 5-25 [1], 18-9 [1]
PLL_CLC 3-16 [1], 18-8 [1]
PMG_CSR 5-4 [1], 18-7 [1]
PMI register address map 18-124 [1]
PMI_CON0 2-28 [1], 18-124 [1]
PMI_CON1 2-29 [1], 18-124 [1]
PMI_CON2 2-30 [1], 18-124 [1]
PMI_ID 2-27 [1], 18-124 [1]
PMU register address map 18-116 [1]
PMU_ID 7-42 [1], 18-116 [1]
Port 0 register address map 18-22 [1]

Port 1 register address map 18-23 [1]
Port 10 register address map 18-32 [1]
Port 2 register address map 18-24 [1]
Port 3 register address map 18-25 [1]
Port 4 register address map 18-26 [1]
Port 5 register address map 18-27 [1]
Port 6 register address map 18-28 [1]
Port 7 register address map 18-29 [1]
Port 8 register address map 18-30 [1]
Port 9 register address map 18-31 [1]
PSW 2-10 [1], 2-11 [1], 18-109 [1]

R

RBCU register address map 18-79 [1]
RBCU_CON 6-37 [1], 18-79 [1]
RBCU_DBADR1 6-48 [1], 18-79 [1]
RBCU_DBADR2 6-48 [1], 18-79 [1]
RBCU_DBADRT 6-54 [1], 18-79 [1]
RBCU_DBBOS 6-49 [1], 18-79 [1]
RBCU_DBBOST 6-55 [1], 18-80 [1]
RBCU_DBCNTL 6-42 [1], 18-79 [1]
RBCU_DBGNTT 6-53 [1], 18-79 [1]
RBCU_DBGRNT 6-47 [1], 18-79 [1]
RBCU_EADD 6-41 [1], 18-79 [1]
RBCU_ECON 6-39 [1], 18-79 [1]
RBCU_EDAT 6-41 [1], 18-79 [1]
RBCU_ID 6-36 [1], 18-79 [1]
RBCU_SRC 6-58 [1], 18-80 [1]
RST_REQ 4-5 [1], 18-7 [1]
RST_SR 4-3 [1], 18-7 [1]
RTID 5-71 [1], 18-8 [1]

S

SBCU_CON 6-37 [1], 18-11 [1]
SBCU_DBADR1 6-48 [1], 18-11 [1]
SBCU_DBADR2 6-48 [1], 18-11 [1]
SBCU_DBADRT 6-54 [1], 18-11 [1]
SBCU_DBBOS 6-49 [1], 18-11 [1]
SBCU_DBBOST 6-55 [1], 18-12 [1]
SBCU_DBCNTL 6-42 [1], 18-11 [1]
SBCU_DBGNTT 6-51 [1], 18-11 [1]
SBCU_DBGRNT 6-45 [1], 18-11 [1]
SBCU_EADD 6-41 [1], 18-11 [1]

- SBCU_ECON 6-39 [1], 18-11 [1]
 - SBCU_EDAT 6-41 [1], 18-11 [1]
 - SBCU_ID 6-36 [1], 18-11 [1]
 - SBCU_SRC 6-58 [1], 18-12 [1]
 - SCU_CON 5-64 [1], 18-8 [1]
 - SCU_EMRSR 5-59 [1], 18-8 [1]
 - SCU_ID 5-69 [1], 18-7 [1]
 - SCU_PETCR 5-39 [1], 18-9 [1]
 - SCU_PETSR 5-40 [1], 18-10 [1]
 - SCU_PTCON 5-52 [1], 18-9 [1]
 - SCU_PTDAT0 5-53 [1], 18-9 [1]
 - SCU_PTDAT1 5-54 [1], 18-9 [1]
 - SCU_PTDAT2 5-55 [1], 18-9 [1]
 - SCU_PTDAT3 5-56 [1], 18-9 [1]
 - SCU_SCLIR 10-24 [1], 18-8 [1]
 - SCU_SCLKFDR 3-42 [1], 18-7 [1]
 - SCU_STAT 5-67 [1], 18-8 [1]
 - SCU_TCCON 5-44 [1], 18-8 [1]
 - SCU_TCLR0 5-46 [1], 18-8 [1]
 - SCU_TCLR1 5-47 [1], 18-8 [1]
 - SSC module registers 20-27 [2]
 - SSC0 register address map 18-81 [1]
 - SSC0_BR 18-81 [1], 20-39 [2]
 - SSC0_CLC 18-81 [1], 20-50 [2]
 - SSC0_CON 18-81 [1], 20-32 [2]
 - SSC0_EFM 18-81 [1], 20-35 [2]
 - SSC0_ESRC 18-82 [1], 20-59 [2]
 - SSC0_FDR 18-81 [1], 20-51 [2]
 - SSC0_FSTAT 18-81 [1], 20-44 [2]
 - SSC0_ID 18-81 [1], 20-29 [2]
 - SSC0_PISEL 18-81 [1], 20-30 [2]
 - SSC0_RB 18-81 [1], 20-45 [2]
 - SSC0_RSRC 18-82 [1], 20-59 [2]
 - SSC0_RXFCON 18-81 [1], 20-40 [2]
 - SSC0_SSOC 18-81 [1], 20-37 [2]
 - SSC0_SSOTC 18-81 [1], 20-38 [2]
 - SSC0_STAT 18-81 [1], 20-34 [2]
 - SSC0_TB 18-81 [1], 20-45 [2]
 - SSC0_TSRC 18-82 [1], 20-59 [2]
 - SSC0_TXFCON 18-81 [1], 20-42 [2]
 - SSC1 register address map 18-82 [1]
 - SSC1_BR 18-82 [1], 20-39 [2]
 - SSC1_CLC 18-82 [1], 20-50 [2]
 - SSC1_CON 18-82 [1], 20-32 [2]
 - SSC1_EFM 18-82 [1], 20-35 [2]
 - SSC1_ESRC 18-83 [1], 20-59 [2]
 - SSC1_FDR 18-82 [1], 20-51 [2]
 - SSC1_ID 18-82 [1], 20-29 [2]
 - SSC1_PISEL 18-82 [1], 20-30 [2]
 - SSC1_RB 18-82 [1], 20-45 [2]
 - SSC1_RSRC 18-83 [1], 20-59 [2]
 - SSC1_SSOC 18-82 [1], 20-37 [2]
 - SSC1_SSOTC 18-82 [1], 20-38 [2]
 - SSC1_STAT 18-82 [1], 20-34 [2]
 - SSC1_TB 18-82 [1], 20-45 [2]
 - SSC1_TSRC 18-83 [1], 20-59 [2]
 - STM register address map 18-13 [1]
 - STM_CAP 15-14 [1], 18-13 [1]
 - STM_CLC 15-9 [1], 18-13 [1]
 - STM_CMCON 15-16 [1], 18-13 [1]
 - STM_CMP0 15-15 [1], 18-13 [1]
 - STM_CMP1 15-15 [1], 18-13 [1]
 - STM_ICR 15-18 [1], 18-13 [1]
 - STM_ID 15-10 [1], 18-13 [1]
 - STM_ISR 15-20 [1], 18-13 [1]
 - STM_SRC0 15-21 [1], 18-13 [1]
 - STM_SRC1 15-21 [1], 18-13 [1]
 - STM_TIM0 15-11 [1], 18-13 [1]
 - STM_TIM1 15-11 [1], 18-13 [1]
 - STM_TIM2 15-12 [1], 18-13 [1]
 - STM_TIM3 15-12 [1], 18-13 [1]
 - STM_TIM4 15-12 [1], 18-13 [1]
 - STM_TIM5 15-13 [1], 18-13 [1]
 - STM_TIM6 15-13 [1], 18-13 [1]
 - SWEVT 18-108 [1]
 - SYSCON 2-10 [1], 18-109 [1]
- T**
- TGADC0 5-32 [1], 18-9 [1], 25-116 [2]
 - TGADC1 5-32 [1], 18-9 [1], 25-116 [2]
 - TR0EVT 2-18 [1], 18-108 [1]
 - TR1EVT 2-18 [1], 18-108 [1]
- W**
- WDT_CON0 16-29 [1], 18-7 [1]
 - WDT_CON1 16-31 [1], 18-7 [1]

WDT_SR 16-32 [1], 18-7 [1]

www.infineon.com

Published by Infineon Technologies AG