Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

enCoRe™ V CY7C643xx, enCoRe™ V LV CY7C604xx

Technical Reference Manual (TRM)

Document No. 001-32519 Rev *H

November 19, 2018

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
www.cypress.com

Copyrights

© Cypress Semiconductor Corporation, 2007-2018. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.

Contents

Section A: Overview 9

1. Pin Information 15

1.1 Pinouts 15
1.1.1 CY7C60413 enCoRe V LV 16-Pin Part Pinout 15
1.1.2 CY7C60445 enCoRe V LV 32-Pin Part Pinout16
1.1.3 CY7C64345, CY7C64343, enCoRe V 32-Pin Part Pinout 17
1.1.4 CY8C20646A/AS/LCY8C20666A/AS/L CY7C64355, CY7C64356 enCoRe V 48-Pin Part Pinout18
1.1.5 CY7C60455, CY7C60456 enCoRe V LV 48-Pin Part Pinout 19
1.1.6 CY8C20066A, CY8CTMG200-00LTXI, CY8CTMG200A-00LTXI PSoC, CY7C64300 enCoRe V and CY7C60400 enCoRe V LV OCD 48-Pin Part Pinout20
1.1.7 32-Pin QFN (with USB) 21
1.1.8 48-Pin SSOP 22

Section B: enCoRe V Core 23

2. CPU Core (M8C) 26

2.1 Overview 26
2.2 Internal Registers 26
2.3 Address Spaces 26
2.4 Instruction Set Summary 27
2.5 Instruction Formats 29
2.5.1 One-Byte Instructions 29
2.5.2 Two-Byte Instructions 29
2.5.3 Three-Byte Instructions 30
2.6 Register Definitions 31
2.6.1 CPU_F Register 31
2.6.2 Related Registers 31

3. Supervisory ROM (SROM) 32

3.1 Architectural Description 32
3.1.1 Additional SROM Feature 33
3.1.2 SROM Function Descriptions 33
3.2 Register Definitions 37

4. RAM Paging 38

4.1 Architectural Description 38
4.1.1 Basic Paging 38
4.1.2 Stack Operations 38
4.1.3 Interrupts 39
4.1.4 MVI Instructions 39
4.1.5 Current Page Pointer 39

4.1.6 Index Memory Page Pointer 40
4.2 Register Definitions 41
4.2.1 TMP_DRx Registers 41
4.2.2 CUR_PP Register 41
4.2.3 STK_PP Register 42
4.2.4 IDX_PP Register 42
4.2.5 MVR_PP Register 42
4.2.6 MVW_PP Register 43
4.2.7 Related Registers 43

5. Interrupt Controller 44
5.1 Architectural Description 44
5.1.1 Posted versus Pending Interrupts 45
5.2 Application Overview 45
5.3 Register Definitions 46
5.3.1 INT_CLR0 Register 46
5.3.2 INT_CLR1 Register 47
5.3.3 INT_CLR2 Register 48
5.3.4 INT_MSK0 Register 49
5.3.5 INT_MSK1 Register 49
5.3.6 INT_MSK2 Register 50
5.3.7 INT_SW_EN Register 50
5.3.8 INT_VC Register 51
5.3.9 Related Registers 51

6. General-Purpose I/O (GPIO) 52
6.1 Architectural Description 52
6.1.1 General Description 53
6.1.2 Digital I/O 53
6.1.3 Analog and Digital Inputs 53
6.1.4 Port 1 Distinctions 53
6.1.5 Port 0 Distinctions 53
6.1.6 GPIO Block Interrupts 54
6.1.7 Data Bypass 55
6.2 Register Definitions 56
6.2.1 PRTxDR Registers 56
6.2.2 PRTxIE Registers 56
6.2.3 PRTxDMx Registers 57
6.2.4 IO_CFG1 Register 58
6.2.5 IO_CFG2 Register 58

7. Analog-to-Digital Converter (ADC) 59
7.1 Architectural Description 59
7.2 Brief Overview of ADC Components and Registers 60
7.2.1 Interface Command/Status Block 60
7.2.2 ADC 60
7.3 ADC Register Definitions - Application Interface 64
7.3.1 ADC Data Register 64
7.3.2 ADC Status Register 64
7.4 Application Overview 65
7.4.1 Use of Application Interface 65
7.4.2 Status Codes 65
7.4.3 ADC Usage Guidelines 65

7.4.4 Typical ADC Operation Procedure 66

8. Internal Main Oscillator (IMO) 67

8.1 Architectural Description 67
8.2 Application Overview 67
8.2.1 Trimming the IMO 67
8.2.2 Engaging Slow IMO 67
8.3 Register Definitions 68
8.3.1 IMO_TR Register 68
8.3.2 IMO_TR1 Register 68
8.3.3 CPU_SCR1 Register 69
8.3.4 OSC_CR2 Register 69
8.3.5 Related Registers 69
8.4 Timing Diagrams 69
8.5 Clocking Strategy 70
8.6 Usage Guidelines 70
8.6.1 Power Down Guidelines 70
8.7 Block Size/Area 70
8.8 Gate Count 70
8.9 Block Pin List 70
8.10 Block Level Interfaces 70
8.11 Initialization 70
8.12 Wounding 70
8.13 On-Chip Debugger Modes 70
8.14 Test Modes 70
8.15 Power Modes 70
8.16 Design Flow 70
8.17 Operating Condition Requirements 71
8.18 DC Specifications 71
8.19 AC Specifications 71

9. Internal Low-speed Oscillator (ILO) 72

9.1 Architectural Description 72
9.2 Register Definitions 73
9.2.1 ILO_TR Register 73

10. External Crystal Oscillator (ECO) 74

10.1 Architectural Description 74
10.2 Application Overview 75
10.3 Register Definitions 76
10.3.1 ECO_ENBUS Register 76
10.3.2 ECO_TRIM Register 76
10.3.3 ECO_CFG Register 76
10.3.4 Related Registers 77
10.4 Usage Modes and Guidelines 77

11. Sleep and Watchdog 78

11.1 Architectural Description 78
11.1.1 Sleep Control Implementation Logic 79
11.1.2 Sleep Timer 81
11.2 Application Overview 81
11.3 Register Definitions 82
11.3.1 RES_WDT Register 82
11.3.2 SLP_CFG Register 82

11.3.3 SLP_CFG2 Register 83
11.3.4 SLP_CFG3 Register 83
11.3.5 Related Registers 83
11.4 Timing Diagrams 84
11.4.1 Sleep Sequence 84
11.4.2 Wakeup Sequence 85
11.4.3 Bandgap Refresh 85
11.4.4 Watchdog Timer 86
11.5 86

12. Regulated I/O 87

12.1 Architectural Description 87
12.1.1 Bias Generator 88
12.1.2 Charge Pump 88
12.1.3 Comparator 88
12.1.4 Replica Structure 88
12.1.5 Pass Transistors 88
12.2 Application Overview 88
12.3 Register Definitions 89
12.3.1 IO_CFG1 Register 89
12.3.2 IO_CFG2 Register 89

13. I/O Analog Multiplexer 90

13.1 Architectural Description 90
13.2 Register Definitions 91
13.2.1 MUX_CRx Registers 91

Section C: System Resources 92

14. Digital Clocks 96

14.1 Architectural Description 96
14.1.1 Internal Main Oscillator 96
14.1.2 Internal Low-speed Oscillator 96
14.1.3 External Clock 97
14.2 Register Definitions 99
14.2.1 USB_MISC_CR Register 99
14.2.2 OUT_P1 Register 99
14.2.3 OSC_CR0 Register 101
14.2.4 OSC_CR2 Register 102

15. I^2C Slave 103

15.1 Architectural Description 103
15.1.1 Basic I^2C Data Transfer 104
15.2 Application Overview 105
15.2.1 Slave Operation 105
15.3 Register Definitions 106
15.3.1 I2C_XCFG Register 106
15.3.2 I2C_ADDR Register 106
15.3.3 I2C_CFG Register 107
15.3.4 I2C_SCR Register 109
15.3.5 I2C_DR Register 110
15.4 Timing Diagrams 111
15.4.1 Clock Generation 111
15.4.2 Basic I/O Timing 111

15.4.3 Status Timing 112
15.4.4 Slave Stall Timing 113

16. System Resets 114

16.1 Architectural Description 114
16.2 Pin Behavior During Reset 114
16.2.1 GPIO Behavior on Power Up 114
16.2.2 Powerup External Reset Behavior 115
16.2.3 GPIO Behavior on External Reset 115
16.3 Register Definitions 116
16.3.1 CPU_SCR1 Register 116
16.3.2 CPU_SCR0 Register 117
16.4 Timing Diagrams 118
16.4.1 Power-On-Reset 118
16.4.2 External Reset 118
16.4.3 Watchdog Timer Reset 118
16.4.4 Reset Details 120
16.5 Power Modes 120

17. POR and LVD 121

17.1 Architectural Description 121
17.2 Register Definitions 122
17.2.1 VLT_CR Register 122
17.2.2 VLT_CMP Register 122

18. SPI 123

18.1 Architectural Description 123
18.1.1 SPI Protocol Function 123
18.1.2 SPI Master Function 124
18.1.3 SPI Slave Function 124
18.1.4 Input Synchronization 125
18.2 Register Definitions 125
18.2.1 SPI_TXR Register 125
18.2.2 SPI_RXR Register 126
18.2.3 SPI_CR Register 127
18.2.4 SPI_CFG Register 128
18.2.5 Related Registers 128
18.3 Timing Diagrams 129
18.3.1 SPI Mode Timing 129
18.3.2 SPIM Timing 130
18.3.3 SPIS Timing 134

19. Programmable Timer 137

19.1 Architectural Description 137
19.1.1 Operation 137
19.2 Register Definitions 139
19.2.1 PT0_CFG Register 139
19.2.2 PT1_CFG Register 139
19.2.3 PT2_CFG Register 140
19.2.4 PTx_DATA0 Register 140
19.2.5 PTx_DATA1 Register 140

20. Full-Speed USB 141

20.1 Architectural Description 141

20.2 Application Description 141
20.2.1 USB SIE 141
20.2.2 USB SRAM 142
20.2.3 Oscillator Lock 144
20.2.4 Transceiver 144
20.2.5 USB Suspend 145
20.2.6 Regulator 145
20.3 Register Definitions 147
20.3.1 USB_SOF0 Register 147
20.3.2 USB_CR0 Register 147
20.3.3 USBIO_CR0 Register 148
20.3.4 USBIO_CR1 Register 148
20.3.5 EP0_CR Register 149
20.3.6 EP0_CNT Register 150
20.3.7 EP0_DRx Register 150
20.3.8 EPx_CNT1 Register 151
20.3.9 EPx_CNT0 Register 152
20.3.10 EPx_CR0 Register 153
20.3.11 PMAx_WA Register 154
20.3.12 PMAx_DR Register 155
20.3.13 PMAx_RA Register 156
20.3.14 USB_CR1 Register 157
20.3.15 USB_MISC_CR Register 157
20.3.16 IMO_TR1 Register 158

Section D: Registers 159

21. Register Reference 163

21.1 Maneuvering Around the Registers 163
21.2 Register Conventions 163
21.3 Bank 0 Registers 164
21.4 Bank 1 Registers 212

Section E: Glossary 239

Section A: Overview

The enCoRe™ V family consists of many On-Chip Controller devices. The CY8C20x46A/46AS/96A/46L/96LCY7C643xx and CY7C604xx enCoRe V devices have fixed analog and digital resources in addition to a fast CPU, flash program memory, and SRAM data memory to support various algorithms.

For the most up-to-date ordering, pinout, packaging, or electrical specification information, refer to the enCoRe V device's datasheet. For the most current technical reference manual information and newest product documentation, go to the Cypress web site at http://www.cypress.com >> Documentation.

This section contains:

- Pin Information on page 15.

Document Organization

This manual is organized into sections and chapters, according to enCoRe V functionality. Each section contains a top-level architectural diagram and a register summary (if applicable). Most chapters within the sections have an introduction, an architectural/application description, register definitions, and timing diagrams. The sections are as follows:

- **Overview** – Presents the top-level architecture, helpful information to get started, and document history and conventions. The enCoRe V device ***pinouts*** are detailed in Pin Information, on page 15.
- **enCoRe V Core** – Describes the heart of the enCoRe V device in various chapters, beginning with an architectural overview and a summary list of registers pertaining to the enCoRe V core.
- **System Resources** – Presents additional enCoRe V system resources, beginning with an overview and a summary list of registers pertaining to system resources.
- **Registers** – Lists all enCoRe V device registers in register mapping tables, and presents bit-level detail of each register in its own Register Reference chapter. Where applicable, detailed register descriptions are also located in each chapter.
- **Glossary** – Defines the specialized terminology used in this manual. Glossary terms are presented in ***bold, italic font*** throughout this manual.
- **Index** – Lists the location of key topics and elements that constitute and empower the enCoRe V devices.

Top-Level Architecture

The enCoRe V block diagram on the next page illustrates the top-level architecture of the CY8C20X46A/46AS/96A/46L/96LCY7C643xx and CY7C604xx devices. Each major grouping in the diagram is covered in this manual in its own section: enCoRe V Core and System Resources. Banding these two main areas together is the communication network of the system ***bus***.

enCoRe V Core

The enCoRe V Core is a powerful engine that supports a rich instruction set. It includes the ***SRAM*** for data storage, an ***interrupt*** controller for easy program execution to new addresses, sleep and watchdog timers, a regulated 3.0-V output option for Port 1 I/Os, and multiple ***clock*** sources that include the IMO (internal main oscillator) and ILO (internal low-speed oscillator) for precision, programmable clocking.

The CPU core, called the M8C, is a powerful processor with speeds up to 24 MHz. The M8C is a four MIPS 8-***bit*** Harvard architecture microprocessor. Within the CPU core are the ***SROM*** and ***Flash*** memory components that provide flexible programming.

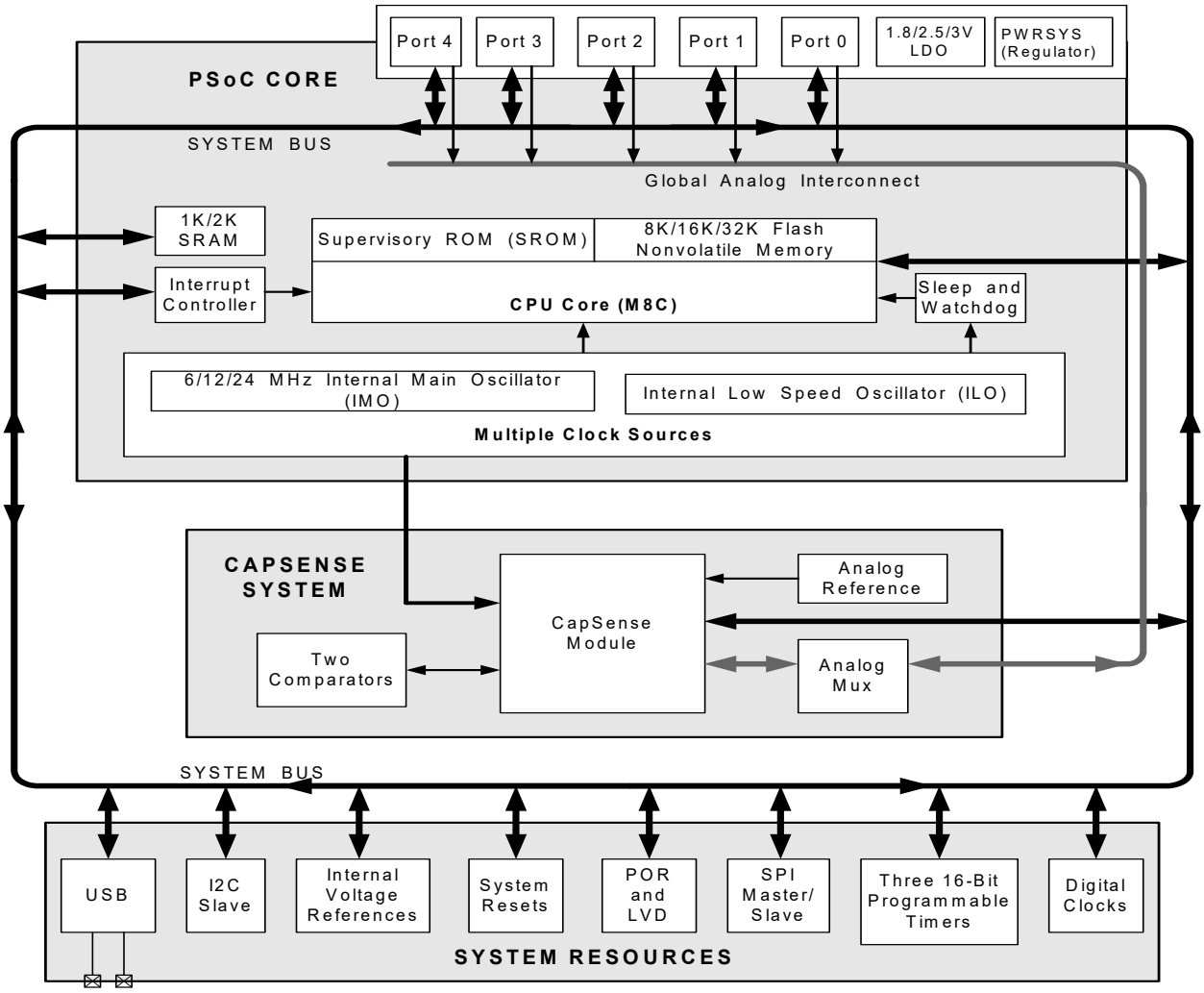
enCoRe V GPIOs provide connection to the CPU and external resources of the device. Each pin's drive mode is selectable from four options, allowing great flexibility in external interfacing. Every pin also has the capability to generate a system interrupt on low level and change from last read.

System Resources

The System Resources provide additional enCoRe V capability. These system resources include:

- Digital clocks to increase flexibility.
- I2C functionality with "no bus stalling."
- Various system resets supported by the M8C.
- Power-on-reset (POR) circuit protection.
- SPI master and slave functionality.
- A programmable timer to provide periodic interrupts.
- Clock boost network providing a stronger signal to switches.
- Full-speed USB interface for USB 2.0 communication with 512 bytes of dedicated buffer memory and an internal 3-V regulator.

enCoRe V Core Top-Level Block Diagram



Getting Started

The quickest path to understanding enCoRe V is by reading the enCoRe V device's datasheet and using *PSoC Designer™ Integrated Development Environment (IDE)*. This manual is useful for understanding the details of the enCoRe V integrated circuit.

Important Note For the most up-to-date Ordering, Packaging, or Electrical Specification information, refer to the individual enCoRe V device's datasheet or go to http://www.cypress.com.

Support

Free support for enCoRe V products is available online at http://www.cypress.com. Resources include Training Seminars, Discussion Forums, Application Notes, TightLink Technical Support Email/Knowledge Base, and Application Support Technicians.

Technical Support can be reached at http://www.cypress.com/support.

Product Upgrades

Cypress provides scheduled upgrades and version enhancements for PSoC Designer free of charge. You can order the upgrades from your distributor on CD-ROM or download them directly from http://www.cypress.com under Software. Also provided are critical updates to system documentation under http://www.cypress.com >> Documentation.

Development Kits

Development Kits are available from the following distributors: Digi-Key, Avnet, Arrow, and Future. The Cypress Online Store contains development kits, ***C*** compilers, and all accessories for enCoRe V development. Go to the Cypress Online Store at http://www.cypress.com under Order >> USB Kits.

Document History

This section serves as a chronicle of the *CY8C20XX6A/AS/LenCoRe™ V and enCoRe™ V LV CY7C643xx and CY7C604xx Technical Reference Manual*.

Technical Reference Manual History

Version/ Release Date	Originator	Description of Change
** September 2007	HMT	First release of the enCoRe™ V and enCoRe™ V LV CY7C643xx and CY7C604xx Technical Reference Manual.
*A June 2008	HMT	Second release of the enCoRe™ V and enCoRe™ V LV CY7C643xx and CY7C604xx Technical Reference Manual.
*B June 2009	FSU	Third release of the enCoRe™ V and enCoRe™ V LV CY7C643xx and CY7C604xx Technical Reference Manual.
*C September 2009	FSU	Fourth release of the enCoRe™ V and enCoRe™ V LV CY7C643xx and CY7C604xx Technical Reference Manual.
*D November 2009	FSU	Multiple fixes, primarily to the sleep and I2C chapters.
*E December 2009	FSU	Multiple fixes, primarily to the External Crystal Oscillator chapter.
*F September 2012	ANTG	Updated external clock source description
*G October 2015	ASRI	Removed all instances of IMODIS related information and provided information for "no glitch protection in the device for an external clock".
*H November 2018	RAJV	Updated the template

Documentation Conventions

There are only four distinguishing font types used in this manual, besides those found in the headings.

- The first is the use of *italics* when referencing a document title or file name.
- The second is the use of ***bold italics*** when referencing a term described in the Glossary of this manual.
- The third is the use of Times New Roman font, distinguishing equation examples.
- The fourth is the use of `Courier New` font, distinguishing code examples.

Register Conventions

The following table lists the register conventions that are specific to this manual. A more detailed set of register conventions is located in the Register Reference chapter on page 163.

Register Conventions

Convention	Example	Description
'x' in a register name	PRTxIE	Multiple instances/address ranges of the same register
R	R : 00	Read register or bit(s)
W	W : 00	Write register or bit(s)
O	RO : 00	Only a read/write register or bit(s).
L	RL : 00	Logical register or bit(s)
C	RC : 00	Clearable register or bit(s)
00	RW : 00	Reset value is 0x00 or 00h
XX	RW : XX	Register is not reset
0,	0,04h	Register is in bank 0
1,	1,23h	Register is in bank 1
x,	x,F7h	Register exists in register bank 0 and register bank 1
Empty, grayed-out table cell		Reserved bit or group of bits, unless otherwise stated

Numeric Naming

Hexadecimal numbers are represented with all letters in uppercase with an appended lowercase 'h' (for example, '14h' or '3Ah') and ***hexadecimal*** numbers may also be represented by a '0x' prefix, the ***C*** coding convention. Binary numbers have an appended lowercase 'b' (for example, 01010100b' or '01000011b'). Numbers not indicated by an 'h' or 'b' are ***decimal***.

Units of Measure

This table lists the units of measure used in this manual.

Units of Measure

Symbol	Unit of Measure
°C	degrees Celsius
dB	decibels
fF	femtofarads
Hz	hertz
k	kilo, 1000
K	2^{10}, 1024
KB	1024 bytes
Kbit	1024 bits
kHz	kilohertz (32.000)
kΩ	kilohms
MHz	megahertz
MΩ	megaohms
µA	microamperes
µF	microfarads
µs	microseconds
µV	microvolts
µVrms	microvolts root-mean-square
mA	milliamperes
ms	milliseconds
mV	millivolts
nA	nanoampheres
ns	nanoseconds
nV	nanovolts
Ω	ohms
pF	picofarads
pp	peak-to-peak
ppm	parts per million
sps	samples per second
σ	sigma: one standard deviation
V	volts

Acronyms

This table lists the acronyms that are used in this manual.

Acronyms

Acronym	Description
ABUS	analog output bus
AC	alternating current
ADC	analog-to-digital converter
API	Application Programming Interface
BR	bit rate
BRA	bus request acknowledge
BRQ	bus request
CI	carry in
CMP	compare
CO	carry out
CPU	central processing unit
CRC	cyclic redundancy check
DAC	digital-to-analog converter
DC	direct current
DI	digital or data input
DMA	direct memory access
DO	digital or data output
ECO	external crystal oscillator
FB	feedback
GIE	global interrupt enable
GPIO	general-purpose I/O
ICE	in-circuit emulator
IDE	integrated development environment
ILO	internal low-speed oscillator
IMO	internal main oscillator
I/O	input/output
IOR	I/O read
IOW	I/O write
IPOR	imprecise power-on-reset
IRQ	interrupt request
ISR	interrupt service routine
ISSP	in system serial programming
IVR	interrupt vector read
LRb	last received bit
LRB	last received byte
LSb	least significant bit
LSB	least significant byte
MISO	master-in-slave-out
MOSI	master-out-slave-in
MSb	most significant bit
MSB	most significant byte
PC	program counter
PCH	program counter high
PCL	program counter low
PD	power down
PMA	PSoC® memory arbiter
POR	power-on-reset

Acronyms *(continued)*

Acronym	Description
PPOR	precision power-on-reset
PRS	pseudo random sequence
PSSDC	power system sleep duty cycle
RAM	random access memory
RETI	return from interrupt
RO	relaxation oscillator
ROM	read-only memory
RW	read/write
SIE	serial interface engine
SE0	single-ended zero
SOF	start of frame
SP	stack pointer
SPI	serial peripheral interconnect
SPIM	serial peripheral interconnect master
SPIS	serial peripheral interconnect slave
SRAM	static random access memory
SROM	supervisory read-only memory
SSADC	single slope ADC
SSC	supervisory system call
TC	terminal count
USB	universal serial bus
WDT	watchdog timer
WDR	watchdog reset
XRES	external reset

1. Pin Information

This chapter lists, describes, and illustrates all pins and pinout configurations for the CY8C20X46A/46AS/96A/46L/96LCY7C643xx and CY7C604xx enCoRe V devices. For up-to-date ordering, pinout, and packaging information, refer to the individual enCoRe V device's datasheet or go to http://www.cypress.com.

1.1 Pinouts

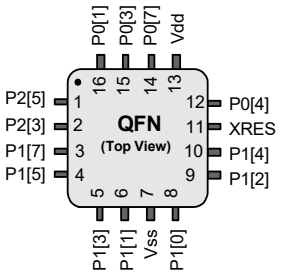
TheCY8C20X46A/46AS/96A/46L/96LCY7C643xx and CY7C604xx enCoRe V devices are available in a variety of packages. Every ***port*** pin (labeled with a "P"), except for ***Vss***, ***Vdd***, and XRES in the following tables and illustrations, is capable of Digital I/O.

1.1.1 CY7C60413 enCoRe V LV 16-Pin Part Pinout

Table 1-1. 16-Pin QFN/COL Part Pinout

Pin No.	Type		Name	Description
	Digital	Analog		
1	IO	I	P2[5]	XTAL Out
2	IO	I	P2[3]	XTAL In
3	IOHR	I	P1[7]	I2C SCL, SPI SS
4	IOHR	I	P1[5]	I2C SDA, SPI MISO
5	IOHR	I	P1[3]	SPI CLK
6	IOHR	I	P1[1]	TC CLK[1], I2C SCL, SPI MOSI
7	Power		Vss	Ground pin
8	IOHR	I	P1[0]	TC DATA[1], I2C SDA, SPI CLK
9	IOHR	I	P1[2]	
10	IOHR	I	P1[4]	EXTCLK
11	Input		XRES	Active high external reset with internal pull down
12	IOH	I	P0[4]	
13	Power		Vdd	Power pin
14	IOH	I	P0[7]	
15	IOH	I	P0[3]	
16	IOH	I	P0[1]	

Devices



Legend A = Analog, I = Input, O = Output, H = 5-mA High Output Drive, R = Regulated Output Option.

[1] These are the ISSP pins, which are not High-Z at POR.

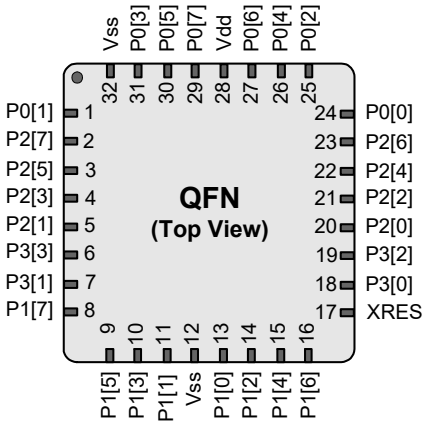
,

1.1.2 CY7C60445 enCoRe V LV 32-Pin Part Pinout

Table 1-2. 32-Pin QFN Part Pinout[2]

Pin No.	Digital	Analog	Name	Description
1	IOH	I	P0[1]	Integrating input
2	IO	I	P2[7]	
3	IO	I	P2[5]	XTAL Out
4	IO	I	P2[3]	XTAL In
5	IO	I	P2[1]	
6	IO	I	P3[3]	
7	IO	I	P3[1]	
8	IOHR	I	P1[7]	I2C SCL, SPI SS
9	IOHR	I	P1[5]	I2C SDA, SPI MISO
10	IOHR	I	P1[3]	SPI CLK
11	IOHR	I	P1[1]	TC CLK[1], I2C SCL, SPI MOSI
12	Power		Vss	Ground pin
13	IOHR	I	P1[0]	TC DATA[1], I2C SDA, SPI CLK
14	IOHR	I	P1[2]	
15	IOHR	I	P1[4]	EXTCLK
16	IOHR	I	P1[6]	
17	Input		XRES	Active high external reset with internal pull down
18	IO	I	P3[0]	
19	IO	I	P3[2]	
20	IO	I	P2[0]	
21	IO	I	P2[2]	
22	IO	I	P2[4]	
23	IO	I	P2[6]	
24	IOH	I	P0[0]	
25	IOH	I	P0[2]	
26	IOH	I	P0[4]	
27	IOH	I	P0[6]	
28	Power		Vdd	Power pin
29	IOH	I	P0[7]	
30	IOH	I	P0[5]	
31	IOH	I	P0[3]	
32	Power		Vss	Ground pin

CY7C60445 enCoRe V LV Devices



Legend A = Analog, I = Input, O = Output, NC = No Connection, H = 5 mA High Output Drive, R = Regulated Output Option.

1 These are the ISSP pins, which are not High-Z at POR.

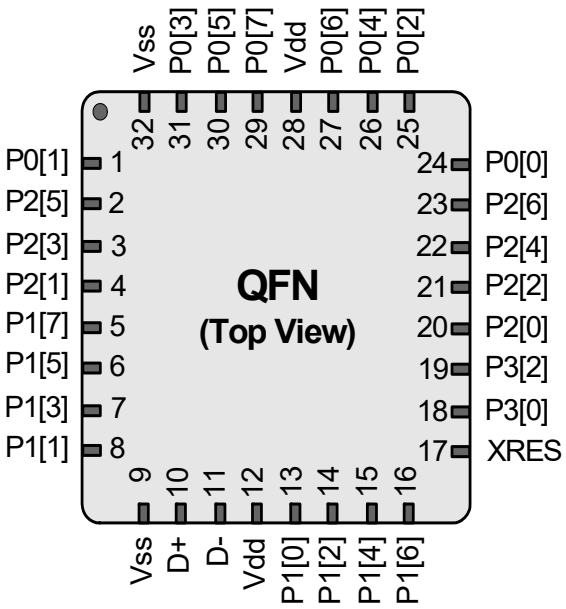
2 The center pad on the QFN package must be connected to ground (Vss) for best mechanical, thermal, and electrical performance. If not connected to ground, it must be electrically floated and not connected to any other signal.

1.1.3 CY7C64345, CY7C64343, enCoRe V 32-Pin Part Pinout

Table 1-3. 32-Pin QFN Part Pinout[2]

Pin No.	Digital	Analog	Name	Description
1	IOH	I	P0[1]	
2	IO	I	P2[5]	XTAL Out
3	IO	I	P2[3]	XTAL In
4	IO	I	P2[1]	
5	IOHR	I	P1[7]	I2C SCL, SPI SS
6	IOHR	I	P1[5]	I2C SDA, SPI MISO
7	IOHR	I	P1[3]	SPI CLK
8	IOHR	I	P1[1]	TC CLK[1], I2C SCL, SPI MOSI
9	Power		Vss	Ground pin
10	IO		D+	USB PHY
11	IO		D-	USB PHY
12	Power		Vdd	Power pin
13	IOHR	I	P1[0]	TC DATA[1], I2C SDA, SPI CLK
14	IOHR	I	P1[2]	
15	IOHR	I	P1[4]	EXTCLK
16	IOHR	I	P1[6]	
17	Input		XRES	Active high external reset with internal pull down
18	IO	I	P3[0]	
19	IO	I	P3[2]	
20	IO	I	P2[0]	
21	IO	I	P2[2]	
22	IO	I	P2[4]	
23	IO	I	P2[6]	
24	IOH	I	P0[0]	
25	IOH	I	P0[2]	
26	IOH	I	P0[4]	
27	IOH	I	P0[6]	
28	Power		Vdd	Power pin
29	IOH	I	P0[7]	
30	IOH	I	P0[5]	
31	IOH	I	P0[3]	
32	Power		Vss	Ground pin

CY7C64345, CY7C64343 enCoRe V Devices



LEGEND A = Analog, I = Input, O = Output, NC = No Connection, H = 5 mA High Output Drive, R = Regulated Output Option.

1 These are the ISSP pins, which are not High Z at POR (Power On Reset).

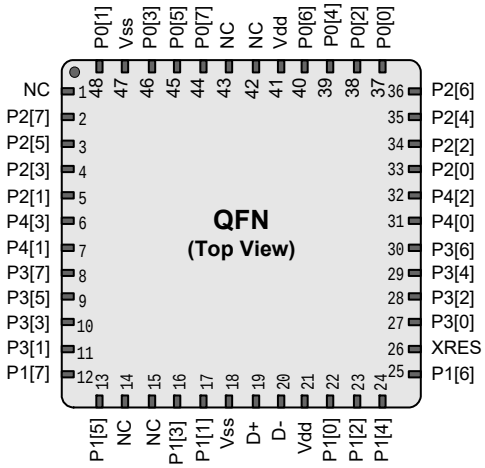
2 The center pad on the QFN package must be connected to ground (Vss) for best mechanical, thermal, and electrical performance. If not connected to ground, it must be electrically floated and not connected to any other signal.

1.1.4 CY8C20646A/AS/LCY8C20666A/AS/L CY7C64355, CY7C64356 enCoRe V 48-Pin Part Pinout

Table 1-4. 48-Pin Part Pinout[2]

Pin No.	Digital	Analog	Name	Description
1			NC	No connection
2	IO	I	P2[7]	
3	IO	I	P2[5]	XTAL Out
4	IO	I	P2[3]	XTAL In
5	IO	I	P2[1]	
6	IO	I	P4[3]	
7	IO	I	P4[1]	
8	IO	I	P3[7]	
9	IO	I	P3[5]	
10	IO	I	P3[3]	
11	IO	I	P3[1]	
12	IOHR	I	P1[7]	I2C SCL, SPI SS
13	IOHR	I	P1[5]	I2C SDA, SPI MISO
14			NC	No connection
15			NC	No connection
16	IOHR	I	P1[3]	SPI CLK
17	IOHR	I	P1[1]	TC CLK[1], I2C SCL, SPI MOSI
18	Power		Vss	Ground pin
19	IO		D +	USB PHY
20	IO		D -	USB PHY
21	Power		Vdd	Power pin
22	IOHR	I	P1[0]	TC DATA[1], I2C SDA, SPI CLK
23	IOHR	I	P1[2]	
24	IOHR	I	P1[4]	EXTCLK
25	IOHR	I	P1[6]	
26	Input		XRES	Active high external reset with internal pull down
27	IO	I	P3[0]	
28	IO	I	P3[2]	
29	IO	I	P3[4]	
30	IO	I	P3[6]	
31	IO	I	P4[0]	
32	IO	I	P4[2]	
33	IO	I	P2[0]	
34	IO	I	P2[2]	
35	IO	I	P2[4]	
36	IO	I	P2[6]	
37	IOH	I	P0[0]	
38	IOH	I	P0[2]	
39	IOH	I	P0[4]	
40	IOH	I	P0[6]	
41	Power		Vdd	Power pin
42			NC	No connection
43			NC	No connection
44	IOH	I	P0[7]	
45	IOH	I	P0[5]	
46	IOH	I	P0[3]	
47	Power		Vss	Ground pin
48	IOH	I	P0[1]	

CY7C64355, CY7C64356 enCoRe VDevices



Legend A = Analog, I = Input, O = Output, NC = No Connection, H = 5 mA High Output Drive, R = Regulated Output Option.

1 These are the ISSP pins, which are not High-Z at POR.

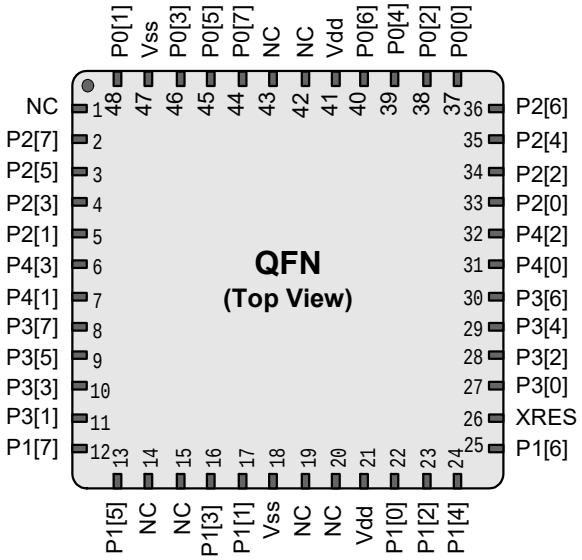
2 The center pad on the QFN package must be connected to ground (Vss) for best mechanical, thermal, and electrical performance. If not connected to ground, it must be electrically floated and not connected to any other signal.

1.1.5 CY7C60455, CY7C60456 enCoRe V LV 48-Pin Part Pinout

Table 1-5. 48-Pin Part Pinout[2]

Pin No.	Digital	Analog	Name	Description
1			NC	No connection
2	IO	I	P2[7]	
3	IO	I	P2[5]	XTAL Out
4	IO	I	P2[3]	XTAL In
5	IO	I	P2[1]	
6	IO	I	P4[3]	
7	IO	I	P4[1]	
8	IO	I	P3[7]	
9	IO	I	P3[5]	
10	IO	I	P3[3]	
11	IO	I	P3[1]	
12	IOHR	I	P1[7]	I2C SCL, SPI SS
13	IOHR	I	P1[5]	I2C SDA, SPI MISO
14			NC	No connection
15			NC	No connection
16	IOHR	I	P1[3]	SPI CLK
17	IOHR	I	P1[1]	TC CLK[1], I2C SCL, SPI MOSI
18	Power		Vss	Ground pin
19			NC	No connection
20			NC	No connection
21	Power		Vdd	Power pin
22	IOHR	I	P1[0]	TC DATA[1], I2C SDA, SPI CLK
23	IOHR	I	P1[2]	
24	IOHR	I	P1[4]	EXTCLK
25	IOHR	I	P1[6]	
26	Input		XRES	Active high external reset with internal pull down
27	IO	I	P3[0]	
28	IO	I	P3[2]	
29	IO	I	P3[4]	
30	IO	I	P3[6]	
31	IO	I	P4[0]	
32	IO	I	P4[2]	
33	IO	I	P2[0]	
34	IO	I	P2[2]	
35	IO	I	P2[4]	
36	IO	I	P2[6]	
37	IOH	I	P0[0]	
38	IOH	I	P0[2]	
39	IOH	I	P0[4]	
40	IOH	I	P0[6]	
41	Power		Vdd	Power pin
42			NC	No connection
43			NC	No connection
44	IOH	I	P0[7]	
45	IOH	I	P0[5]	
46	IOH	I	P0[3]	
47	Power		Vss	Ground pin
48	IOH	I	P0[1]	

CY7C60455, CY7C60456 enCoRe V LV Devices



LEGEND A = Analog, I = Input, O = Output, NC = No Connection, H = 5 mA High Output Drive, R = Regulated Output Option.

1 These are the ISSP pins, which are not High Z at POR (Power On Reset).

2 The center pad on the QFN package must be connected to ground (Vss) for best mechanical, thermal, and electrical performance. If not connected to ground, it must be electrically floated and not connected to any other signal.

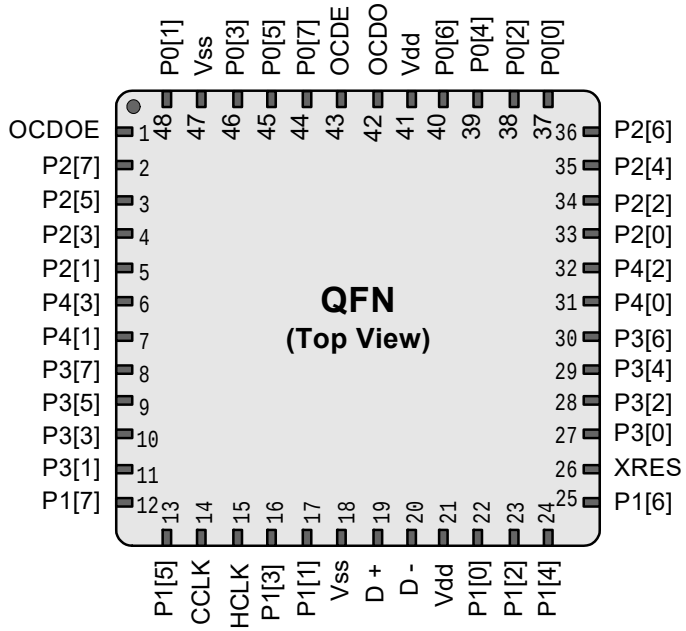
1.1.6 CY8C20066A, CY8CTMG200-00LTXI, CY8CTMG200A-00LTXI PSoC, CY7C64300 enCoRe V and CY7C60400 enCoRe V LV OCD 48-Pin Part Pinout

The 48-pin QFN part is for on-chip debugging (OCD). Note that this part is only used for in-circuit debugging. **It is NOT available for production**.

Table 1-6. 48-Pin OCD Part Pinout[2]

Pin No.	Digital	Analog	Name	Description
1			OCDOE	OCD directional pin
2	IO	I	P2[7]	
3	IO	I	P2[5]	XTAL Out
4	IO	I	P2[3]	XTAL In
5	IO	I	P2[1]	
6	IO	I	P4[3]	
7	IO	I	P4[1]	
8	IO	I	P3[7]	
9	IO	I	P3[5]	
10	IO	I	P3[3]	
11	IO	I	P3[1]	
12	IOHR	I	P1[7]	I2C SCL, SPI SS
13	IOHR	I	P1[5]	I2C SDA, SPI MISO
14			CCLK	OCD CPU CLK OUTPUT
15			HCLK	OCD HIGH SPEED CLK
16	IOHR	I	P1[3]	SPI CLK
17	IOHR	I	P1[1]	TC CLK[1], I2C SCL, SPI MOSI
18	Power		Vss	Ground pin
19	IO		D+	USB PHY
20	IO		D–	USB PHY
21	Power		Vdd	Power pin
22	IOHR	I	P1[0]	TC DATA[1], I2C SDA, SPI CLK
23	IOHR	I	P1[2]	
24	IOHR	I	P1[4]	EXTCLK
25	IOHR	I	P1[6]	
26	Input		XRES	Active high external reset with internal pull down
27	IO	I	P3[0]	
28	IO	I	P3[2]	
29	IO	I	P3[4]	
30	IO	I	P3[6]	
31	IO	I	P4[0]	
32	IO	I	P4[2]	
33	IO	I	P2[0]	
34	IO	I	P2[2]	
35	IO	I	P2[4]	
36	IO	I	P2[6]	
37	IOH	I	P0[0]	
38	IOH	I	P0[2]	
39	IOH	I	P0[4]	
40	IOH	I	P0[6]	

CY8C20066A, CY8CTMG200-00LTXI, CY8CTMG200A-00LTXI, CY7C64300, CY7C60400 enCoRe V OCD Devices



NOT FOR PRODUCTION – OCD Part

Pin No.	Digital	Analog	Name	Description
41	Power		Vdd	Power pin
42			OCDO	OCD even data I/O
43			OCDE	OCD odd data output
44	IOH	I	P0[7]	
45	IOH	I	P0[5]	
46	IOH	I	P0[3]	
47	Power		Vss	Ground pin
48	IOH	I	P0[1]	

Legend A = Analog, I = Input, O = Output, NC = No Connection, H = 5-mA High Output Drive, R = Regulated Output Option.

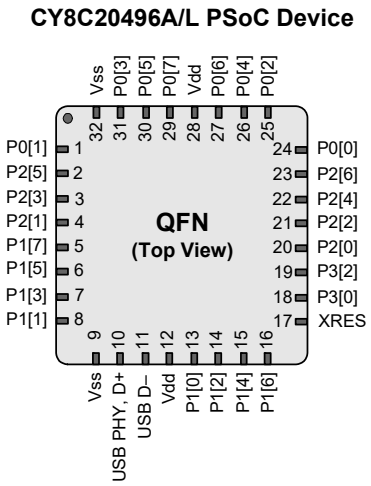
1 ISSP pin which is not High-Z at POR.

2 The center pad (CP) on the QFN package must be connected to ground (V_{SS}) for best mechanical, thermal, and electrical performance. If not connected to ground, it must be electrically floated and not connected to any other signal.

1.1.7 32-Pin QFN (with USB)

Table 1-7. Pin Definitions – CY8C20496A/L PSoC Device[2]

Pin No.	Type		Name	Description
	Digital	Analog		
1	IOH	I	P0[1]	Integrating Input
2	I/O	I	P2[5]	XTAL Out
3	I/O	I	P2[3]	XTAL In
4	I/O	I	P2[1]	
5	IOHR	I	P1[7]	I^2C SCL, SPI SS
6	IOHR	I	P1[5]	I^2C SDA, SPI MISO
7	IOHR	I	P1[3]	SPI CLK
8	IOHR	I	P1[1]	ISSP CLK[1], I^2C SCL, SPI MOSI
9	Power		V_{SS}	Ground Pin
10	I		D+	USB D+
11	I		D–	USB D–
12	Power		V_{DD}	Power pin
13	IOHR	I	P1[0]	ISSP DATA[1], I^2C SDA, SPI CLKI[3]
14	IOHR	I	P1[2]	
15	IOHR	I	P1[4]	Optional external clock input (EXTCLK)
16	IOHR	I	P1[6]	
17	Input		XRES	Active high external reset with internal pull-down
18	I/O	I	P3[0]	
19	I/O	I	P3[2]	
20	I/O	I	P2[0]	
21	I/O	I	P2[2]	
22	I/O	I	P2[4]	
23	I/O	I	P2[6]	
24	IOH	I	P0[0]	
25	IOH	I	P0[2]	
26	IOH	I	P0[4]	
27	IOH	I	P0[6]	
28	Power		V_{DD}	Power Pin
29	IOH	I	P0[7]	
30	IOH	I	P0[5]	
31	IOH	I	P0[3]	Integrating Input
32	Power		V_{SS}	Ground Pin



Legend A = Analog, I = Input, O = Output, OH = 5 mA High Output Drive, R = Regulated Output.

1 On power-up, the SDA(P1[0]) drives a strong high for 256 sleep clock cycles and drives resistive low for the next 256 sleep clock cycles. The SCL(P1[1])line drives resistive low for 512 sleep clock cycles and both the pins transition to high-impedance state. On reset, after XRES deasserts, the SDA and the SCL lines drive resistive low for 8 sleep clock cycles and transition to high-impedance state. Hence, during power-up or reset event, P1[1] and P1[0] may disturb the I2C bus. Use alternate pins if you encounter issues.

2 The center pad on the QFN package must be connected to ground (Vss) for best mechanical, thermal, and electrical performance. If not connected to ground, it must be electrically floated and not connected to any other signal.

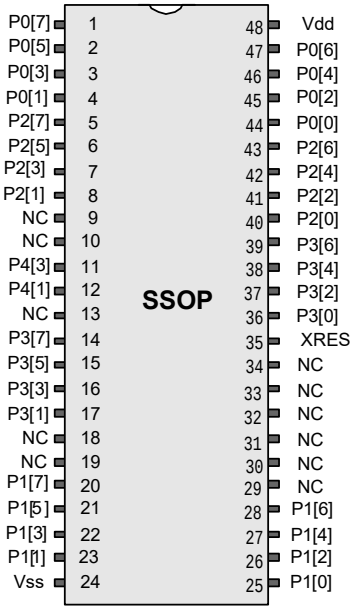
3 Alternate SPI clock.

1.1.8 48-Pin SSOP

Table 1-8. Pin Definitions – CY8C20536A, CY8C20546A, and CY8C20566A PSoC Device

Pin No.	Digital	Analog	Name	Description
1	IOH	I	P0[7]	
2	IOH	I	P0[5]	
3	IOH	I	P0[3]	Integrating Input
4	IOH	I	P0[1]	Integrating Input
5	I/O	I	P2[7]	
6	I/O	I	P2[5]	XTAL Out
7	I/O	I	P2[3]	XTAL In
8	I/O	I	P2[1]	
9			NC	No connection
10			NC	No connection
11	I/O	I	P4[3]	
12	I/O	I	P4[1]	
13			NC	No connection
14	I/O	I	P3[7]	
15	I/O	I	P3[5]	
16	I/O	I	P3[3]	
17	I/O	I	P3[1]	
18			NC	No connection
19			NC	No connection
20	IOHR	I	P1[7]	I^2C SCL, SPI SS
21	IOHR	I	P1[5]	I^2C SDA, SPI MISO
22	IOHR	I	P1[3]	SPI CLK
23	IOHR	I	P1[1]	ISSP CLK[1], I^2C SCL, SPI MOSI
24			V_{SS}	Ground Pin
25	IOHR	I	P1[0]	ISSP DATA[1], I^2C SDA, SPI CLK[2]
26	IOHR	I	P1[2]	
27	IOHR	I	P1[4]	Optional external clock input (EXT CLK)
28	IOHR	I	P1[6]	
29			NC	No connection
30			NC	No connection
31			NC	No connection
32			NC	No connection
33			NC	No connection
34			NC	No connection
35			XRES	Active high external reset with internal pull-down
36	I/O	I	P3[0]	
37	I/O	I	P3[2]	
38	I/O	I	P3[4]	
39	I/O	I	P3[6]	
40	I/O	I	P2[0]	
41	I/O	I	P2[2]	
42	I/O	I	P2[4]	
43	I/O	I	P2[6]	
44	IOH	I	P0[0]	
45	IOH	I	P0[2]	
46	IOH	I	P0[4]	
47	IOH	I	P0[6]	
48	Power		V_{DD}	Power Pin

CY8C20536A, CY8C20546A, and CY8C20566A PSoC Device



Legend A = Analog, I = Input, O = Output, NC = No Connection, H = 5 mA High Output Drive, R = Regulated Output Option.

[1] On power-up, the SDA(P1[0]) drives a strong high for 256 sleep clock cycles and drives resistive low for the next 256 sleep clock cycles. The SCL(P1[1])line drives resistive low for 512 sleep clock cycles and both the pins transition to high-impedance state. On reset, after XRES deasserts, the SDA and the SCL lines drive resistive low for 8 sleep clock cycles and transition to high-impedance state. Hence, during power-up or reset event, P1[1] and P1[0] may disturb the I2C bus. Use alternate pins if you encounter issues.

[2] Alternate SPI clock.

Section B: enCoRe V Core

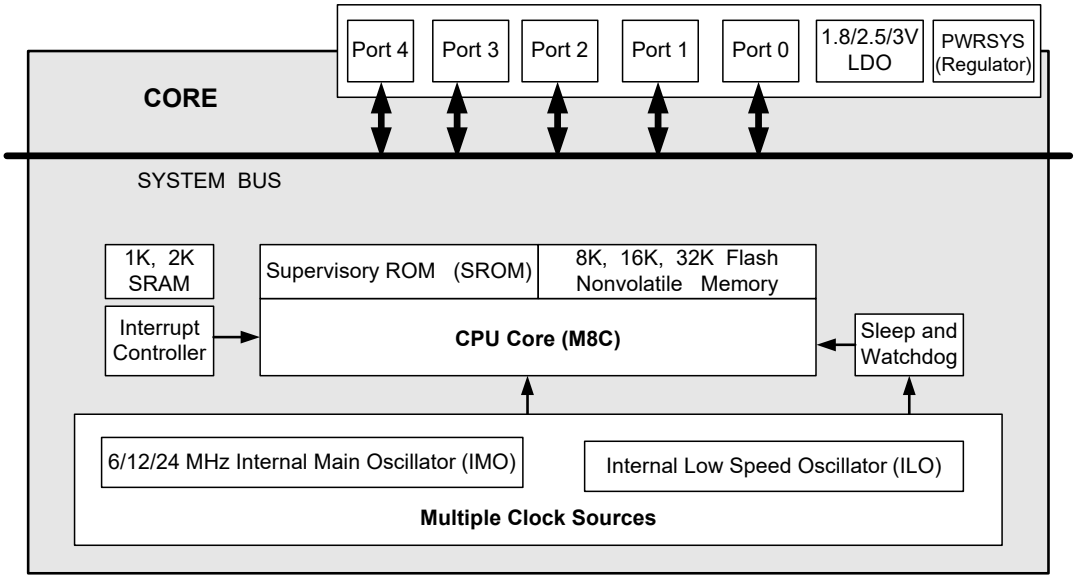
The enCoRe V Core section discusses the core components of an enCoRe V device with a base part number of CY7C643xx and CY7C604xx and the registers associated with those components. The core section covers the heart of the enCoRe V device, which includes the M8C ***microcontroller***; SROM, interrupt controller, GPIO, and ***SRAM*** paging; multiple clock sources such as IMO and ILO; and sleep and watchdog functionality. This section includes these chapters:

- CPU Core (M8C) on page 26.
- Supervisory ROM (SROM) on page 32.
- RAM Paging on page 38.
- Interrupt Controller on page 44.
- General-Purpose I/O (GPIO) on page 52.
- Internal Main Oscillator (IMO) on page 67.
- Internal Low-speed Oscillator (ILO) on page 72.
- External Crystal Oscillator (ECO), on page 74
- Sleep and Watchdog on page 78.

Top-Level Core Architecture

This figure displays the top-level architecture of the enCoRe V core. Each component of the figure is discussed at length in this section.

enCoRe V Core Block Diagram



Core Register Summary

This table lists all the enCoRe V registers for the CPU core in ***address*** order within their system resource configuration. The grayed out bits are reserved bits. If you write these bits, always write them with a value of '0'. For the core registers, the first 'x' in some ***register*** addresses represents either bank 0 or bank 1. These registers are listed throughout this manual in bank 0, even though they are also available in bank 1.

Summary Table of the Core Registers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
M8C REGISTER (page 26)										
x,F7h	CPU_F	PgMode[1:0]		XIO_1	XIO		Carry	Zero	GIE	RL : 02
RAM PAGING (SRAM) REGISTERS (page 38)										
x,6Ch	TMP_DR0	Data[7:0]								RW : 00
x,6Dh	TMP_DR1	Data[7:0]								RW : 00
x,6Eh	TMP_DR2	Data[7:0]								RW : 00
x,6Fh	TMP_DR3	Data[7:0]								RW : 00
0,D0h	CUR_PP						Page Bits[2:0]			RW : 0
0,D1h	STK_PP						Page Bits[2:0]			RW : 0
0,D3h	IDX_PP						Page Bits[2:0]			RW : 0
0,D4h	MVR_PP						Page Bits[2:0]			RW : 0
0,D5h	MVW_PP						Page Bits[2:0]			RW : 0
INTERRUPT CONTROLLER REGISTERS (page 44)										
0,DAh	INT_CLR0	I2C	Sleep	SPI	GPIO	Timer0	Reserved	Reserved	V Monitor	RW : 00
0,DBh	INT_CLR1	Endpoint3	Endpoint2	Endpoint1	Endpoint0	USB SOF	USB Bus Rst	Timer2	Timer1	RW : 00
0,DCh	INT_CLR2			USB_WAKE	Endpoint8	Endpoint7	Endpoint6	Endpoint5	Endpoint4	RW : 00
0,DEh	INT_MSK2			USB Wakeup	Endpoint8	Endpoint7	Endpoint6	Endpoint5	Endpoint4	RW : 00
0,DFh	INT_MSK1	Endpoint3	Endpoint2	Endpoint1	Endpoint0	USB SOF	USB Bus Reset	Timer2	Timer1	RW : 00
0,E0h	INT_MSK0	I2C	Sleep	SPI	GPIO	Timer0	Reserved	Reserved	V Monitor	RW : 00
0,E1h	INT_SW_EN								ENSWINT	RW : 0
0,E2h	INT_VC	Pending Interrupt[7:0]								RC : 00
GENERAL-PURPOSE I/O (GPIO) REGISTERS (page 56)										
0,00h	PRT0DR	Data[7:0]								RW : 00
0,01h	PRT0IE	Interrupt Enables[7:0]								RW : 00
0,04h	PRT1DR	Data[7:0]								RW : 00
0,05h	PRT1IE	Interrupt Enables[7:0]								RW : 00
0,08h	PRT2DR	Data[7:0]								RW : 00
0,09h	PRT2IE	Interrupt Enables[7:0]								RW : 00
0,0Ch	PRT3DR	Data[7:0]								RW : 00
0,0Dh	PRT3IE	Interrupt Enables[7:0]								RW : 00
1,00h	PRT0DM0	Drive Mode 0[7:0]								RW : 00
1,01h	PRT0DM1	Drive Mode 1[7:0]								RW : FF
1,04h	PRT1DM0	Drive Mode 0[7:0]								RW : 00
1,05h	PRT1DM1	Drive Mode 1[7:0]								RW : FF
1,08h	PRT2DM0	Drive Mode 0[7:0]								RW : 00
1,09h	PRT2DM1	Drive Mode 1[7:0]								RW : FF
1,0Ch	PRT3DM0	Drive Mode 0[7:0]								RW : 00
1,0Dh	PRT3DM1	Drive Mode 1[7:0]								RW : FF
0,10h	PRTxDR	Data[7:0]								RW : 00
0,11h	PRTxIE	Interrupt Enables[7:0]								RW : 00
1,10h	PRTxDM0	Drive Mode 0[7:0]								RW : 00
1,11h	PRTxDM1	Drive Mode 0[7:0]								RW : 00

Summary Table of the Core Registers *(continued)*

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,DCh	IO_CFG1	StrongP		Range[1:0]		P1_LOW_THRS	SPICLK_ON_P10	REG_EN	IOINT	RW : 00
INTERNAL MAIN OSCILLATOR (IMO) REGISTER (page 68)										
1,E8h	IMO_TR	Trim[7:0]								W: 00
1,FAh	IMO_TR1						Fine Trim[2:0]			RW : 00
x,FEh	CPU_SCR1	IRESS			SLIM[1:O]				IRAMDIS	# : 00
1,E2h	OSC_CR2				CLK48MEN		EXTCLKEN	RSVD		RW : 00
INTERNAL LOW-SPEED OSCILLATOR (ILO) REGISTER (page 73)										
1,E9h	ILO_TR		PD_MODE	ILOFREQ	SATBIASB	Freq Trim[3:0]				RW : 18
EXTERNAL CRYSTAL OSCILLATOR (ECO) REGISTERS (page 74)										
1,D2h	ECO_ENBUS						ECO_ENBUS[2:0]			RW : 07
1,D3h	ECO_TRIM				ECO_XGM[2:0]			ECO_LP[1:0]		RW : 00
1,E1h	ECO_CFG						ECO_LPM	ECO_EXW	ECO_EX	RW : 00
SLEEP AND WATCHDOG REGISTERS (page 82)										
0,E3h	RES_WDT	WDSL_Clear[7:0]								W : 00
1,EBh	SLP_CFG	PSSDC[1:0]								RW : 0
1,ECh	SLP_CFG2					ALT_Buzz [1:0]		I2C_ON	LSO_OFF	RW : 00
1,EDh	SLP_CFG3		DBL_TAPS	T2TAP [1:0]		T1TAP [1:0]		T0TAP [1:0]		RW : 0x7F

Legend
L The and f, expr; or f, expr; and xor f, expr instructions can be used to modify this register.
x An “x” before the comma in the address field indicates that this register can be accessed or written to no matter what bank is used.
C Clearable register or bit(s).
R Read register or bit(s).
W Write register or bit(s).

2. CPU Core (M8C)

This chapter explains the CPU Core, called the M8C, and its associated register. It covers the internal M8C registers, address spaces, ***instruction*** set and formats. For additional information concerning the M8C instruction set, refer to the *PSoC Designer Assembly Language User Guide* available at http://www.cypress.com. For a quick reference of all enCoRe V registers in address order, refer to the Register Reference chapter on page 163.

2.1 Overview

The ***M8C*** is a four MIPS 8-bit Harvard architecture microprocessor. Selectable processor clock speeds up to 24 MHz enable you to tune the M8C to a particular application's performance and power requirements. The M8C supports a rich instruction set that allows for efficient low-level language support.

2.2 Internal Registers

The M8C has five internal registers that are used in program execution. Here is a list of these registers.

- Accumulator (A)
- Index (X)
- Program Counter (PC)
- Stack Pointer (SP)
- Flags (F)

All the internal M8C registers are 8 bits in width, except for the PC, which is 16 bits wide. Upon ***reset***, A, X, PC, and SP are reset to 00h. The Flag register (F) is reset to 02h, indicating that the Z ***flag*** is ***set***.

With each ***stack*** operation, the SP is automatically incremented or decremented so that it always points to the next stack ***byte*** in RAM. If the last byte in the stack is at address FFh, the ***stack pointer*** wraps to RAM address 00h. It is the ***firmware*** developer's responsibility to ensure that the stack does not overlap with user-defined variables in RAM.

With the exception of the F register, the M8C internal registers are not accessible via an explicit register address. The internal M8C registers are accessed using these instructions:

- `MOV A, expr`
- `MOV X, expr`
- `SWAP A, SP`
- `OR F, expr`
- `JMP LABEL`

The F register is read by using address F7h in either register bank.

2.3 Address Spaces

The M8C has three address spaces: ***ROM***, ***RAM***, and registers. The ROM address space includes the Supervisory ROM (SROM) and the flash. The ROM address space is accessed through its own address and ***data bus***.

The ROM address space is composed of the SROM and the on-chip flash program store. Flash is organized into 128-byte blocks. Program store page boundaries are not an issue because the M8C automatically increments the 16-bit PC on every instruction. This makes the block boundaries invisible to user code. Instructions occurring on a 128-byte flash page boundary (with the exception of `JMP` instructions) incur an extra M8C clock cycle, because the upper byte of the PC is incremented.

The register address space is used to configure the enCoRe Vmicrocontroller's programmable blocks. It consists of two banks of 256 bytes each. To switch between banks, the XIO bit in the Flag register is set or cleared (set for Bank1, cleared for Bank0). The common convention is to leave the bank set to Bank0 (XIO cleared), switch to Bank1 as needed (set XIO), then switch back to Bank0.

2.4 Instruction Set Summary

The instruction set is summarized in both Table 2-1 and Table 2-2 (in numeric and ***mnemonic*** order, respectively), and serves as a quick reference. If more information is needed, the Instruction Set Summary tables are described in detail in the *PSoC Designer Assembly Language User Guide* (visit http://www.cypress.com).

Table 2-1. Instruction Set Summary Sorted Numerically by Opcode

Opcode Hex	Cycles	Bytes	Instruction Format	Flags	Opcode Hex	Cycles	Bytes	Instruction Format	Flags	Opcode Hex	Cycles	Bytes	Instruction Format	Flags
00	15	1	SSC		2D	8	2	OR [X+expr], A	Z	5A	5	2	MOV [expr], X	
01	4	2	ADD A, expr	C, Z	2E	9	3	OR [expr], expr	Z	5B	4	1	MOV A, X	Z
02	6	2	ADD A, [expr]	C, Z	2F	10	3	OR [X+expr], expr	Z	5C	4	1	MOV X, A	
03	7	2	ADD A, [X+expr]	C, Z	30	9	1	HALT		5D	6	2	MOV A, reg[expr]	Z
04	7	2	ADD [expr], A	C, Z	31	4	2	XOR A, expr	Z	5E	7	2	MOV A, reg[X+expr]	Z
05	8	2	ADD [X+expr], A	C, Z	32	6	2	XOR A, [expr]	Z	5F	10	3	MOV [expr], [expr]	
06	9	3	ADD [expr], expr	C, Z	33	7	2	XOR A, [X+expr]	Z	60	5	2	MOV reg[expr], A	
07	10	3	ADD [X+expr], expr	C, Z	34	7	2	XOR [expr], A	Z	61	6	2	MOV reg[X+expr], A	
08	4	1	PUSH A		35	8	2	XOR [X+expr], A	Z	62	8	3	MOV reg[expr], expr	
09	4	2	ADC A, expr	C, Z	36	9	3	XOR [expr], expr	Z	63	9	3	MOV reg[X+expr], expr	
0A	6	2	ADC A, [expr]	C, Z	37	10	3	XOR [X+expr], expr	Z	64	4	1	ASL A	C, Z
0B	7	2	ADC A, [X+expr]	C, Z	38	5	2	ADD SP, expr		65	7	2	ASL [expr]	C, Z
0C	7	2	ADC [expr], A	C, Z	39	5	2	CMP A, expr	if (A=B) Z=1 if (A<B) C=1	66	8	2	ASL [X+expr]	C, Z
0D	8	2	ADC [X+expr], A	C, Z	3A	7	2	CMP A, [expr]		67	4	1	ASR A	C, Z
0E	9	3	ADC [expr], expr	C, Z	3B	8	2	CMP A, [X+expr]		68	7	2	ASR [expr]	C, Z
0F	10	3	ADC [X+expr], expr	C, Z	3C	8	3	CMP [expr], expr		69	8	2	ASR [X+expr]	C, Z
10	4	1	PUSH X		3D	9	3	CMP [X+expr], expr		6A	4	1	RLC A	C, Z
11	4	2	SUB A, expr	C, Z	3E	10	2	MVI A, [[expr]++]	Z	6B	7	2	RLC [expr]	C, Z
12	6	2	SUB A, [expr]	C, Z	3F	10	2	MVI [[expr]++], A		6C	8	2	RLC [X+expr]	C, Z
13	7	2	SUB A, [X+expr]	C, Z	40	4	1	NOP		6D	4	1	RRC A	C, Z
14	7	2	SUB [expr], A	C, Z	41	9	3	AND reg[expr], expr	Z	6E	7	2	RRC [expr]	C, Z
15	8	2	SUB [X+expr], A	C, Z	42	10	3	AND reg[X+expr], expr	Z	6F	8	2	RRC [X+expr]	C, Z
16	9	3	SUB [expr], expr	C, Z	43	9	3	OR reg[expr], expr	Z	70	4	2	AND F, expr	C, Z
17	10	3	SUB [X+expr], expr	C, Z	44	10	3	OR reg[X+expr], expr	Z	71	4	2	OR F, expr	C, Z
18	5	1	POP A	Z	45	9	3	XOR reg[expr], expr	Z	72	4	2	XOR F, expr	C, Z
19	4	2	SBB A, expr	C, Z	46	10	3	XOR reg[X+expr], expr	Z	73	4	1	CPL A	Z
1A	6	2	SBB A, [expr]	C, Z	47	8	3	TST [expr], expr	Z	74	4	1	INC A	C, Z
1B	7	2	SBB A, [X+expr]	C, Z	48	9	3	TST [X+expr], expr	Z	75	4	1	INC X	C, Z
1C	7	2	SBB [expr], A	C, Z	49	9	3	TST reg[expr], expr	Z	76	7	2	INC [expr]	C, Z
1D	8	2	SBB [X+expr], A	C, Z	4A	10	3	TST reg[X+expr], expr	Z	77	8	2	INC [X+expr]	C, Z
1E	9	3	SBB [expr], expr	C, Z	4B	5	1	SWAP A, X	Z	78	4	1	DEC A	C, Z
1F	10	3	SBB [X+expr], expr	C, Z	4C	7	2	SWAP A, [expr]	Z	79	4	1	DEC X	C, Z
20	5	1	POP X		4D	7	2	SWAP X, [expr]		7A	7	2	DEC [expr]	C, Z
21	4	2	AND A, expr	Z	4E	5	1	SWAP A, SP	Z	7B	8	2	DEC [X+expr]	C, Z
22	6	2	AND A, [expr]	Z	4F	4	1	MOV X, SP		7C	13	3	LCALL	
23	7	2	AND A, [X+expr]	Z	50	4	2	MOV A, expr	Z	7D	7	3	LJMP	
24	7	2	AND [expr], A	Z	51	5	2	MOV A, [expr]	Z	7E	10	1	RETI	C, Z
25	8	2	AND [X+expr], A	Z	52	6	2	MOV A, [X+expr]	Z	7F	8	1	RET	
26	9	3	AND [expr], expr	Z	53	5	2	MOV [expr], A		8x	5	2	JMP	
27	10	3	AND [X+expr], expr	Z	54	6	2	MOV [X+expr], A		9x	11	2	CALL	
28	11	1	ROMX	Z	55	8	3	MOV [expr], expr		Ax	5	2	JZ	
29	4	2	OR A, expr	Z	56	9	3	MOV [X+expr], expr		Bx	5	2	JNZ	
2A	6	2	OR A, [expr]	Z	57	4	2	MOV X, expr		Cx	5	2	JC	
2B	7	2	OR A, [X+expr]	Z	58	6	2	MOV X, [expr]		Dx	5	2	JNC	
2C	7	2	OR [expr], A	Z	59	7	2	MOV X, [X+expr]		Ex	7	2	JACC	
										Fx	13	2	INDEX	Z

Note 1 Interrupt acknowledge to Interrupt Vector table = 13 cycles.

Note 2 The number of cycles required by an instruction is increased by one for instructions that span 128 byte page boundaries in the flash memory space.

Table 2-2. Instruction Set Summary Sorted Alphabetically by Mnemonic

Opcode Hex	Cycles	Bytes	Instruction Format	Flags	Opcode Hex	Cycles	Bytes	Instruction Format	Flags	Opcode Hex	Cycles	Bytes	Instruction Format	Flags
09	4	2	ADC A, expr	C, Z	76	7	2	INC [expr]	C, Z	20	5	1	POP X	
0A	6	2	ADC A, [expr]	C, Z	77	8	2	INC [X+expr]	C, Z	18	5	1	POP A	Z
0B	7	2	ADC A, [X+expr]	C, Z	Fx	13	2	INDEX	Z	10	4	1	PUSH X	
0C	7	2	ADC [expr], A	C, Z	Ex	7	2	JACC		08	4	1	PUSH A	
0D	8	2	ADC [X+expr], A	C, Z	Cx	5	2	JC		7E	10	1	RETI	C, Z
0E	9	3	ADC [expr], expr	C, Z	8x	5	2	JMP		7F	8	1	RET	
0F	10	3	ADC [X+expr], expr	C, Z	Dx	5	2	JNC		6A	4	1	RLC A	C, Z
01	4	2	ADD A, expr	C, Z	Bx	5	2	JNZ		6B	7	2	RLC [expr]	C, Z
02	6	2	ADD A, [expr]	C, Z	Ax	5	2	JZ		6C	8	2	RLC [X+expr]	C, Z
03	7	2	ADD A, [X+expr]	C, Z	7C	13	3	LCALL		28	11	1	ROMX	Z
04	7	2	ADD [expr], A	C, Z	7D	7	3	LJMP		6D	4	1	RRC A	C, Z
05	8	2	ADD [X+expr], A	C, Z	4F	4	1	MOV X, SP		6E	7	2	RRC [expr]	C, Z
06	9	3	ADD [expr], expr	C, Z	50	4	2	MOV A, expr	Z	6F	8	2	RRC [X+expr]	C, Z
07	10	3	ADD [X+expr], expr	C, Z	51	5	2	MOV A, [expr]	Z	19	4	2	SBB A, expr	C, Z
38	5	2	ADD SP, expr		52	6	2	MOV A, [X+expr]	Z	1A	6	2	SBB A, [expr]	C, Z
21	4	2	AND A, expr	Z	53	5	2	MOV [expr], A		1B	7	2	SBB A, [X+expr]	C, Z
22	6	2	AND A, [expr]	Z	54	6	2	MOV [X+expr], A		1C	7	2	SBB [expr], A	C, Z
23	7	2	AND A, [X+expr]	Z	55	8	3	MOV [expr], expr		1D	8	2	SBB [X+expr], A	C, Z
24	7	2	AND [expr], A	Z	56	9	3	MOV [X+expr], expr		1E	9	3	SBB [expr], expr	C, Z
25	8	2	AND [X+expr], A	Z	57	4	2	MOV X, expr		1F	10	3	SBB [X+expr], expr	C, Z
26	9	3	AND [expr], expr	Z	58	6	2	MOV X, [expr]		00	15	1	SSC	
27	10	3	AND [X+expr], expr	Z	59	7	2	MOV X, [X+expr]		11	4	2	SUB A, expr	C, Z
70	4	2	AND F, expr	C, Z	5A	5	2	MOV [expr], X		12	6	2	SUB A, [expr]	C, Z
41	9	3	AND reg[expr], expr	Z	5B	4	1	MOV A, X	Z	13	7	2	SUB A, [X+expr]	C, Z
42	10	3	AND reg[X+expr], expr	Z	5C	4	1	MOV X, A		14	7	2	SUB [expr], A	C, Z
64	4	1	ASL A	C, Z	5D	6	2	MOV A, reg[expr]	Z	15	8	2	SUB [X+expr], A	C, Z
65	7	2	ASL [expr]	C, Z	5E	7	2	MOV A, reg[X+expr]	Z	16	9	3	SUB [expr], expr	C, Z
66	8	2	ASL [X+expr]	C, Z	5F	10	3	MOV [expr], [expr]		17	10	3	SUB [X+expr], expr	C, Z
67	4	1	ASR A	C, Z	60	5	2	MOV reg[expr], A		4B	5	1	SWAP A, X	Z
68	7	2	ASR [expr]	C, Z	61	6	2	MOV reg[X+expr], A		4C	7	2	SWAP A, [expr]	Z
69	8	2	ASR [X+expr]	C, Z	62	8	3	MOV reg[expr], expr		4D	7	2	SWAP X, [expr]	
9x	11	2	CALL		63	9	3	MOV reg[X+expr], expr		4E	5	1	SWAP A, SP	Z
39	5	2	CMP A, expr	if (A=B) Z=1 if (A<B) C=1	3E	10	2	MVI A, [[expr]++]	Z	47	8	3	TST [expr], expr	Z
3A	7	2	CMP A, [expr]		3F	10	2	MVI [[expr]++], A		48	9	3	TST [X+expr], expr	Z
3B	8	2	CMP A, [X+expr]		40	4	1	NOP		49	9	3	TST reg[expr], expr	Z
3C	8	3	CMP [expr], expr		29	4	2	OR A, expr	Z	4A	10	3	TST reg[X+expr], expr	Z
3D	9	3	CMP [X+expr], expr		2A	6	2	OR A, [expr]	Z	72	4	2	XOR F, expr	C, Z
73	4	1	CPL A	Z	2B	7	2	OR A, [X+expr]	Z	31	4	2	XOR A, expr	Z
78	4	1	DEC A	C, Z	2C	7	2	OR [expr], A	Z	32	6	2	XOR A, [expr]	Z
79	4	1	DEC X	C, Z	2D	8	2	OR [X+expr], A	Z	33	7	2	XOR A, [X+expr]	Z
7A	7	2	DEC [expr]	C, Z	2E	9	3	OR [expr], expr	Z	34	7	2	XOR [expr], A	Z
7B	8	2	DEC [X+expr]	C, Z	2F	10	3	OR [X+expr], expr	Z	35	8	2	XOR [X+expr], A	Z
30	9	1	HALT		43	9	3	OR reg[expr], expr	Z	36	9	3	XOR [expr], expr	Z
74	4	1	INC A	C, Z	44	10	3	OR reg[X+expr], expr	Z	37	10	3	XOR [X+expr], expr	Z
75	4	1	INC X	C, Z	71	4	2	OR F, expr	C, Z	45	9	3	XOR reg[expr], expr	Z
										46	10	3	XOR reg[X+expr], expr	Z

Note 1 Interrupt acknowledge to Interrupt Vector table = 13 cycles.

Note 2 The number of cycles required by an instruction is increased by one for instructions that span 128 byte page boundaries in the flash memory space.

2.5 Instruction Formats

The M8C has a total of seven instruction formats that use instruction lengths of one, two, and three bytes. All instruction bytes are taken from the program memory (flash), using an address and data bus that are independent from the address and data buses used for register and RAM access.

While examples of instructions are given in this section, refer to the *PSoC Designer Assembly Language User Guide* for detailed information on individual instructions.

2.5.1 One-Byte Instructions

Many instructions, such as some of the MOV instructions, have single-byte forms because they do not use an address or data as an operand. As shown in Table 2-3, one-byte instructions use an 8-bit opcode. The set of one-byte instructions are divided into four categories, according to where their results are stored.

Table 2-3. One-Byte Instruction Format

Byte 0
8-Bit Opcode

The first category of one-byte instructions are those that do not update any register or RAM. Only the one-byte NOP and SSC instructions fit this category. While the ***program counter*** is incremented as these instructions execute, they do not cause any other internal M8C registers to update, nor do these instructions directly affect the register space or the RAM address space. The SSC instruction causes SROM code to run, which modifies RAM and the M8C internal registers.

The second category contains the two PUSH instructions. The PUSH instructions are unique because they are the only one-byte instructions that modify a RAM address. These instructions automatically increment the SP.

The third category contains the HALT instruction. The HALT instruction is unique because it is the only one-byte instruction that modifies a user register. The HALT instruction modifies user register space address FFh (CPU_SCR0 register).

The final category for one-byte instructions are those that update the internal M8C registers. This category holds the largest number of instructions: ASL, ASR, CPL, DEC, INC, MOV, POP, RET, RETI, RLC, ROMX, RRC, SWAP. These instructions cause the A, X, and SP registers or SRAM to update.

2.5.2 Two-Byte Instructions

The majority of M8C instructions are two bytes in length. While these instructions are divided into categories identical to the one-byte instructions, this does not provide a useful distinction between the three two-byte instruction formats that the M8C uses.

Table 2-4. Two-Byte Instruction Formats

Byte 0	Byte 1
4-Bit Opcode	12-Bit Relative Address
8-Bit Opcode	8-Bit Data
8-Bit Opcode	8-Bit Address

The first two-byte instruction format, shown in the first row of Table 2-4, is used by short jumps and calls: CALL, JMP, JACC, INDEX, JC, JNC, JNZ, JZ. This instruction format uses only four bits for the instruction opcode, leaving 12 bits to store the relative destination address in a two's-complement form. These instructions can change program execution to an address relative to the current address by –2048 or +2047.

The second two-byte instruction format, shown in the second row of Table 2-4, is used by instructions that employ the Source Immediate addressing ***mode*** (see the *PSoC Designer Assembly Language User Guide*). The destination for these instructions is an internal M8C register, while the source is a constant value. An example of this type of instruction is `ADD A, 7`.

The third two-byte instruction format, shown in the third row of Table 2-4, is used by a wide range of instructions and addressing modes. The following is a list of the addressing modes that use this third two-byte instruction format:

- Source Direct (`ADD A, [7]`)
- Source Indexed (`ADD A, [X+7]`)
- Destination Direct (`ADD [7], A`)
- Destination Indexed (`ADD [X+7], A`)
- Source Indirect Post Increment (`MVI A, [7]`)
- Destination Indirect Post Increment (`MVI [7], A`)

For more information on addressing modes see the *PSoC Designer Assembly Language User Guide*.

2.5.3 Three-Byte Instructions

The three-byte instruction formats are the second most prevalent instruction formats. These instructions need three bytes because they either move data between two addresses in the user accessible address space (registers and RAM) or they hold 16-bit absolute addresses as the destination of a long jump or long call.

Table 2-5. Three-Byte Instruction Formats

Byte 0	Byte 1	Byte 2
8-Bit Opcode	16-Bit Address (MSB, LSB)	
8-Bit Opcode	8-Bit Address	8-Bit Data
8-Bit Opcode	8-Bit Address	8-Bit Address

The first instruction format, shown in the first row of Table 2-5, is used by the `LJMP` and `LCALL` instructions. These instructions change program execution unconditionally to an absolute address. The instructions use an 8-bit opcode, leaving room for a 16-bit destination address.

The second three-byte instruction format, shown in the second row of Table 2-5, is used by the following two addressing modes:

- Destination Direct Source Immediate (`ADD [7], 5`)
- Destination Indexed Source Immediate (`ADD [X+7], 5`)

The third three-byte instruction format, shown in the third row of Table 2-5, is for the Destination Direct Source Direct addressing mode, which is used by only one instruction. This instruction format uses an 8-bit opcode followed by two 8-bit addresses. The first address is the destination address in RAM, while the second address is the source address in RAM. The following is an example of this instruction:

```
MOV [7], [5]
```

2.6 Register Definitions

The following register is associated with the CPU Core (M8C). The register description has an associated register table showing the bit structure. The bits that are grayed out in the table are reserved bits and are not detailed in the register description that follows. Always write reserved bits with a value of '0'.

2.6.1 CPU_F Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
x,F7h	CPU_F	PgMode[1:0]			XIO		Carry	Zero	GIE	RL : 02

Legend
L The AND F, expr; OR F, expr; and XOR F, expr flag instructions can be used to modify this register.
x An "x" before the comma in the address field indicates that this register can be read or written to no matter what bank is used.

The M8C Flag Register (CPU_F) provides read access to the M8C flags.

Bits 7 and 6: PgMode[1:0]. PgMode determines how the CUR_PP, STK_PP, and IDX_PP registers are used in forming effective RAM addresses for Direct Address mode and Indexed Address mode operands. PgMode also determines whether the stack page is determined by the STK_PP or IDX_PP register. (See the Register Definitions on page 41 in the RAM Paging chapter.)

Bit 4: XIO. The I/O bank select bit, also known as the register bank select bit, is used to select the register bank that is active for a register read or write. This bit allows the enCoRe V device to have 512 8-bit registers and is thought of as the ninth address bit for registers. The address space accessed when the XIO bit is set to '0' is called ***user space***, while address space accessed when the XIO bit is set to '1' is called ***configuration space***.

Bit 2: Carry. The Carry flag bit is set or cleared in response to the result of several instructions. It is also manipulated by the flag logic opcodes (for example, OR F, 4). See the *PSoC Designer Assembly Language User Guide* for more details.

Bit 1: Zero. The Zero flag bit is set or cleared in response to the result of several instructions. It is also manipulated by the flag logic opcodes (for example, OR F, 2). See the *PSoC Designer Assembly Language User Guide* for more details.

Bit 0: GIE. The state of the Global Interrupt Enable bit determines whether interrupts (by way of the interrupt request (IRQ)) are recognized by the M8C. This bit is set or cleared using the flag logic instructions (for example, OR F, 1). GIE is also automatically cleared when an interrupt is processed, after the flag byte is stored on the stack, preventing nested interrupts. If wanted, set the bit in an ***interrupt service routine (ISR)***. For GIE=1, the M8C samples the IRQ input for each instruction. For GIE=0, the M8C ignores the IRQ.

For additional information, refer to the CPU_F register on page 208.

2.6.2 Related Registers

The following registers are related to the M8C block:

- CPU_SCR1 register on page 210.
- CPU_SCR0 register on page 211.

3. Supervisory ROM (SROM)

This chapter discusses the Supervisory ROM (SROM) functions. For a quick reference of all enCoRe V registers in address order, refer to the Register Reference chapter on page 163.

3.1 Architectural Description

The SROM holds code that boots a enCoRe V device, calibrates circuitry, and performs flash operations. The functions provided by the SROM are called from code stored in the flash or by device programmers.

The SROM is used to boot the part and provide interface functions to the flash blocks. Table 3-1 lists the SROM functions. The SROM functions are accessed by executing the Supervisory System Call instruction (SSC), which has an opcode of 00h. Before executing the SSC, the M8C's accumulator needs to load with the wanted SROM function code from Table 3-1.

Attempting to access undefined functions (Reserved functions) causes a HALT. The SROM functions execute code with calls; therefore, the functions require stack space. With the exception of Reset, all of the SROM functions have a parameter block in SRAM that you must configure before executing the SSC.

Table 3-2 lists all possible parameter block variables. The meaning of each parameter, with regards to a specific SROM function, is described later in this chapter. Because the SSC instruction clears the CPU_F PgMode bits, all parameter block variable addresses are in SRAM Page 0. The CPU_F value is automatically restored at the end of the SROM function.

The MVR_PP and MVW_PP pointers are not disabled by clearing the CPU_F PgMode bits. Therefore, the POINTER parameter is interpreted as an address in the page indicated by the MVI page pointers, when the supervisory operation is called. This allows the data buffer used in the supervisory operation to be located in any SRAM page. (See the RAM Paging chapter on page 38 for more details regarding the MVR_PP and MVW_PP pointers.)

Table 3-1. List of SROM Functions

Function Code	Function Name	Required Stack Space	Page
00h	SWBootReset	0	33
01h	ReadBlock	7	34
02h	WriteBlock	7	34
03h	EraseBlock	5	35
06h	TableRead	3	35
07h	CheckSum	4	36
08h	Calibrate0	4	36
09h	Calibrate1	3	36
0Ah	WriteAndVerify	7	36
0Fh	HWBootReset	3	37

Note ProtectBlock and EraseAll (described on page 35) SROM functions are not listed in this table because they are dependent on external programming.

Table 3-2. SROM Function Variables

Variable Name	SRAM Address
KEY1/RETURN CODE	0,F8h
KEY2	0,F9h
BLOCKID	0,FAh
POINTER	0,FBh
CLOCK	0,FCh
Reserved	0,FDh
DELAY	0,FEh
Reserved	0,FFh

Note CLOCK and DELAY are ignored and are reserved for future use.

Two important variables that are used for all functions are KEY1 and KEY2. These variables are used to help discriminate between valid SSCs and inadvertent SSCs. KEY1 must always have a value of 3Ah, while KEY2 must have the same value as the stack pointer when the SROM function begins execution. This is the SP (Stack Pointer) value when the SSC opcode is executed, plus three. For all SROM functions except SWBootReset, if either of the keys do not match the expected values, the M8C halts. The SWBootReset function does not check the key values. It only checks to see if the accumulator's value is 00h.

The following code example puts the correct value in KEY1 and KEY2. The code is preceded by a HALT, to force the

program to jump directly into the setup code and not accidentally run into it.

```
1.        halt
2.        SSC0P: mov [KEY1], 3ah
3.        mov X, SP
4.        mov A, X
5.        add A, 3
6.        mov [KEY2], A
```

3.1.1 Additional SROM Feature

Return Codes: These aid in the determination of success or failure of a particular function. The return code is stored in KEY1's position in the parameter block. The Checksum and TableRead functions do not have return codes because KEY1's position in the parameter block is used to return other data.

Table 3-3. SROM Return Code Meanings

Return Code Value	Description
00h	Success.
01h	Function not allowed because of block level protection.
02h	Software reset without hardware reset.
03h	Fatal error, SROM halted.
04h	Write and Verify error.
06h	Failure of Smartwrite parameters CheckSum

Note Read, write, and erase operations may fail if the target block is read or write protected. Block protection levels are set during device programming and cannot be modified from code in the enCoRe V device.

3.1.2 SROM Function Descriptions

3.1.2.1 SWBootReset Function

The SROM function SWBootReset is responsible for transitioning the device from a reset state to running user code. See Chapter "System Resets" on page 114 for more information on what events causes the SWBootReset function to execute.

The SWBootReset function executes whenever the SROM is entered with an M8C accumulator value of 00h; the SRAM parameter block is not used as an input to the function. This happens, by design, after a hardware reset because the M8C's accumulator is reset to 00h or when user code executes the SSC instruction with an accumulator value of 00h.

If the checksum of the calibration data is valid, the SWBootReset function ends by setting the internal M8C registers (CPU_SP, CPU_PC, CPU_X, CPU_F, CPU_A) to 00h, writing 00h to most SRAM addresses in SRAM Page 0, and then begins to execute user code at address 0000h. See Table 3-5 and the following paragraphs for more information on which SRAM addresses are modified. If the checksum is not valid, an internal reset is executed and the boot process starts over. If this condition occurs, the internal reset status bit (IRESS) is set in the CPU_SCR1 register.

In devices with more than 256 bytes of SRAM, no SRAM is modified by the SWBootReset function in SRAM pages numbered higher than '0'.

Table 3-5 documents the value of all the SRAM addresses in Page 0 after a successful SWBootReset. A value of "xx" indicates that the SRAM address is not modified by the SWBootReset function. A hex value indicates that the address always has the indicated value after a successful SWBootReset. A "??" indicates that the value, after a SWBootReset, is determined by the value of the IRAMDIS bit in the CPU_SCR1 register. If IRAMDIS is not set, these addresses are initialized to 00h. If IRAMDIS is set, these addresses are not modified by a SWBootReset after a watchdog reset.

The IRAMDIS bit allows the preservation of variables even if a watchdog reset (WDR) occurs. The IRAMDIS bit is reset by all system resets except watchdog reset. Therefore, this bit is only useful for watchdog resets and not general resets.

Table 3-5. SRAM Map Post SWBootReset (00h)

Address	0	1	2	3	4	5	6	7
	8	9	A	B	C	D	E	F
0x0_	0x00	0x00	0x00	??	??	??	??	??
	??	??	??	??	??	??	??	??
0x1_	??	??	??	??	??	??	??	??
	??	??	??	??	??	??	??	??
0x2_	??	??	??	??	??	??	??	??
	??	??	??	??	??	??	??	??
0x3_	??	??	??	??	??	??	??	??
	??	??	??	??	??	??	??	??
0x4_	??	??	??	??	??	??	??	??
	??	??	??	??	??	??	??	??
0x5_	??	??	??	??	??	??	??	??
	??	??	??	??	??	??	??	??
0x6_	??	??	??	??	??	??	??	??
	??	??	??	??	??	??	??	??
0x7_	??	??	??	??	??	??	??	??
	??	??	??	??	??	??	??	??
0x8_	??	??	??	??	??	??	??	??
	??	??	??	??	??	??	??	??
0x9_	??	??	??	??	??	??	??	??
	??	??	??	??	??	??	??	??
0xA_	??	??	??	??	??	??	??	??
	??	??	??	??	??	??	??	??
0xB_	??	??	??	??	??	??	??	??
	??	??	??	??	??	??	??	??
0xC_	??	??	??	??	??	??	??	??
	??	??	??	??	??	??	??	??
0xD_	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
0xE_	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
0xF_	0x00	0x00	0x00	0x00	0x00	0x00	??	??
	0x00 0x02 0x06	Xx	0x00	0x00	0xn	xx	0x00	0x00

Address F8h is the return code byte for all SROM functions (except Checksum and TableRead); for this function, the only acceptable values are 00h, 02h, and 06h. Address FCh is the fail count variable. After POR, WDR, or XRES, the variable is initialized to 00h by the SROM. Each time the Checksum fails, the fail count is incremented. Therefore, if it takes two passes through SWBootReset to get a good Checksum, the fail count is 01h.

3.1.2.2 ReadBlock Function

The ReadBlock function is used to read 128 contiguous bytes from flash: a block. The enCoRe V device has 32 KB of flash and has two hundred fifty-six 128-byte blocks. Valid block IDs are 00h to FFh.

Table 3-6. Flash Memory Organization

enCoRe V Device	Amount of Flash	Amount of SRAM	Number of Blocks per Bank	Number of Banks
CY7C6xxxx,	32 KB	2K Bytes	256	1

The first thing the ReadBlock function does is check the protection bits to determine if the wanted BLOCKID is readable. If read protection is turned on, the ReadBlock function exits setting the accumulator and KEY2 back to 00h. KEY1 has a value of 01h indicating a read failure.

If read protection is not enabled, the function reads 128 bytes from the flash using a `ROMX` instruction and stores the results in SRAM using an `MVI` instruction. The 128 bytes are stored in SRAM, beginning at the address indicated by the value of the POINTER parameter. When the ReadBlock completes successfully, the accumulator, KEY1, and KEY2 has a value of 00h.

Note An `MVI [expr],` A stores the flash block contents in SRAM meaning that you can use the MVW_PP register to indicate which SRAM pages receive the data.

Table 3-7. ReadBlock Parameters (01h)

Name	Address	Type	Description
MVW_PP	0,D5h	Register	`MVI` write page pointer register.
KEY1	0,F8h	RAM	3Ah.
KEY2	0,F9h	RAM	Stack Pointer value+3, when `SSC` is executed.
BLOCKID	0,FAh	RAM	Flash block number.
POINTER	0,FBh	RAM	Addresses in SRAM to store returned data.

3.1.2.3 WriteBlock Function

The WriteBlock function stores data in the flash. No verification of the data is performed, but execution time is about 1 ms less than the WriteAndVerify function. The WriteAndVerify function is the recommended method for altering the data in one flash block (see “WriteAndVerify Function” on page 36). Data moves 128 bytes at a time from SRAM to flash. This is a two-step process, the first step is to load the page latch with 128 bytes of data and it is followed by the programming of the corresponding block of flash. No erase is needed before WriteBlock.

If write protection is turned on, then the WriteBlock function exits, setting the accumulator and KEY2 back to 00h. KEY1 has a value of 01h, indicating a write failure. Write protection is set when the enCoRe V device is programmed externally and cannot be changed through the `SSC` function.

The BLOCKID of the flash block, where the data is stored, must be determined and stored at SRAM address FAh. Valid block IDs are 00h to FFh.

An `MVI A, [expr]` instruction is used to move data from SRAM into flash. Therefore, use the `MVI` read pointer (MVR_PP register) to specify which SRAM page from which data is pulled. Using the `MVI` read pointer and the parameter blocks POINTER value allows the SROM WriteBlock function to move data from any SRAM page into any flash block.

The SRAM address, the first of the 128 bytes to store in flash, is indicated using the POINTER variable in the parameter block (SRAM address FBh).

Table 3-8. WriteBlock Parameters (02h)

Name	Address	Type	Description
MVR_PP	0,D4h	Register	`MVI` read page pointer register.
KEY1	0,F8h	RAM	3Ah.
KEY2	0,F9h	RAM	Stack Pointer value+3, when SSC is executed.
BLOCKID	0,FAh	RAM	Flash block number.
POINTER	0,FBh	RAM	First of 128 addresses in SRAM, where the data to be stored in flash, is located before calling WriteBlock.

3.1.2.4 EraseBlock Function

The EraseBlock function is not recommended for use. The functionality is redundant with the WriteBlock and WriteAndVerify functions. The only practical use is for clearing all data in a 128 byte block of contiguous bytes in flash to 00h. If used, it should not be called repeatedly on the same block. It may be used between WriteAndVerify or WriteBlock operations.

If write protection is turned on, then the EraseBlock function exits, setting the accumulator and KEY2 back to 00h. KEY1 has a value of 01h, indicating a write failure.

To set up the parameter block for the EraseBlock function, store the correct key values in KEY1 and KEY2. The block number to erase must be stored in the BLOCKID variable.

Table 3-9. EraseBlock Parameters (03h)

Name	Address	Type	Description
KEY1	0,F8h	RAM	3Ah.
KEY2	0,F9h	RAM	Stack Pointer value+3, when SSC is executed.
BLOCKID	0,FAh	RAM	Flash block number.

3.1.2.5 ProtectBlock Function

The enCoRe V devices offer flash protection on a block-by-block basis. Table 3-10 lists the protection modes available. In the table, ER and EW indicate the ability to perform external reads and writes (that is, by an external programmer). For internal writes, IW is used. Internal reading is always permitted by way of the `ROMX` instruction. An SR indicates the ability to read by way of the SROM ReadBlock function.

In this table, note that all protection is removed by EraseAll.

Table 3-10. Protect Block Modes

Mode	Settings	Description	In PSoC Designer
00b	SR ER EW IW	Unprotected	U = Unprotected
01b	$\overline{SR}$ ER EW IW	Read protect	F = Factory upgrade
10b	$\overline{SR}$ $\overline{ER}$ $\overline{EW}$ IW	Disable external write	R = Field upgrade
11b	$\overline{SR}$ $\overline{ER}$ $\overline{EW}$ $\overline{IW}$	Disable internal write	W = Full protection

Table 3-11. Protection Level Bit Packing

7	6	5	4	3	2	1	0
Block n+3		Block n+2		Block n+1		Block n	

3.1.2.6 TableRead Function

The TableRead function gives the user access to part-specific data stored in the flash during manufacturing. The flash for these tables is separate from the program flash and is not directly accessible. It also returns a revision ID for the die (do not confuse this with the silicon ID stored in the Table 0 row in Table 3-14).

A summary of the information stored in the tables for the flash is contained in Table 3-14.CY8C20X66A/AS/LCY8C20X46A/46AS/96A/46L/96L

Table 3-12. TableRead Parameters (06h)

Name	Address	Type	Description
KEY1	0,F8h	RAM	3Ah.
KEY2	0,F9h	RAM	Stack Pointer value+3, when SSC is executed.
BLOCKID	0,FAh	RAM	Table number to read.

3.1.2.7 EraseAll Function

The EraseAll function performs a series of steps that destroys the user data in the flash banks and resets the protection block in each flash bank to all zeros (the unprotected state). This function is only executed by an external programmer. If EraseAll is executed from code, the M8C HALTs without touching the flash or protections. See Table 3-13. The three other hidden blocks above the protection block, in each flash bank, are not affected by the EraseAll.

Table 3-13. EraseAll Parameters (05h)

Name	Address	Type	Description
KEY1	0,F8h	RAM	3Ah.
KEY2	0,F9h	RAM	Stack Pointer value+3, when SSC is executed.

Table 3-14. Flash Tables with Assigned Values

Table	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Table 0	Silicon ID High Byte Expected Numbers = # Bits Used to Encode = Max Values (including 0) = Bits Targeted =		Reserved					
Table 1	IMO 6 MHz trim	IMO 12 MHz trim	IMO 24 MHz trim	IMO 24 MHz USB trim (high power)	Reserved			
Table 2	Reserved							
Table 3	Reserved							

3.1.2.8 Checksum Function

The Checksum function calculates a 16-bit checksum over a user specifiable number of blocks, within a single flash bank starting at block zero. The BLOCKID parameter is used to pass in the number of blocks to checksum. A BLOCKID value of '1' calculates the checksum of only block 0, while a BLOCKID value of '0' calculates the checksum of the entire flash bank.

The 16-bit checksum is returned in KEY1 and KEY2. The parameter KEY1 holds the lower 8 bits of the checksum and the parameter KEY2 holds the upper 8 bits of the checksum.

Table 3-15. Checksum Parameters (07h)

Name	Address	Type	Description
KEY1	0,F8h	RAM	3Ah.
KEY2	0,F9h	RAM	Stack Pointer value+3, when SSC is executed.
BLOCKID	0,FAh	RAM	Number of flash blocks from which to calculate the checksum.

3.1.2.9 Calibrate0 Function

This function may be executed at any time to set all calibration values. However, it is unnecessary to call this function; it is simply documented for completeness. The calibration values are accessed using the TableRead function, which is described in the section TableRead Function, on page 35.

Table 3-16. Calibrate0 Parameters (08h)

Name	Address	Type	Description
KEY1	0,F8h	RAM	3Ah.
KEY2	0,F9h	RAM	Stack Pointer value+3, when SSC is executed.

3.1.2.10 Calibrate1 Function

While the Calibrate1 function is a completely separate function from Calibrate0, they perform the same task, which is to transfer the calibration values stored in a special area of flash to their appropriate registers. What is unique about Calibrate1 is that it calculates a checksum of the calibration data and, if that checksum is determined as invalid, Calibrate1 causes a ***hardware reset*** by generating an internal reset. If this occurs, it is indicated by setting the Internal Reset Status bit (IRESS) in the CPU_SCR1 register.

The Calibrate1 function uses SRAM to calculate a checksum of the calibration data. The POINTER value is used to indicate the address of a 38-byte buffer used by this function. When the function completes, the 38 bytes are set to 00h.

An `MVI A, [expr]` and an `MVI [expr], A` instruction are used to move data between SRAM and flash. Therefore, the `MVI` write pointer (MVW_PP) and the `MVI` read pointer (MVR_PP) must be specified to the same SRAM page to control the page of RAM used for the operations.

Calibrate1 was created as a sub-function of SWBootReset and the Calibrate1 function code was added to provide direct access. For more information on how Calibrate1 works, see SWBootReset Function on page 33.

This function may be executed at any time to reset all calibration values. However, it is unnecessary to call this function; it is simply documented for completeness. The calibration values are accessed using the TableRead function, which is described in the section TableRead Function on page 35.

Table 3-17. Calibrate1 Parameters (09h)

Name	Address	Type	Description
KEY1	0,F8h	RAM	3Ah.
KEY2	0,F9h	RAM	Stack Pointer value+3, when SSC is executed.
POINTER	0,FBh	RAM	First of 30 SRAM addresses used by this function.
MVR_PP	0,D4h	Register	`MVI` write page pointer.
MVW_PP	0,D5h	Register	`MVI` read page pointer.

3.1.2.11 WriteAndVerify Function

WriteAndVerify is the recommend function for modifying one block of data in flash. The WriteAndVerify function works exactly the same as the WriteBlock function except that the flash data is verified after the Write. The execution time is about 1 ms longer than WriteBlock (but still within the Twrite

spec). The function performs a three-step process. In the first step, 128 bytes of data are moved from SRAM to the flash. In the second step, flash is programmed with the data. In the final step, the flash data is compared against the input data values, thus verifying that the write was successful. The write and verify is one SROM operation; therefore, the SROM is not exited until the verify is completed.

The parameters for this block are identical to the WriteBlock (see WriteBlock Function on page 34). If the verify operation fails, the 04h error code is returned at SRAM address F8h

Table 3-18. WriteAndVerify Parameters (0Ah)

Name	Address	Type	Description
KEY1	0,F8h	RAM	3Ah.
KEY2	0,F9h	RAM	Stack Pointer value+3, when SSC is executed.
BLOCKID	0,FAh	RAM	Flash block number.
POINTER	0,FBh	RAM	First of 128 addresses in SRAM, where the data to be stored in flash, is located before calling WriteBlock.

3.1.2.12 HWBootReset Function

The HWBootReset function is used to force a hardware reset. A hardware reset causes all registers to go back to their POR state. Then, the SROM SWBootReset function executes, followed by flash code execution beginning at address 0x0000.

The HWBootReset function only requires that the CPU_A, KEY1, and KEY2 be set up correctly. As with all other SROM functions, if the setup is incorrect, the SROM executes a HALT. Then, either a POR, XRES, or WDR is needed to clear the HALT. See Chapter “System Resets” on page 114 for more information.

Table 3-19. HWBootReset Parameters (0Fh)

Name	Address	Type	Description
KEY1	0,F8h	RAM	3Ah.
KEY2	0,F9h	RAM	Stack Pointer value+3, when SSC is executed.

3.2 Register Definitions

This chapter has no register detail information because there are no registers directly assigned to the Supervisory ROM.

4. RAM Paging

This chapter explains the enCoRe V device's use of RAM Paging and its associated registers. For a complete table of the RAM paging registers, refer to the Summary Table of the Core Registers on page 24. For a quick reference of all enCoRe V registers in address order, refer to the Register Reference chapter on page 163.

4.1 Architectural Description

The M8C is an 8-bit CPU with an 8-bit memory address bus. The memory address bus allows the M8C to access up to 256 bytes of SRAM, to increase the amount of available SRAM and preserve the M8C ***assembly*** language. The enCoRe V device has 1K and 2K bytes of SRAM with eight pages of memory architecture.

To take full advantage of the paged memory architecture of the enCoRe V device, you use several registers and manage two CPU_F register bits. However, the power-on-reset (POR) value for all of the paging registers and CPU_F bits is zero. This places the enCoRe V device in a mode identical to devices with only 256 bytes of SRAM. There is no need to understand all of the paging registers to take advantage of the additional SRAM available in some devices. To use the additional SRAM pages, modify the memory paging logic reset state.

The memory paging architecture consists of five areas:

- Stack Operations
- Interrupts
- MVI Instructions
- Current Page Pointer
- Indexed Memory Page Pointer

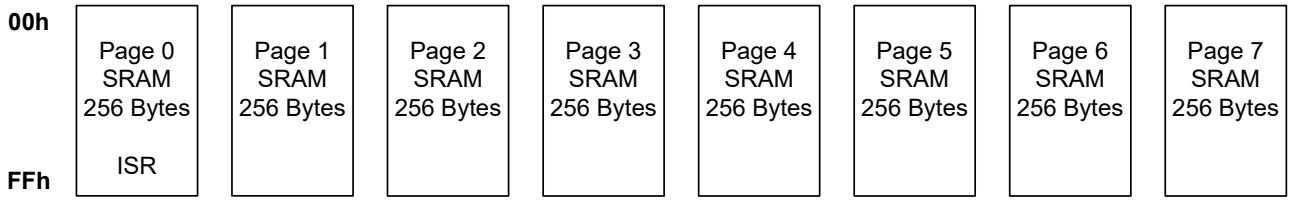
The first three of these areas do not depend upon the CPU_F register's PgMode bits and are covered in the subsections after Basic Paging. The function of the last two depend upon the CPU_F PgMode bits and are covered last.

4.1.1 Basic Paging

To increase the amount of SRAM, the M8C accesses memory page bits. The memory page bits are located in the CUR_PP register and allow selection of one of eight SRAM pages. In addition to setting the page bits, Page mode is enabled by setting the CPU_F[7] bit. If Page mode is not enabled, the page bits are ignored and all non-stack memory access is directed to Page 0.

After Page mode is enabled and the page bits are set, all instructions that operate on memory access the SRAM page indicated by the page bits. The exceptions to this are the instructions that operate on the stack and the `MVI` instructions: `PUSH`, `POP`, `LCALL`, `RETI`, `RET`, `CALL`, and `MVI`. See the description of Stack Operations and MVI Instructions for a more detailed discussion.

Figure 4-1. Data Memory Organization



4.1.2 Stack Operations

As mentioned previously, the paging architecture's reset state puts the enCoRe Vin a mode identical to that of a 256-byte device. Therefore, upon reset, all memory accesses are to Page 0. The SRAM page that stack operations use is determined by the value of the three least significant bits (LSb) of the Stack Page Pointer register (STK_PP). Stack operations have no dependency on the PgMode bits in the CPU_F register. Stack operations are those that use the Stack Pointer (SP) to calculate their affected address. Refer

to the *PSoC Designer Assembly Language User Guide* for more information on all M8C instructions.

Stack memory accesses are a special case. If they were not, the stack could fragment across several pages. To prevent the stack from fragmenting, all instructions that operate on the stack automatically use the page indicated by the STK_PP register. Therefore, if the program encounters a CALL, the enCoRe V device automatically pushes the program counter onto the stack page indicated by STK_PP. After the program counter is pushed, the SRAM paging mode automatically switches back to the precall mode. All other stack operations, such as RET and POP, follow the same rule as CALL. The stack is confined to a single SRAM page and the Stack Pointer wraps from 00h to FFh and FFh to 00h. The user code must ensure that the stack is not damaged because of stack wrapping.

Because the value of the STK_PP register can change at any time, it is theoretically possible to manage the stack in such a way as to allow it to grow beyond one SRAM page or manage multiple stacks. However, the only supported use of the STK_PP register is when its value is set before the first stack operation and not changed again.

4.1.3 Interrupts

Interrupts, in a multipage SRAM enCoRe V device, operate the same as interrupts in a 256-byte device. However, because the CPU_F register is automatically set to 00h on an interrupt and because of the nonlinear nature of interrupts in a system, other parts of the memory paging architecture can be affected.

Interrupts are an abrupt change in program flow. If no special action is taken on interrupts by the enCoRe V device, the ***interrupt service routine (ISR)*** could be thrown into any SRAM page. To prevent this problem, the special addressing modes for all memory accesses, except for stack and MVI, are disabled when an ISR is entered. The special addressing modes are disabled when the CUP_F register is cleared. At the end of the ISR, the previous SRAM addressing mode is restored when the CPU_F register value is restored by the RETI instruction.

All interrupt service ***routine*** code starts execution in SRAM Page 0. If the ISR must change to another SRAM page, do this by changing the values of the CPU_F[7:6] bits to enable the special SRAM addressing modes. However, any change made to the CUR_PP, IDX_PP, or STK_PP registers persists after the ISR returns. Therefore, have the ISR save the current value of any paging register it modifies and restore its value before the ISR returns.

4.1.4 MVI Instructions

MVI instructions use data page pointers of their own (MVR_PP and MVW_PP). This allows a data buffer to be located away from other program variables, but accessible without changing the Current Page Pointer (CUR_PP).

An MVI instruction performs three memory operations. Both forms of the MVI instruction access an address in SRAM that holds the data pointer (a memory read 1st access), incrementing that value and then storing it back in SRAM (a memory write 2nd access). This pointer value must reside in the current page, just as all other nonstack and nonindexed operations on memory. However, the third memory operation uses the MVx_PP register. This third memory access is either a read or a write, depending upon which MVI instruction is used. The MVR_PP pointer is used for the MVI instruction that moves data into the accumulator. The MVW_PP pointer is used for the MVI instruction that moves data from the accumulator into SRAM. The MVI pointers are always enabled, regardless of the state of the Flag register page bits (CPU_F register).

4.1.5 Current Page Pointer

The Current Page Pointer determines which SRAM page is used for all memory accesses. Normal memory accesses are those not covered by other pointers including all nonstack, non-MVI, and nonindexed memory access instructions. The normal memory access instructions have the SRAM page they operate on determined by the value of the CUR_PP register. By default, the CUR_PP register has no affect on the SRAM page that is used for normal memory access, because all normal memory access is forced to SRAM Page 0.

The upper bit of the PgMode bits in the CPU_F register determine if the CUR_PP register affects normal memory access. When the upper bit of the PgMode bits is set to ‘0’, all normal memory access is forced to SRAM Page 0. This mode is automatically enabled when an Interrupt Service Routine (ISR) is entered. This is because, before the ISR is entered, the M8C pushes the current value of the CPU_F register onto the stack and then clears the CPU_F register. Therefore, by default, any normal memory access in an ISR is guaranteed to occur in SRAM Page 0.

When the RETI instruction is executed to end the ISR, the previous value of the CPU_F register is restored, returning to the previous page mode. This is the default ISR behavior and it is possible to change the PgMode bits in the CPU_F register while in an ISR. If the PgMode bits are changed while in an ISR, the pre-ISR value is still restored by the RETI; but if the CUR_PP register is changed in the ISR, the ISR is also required to restore the value before executing the RETI instruction.

When the upper bit of the PgMode bits is set to ‘1‘, all normal memory access is forced to the SRAM page indicated by the CUR_PP register value. Table 4-1 summarizes the PgMode bit values and the corresponding Memory Paging mode.

4.1.6 Index Memory Page Pointer

The Source Indexed and Destination Indexed addressing modes to SRAM are treated as a unique addressing mode in a enCoRe V device with more than one page of SRAM. An example of an indexed addressing mode is the `MOV A, [X+expr]` instruction. Register access has indexed addressing as well; however, those instructions are not affected by the SRAM paging architecture.

Important Note If you are not using assembly to program a enCoRe V device, be aware that the ***compiler*** writer may restrict the use of some memory paging modes. Review the conventions in your compiler's user guide for more information on restrictions or conventions associated with memory paging modes.

Indexed SRAM accesses operate in one of three modes:

- Index memory access modes are forced to SRAM Page 0.
- Index memory access modes are directed to the SRAM page indicated by the value in the STK_PP register.
- Index memory access is forced to the SRAM page indicated by the value in the IDX_PP register.

The mode is determined by the value of the PgMode bits in the CPU_F register. However, the final SRAM page that is used also requires setting either the Stack Page Pointer (STK_PP) register or the Index Page Pointer (IDX_PP) register. Table 4-1 shows the three indexed memory access modes. The third column of the table is provided for reference only.

Table 4-1. CPU_F PgMode Bit Modes

CPU_F PgMode BIts	Current SRAM Page	Indexed SRAM Page	Typical Use
00b	0	0	ISR*
01b	0	STK_PP	ISR with variables on stack
10b	CUR_PP	IDX_PP	
11b	CUR_PP	STK_PP	

* Mode used by SROM functions initiated by the SSC instruction.

After reset, the PgMode bits are set to 00b. In this mode, index memory accesses are forced to SRAM Page 0, just as they are in a enCoRe V device with only 256 bytes of SRAM. This mode is also automatically enabled when an interrupt occurs in a enCoRe V device and is considered the default ISR mode. This is because before the ISR is entered, the M8C pushes the current value of the CPU_F register onto the stack and then clears the CPU_F register. Thus, by default, any indexed memory access in an ISR is guaranteed to occur in SRAM Page 0. When the `RETI` instruction executes to end the ISR, the previous value of the CPU_F register is restored as is the previous page mode. Note that this ISR behavior is the default and that the PgMode bits in the CPU_F register may be changed while in an ISR. If the PgMode bits are changed while in an ISR, the pre-ISR value is still restored by the `RETI`; but if the STK_PP or IDX_PP registers are changed in the ISR, the ISR is also required to restore the values before executing the `RETI` instruction.

The most likely PgMode bit change, while in an ISR, is from the default value of 00b to 01b. In the 01b mode, indexed memory access is directed to the SRAM page indicated by the value of the STK_PP register. By using the PgMode, modification of the STK_PP register value is unnecessary. The STK_PP register determines on which SRAM page the stack is located. The 01b paging mode is intended to provide easy access to the stack, while in an ISR, by setting the CPU_X register (just X in instruction format) equal to value of SP using `MOV X, SP` instruction.

The two previous paragraphs covered two of the three indexed memory access modes: STK_PP and forced to SRAM Page 0. Note, as shown in Table 4-1, that the STK_PP mode for indexed memory access is available under two PgMode settings. The 01b mode is intended for ISR use and the 11b mode is intended for non-ISR use. The third indexed memory access mode requires the PgMode bits to be set to 10b. In this mode, indexed memory access is forced to the SRAM page indicated by the value of the IDX_PP register.

4.2 Register Definitions

The following registers are associated with RAM Paging and are listed in address order. The register descriptions have an associated register table showing the bit structure for that register. The bits in the tables that are grayed out are reserved bits and are not detailed in the register descriptions that follow. Always write reserved bits with a value of '0'. For a complete table of RAM Paging registers, refer to the Summary Table of the Core Registers on page 24.

4.2.1 TMP_DRx Registers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
x,6xh	TMP_DRx	Data[7:0]								RW : 00

Legend
x An 'x' before the comma in the address field indicates that this register can be read or written to no matter what bank is used. An "x" after the comma in the address field indicates that there are multiple instances of the register.

The Temporary Data Registers (TMP_DR0, TMP_DR1, TMP_DR2, and TMP_DR3) enhance the performance in multiple SRAM page enCoRe V devices.

These registers have no predefined function (for example, the compiler and hardware do not use these registers) and exist for the user to define.

Bits 7 to 0: Data[7:0]. Due to the paged SRAM architecture of enCoRe V devices with more than 256 bytes of SRAM, a value in SRAM is not always accessible without first changing the current page. The TMP_DRx registers are readable and writable registers that are provided to improve the performance of multiple SRAM page enCoRe V devices, by supplying some register space for data that is always accessible.

For an expanded listing of the TMP_DRx registers, refer to the Summary Table of the Core Registers on page 24. For additional information, refer to the TMP_DRx register on page 219.

4.2.2 CUR_PP Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,D0h	CUR_PP						Page Bits[2:0]			RW : 00

The Current Page Pointer Register (CUR_PP) sets the effective SRAM page for normal memory accesses in a multi-SRAM page enCoRe V device.

Bits 2 to 0: Page Bits[2:0]. These bits affect the SRAM page that is accessed by an instruction when the CPU_F[7:6] bits have a value of either 10b or 11b. Source Indexed, Destination Indexed addressing modes, and stack instructions, are never affected by the value of the CUR_PP register. See the STK_PP and IDX_PP registers for more information.

The Source Indirect Post Increment and Destination Indirect Post Increment addressing modes, better know as `MVI`, are only partially affected by the value of the CUR_PP register. For `MVI` instructions, the pointer address is in the SRAM page indicated by CUR_PP, but the address pointed to may be in another SRAM page.

See the MVR_PP and MVW_PP register descriptions for more information.

For additional information, refer to the CUR_PP register on page 188.

4.2.3 STK_PP Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,D1h	STK_PP						Page Bits[2:0]			RW : 0

The Stack Page Pointer Register (STK_PP) is used to set the effective SRAM page for stack memory accesses in a multi-SRAM page enCoRe V device.

Bits 2 to 0: Page Bits[2:0]. These bits have the potential to affect two types of memory access.

The purpose of this register is to determine on which SRAM page to store the stack. In the reset state, this register's value is 00h and the stack is in SRAM Page 0. However, if the STK_PP register value is changed, the next stack operation occurs on the SRAM page indicated by the new STK_PP value. Therefore, set the value of this register early in the program and never change it. If the program changes the STK_PP value after the stack grows, the program must ensure that the STK_PP value is restored when needed.

Note The impact of the STK_PP register on the stack is independent of the SRAM Paging bits in the CPU_F register.

The second type of memory accesses that the STK_PP register affects are indexed memory accesses when the CPU_F[7:6] bits are set to 11b. In this mode, Source Indexed and Destination Indexed memory accesses are directed to the stack SRAM page, rather than the SRAM page indicated by the IDX_PP register or SRAM Page 0.

For additional information, refer to the STK_PP register on page 189.

4.2.4 IDX_PP Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,D3h	IDX_PP						Page Bits[2:0]			RW : 0

The Index Page Pointer Register (IDX_PP) sets the effective SRAM page for indexed memory accesses in a multi-SRAM page enCoRe V device.

Bits 2 to 0: Page Bits[2:0]. These bits allow instructions, which use the Source Indexed and Destination Indexed address modes, to operate on an SRAM page that is not equal to the current SRAM page. However, the effect this register has on indexed addressing modes is only enabled when the CPU_F[7:6] is set to 10b.

When CPU_F[7:6] is set to 10b and an indexed memory access is made, the access is directed to the SRAM page indicated by the value of the IDX_PP register.

See the STK_PP register description for more information on other indexed memory access modes. For additional

4.2.5 MVR_PP Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,D4h	MVR_PP						Page Bits[2:0]			RW : 0

The MVI Read Page Pointer Register (MVR_PP) sets the effective SRAM page for MVI read memory accesses in a multi-SRAM page enCoRe V device.

Bits 2 to 0: Page Bits[2:0]. These bits are only used by the `MVI A, [expr]` instruction, not to be confused with the `MVI [expr], A` instruction covered by the MVW_PP register. This instruction is considered a read because data is transferred from SRAM to the microprocessor's A register (CPU_A).

When an `MVI A, [expr]` instruction is executed in a device with more than one page of SRAM, the SRAM address that is read by the instruction is determined by the value of the least significant bits in this register. However, the pointer for the `MVI A, [expr]` instruction is always located in the current SRAM page. See the *PSoC Designer Assembly Language User Guide* for more information on the `MVI A, [expr]` instruction.

The function of this register and the `MVI` instructions are independent of the SRAM Paging bits in the CPU_F register. For additional information, refer to the MVR_PP register on page 191.

4.2.6 MVW_PP Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,D5h	MVW_PP						Page Bits[2:0]			RW : 0

The MVI Write Page Pointer Register (MVW_PP) sets the effective SRAM page for MVI write memory accesses in a multi-SRAM page enCoRe V device.

Bits 2 to 0: Page Bits[2:0]. These bits are only used by the `MVI [expr], A` instruction, not to be confused with the `MVI A, [expr]` instruction covered by the MVR_PP register. This instruction is considered a write because data is transferred from the microprocessor's A register (CPU_A) to SRAM.

When an `MVI [expr], A` instruction is executed in a device with more than one page of SRAM, the SRAM address that is written by the instruction is determined by the value of the least significant bits in this register. However, the pointer for the `MVI [expr], A` instruction is always located in the current SRAM page. See the *PSoC Designer Assembly Language User Guide* for more information on the `MVI [expr], A` instruction.

The function of this register and the MVI instructions are independent of the SRAM Paging bits in the CPU_F register. For additional information, refer to the MVW_PP register on page 192.

4.2.7 Related Registers

- CPU_F Register on page 31.

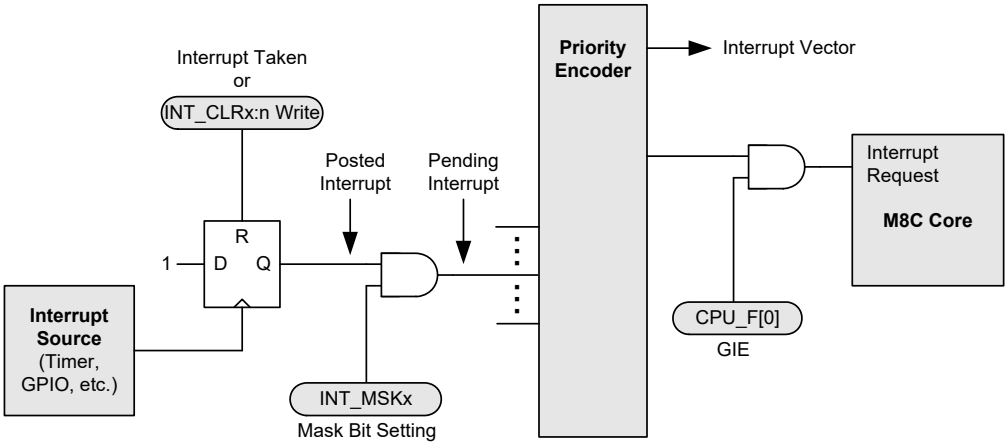
5. Interrupt Controller

This chapter presents the Interrupt Controller and its associated registers. The interrupt controller provides a mechanism for a hardware resource in enCoRe V devices to change program execution to a new address without regard to the current task being performed by the code being executed. For a quick reference of all enCoRe V registers in address order, refer to the Register Reference chapter on page 163.

5.1 Architectural Description

Figure 5-1 shows a block diagram of the Interrupt Controller, illustrating the concepts of ***posted interrupts*** and ***pending interrupts***.

Figure 5-1. Interrupt Controller Block Diagram



This is the sequence of events that occur during interrupt processing.

1. An interrupt becomes active, either because (a) the interrupt condition occurs (for example, a timer expires), (b) a previously posted interrupt is enabled through an update of an interrupt ***mask*** register, or (c) an interrupt is pending and GIE is set from '0' to '1' in the CPU Flag register.
2. The current executing instruction finishes.
3. The internal interrupt service routine (ISR) executes, taking 13 cycles. During this time, the following actions occur:
   - The PCH, PCL, and Flag register (CPU_F) are pushed onto the stack (in that order).
   - The CPU_F register clears. Because this clears the GIE bit to '0', additional interrupts are temporarily disabled.
   - The PCH (PC[15:8]) is cleared to zero.
   - The interrupt vector is read from the interrupt controller and its value is placed into PCL (PC[7:0]). This sets the program counter to point to the appropriate address in the interrupt table (for example, 0014h for the GPIO interrupt).
4. Program execution vectors to the interrupt table. Typically an `LJMP` instruction in the interrupt table sends execution to the user's interrupt service routine for this interrupt. (See Instruction Set Summary on page 27.)
5. The ISR executes. Interrupts are disabled because GIE = 0. In the ISR, interrupts can be re-enabled if necessary by setting GIE = 1 (take care to avoid stack overflow in this case).
6. The ISR ends with an `RETI` instruction. This pops the Flag register, PCL, and PCH from the stack, restoring those registers. The restored Flag register re-enables interrupts because GIE = 1 again.

7. Execution resumes at the next instruction, after the instruction that occurred before the interrupt. However, if there are more pending interrupts, the subsequent interrupts are processed before the next normal program instruction.

Interrupt Latency. The time between the assertion of an enabled interrupt and the start of its ISR is calculated using this equation:

Equation 1

$$\begin{gathered}\textit{Latency} = \\ \textit{Time for current instruction to finish} + \\ \textit{Time for M8C to change program counter to interrupt address} + \\ \textit{Time for LJMP instruction in interrupt table to execute.}\end{gathered}$$

For example, if the 5-cycle `JMP` instruction is executing when an interrupt becomes active, the total number of CPU clock cycles before the ISR begins is:

Equation 2

$$\begin{gathered}\textit{(1 to 5 cycles for JMP to finish)} + \\ \textit{(13 cycles for interrupt routine)} + \\ \textit{(7 cycles for LJMP)} = \textit{21 to 25 cycles.}\end{gathered}$$

In this example, at 24 MHz, 25 clock cycles take 1.042 µs.

Interrupt Priority. Interrupt priorities come into consideration when more than one interrupt is pending during the same instruction cycle. In this case, the Priority Encoder (see Figure 5-1) generates an interrupt vector for the highest priority pending interrupt.

5.1.1 Posted versus Pending Interrupts

An interrupt is posted when its interrupt conditions occur. This results in the flip-flop in Figure 5-1 clocking in a 1. The interrupt remains posted until the interrupt is taken or until it is cleared by writing to the appropriate INT_CLRx register.

A posted interrupt is not pending unless it is enabled by setting its interrupt mask bit (in the appropriate INT_MSKx register). All pending interrupts are processed by the Priority Encoder to determine the highest priority interrupt taken by the M8C if the Global Interrupt Enable bit is set in the CPU_F register.

Disabling an interrupt by clearing its interrupt mask bit (in the INT_MSKx register) does not clear a posted interrupt, nor does it prevent an interrupt from posting. It simply prevents a posted interrupt from becoming pending.

It is especially important to understand the functionality of clearing posted interrupts, if the configuration of the enCoRe V device is changed by the application.

For example, if a block has a posted interrupt when it is enabled and then disabled, the posted interrupt remains. Therefore, it is good practice to use the INT_CLR register to clear posted interrupts before enabling or re-enabling a block.

5.2 Application Overview

The interrupt controller and its associated registers allow the user's code to respond to an interrupt from almost every functional block in enCoRe V devices. Interrupts for all the digital blocks and each of the analog columns are available, as well as interrupts for supply voltage, sleep, variable clocks, and a general GPIO (pin) interrupt.

The registers associated with the interrupt controller allow the disabling of interrupts either globally or individually. The registers also provide a mechanism by which a user can ***clear*** all pending and posted interrupts or clear individual posted or pending interrupts. A ***software*** mechanism is provided to set individual interrupts. Setting an interrupt by way of software is very useful during code development, when one may not have the complete hardware system necessary to generate a real interrupt.

The following table lists the interrupts and priorities that are available in the enCoRe V devices.

Table 5-1. Device Interrupts

Interrupt Priority	Interrupt Address	Interrupt Name
0 (Highest)	0000h	Reset
1	0004h	Supply voltage monitor
2	0008h	Reserved
3	000Ch	Reserved
4	0010h	Timer0
5	0014h	GPIO
6	0018h	SPI
7	001Ch	I2C
8	0020h	Sleep Timer
9	0024h	Timer1
10	0028h	Timer2

5.3 Register Definitions

The following registers are associated with the Interrupt Controller and are listed in address order. The register descriptions have an associated register table showing the bit structure for that register. The bits in the tables that are grayed out are reserved bits and are not detailed in the register descriptions that follow. Always write reserved bits with a value of '0'. For a complete table of Interrupt Controller registers, refer to the Summary Table of the Core Registers on page 24.

5.3.1 INT_CLR0 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,DAh	INT_CLR0	I2C	Sleep	SPI	GPIO	Timer0	Reserved	Reserved	V Monitor	RW : 00

The Interrupt Clear Register 0 (INT_CLR0) enables the individual interrupt sources' ability to clear posted interrupts.

The INT_CLR0 register is similar to the INT_MSK0 register in that it holds a bit for each interrupt source. Functionally the INT_CLR0 register is similar to the INT_VC register, although its operation is completely independent. When the INT_CLR0 register is read, any bits that are set indicate an interrupt was posted for that hardware resource. Reading this register gives the user the ability to determine all posted interrupts.

The Enable Software Interrupt (ENSWINT) bit in the INT_SW_EN register determines how an individual bit value, written to an INT_CLR0 register, is interpreted. When ENSWINT is cleared (the default state), writing 1's to the INT_CLR0 register has no effect. However, writing 0's to the INT_CLR0 register, when ENSWINT is cleared, causes the corresponding interrupt to clear. If the ENSWINT bit is set, any 0's written to the INT_CLR0 register are ignored. However, 1's written to the INT_CLR0 register, while ENSWINT is set, cause an interrupt to post for the corresponding interrupt.

Software interrupts aid in debugging interrupt service routines by eliminating the need to create system level interactions that are sometimes necessary to create a hardware-only interrupt.

Bit 7: I2C. This bit allows posted I2C interrupts to be read, cleared, or set.

Bit 6: Sleep. This bit allows posted sleep interrupts to be read, cleared, or set.

Bit 5: SPI. This bit allows posted SPI interrupts to be read, cleared, or set.

Bit 4: GPIO. This bit allows posted GPIO interrupts to be read, cleared, or set.

Bit 3: Timer0. This bit allows posted timer interrupts to be read, cleared, or set.

Bit 0: V Monitor. This bit allows posted voltage monitor interrupts to be read, cleared, or set.

For additional information, refer to the INT_CLR0 register on page 196.

5.3.2 INT_CLR1 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,DBh	INT_CLR1	Endpoint3	Endpoint2	Endpoint1	Endpoint0	USB SOF	USB Bus Rst	Timer2	Timer1	RW : 00

This register enables the individual interrupt sources' ability to clear posted interrupts.

When bits in this register are read, a '1' is returned for every bit position that has a corresponding posted interrupt. When bits in this register are written with a '0' and ENSWINT is not set, posted interrupts are cleared at the corresponding bit positions. If there is no posted interrupt, there is no effect. When bits in this register are written with a '1' and ENSWINT is set, an interrupt is posted in the interrupt controller.

Bit 7: Endpoint3. Read '0', no posted interrupt for USB Endpoint3. Read '1', posted interrupt present for USB Endpoint3.
Write 0 AND ENSWINT = 0. Clear posted interrupt if it exists.
Write 1 AND ENSWINT = 0. No effect.
Write 0 AND ENSWINT = 1. No effect.
Write 1 AND ENSWINT = 1. Post an interrupt for USB Endpoint3.

Bit 6: Endpoint2. Read '0', no posted interrupt for USB Endpoint2. Read '1', posted interrupt present for USB Endpoint2.
Write 0 AND ENSWINT = 0. Clear posted interrupt if it exists.
Write 1 AND ENSWINT = 0. No effect.
Write 0 AND ENSWINT = 1. No effect.
Write 1 AND ENSWINT = 1. Post an interrupt for USB Endpoint2.

Bit 5: Endpoint1. Read '0', no posted interrupt for USB Endpoint1. Read '1', posted interrupt present for USB Endpoint1.
Write 0 AND ENSWINT = 0. Clear posted interrupt if it exists.
Write 1 AND ENSWINT = 0. No effect.
Write 0 AND ENSWINT = 1. No effect.
Write 1 AND ENSWINT = 1. Post an interrupt for USB Endpoint1.

Bit 4: Endpoint0. Read '0', no posted interrupt for USB Endpoint0. Read '1', posted interrupt present for USB Endpoint0.
Write 0 AND ENSWINT = 0. Clear posted interrupt if it exists.
Write 1 AND ENSWINT = 0. No effect.
Write 0 AND ENSWINT = 1. No effect.
Write 1 AND ENSWINT = 1. Post an interrupt for USB Endpoint0.

Bit 3: USB SOF. Read '0', no posted interrupt for USB Start of Frame (SOF). Read '1', posted interrupt present for USB Start of Frame (SOF).
Write 0 AND ENSWINT = 0. Clear posted interrupt if it exists.
Write 1 AND ENSWINT = 0. No effect.
Write 0 AND ENSWINT = 1. No effect.
Write 1 AND ENSWINT = 1. Post an interrupt for USB Start of Frame (SOF).

Bit 2: USB Bus Rst. Read '0', no posted interrupt for USB Bus Reset. Read '1', posted interrupt present for USB Bus Reset.
Write 0 AND ENSWINT = 0. Clear posted interrupt if it exists.
Write 1 AND ENSWINT = 0. No effect.
Write 0 AND ENSWINT = 1. No effect.
Write 1 AND ENSWINT = 1. Post an interrupt for USB Bus Reset.

Bit 1: Timer2. Read '0', no posted interrupt for Timer2. Read '1', posted interrupt present for Timer2.
Write 0 AND ENSWINT = 0. Clear posted interrupt if it exists.
Write 1 AND ENSWINT = 0. No effect.
Write 0 AND ENSWINT = 1. No effect.
Write 1 AND ENSWINT = 1. Post an interrupt for Timer2.

Bit 0: Timer1. Read '0', no posted interrupt for Timer1. Read '1', posted interrupt present for Timer1.
Write 0 AND ENSWINT = 0. Clear posted interrupt if it exists.
Write 1 AND ENSWINT = 0. No effect.
Write 0 AND ENSWINT = 1. No effect.
Write 1 AND ENSWINT = 1. Post an interrupt for Timer1.

For additional information, refer to the INT_CLR1 register on page 198.

5.3.3 INT_CLR2 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,DCh	INT_CLR2			USB_WAKE	Endpoint8	Endpoint7	Endpoint6	Endpoint5	Endpoint4	RW : 00

This register enables the individual interrupt sources' ability to clear posted interrupts.

When bits in this register are read, a '1' is returned for every bit position that has a corresponding posted interrupt. When bits in this register are written with a '0' and ENSWINT is not set, posted interrupts are cleared at the corresponding bit positions. If there was not a posted interrupt, there is no effect. When bits in this register are written with a '1' and ENSWINT is set, an interrupt is posted in the interrupt controller.

Bit 5: USB_WAKE. Read ‘0’, no posted interrupt for USB wakeup. Read ‘1’, posted interrupt present for USB wakeup.
Write 0 AND ENSWINT = 0. Clear posted interrupt if it exists.
Write 1 AND ENSWINT = 0. No effect.
Write 0 AND ENSWINT = 1. No effect.
Write 1 AND ENSWINT = 1. Post an interrupt for USB wakeup.

Bit 4: Endpoint8. Read ‘0’, no posted interrupt for USB Endpoint8. Read ‘1’, posted interrupt present for USB Endpoint8.
Write 0 AND ENSWINT = 0. Clear posted interrupt if it exists.
Write 1 AND ENSWINT = 0. No effect.
Write 0 AND ENSWINT = 1. No effect.
Write 1 AND ENSWINT = 1. Post an interrupt for USB Endpoint8.

Bit 3: Endpoint7. Read ‘0’, no posted interrupt for USB Endpoint7. Read ‘1’, posted interrupt present for USB Endpoint7.
Write 0 AND ENSWINT = 0. Clear posted interrupt if it exists.
Write 1 AND ENSWINT = 0. No effect.
Write 0 AND ENSWINT = 1. No effect.
Write 1 AND ENSWINT = 1. Post an interrupt for USB Endpoint7.

Bit 2: Endpoint6. Read ‘0’, no posted interrupt for USB Endpoint6. Read ‘1’, posted interrupt present for USB Endpoint6.
Write 0 AND ENSWINT = 0. Clear posted interrupt if it exists.
Write 1 AND ENSWINT = 0. No effect.
Write 0 AND ENSWINT = 1. No effect.
Write 1 AND ENSWINT = 1. Post an interrupt for USB Endpoint6.

Bit 1: Endpoint5. Read ‘0’, no posted interrupt for USB Endpoint5. Read ‘1’, posted interrupt present for USB Endpoint5.
Write 0 AND ENSWINT = 0. Clear posted interrupt if it exists.
Write 1 AND ENSWINT = 0. No effect.
Write 0 AND ENSWINT = 1. No effect.
Write 1 AND ENSWINT = 1. Post an interrupt for USB Endpoint5.

Bit 0: Endpoint4. Read ‘0’, no posted interrupt for USB Endpoint4. Read ‘1’, posted interrupt present for USB Endpoint4.
Write 0 AND ENSWINT = 0. Clear posted interrupt if it exists.
Write 1 AND ENSWINT = 0. No effect.
Write 0 AND ENSWINT = 1. No effect.
Write 1 AND ENSWINT = 1. Post an interrupt for USB Endpoint4.

For additional information, refer to the INT_CLR2 register on page 200.

5.3.4 INT_MSK0 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,E0h	INT_MSK0	I2C	Sleep	SPI	GPIO	Timer0	Reserved	Reserved	V Monitor	RW : 00

The Interrupt Mask Register (INT_MSK0) enables the individual interrupt sources' ability to create pending interrupts.

If cleared, each bit in an INT_MSK0 register prevents a posted interrupt from becoming a pending interrupt (input to the Priority Encoder). However, an interrupt can still post even if its mask bit is zero. All INT_MSK0 bits are independent of all other INT_MSK0 bits.

If an INT_MSK0 bit is set, the interrupt source associated with that mask bit may generate an interrupt that becomes a pending interrupt. For example, if INT_MSK0[4] is set and at least one GPIO pin is configured to generate an interrupt, the interrupt controller allows a GPIO interrupt request to post and become a pending interrupt to which the M8C responds. If a higher priority interrupt is generated before the M8C responds to the GPIO interrupt, the higher priority interrupt is responded to before the GPIO interrupt.

Each interrupt source may require configuration at a block level. Refer to the corresponding chapter for each interrupt for any additional configuration information.

Bit 7: I2C. This bit allows I2C interrupts to be enabled or masked.

Bit 6: Sleep. This bit allows sleep interrupts to be enabled or masked.

Bit 5: SPI. This bit allows SPI interrupts to be enabled or masked.

Bit 4: GPIO. This bit allows GPIO interrupts to be enabled or masked.

Bit 3: Timer0. This bit allows Timer0 interrupts to be enabled or masked.

Bit 0: V Monitor. This bit allows voltage monitor interrupts to be enabled or masked.

For additional information, refer to the INT_MSK0 register on page 204.

5.3.5 INT_MSK1 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,DFh	INT_MSK1	Endpoint3	Endpoint2	Endpoint1	Endpoint0	USB SOF	USB Bus Reset	Timer2	Timer1	RW : 00

This register enables the individual sources' ability to create pending interrupts.

When an interrupt is masked off, the mask bit is '0'. The interrupt continues to post in the interrupt controller. Clearing the mask bit only prevents a posted interrupt from becoming a pending interrupt.

Bit 7: Endpoint3. '0' is mask USB Endpoint3 interrupt. '1' is unmask USB Endpoint3 interrupt.

Bit 6: Endpoint2. '0' is mask USB Endpoint2 interrupt. '1' is unmask USB Endpoint2 interrupt.

Bit 5: Endpoint1. '0' is mask USB Endpoint1 interrupt. '1' is unmask USB Endpoint1 interrupt.

Bit 4: Endpoint0. '0' is mask USB Endpoint0 interrupt. '1' is unmask USB Endpoint0 interrupt.

Bit 3: USB SOF. '0' is mask USB SOF interrupt. '1' is unmask USB SOF interrupt.

Bit 2: USB Bus Reset (K). '0' is mask USB Bus Reset interrupt. '1' is unmask USB Bus Reset interrupt.

Bit 1: Timer2. '0' is mask Timer2 interrupt. '1' is unmask Timer2 interrupt.

Bit 0: Timer1. '0' is mask Timer1 interrupt. '1' is unmask Timer1 interrupt.

For additional information, refer to the INT_MSK1 register on page 203.

5.3.6 INT_MSK2 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,DEh	INT_MSK2			USB Wakeup	Endpoint8	Endpoint7	Endpoint6	Endpoint5	Endpoint4	RW : 00

This register is used to enable the individual sources' ability to create pending interrupts.

When an interrupt is masked off, the mask bit is '0'. The interrupt still posts in the interrupt controller. Therefore, clearing the mask bit only prevents a posted interrupt from becoming a pending interrupt.

Bit 5: USB Wakeup. ’0’ is mask USB Wakeup interrupt. ‘1’ is unmask USB Wakeup interrupt.

Bit 4: Endpoint8. ’0’ is mask USB Endpoint8 interrupt. ‘1’ is unmask USB Endpoint8 interrupt.

Bit 3: Endpoint7. ’0’ is mask USB Endpoint7 interrupt. ‘1’ is unmask USB Endpoint7 interrupt.

Bit 2: Endpoint6. ’0’ is mask USB Endpoint6 interrupt. ‘1’ is unmask USB Endpoint6 interrupt.

Bit 1: Endpoint5. ’0’ is mask USB Endpoint5 interrupt. ‘1’ is unmask USB Endpoint5 interrupt.

Bit 0: Endpoint4. ’0’ is mask USB Endpoint4 interrupt. ‘1’ is unmask USB Endpoint4 interrupt.

For additional information, refer to the INT_MSK2 register on page 202.

5.3.7 INT_SW_EN Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,E1h	INT_SW_EN								ENSWINT	RW : 0

The Interrupt Software Enable Register (INT_SW_EN) is used to enable software interrupts.

Bit 0: ENSWINT. This bit is a special non-mask bit that controls the behavior of the INT_CLR0 register. See the INT_CLR0 register in this section for more information.

For more details, see INT_SW_EN register on page 205.

5.3.8 INT_VC Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,E2h	INT_VC	Pending Interrupt[7:0]								RC : 00

Legend

Clearable register or bits.

The Interrupt Vector Clear Register (INT_VC) returns the next pending interrupt and clears all pending interrupts when written.

Bits 7 to 0: Pending Interrupt[7:0]. When the register is read, the ***least significant byte (LSB)*** of the highest priority pending interrupt is returned. For example, if the GPIO and I2C interrupts were pending and the INT_VC register was read, the value 14h is read. However, if no interrupts were pending, the value 00h is returned. This is the reset vector in the interrupt table; however, reading 00h from the INT_VC register is not considered an indication that a system reset is pending. Rather, reading 00h from the INT_VC register simply indicates that there are no pending interrupts. The highest priority interrupt, indicated by the value returned by a read of the INT_VC register, is removed from the list of pending interrupts when the M8C services an interrupt.

Reading the INT_VC register has limited usefulness. If interrupts are enabled, a read to the INT_VC register is not able to determine that an interrupt was pending before the interrupt was actually taken. However, while in an interrupt service routine, a user may wish to read the INT_VC register to see the next interrupt. When the INT_VC register is written with any value, all pending and posted interrupts are cleared by asserting the clear line for each interrupt.

For additional information, refer to the INT_VC register on page 206.

5.3.9 Related Registers

- CPU_F on page 208.

6. General-Purpose I/O (GPIO)

This chapter discusses the general-purpose I/O (GPIO) and its associated registers, which is the circuit responsible for interfacing to the I/O pins of a enCoRe V device. The GPIO blocks provide the interface between the M8C core and the outside world. They offer a large number of configurations to support several types of input/output (IO) operations for both digital and analog systems. For a complete table of the GPIO registers, refer to enCoRe V Core on page 23. For a quick reference of all enCoRe Vregisters in address order, refer to the Register Reference chapter on page 163.

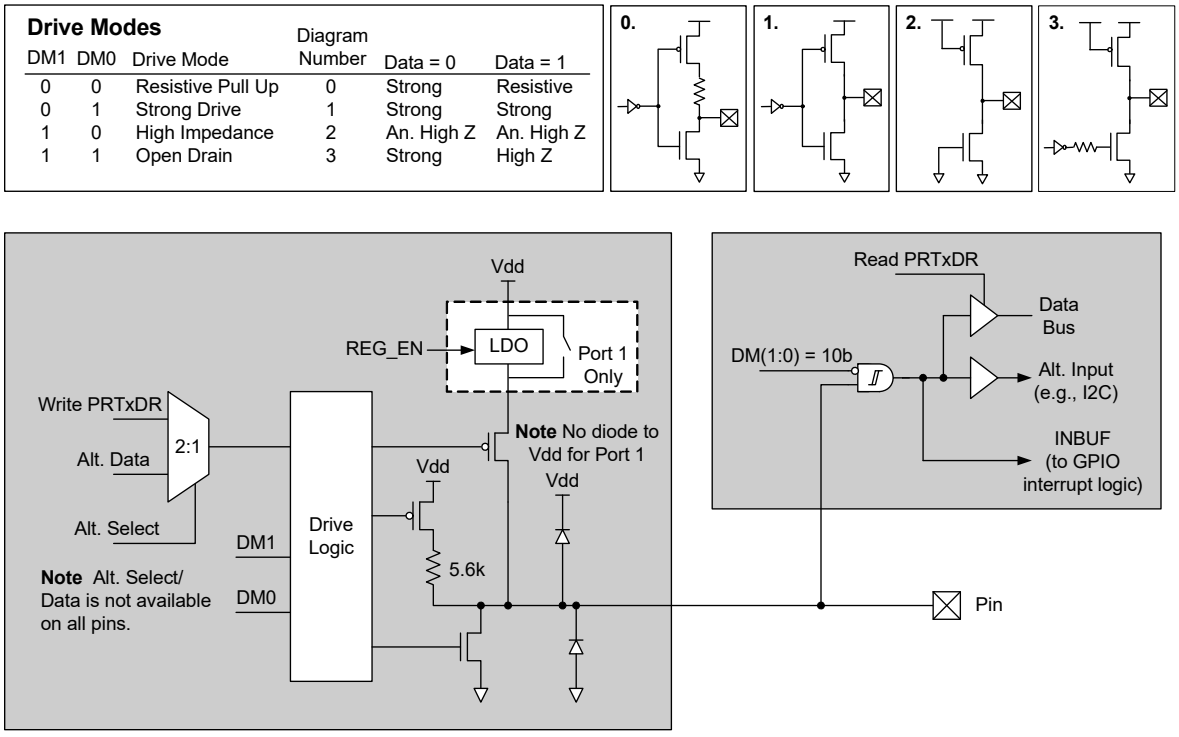
6.1 Architectural Description

The GPIO in the CY7C643xx and CY7C604xx devices are all uniform, except that Port 0 and Port 1 GPIO have stronger high drive. In addition to higher drive strength, Port 1 GPIO have an option for regulated output level. These distinctions are discussed in more detail in Port 1 Distinctions on page 53 and Port 0 Distinctions on page 53.

Figure 6-1. GPIO Block Diagram

Drive Modes

DM1	DM0	Drive Mode	Diagram Number	Data = 0	Data = 1
0	0	Resistive Pull Up	0	Strong	Resistive
0	1	Strong Drive	1	Strong	Strong
1	0	High Impedance	2	An. High Z	An. High Z
1	1	Open Drain	3	Strong	High Z



6.1.1 General Description

The GPIO contains input buffers, output drivers, and configuration logic for connecting the enCoRe V device to the outside world.

I/O ports are arranged with (up to) 8 bits per port. Each full port contains eight identical GPIO blocks. Each GPIO block is used for the following types of I/O:

- Digital I/O (digital input and output controlled by software)
- Analog I/O

Each I/O pin also has several drive modes, and interrupt capabilities. All GPIO pins provide both digital I/O and analog input capability.

Certain pins contain an option to bypass the normal data path and output from an internal source. An example is I2C outputs. These are described in Data Bypass on page 55.

6.1.2 Digital I/O

One of the basic operations of the GPIO ports is to allow the M8C to send information out of the enCoRe V device and get information into the M8C from outside the device. This is accomplished using the port data register (PRTxDR). Writes from the M8C to the PRTxDR register store the data state, one bit per GPIO. In the standard non-bypass mode, the pin drivers drive the pin in response to this data bit, with a drive strength determined by the Drive mode setting (see Figure 6-1). The actual voltage on the pin depends upon the Drive mode and the external load.

The M8C reads the value of a port by reading the PRTxDR register address. When the M8C reads the PRTxDR register address, the current value of the pin voltage is translated into a logic value and returned to the M8C. Note that the pin voltage can represent a different logic value than the last value written to the PRTxDR register. This is an important distinction to remember in situations such as the use of a read modify write to a PRTxDR register. Examples of read modify write instructions include AND, OR, and XOR.

The following is an example of how a read modify write, to a PRTxDR register, can have an unexpected and even indeterminate result in certain systems. Consider a scenario where all bits of Port 1 on the enCoRe V device are in the strong 0 resistive 1 Drive mode; so that in some cases, the system the enCoRe V is in may pull down one of the bits by an external driver.

```
mov   reg[PRT1DR], 0xFF
and   reg[PRT1DR], 0x7F
```

In the first line of this code, writing a FFh to the port causes the enCoRe V to drive all pins high through a resistor. This does not affect any bits that are strongly driven low by the system the enCoRe V is in. However, in the second line of code, it cannot guarantee that only bit 7 is the one set to a strong 0 (zero). Because the AND instruction first reads the port, any bits that are currently driven low externally are read as a 0. These zeros are then written back to the port. When this happens, the pin goes into a strong 0 state; therefore, if the external low drive condition ends in the system, the enCoRe Vkeeps the pin value at a logic 0.

6.1.3 Analog and Digital Inputs

The High Impedance Analog mode turns off the Schmitt trigger on the input path, which may reduce power consumption and decrease internal switching noise when using a particular I/O as an analog input.

All modes, except High-impedance Analog, allow digital inputs. The most useful digital input modes are Resistive Pull Up (DM1, DM0 = 00b with Data = 1) or a fully high-impedance input using open drain (DM1, DM0 = 11b with Data = 1).

6.1.4 Port 1 Distinctions

Port 1 has two differences from the other GPIO ports. It has stronger high drive (as does Port 0) and it has an option for regulating all outputs to a 3 V/2.5 V/1.8 V level when in strong drive mode. Refer to the device datasheet for the different current sourcing specifications of Port 1.

By setting the REG_EN bit in the IO_CFG1 register, Port 1 can be configured to drive strong high to a regulated 3 V/ 2.5 V/1.8 V level. If REG_EN is set low, Port 1 pins drive to Vdd in strong drive mode.

In Resistive High Drive mode ([DM1, DM0] = 00), the pins pull up to the ***chip*** Vdd level regardless of the regulator setting for this port. Only Strong Drive mode allows for the outputs to be driven to the regulated level. When the REG_EN bit is set high, pins configured for strong drive to regulated level, while those in resistive pull-up mode drive to Vdd.

In their default state, all Port 1 I/O prevent DC current from flowing into the pin when the pin voltage is above the chip Vdd. This feature resolves the problem where the enCoRe Vholds down the system I2C bus or provides a current leakage path from a powered peripheral during enCoRe V power down or reset.

The open drain driver is capable of sinking 24 mA current (required for sinking LEDs used for backlighting) and maintaining a logic low state.

Regulated output level can be selected by bits 4 and 5 in the IO_CFG1 register. For 3-V output level, the chip Vdd should be greater than 3.1 V. For 2.5-V output, the chip Vdd should be greater than 2.7 V, and for 1.8-V output level, chip Vdd should be greater than 2.5 V.

6.1.5 Port 0 Distinctions

Port 0 has a stronger high drive. However, unlike Port 1, it does not have an option to regulate the outputs when in

strong drive mode. Refer to the device datasheet for the different current sourcing specifications of Port 0.

6.1.6 GPIO Block Interrupts

You can individually configure each GPIO pin for interrupt capability. Pins are configured by pin interrupt enables and also by a chip wide selection for interrupt state with this global selection. Pins can be set to interrupt when the pin is low or when it changes from the last time it was read. The block provides an open drain interrupt output (INTO) that is connected to other GPIO blocks in a wire-OR fashion.

All pin interrupts that are wire-ORed together are tied to the same system GPIO interrupt. Therefore, if interrupts are enabled on multiple pins, the user's interrupt service routine must provide a mechanism to determine which pin was the source of the interrupt.

Using a GPIO interrupt requires these steps:

1. Set the Interrupt mode (IOINT bit in the IO_CFG1 register).
2. Enable the bit interrupt in the GPIO block.
3. Set the mask bit for the (global) GPIO interrupt.
4. Assert the overall Global Interrupt Enable.

The first step sets a common interrupt mode for all pins.

The second step is set at the GPIO pin level (that is, at each port pin), by way of the PRTxIE registers.

The last two steps are common to all interrupts and described in the Interrupt Controller chapter on page 44.

At the GPIO block level, asserting the INTO line depends only on the bit interrupt enable and the state of the pin relative to the chosen Interrupt mode. At the enCoRe V device level, due to their wire-OR nature, the GPIO interrupts are neither true edge sensitive interrupts nor true level sensitive interrupts. They are considered edge-sensitive for asserting, but level sensitive for release of the wire-OR interrupt line.

If no GPIO interrupts are asserting, a GPIO interrupt occurs whenever a GPIO pin interrupt enable is set and the GPIO pin transitions (if not already transitioned) appropriately high or low to match the interrupt mode configuration. After this happens, the INTO line pulls low to assert the GPIO interrupt. This assumes the other system level enables are on, such as setting the global GPIO interrupt enable and the Global Interrupt Enable. Setting the pin interrupt enable may immediately assert INTO, if the Interrupt mode conditions are already being met at the pin.

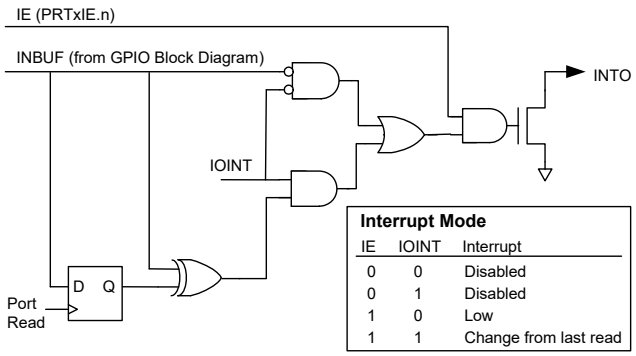
After INTO pulls low, it continues to hold INTO low until one of these conditions change:

- The pin interrupt enable is cleared.
- The voltage at pin transitions to the opposite state.
- In interrupt-on-change mode, the GPIO data register is read thus setting the local interrupt level to the opposite state.
- The Interrupt mode is changed so that the current pin state does not create an interrupt.

After one of these conditions is met, the INTO releases. At this point, another GPIO pin (or this pin again) can assert its INTO pin, pulling the common line low to assert a new interrupt.

Note the following behavior from this level release feature. If one pin is asserting INTO and then a second pin asserts its INTO, when the first pin releases its INTO, the second pin is already driving INTO and thus no change is seen (that is, no new interrupt is asserted on the GPIO interrupt). Take care, using polling or the states of the GPIO pin and Global Interrupt Enables, to catch all interrupts among a set of wire-OR GPIO blocks.

Figure 6-2. GPIO Interrupt Logic Diagram



6.1.6.1 Interrupt Modes

GPIO interrupts use the IOINT bit from the IO_CFG1 register. The setting of IOINT determines the interrupt mode for all GPIO.

Interrupt mode IOINT=0 means that the block asserts the GPIO interrupt line (INTO) when the pin voltage is low, if the block's bit interrupt enable line is set (high).

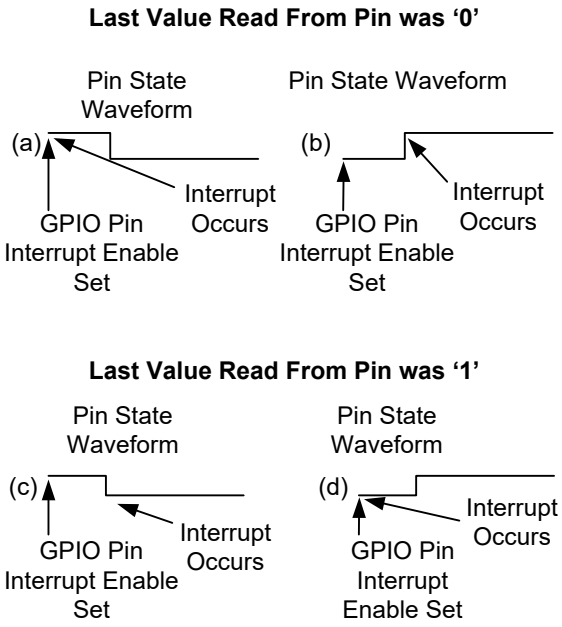
Interrupt mode IOINT=1 means that the block asserts the interrupt line (INTO) when the pin voltage is the opposite of the last state read from the pin, if the block's bit interrupt enable line is set high. This mode switches between low mode and high mode, depending upon the last value read from the port during reads of the data register (PRTxDR). If the last value read from the GPIO was 0, the GPIO pin is in Interrupt High mode. If the last value read from the GPIO was '1', the GPIO is in Interrupt Low mode.

Table 6-1. GPIO Interrupt Modes

IE	IOINT	Description
0	0	Bit interrupt disabled, INTO deasserted
0	1	Bit interrupt disabled, INTO d-asserted
1	0	Assert INTO when PIN = low
1	1	Assert INTO when PIN = change from last read

Figure 6-3 assumes that the GIE is set, GPIO interrupt mask is set, and that the IOINT bit was set to high. The Change Interrupt mode relies on the value of an internal read register to determine if the pin state changed. Therefore, the port that contains the GPIO in question must be read during every interrupt service routine. If the port is not read, the Interrupt mode acts as if it is in high mode when the latch value is '0' and low mode when the latch value is '1'.

Figure 6-3. GPIO Interrupt Mode IOINT = 1



6.1.7 Data Bypass

GPIO pins are configured to either output data through CPU writes to the PRTxDR registers or to bypass the port's data register and output data from internal functions instead. The bypass path is shown in Figure 6-1 by the Alt Data input, which is selected by the Alt Select input. These data bypass options are selected in one of two ways.

- For internal functions such as I2C and SPI, the hardwire automatically selects the bypass mode for the required pins when the function is enabled.
- For all bypass modes, the wanted drive mode of the pin must be configured separately for each pin, with the PRTxDM1 and PRTxDM0 registers.

6.2 Register Definitions

The following registers are associated with the general-purpose I/O (GPIO) and are listed in address order. The register descriptions have an associated register table showing the bit structure for that register. The bits in the tables that are grayed out are reserved bits and are not detailed in the register descriptions that follow. Always write reserved bits with a value of 0. For a complete table of general-purpose I/O registers, refer to the Core Register Summary on page 24.

For a selected GPIO block, the individual registers are addressed in the Core Register Summary on page 24. In the register names, the 'x' is the port number, configured at the enCoRe V device level (x = 0 to 4 typically). All register values are readable, except for the PRTxDR register; reads of this register return the pin state instead of the register bit state.

6.2.1 PRTxDR Registers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,xxh	PRTxDR	Data[7:0]								RW : 00

Legend
xx An "x" after the comma in the address field indicates that there are multiple instances of the register. For an expanded address listing of these registers, refer to the Core Register Summary on page 24.

The Port Data Register (PRTxDR) allows for write or read access of the current logical equivalent of the voltage on the pin.

Bits 7 to 0: Data[7:0]. Writing the PRTxDR register bits set the output drive state for the pin to high (for Data = 1) or low (Data = 0), unless a bypass mode is selected (see Data Bypass on page 55).

Reading the PRTxDR register returns the actual pin state, as seen by the input buffer. This may not be the same as the expected output state, if the load pulls the pin more strongly than the pin's configured output drive. See Digital I/O on page 53 for a detailed discussion of digital I/O.

For additional information, refer to the PRTxDR register on page 164.

6.2.2 PRTxIE Registers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,xxh	PRTxIE	InterruptEnables[7:0]								RW : 00

Legend
xx An "x" after the comma in the address field indicates that there are multiple instances of the register. For an expanded address listing of these registers, refer to the Core Register Summary on page 24.

The Port Interrupt Enables (PRTxIE) registers enable or disable interrupts from individual GIPIO pins.

Bits 7 to 0: InterruptEnables[7:0]. These bits enable the corresponding port pin interrupt. Only four LSB pins are used because this port has four pins.

'0' is port pin interrupt disabled for the corresponding pin.

'1' is port pin interrupt enabled for the corresponding pin. Interrupt mode is determined by the IOINT bit in the IO_CFG1 register.

For additional information, refer to the PRTxDR register on page 164.

6.2.3 PRTxDMx Registers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,xxh	PRTxDM0	Drive Mode 0[7:0]								RW : 00
1,xxh	PRTxDM1	Drive Mode 1[7:0]								RW : FF

Legend
xx An “x” after the comma in the address field indicates that there are multiple instances of the register. For an expanded address listing of these registers, refer to the Core Register Summary on page 24.

The Port Drive Mode Bit Registers (PRTxDM0 and PRTxDM1) specify the Drive mode for GPIO pins.

Bits 7 to 0: Drive Mode x[7:0]. In the PRTxDMx registers there are four possible drive modes for each port pin. Two mode bits are required to select one of these modes, and these two bits are spread into two different registers (PRTxDM0 and PRTxDM1). The bit position of the affected port pin (for example, Pin[2] in Port 0) is the same as the bit position of each of the two drive mode register bits that control the Drive mode for that pin (for example, bit[2] in PRT0DM0 and bit[2] in PRT0DM1). The two bits from the two registers are treated as a group. These are referred to as DM1 and DM0, or together as DM[1:0]. Drive modes are shown in Table 6-3.

For analog I/O, set the drive mode to the High-Z analog mode, 10b. The 10b mode disables the block’s digital input buffer so no crowbar current flows, even when the analog input is not close to either power rail. If the 10b drive mode is used, the pin is always read as a zero by the CPU and the pin cannot generate a useful interrupt. (It is not strictly required that you select High-Z mode for analog operation.)

When digital inputs are needed on the same pin as analog inputs, use the 11b Drive mode with the corresponding data bit (in the PRTxDR register) set high.

Drive Modes		Pin State	Description
DM1	DM0		
0	0	Resistive pull-up	Resistive high, strong low
0	1	Strong drive	Strong high, strong low
1	0	High-impedance, analog (**reset state**)	High-Z high and low, digital input disabled (for zero power) (**reset state**)
1	1	Open drain low	High-Z high (digital input enabled), strong low.

The GPIO provides a default drive mode of high-impedance, analog (High-Z). This is achieved by forcing the reset state of all PRTxDM1 registers to FFh.

For additional information, refer to PRTxDM0 register on page 212 and PRTxDM1 register on page 213.

6.2.4 IO_CFG1 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,DCh	IO_CFG1	StrongP		Range[1:0]		P1_LOW_THRS	SPICLK_ON_P10	REG_EN	IOINT	RW : 00

The Input/Output Configuration Register 1 (IO_CFG1) configures the Port 1 output regulator and set the Interrupt mode for all GPIO.

Bit 7: StrongP. Setting this bit increases the drive strength and edge ratio for high outputs.

Bit 5 and 4: Range[1:0]. These bits select the regulator output level for Port 1. Available levels are 3.0 V, 1.8 V, and 2.5 V.

Selects the high output level for Port 1 outputs.

Range[1:0]	Output Level
00	3.0 volts
01	3.0 volts
10	1.8 volts
11	2.5 volts

Bit 3 P1_LOW_THRS. This bit reduces the threshold voltage of the P1 port input buffers so that there are no compatibility issues when Port 1 is communicating at regulated voltage levels.

'0' is standard threshold of VIH, VIL. '1' is reduce threshold of VIH, VIL.

Bit 2: SPICLK_ON_P10. When this bit is set to '1', the SPI clock is mapped to Port 1 pin 0. Otherwise, it is mapped to Port 1 pin 3.

Bit 1: REG_EN. The Register Enable bit (REG_EN) controls the regulator on Port 1 outputs.

Bit 0: IO INT. This bit sets the GPIO Interrupt mode for all pins in the CY7C643xx and CY7C604xx enCoRe V devices. GPIO interrupts are controlled at each pin by the PRTxIE registers, and also by the global GPIO bit in the INT_MSK0 register.

For additional information, refer to the IO_CFG1 register on page 224.

6.2.5 IO_CFG2 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,DEh	IO_CFG2			REG_LEVEL[2:0]				REG_CLOCK[1:0]		RW : 00

The Input/Output Configuration Register 2 (IO_CFG2) selects output regulated supply and clock rates.

Bits 5 to 3: REG_LEVEL[2:0]. These bits select output regulated supply.

REG_LEVEL[2:0]	Approx. Regulated Supply (V)		
000	3	2.5	1.8
001	3.1	2.6	1.9
010	3.2	2.7	2.0
011	3.3	2.8	2.1
100	3.4	2.9	2.2
101	3.5	3.0	2.3
110	3.6	3.1	2.4
111	3.7	3.2	2.5

Bits 1 to 0: REG_CLOCK[1:0]. The Regulated I/O charge pump can operate with a maximum clock speed of 12 MHZ. The REG_CLOCK[1:0] bits select clocking options for the regulator. Setting REG_CLOCK[1:0] to '10' should be used with 24-MHz SYSCLK and '01' should be used with 6-/12-MHz SYSCLK.

REG_CLOCK[1:0]	SYSCLK Clock Rate
10	24 MHz
01	6/12 MHz

For additional information, refer to the IO_CFG2 register on page 227.

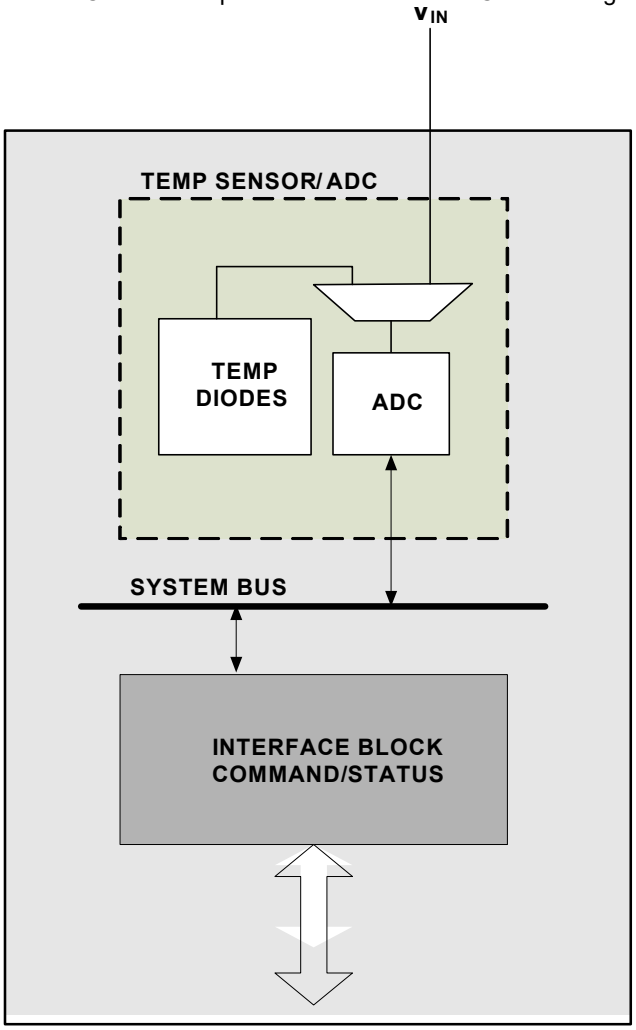
7. Analog-to-Digital Converter

This chapter discusses the Analog-to-Digital Converter (ADC) and its associated registers. For a complete table of the ADC registers, refer to the Core Register Summary on page 24. For a quick reference of all enCoRe V registers in address order, refer to the Register Reference chapter on page 163.

7.1 Architectural Description

The ADC on enCoRe V devices is an independent block with a state machine interface to control accesses to the block. The ADC is housed together with the temperature sensor core. Figure 7-1.

Figure 7-1. enCoRe V Temperature Sensor and ADC Block Diagram



7.2 Brief Overview of ADC Components and Registers

This section provides an overview of the functions of the ADC block and the application interface.

7.2.1 Interface Command/Status Block

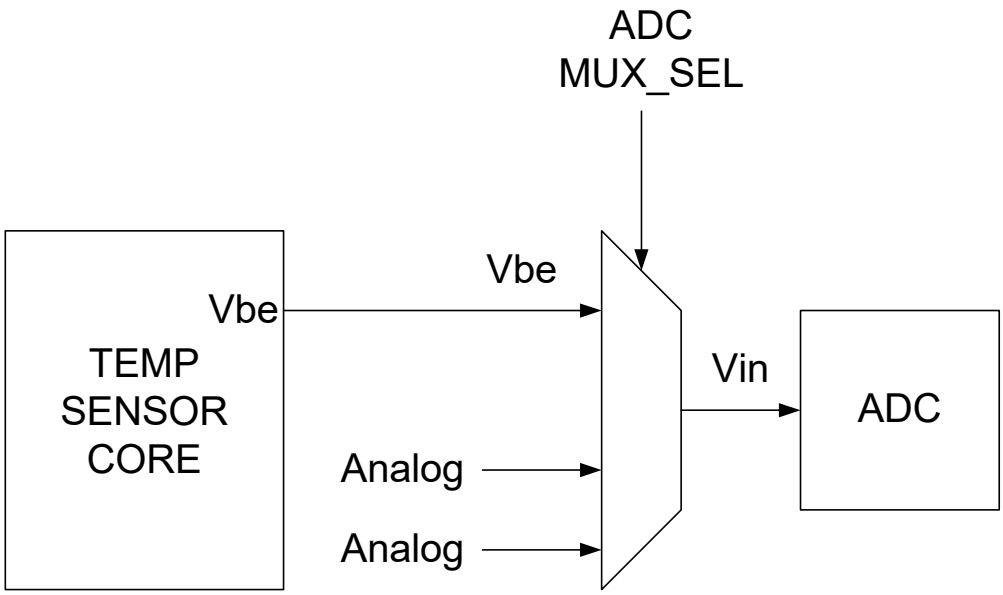
The Interface command and status block provides the application interface logic. This block has a state machine that is used to generate the status and other relevant signals for all instructions.

7.2.2 ADC

The ADC is a part of the temperature control block.

The ADC in enCoRe V can be connected to the Temperature Sensor Core or the Analog Mux Bus as shown in Figure 7-2. As a default operation, the ADC is connected to the temperature sensor to give digital values of the temperature.

Figure 7-2. Temperature Sensor/Analog Bus Connection to ADC



The ADC User Module contains an integrator block and one comparator with positive and negative input set by the MUXes. The input to the integrator stage comes from the Analog Global Input Mux or the temperature sensor with full scale input being 0V to 1.3V.

In the ADC only configuration (the ADC MUX selects the Analog Mux Bus, not the default temperature sensor connection), an external voltage can be connected to the input of the modulator for voltage conversion. The ADC is run for a number of cycles set by the timer, depending upon the resolution of the ADC desired by the user. A counter counts the number of trips by the comparator, which is proportional to the input voltage. The Temp Sensor block clock speed is 36 MHz and is divided down to 1 to 12 MHz for ADC operation.

7.2.2.1 ADC Register Definitions on page 60 shows the registers that need to be configured for this conversion. A minimum of 2 µs wait time is required for the modulator to be enabled after the other registers are configured. Therefore, writes to the Modulator Control Registers (MOD_CR0 and MOD_CR1) should be held for at least 2 µs after all the other registers are configured.

The registers specified below are controlled by the writing to the interface commmand/status block.

7.2.2.1 ADC Register Definitions

The following registers (listed in table) are associated with the ADC block in the temperature sensor core of enCoRe V devices and are listed in address order. The register descriptions have an associated register table showing the bit structure for that register. The bits in the tables that are grayed out are reserved bits and are not detailed in the register descriptions that follow. Always write reserved bits with a value of ‘0’.

Modulator Control Register 0

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
3Bh	MOD_CR0	MOD_EN	Reserved						TIMER_EN	RW : 00

This is the Control Register 0 for the modulator in the ADC block.

Bit 7: Modulator Enable. ‘0‘ is disable. ‘1’ is enable.

Bits 6 to 1: MOD_CR0[6:1]. These 6 bits are reserved and are ‘0’ by default.

Bit 0: Timer/Counter Enable. ‘0‘ is disable Timer/Counter. ‘1’ is enable Timer/Counter.

Modulator Control Register 1

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
3Ch	MOD_CR1	Reserved				INT_EN	ADC Input Select		Reserved	RW : 00

This is the Control Register 1 for the modulator in the ADC block.

Bits 7 to 4: MOD_CR1[7:4]. These 4 bits are reserved and are ‘0’ by default.

Bit 3: ADC Interrupt Enable. ‘0‘ is disable the interrupt. ‘1’ is enable ADC interrupt.

The application M8C recognizes these interrupts as ADC interface interrupts .

Bits 2 to 1: ADC Input Select. These bits select the input to the ADC.

‘00‘ - Temp Sensor Ouput

‘01‘ - Positive Reference Voltage

‘10‘ - Analog Bus

‘11‘ - External Port Input

Bit 0: Reserved. This bit is reserved and is ‘0’ by default.

Analog Clock Configuration Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
3Dh	ACLK	Reserved				Clock Divider Value				RW : 00

This is the configuration register for the analog clock in the Temperature Sensor/ADC core.

Bits 7 to 4: ACLK[7:4]. These 4 bits are reserved and are ‘0’ by default.

Bits 3 to 0: Clock Divider. These bits set the divider value for the ADC clock. The temperature sensor block is clocked at 36 MHz. This value of 36 MHz is then divided by the value set by these 4 bits.

‘0000‘ - Divide by 1

‘0001‘ - Divide by 2

‘0010‘ - Divide by 3

‘0011‘ - Divide by 4

‘0100‘ - Divide by 5

‘0101‘ - Divide by 6

‘0110‘ - Divide by 7

‘0111‘ - Divide by 8

‘1000‘ - Divide by 9

‘1001‘ - Divide by 10

‘1010‘ - Divide by 11

‘1011‘ - Divide by 12

‘1100‘ - Divide by 13

‘1101‘ - Divide by 14

‘1110‘ - Divide by 15

‘1111‘ - Divide by 16

Temperature Sensor Control Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
3Eh	TS_CR0	Reserved							TS_EN	RW : 00

This is the Control Register for the temperature sensor.

Bits 7 to 1: TS_CR0[7:4]. These 7 bits are reserved and are ‘0’ by default.

Bit 0: Temperature Sensor Enable. ‘0‘ is disable the temperature sensor. ‘1’ is enable the temperature sensor.

This bit must be set for ADC operation as this controls the analog ground reference generation.

ADC Count Low Byte Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
3Fh	ADC_CNTL	Data [7:0]								RW : 00

This register holds the lower byte count value of the ADC operation.

Bits 7 to 0: ADC_CNTL[7:0]. Contains the LSB 8 bits of the ADC counter value.

ADC Count High Byte Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
40h	ADC_CNTH	Data [7:0]								RW : 00

This register holds the upper byte count value of the ADC operation.

Bits 7 to 0: ADC_CNTL[7:0]. Contains the MSB 8 bits of the ADC counter value.

Timer Period Low Byte Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
41h	TMR Period L	Data [7:0]								RW : 00

This register holds the lower byte value of the timer period for the ADC operation.

Bits 7 to 0: Timer Period L [7:0]. Contains the LSB 8 bits of the ADC timer period value.

Timer Period High Byte Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
42h	TMR Period H	Data [7:0]								RW : 00

This register holds the upper byte value of the timer period for the ADC operation.

Bits 7 to 0: Timer Period H [7:0]. Contains the MSB 8 bits of the ADC timer period value.

Comparator Control Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
55h	CMP_CR0	Reserved				Comparator Control bits				RW : 00

This register controls the comparator and its inputs.

Bits 7 to 4: CMP_CR0[7:4]. These 4 bits are reserved and are ‘0’ by default.

Bits 3 to 0: Comparator Control. These bits set the input to the comparator module. For ADC operation this must be set to a value of ‘0000’.

Note Value ‘0000’ is required for ADC operation.

7.3 ADC Register Definitions - Application Interface

The following two registers are associated with the Analog to Digital Converter (ADC) interface and are listed in address order. The register descriptions have an associated register table showing the bit structure for that register. The bits in the tables that are grayed out are reserved bits and are not detailed in the register descriptions that follow. Always write reserved bits with a value of '0'. For a complete table of registers, refer to the Core Register Summary on page 24.

The two registers, ADC Data Register and the ADC Status Register, are the main interface between the application processor M8C and the Temperature Sensor/ADC. Control to the ADC registers has to happen through the following two registers. For example, if a value has to be written to the ADC register TS_CR0, the ADC_DATA register is to be written with command byte, the register address, and the data. Section 7.4 of this chapter provides detailed information on the interface between the ADC and the application processor and how control and data transfer to and from the ADC is affected.

7.3.1 ADC Data Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,E5h	ADC_DATA	Data[7:0]								RW : 00

This is the ADC Data register. It holds the data to write to the ADC. When read, this register returns the data from the ADC.

Bits 7 to 0 Data[7:0]. The 8 bits of data that is written to or read by the application processor M8C through the command/status interface. These 8 bits are '0' by default.

7.3.2 ADC Status Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,E6h	ADC_STATUS	Status Code[5:0]						Data Ready	INS_BUSY	RW : 00

This is the ADC Status register.

Bits 7 to 2 Status Code[5:0]. These bits are '0' by default. The status of the ADC operation is updated in these 6 bits. If there is any error, the error status is updated.

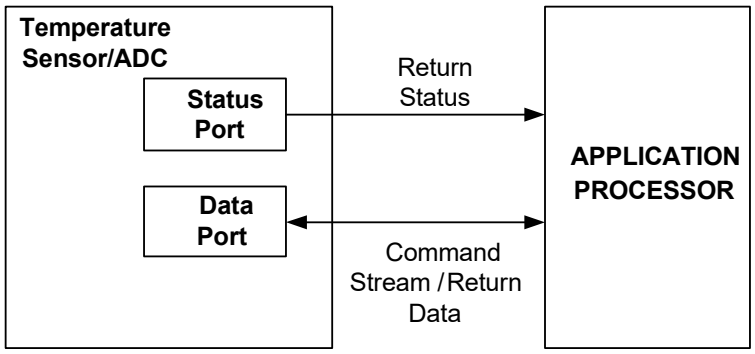
Bit: 1 Data Ready. '0' is Temperature Senosr/ADC processing the data. '1' is Temperature Senosr/ADC ready to accept the next data byte.

Bit 0: Instruction Busy. '0' is ADC waiting for the next instruction.'1' is ADC processing the current instruction.

7.4 Application Overview

As shown in Figure 7-3, the Temperature Sensor/ADC contains one read/write data port and one read only status port. Reads from the status port return terminal conditions. The application processor/controller presents the data port with a stream of bytes formatted to implement the desired commands.

Figure 7-3. ADC Application Interface



7.4.1 Use of Application Interface

In a typical programming operation, a command byte is written. After the first byte of information is driven to the ADC (which is the instruction opcode), the ADC drives the 0th bit of the ADC_STATUS register as '1' indicating that ADC is busy in processing the instruction. The next byte of information should be given to the ADC only if the 1st bit of status is set to '1'. This bit is also set when the read data is ready on the rd_data port of the ADC. This bit is monitored by the host processor/controller to give the next byte of data.

7.4.2 Status Codes

The ADC returns a 6-bit status code (bits 7:2 in the Status register ADC_STATUS) after an instruction is executed. The status code when not 0 indicates an error condition. Table 7-1shows the three error codes that are associated with the ADC operation.

The errored status code is only cleared after the next command instruction is written into the ADC.

Table 7-1. Status Code for ADC

Status Codes bits [7:2]	Meaning
0x0	Operation Successful
0x3	Invalid Instruction Code
0xC	Error Status Flag

7.4.3 ADC Usage Guidelines

The temperature sensor block needs to be powered up by enabling the TS_EN before enabling the modulator. This is required as this bit sets the analog ground reference for the ADC. A minimum of 15 µs of start-up time is required for the sensor core to settle to its stable voltage level.

ADC is required to be reset before every conversion to discharge the accumulator capacitor. This is done by disabling the modulator and holding it in this state for a minimum of 5 µs. Programming flexibility for resolution, MUX inputs to the modulator and comparator are provided.

The Table 7-2 shows the configuration required for a 10-bit conversion at 12 MHz sampling frequency.

Table 7-2. Configuration Values - 12 MHz, 10-Bit Conversion

Register Address	Register Name	Configuration Value
0x3C	MOD_CR1	0Ch
0x3D	ACLK	02h
0x3E	TS_CR0	01h
0x41	Timer Period L	FFh
0x42	Timer Period H	03h
0x55	CMP_CR0	00h
0x3B	MOD_CR0	81h

Enabling the modulator and timer starts the clocks to the ADC block and thereby the conversion. It is recommended that the modulator and the timer/counter be enabled at the same time to start the ADC conversion. A minimum of 2 µs wait time is required for the modulator to be enabled after the other registers are configured. It is desired that for reliable operation, register MOD_CR0 (3Bh) be written to last after all the others.

7.4.4 Typical ADC Operation Procedure

The ADC registers in the temperature sensor block are to be controlled using the command interface as per details provided in Section 7.4. The following steps with the register values as shown in Table 7-2 configure the ADC for 12 MHz, 10-bit conversion.

- Configure the GPIO for an external input or route an internal voltage through the analog global.
- Configure registers as mentioned in Table 7-2. Select analog input source from MOD_CR1 register.
- Select required clock speed with ACLK register.
- TS_EN also controls the analog ground reference generation. This bit is also required to be set for ADC only operation.
- Select comparator inputs. The default setting of 00h is the required setting for this operation.
- Configure the timer period register based on the resolution required. Each conversion is required to run 2n cycles (where n is the resolution).
- Hold the settings for 15 μs to allow the temperature sensor core output to settle before enabling the timer/counter and the modulator through MOD_CR0 register
- If the ADC interrupt is enabled, an interrupt occurs at the end of conversion and the counter register holds the equivalet digital reading.
- For a new conversion, disable the modulator and timer/counter and wait for 5μs before re-enabling them.

8. Internal Main Oscillator (IMO)

This chapter presents the Internal Main Oscillator (IMO) and its associated registers. The IMO produces clock signals of 6, 12, and 24 MHz. For a complete table of the IMO registers, refer to the Summary Table of the Core Registers on page 24. For a quick reference of all enCoRe V registers in address order, refer to the Register Reference chapter on page 163.

8.1 Architectural Description

The Internal Main Oscillator (IMO) outputs a clock that is normally driven to the main system clock, SYSCLK. The IMO clock frequency can be configured as 6, 12, or 24 MHz.

The accuracy of the internal IMO clock is approximately ±5% over temperature and voltage variation. No external components are required to achieve this level of accuracy. The IMO provides higher accuracies when enabled for locking to USB traffic during USB operation. See Full-Speed USB chapter on page 141 for more information. The frequency doubler circuit, which produces SYSCLKX2, can be disabled to save power.

Registers for controlling these operations are found in the Digital Clocks chapter on page 96.

Table 8-1. IMO Frequencies

SLIMO	CY7C6xxxx
00	12
01	6
10	24
11	Reserved

8.2 Application Overview

Device power may be optimized by selecting among the 24, 12, or 6 MHz settings using the SLIMO bits in the CPU_SCR1 register in conjunction with associated trim values in the IMO_TR register. Both methods are described later in this document.

8.2.1 Trimming the IMO

An 8-bit register (IMO_TR) is used to trim the IMO. Bit 0 is the LSB and bit 7 is the MSB. The trim step size is approximately 60 kHz at the 24-MHz clock setting. A factory trim setting is loaded into the IMO_TR register at boot time.

8.2.2 Engaging Slow IMO

Writing to the SLIMO bits of the CPU_SCR1 register enables the Slow IMO feature. SLIMO settings for 6 and 12 MHz are listed in Table 8-1. When changing frequency ranges, the associated factory trim value must be loaded into the IMO_TR register. The IMO immediately changes to the new frequency. Factory trim settings are stored in flash for the frequencies listed in Table 8-1.

8.3 Register Definitions

The following registers are associated with the Internal Main Oscillator (IMO). The register descriptions have associated register tables showing the bit structure for that register. The bits in the tables that are grayed out are reserved bits and are not detailed in the register descriptions that follow. Always write reserved bits with a value of '0'. For a complete table showing all oscillator registers, refer to the Summary Table of the Core Registers on page 24.

8.3.1 IMO_TR Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,E8h	IMO_TR	Trim[7:0]								RW : 00

The Internal Main Oscillator Trim Register (IMO_TR) manually centers the oscillator's output to a target frequency.

This register is loaded with a factory trim value at boot. When changing frequency ranges, the matching frequency trim value must be loaded into this register.

A TableRead command to the Supervisory ROM returns the trim values to the SRAM. EraseAll Parameters (05h), on page 35 has information on the location of various trim settings stored in flash tables. Firmware needs to read the right trim value for desired frequency and update the IMO_TR register. The IMO_TR register must be changed at the lower frequency range setting.

For additional information, refer to the IMO_TR register on page 233

8.3.2 IMO_TR1 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,FAh	IMO_TR1						Fine Trim[2:0]			RW : 0

The Internal Main Oscillator Trim Register 1 (Encore V IMO_TR1) adjusts the IMO frequency.

Bits 2 to 0: Fine Trim[2:0]. These bits provide a fine tuning capability to the IMO trim. These three bits are the 3 LSB of the IMO trim with the IMO_TR register supplying the 8 MSB. A larger value in this register will increase the speed of the oscillator. The value in these bits varies the IMO frequency: approximately 7.5 kHz/step. When the EnableLock bit is set in the USB_CR1 register, firmware writes to this register are disabled.

For additional information, refer to the IMO_TR1 register on page 238.

8.3.3 CPU_SCR1 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
x,FEh	CPU_SCR1	IRESS			SLIMO[1:0]				IRAMDIS	# : 0

Legend
x An "x" before the comma in the address field indicates that this register can be read or written to no matter what bank is used.
Access is bit specific. Refer to the Register Reference chapter on page 163 for additional information.

The System Status and Control Register 1 (CPU_SCR1) conveys the status and control of events related to internal resets and watchdog reset.

Bit 7: IRESS. The Internal Reset Status bit is a read-only bit that determines if the booting process occurred more than once.

When this bit is set, it indicates that the SROM SWBootReset code ran more than once. If this bit is not set, the SWBootReset ran only once. In either case, the SWBootReset code does not allow execution from code stored in flash until the M8C core is in a safe operating mode with respect to supply voltage and flash operation. There is no need for concern when this bit is set. It is provided for systems that may be sensitive to boot time, so that they can determine if the normal one pass boot time was exceeded. For more information on the SWBootReest code, see the Supervisory ROM (SROM) chapter on page 32.

Bit 4 to 3: SLIMO[1:0]. These bits set the IMO frequency range. See the following table for more information.

These changes allow optimization of speed and power. The IMO trim value must also be changed when SLIMO is changed (see Engaging Slow IMO on page 67). When not in external clocking mode, the IMO is the source for SYSCLK; therefore, when the speed of the IMO changes so does SYSCLK.

SLIMO	CY7C6xxxx
00	12
01	6
10	24
11	Reserved

Bit 0: IRAMDIS. Initialize RAM Disable. This bit is a control bit that is readable and writeable. The ***default value*** for this bit is '0', which indicates that the maximum amount of SRAM must be initialized upon watchdog reset to a value of 00h. When the bit is '1', the minimum amount of SRAM is initialized after a watchdog reset.

For additional information, refer to the CPU_SCR1 register on page 210.

8.3.4 OSC_CR2 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,E2h	OSC_CR2				CLK48MEN		EXTCLKEN	RSVD		RW : 00

The Oscillator Control Register 2 (OSC_CR2) configures various features of internal clock sources and clock nets.

Bit 4: CLK48MEN. This is the 48-MHz clock enable bit. '0' disables the bit and '1' enables the bit. This register setting applies only when the device is not in OCD mode. When in OCD mode, the 48-MHz clock is always active.

Bit 2: EXTCLKEN. When the EXTCLKEN bit is set, the external clock becomes the source for the internal clock tree, SYSCLK, which drives most device clocking functions. All external and internal signals, including the low-speed oscillator, are synchronized to this clock source. The external clock input is located on P1[4]. When using this input, the pin drive mode must be set to High-Z (not High-Z analog), such as drive mode 11b with PRT1DR bit 4 set high.

Bit 1: RSVD. This is a reserved bit. It should always be 0..

For additional information, refer to the OSC_CR2 register on page 230.

8.3.5 Related Registers

- OSC_CR2 Register on page 102.
- CPU_SCR1 Register on page 116.

8.4 Timing Diagrams

The IMO startup time is 1.5 µs. The worst case skew between SYSCLK and SYSCLKX2 in Krypton is 470 ps. BROS 001-13648 has more information.

8.5 Clocking Strategy

No clocks are needed for this block integration in the chip.

8.6 Usage Guidelines

Oscillator clock frequency is adjusted to it operating frequency at a given process corner using the trim bits of the DAC.

8.6.1 Power Down Guidelines

The 36 MHz oscillators for SPC control shares resources with main oscillator, therefore, 36 MHz clock has to be powered down prior to main oscillator is powered down. The PD applied to the main oscillator will power down the 36MHz output but the state of the SPC clock may not be known as it powers down the entire circuit when SYSCLK clock goes low irrespective of the state of the SPC clock. This ensures circuit powers down when state of the clock is low.

Current/Voltage Reference guidelines.

Output clock frequency is very sensitive to band gap reference current source. A trimmed (over process corner) and stable (across V-T) current reference/voltage reference should be used for stable output frequency.

A high accuracy current reference of 10 µA is required. The accuracy of oscillator largely depends on the accuracy of the current reference input.Prerequisite IP

8.7 Block Size/Area

Table 8-2. IMO Block Size

	BROS	BR1	BR2	BR3	BR4
Block Size (um x um)					
Block Area (um^2)					

8.8 Gate Count

Table 8-3. IMO Gate Count

	BROS	BR1	BR2	BR3	BR4
Gate Count					

8.9 Block Pin List

Table 8-4. IMO Signals

Name	Type	Direction	Description
General			
vpwr	Supply	Input	Power Supply
vgnd	Supply	Input	Ground
F2xOFF	CMOS	Input	Powers down the frequency doubler. Active High.
RESET	CMOS	Input	Reset, forces clock outputs to ground. Active High.
VREF1	Analog	Input	0.6V voltage reference
VREF2	Analog	Input	1.2V voltage reference
IIN	Analog	Input	Bandgap current reference used as input to the DAC
PDC	CMOS	Input	System-wide power down. Powers down CLKout and Doubler. Active High.
PDX	CMOS	Input	Powers down CLKout and Doubler. Active High.
XCLK	CMOS	Input	External clock input
XCSEL	CMOS	Input	CLKMUX control. Selects XCLK when set high
SYSCLK	CMOS	Output	6/12 MHz clock output
SYSCLKx2	CMOS	Output	Doubled (12/24 MHz) clock output

8.10 Block Level Interfaces

8.11 Initialization

8.12 Wounding

This block cannot be wounded.

8.13 On-Chip Debugger Modes

There are no on-chip debugger modes associated with this block.

8.14 Test Modes

There are no test modes associated with this block.

8.15 Power Modes

8.16 Design Flow

This block uses an analog design flow.

8.17 Operating Condition Requirements

Table 8-5. IMO Operating Condition Requirements

Operating Condition	Description	Specification		Units
		Min	Max	

8.18 DC Specifications

Table 8-6. DC IMO Specifications

DC Param.	Description	Block Specification			Design Target		Simulation Results		BR4 Char	Units
		Min	Typ	Max	Min	Max	Min	Max	Actual	Corner, Temp., Vdd

8.19 AC Specifications

Table 8-7. AC IMO Specifications

AC Param.	Description	Block Specification			Design Target		Simulation Results		BR4 Char	Units
		Min	Typ	Max	Min	Max	Min	Max	Actual	Corner, Temp., Vdd

9. Internal Low-speed Oscillator (ILO)

This chapter briefly explains the Internal Low-speed Oscillator (ILO) and its associated register. The Internal Low-speed Oscillator produces a 32-kHz or 1-kHz clock. For a quick reference of all enCoRe V registers in address order, refer to the Register Reference chapter on page 163.

9.1 Architectural Description

The Internal Low-speed Oscillator (ILO) is an oscillator with a nominal frequency of 32 kHz or 1 kHz. It is used to generate sleep wakeup interrupts and watchdog resets. This oscillator is also used as a clocking source for the digital blocks. This block operates with a small internal bias current and produces an output clock of either 1 kHz or 32 kHz, configurable by the user. The ILO is trimmed for 32 kHz in production devices. There is no trim for 1 kHz, hence, high variation is expected from nominal value.

The block operates by charging a capacitor with a current, to a reference level. When reached, the capacitor is discharged to ground. This process repeats to provide the oscillator (half) period.

9.2 Register Definitions

The following register is associated with the Internal Low-speed Oscillator (ILO). The register description has an associated register table showing the bit structure. The bits in the table that are grayed out are reserved bits and are not detailed in the register description that follows. Always write reserved bits with a value of '0'.

9.2.1 ILO_TR Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,E9h	ILO_TR		PD_MODE	ILOFREQ		Freq Trim[3:0]				RW : 18

The Internal Low-speed Oscillator Trim Register (ILO_TR) sets the adjustment for the internal low-speed oscillator.

Bit 6: PD_MODE. This bit selects power down mode. Setting this bit high disables the oscillator and current bias when the ILO is powered down, which results in slower startup time. Setting this bit low keeps the small current bias running when the ILO is powered down, which results in faster startup time.

Bit 5: ILOFREQ. When this bit is set, the oscillator operates at a nominal frequency of 1 kHz, otherwise, it runs at the default 32 kHz.

Bits 3 to 0: FREQ_TRIM[3:0]. These bits trim the oscillator frequency. The device-specific value, placed in the trim bits of this register at boot time, is based on factory testing. ***Do not alter the values in the register***.

For additional information, refer to the ILO_TR register on page 234.

10. External Crystal Oscillator (ECO)

This chapter briefly explains the External Crystal Oscillator (ECO) and its associated registers. The 32.768-kHz external crystal oscillator circuit allows the user to replace the internal low-speed oscillator with a more precise time source. For a quick reference of all enCoRe V registers in address order, refer to the Register Reference chapter on page 163.

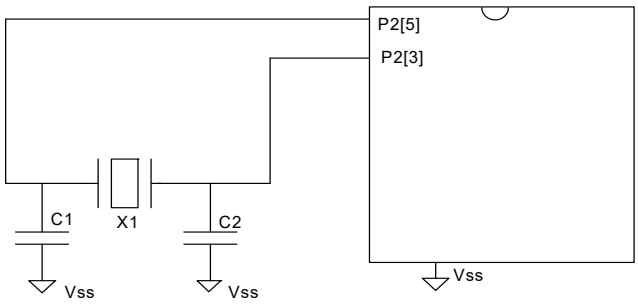
10.1 Architectural Description

The External Crystal Oscillator (ECO) circuit requires only the following external components: an inexpensive watch crystal and two small value capacitors. The XTALIn (P2[3]) and XTALOut (P2[5]) pins connect to a 32.768-kHz watch crystal and the two external capacitors bypass these pins to ground. Figure 10-1 shows the external connections needed to implement the ECO. See the Application Overview on page 75 for information on enabling the ECO. Transitions between the internal and external oscillator domains may produce glitches on the clock bus.

During the process of activating the ECO, there must be a hold-off period before using it as the 32.768-kHz source. This hold-off period is partially implemented in hardware using the sleep timer. Firmware must set up a sleep period of one second (maximum ECO settling time), and then enable the ECO in the OSC_CR0 register. At the one second timeout (the sleep interrupt), the switch is made by hardware to the ECO. If the ECO is subsequently deactivated, the ILO will again be activated and the switch is made back to the ILO immediately.

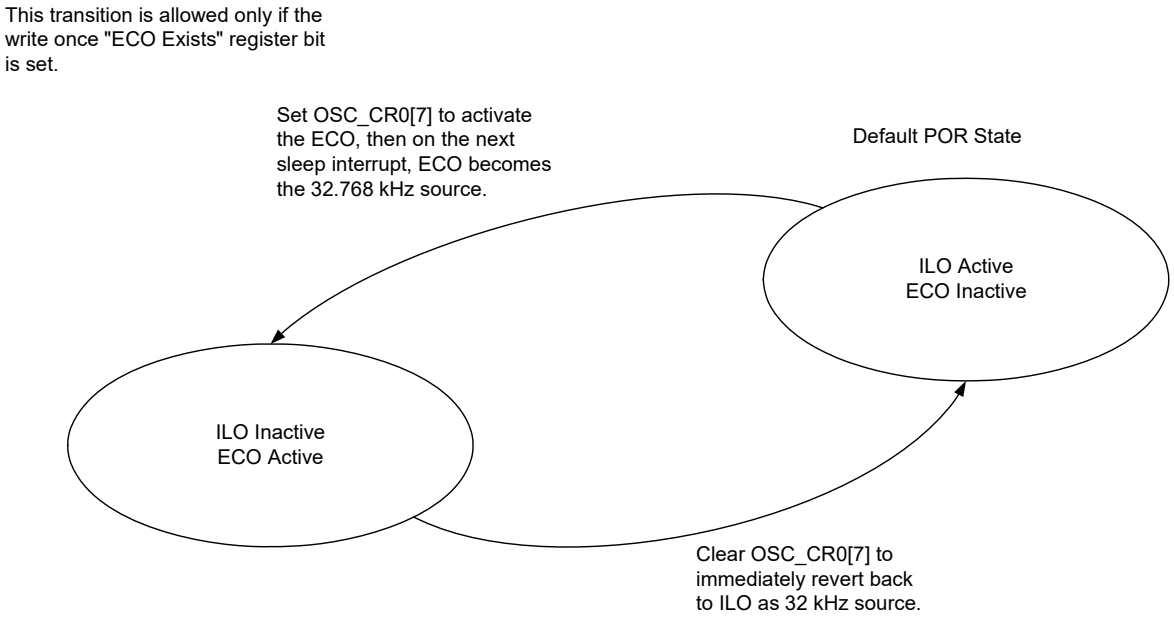
The ECO Exists bit (ECO EX, bit 0 of ECO_CONFIG) is used to control whether the switch-over is allowed or locked. This is a write-once bit. It is written early in code execution after a power-on-reset (POR) or external reset (XRES) event. A '1' in this bit indicates to the hardware that a crystal exists in the system, and firmware is allowed to switch back and forth between ECO and ILO operation. If the bit is '0', switch-over to the ECO is locked out.

Figure 10-1. External Components for the ECO



The ECO Exists Written bit (ECO EXW, bit 1 of ECO_CONFIG) is read only and is set on the first write to this register. When this bit is '1', it indicates that the state of ECO EX is locked. This is illustrated in Figure 10-2.

Figure 10-2. State Transition Between ECO and ILO



10.2 Application Overview

To use a 32.768-kHz external crystal, the GPIO pins that connect to the crystal must be set to the High-impedance drive mode. See the General-Purpose I/O (GPIO) chapter on page 52 for information on GPIOs and their drive modes.

The firmware steps involved in switching between the ILO to the 32.768-kHz ECO are as follows.

At reset, the device begins operation using the ILO.

1. Set the ECO EX bit to allow crystal operation.
2. Modify bits [2:0] in the External Crystal Oscillator ENBUS Register to be 011b.
3. Select a sleep interval of one second, using bits[4:3] in the Oscillator Control Register 0 (OSC_CR0), as the oscillator stabilization interval.
4. Set bit [7] in the Oscillator Control Register 0 (OSC_CR0) to '1' to enable the external crystal oscillator.
5. The ECO becomes the selected source at the end of the one-second interval on the edge created by the sleep interrupt logic. The one-second interval gives the oscillator time to stabilize before it becomes the active source. The sleep interrupt need not be enabled for the switchover to occur. Reset the sleep timer (if this does not interfere with any ongoing realtime clock operation), to guarantee the interval length. Note that the ILO continues to run until the oscillator is automatically switched over by the sleep timer interrupt.

Note The ILO switches back instantaneously by writing the 32-kHz Select Control bit to '0'.

Note Transitions between oscillator domains may produce glitches on the 32-kHz clock bus. Functions that require accuracy on the 32-kHz clock should be enabled after the transition in oscillator domains.

10.3 Register Definitions

These registers are associated with the external crystal oscillator.

10.3.1 ECO_ENBUS Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,D2h	ECO_ENBUS						ECO_ENBUS[2:0]			RW : 07

The ECO_ENBUS register is used to disable and enable the external crystal oscillator (ECO).

Bits 2 to 0 ECO_ENBUS[2:0]. 111b – Default. Disables the external crystal oscillator (ECO).

011b – Allows the ECO to be enabled by bits in the ECO_CFG register.

Other values are reserved. See the Application Overview on page 75 for the proper sequence for enabling the ECO.

For additional information, refer to the ECO_ENBUS register on page 221.

10.3.2 ECO_TRIM Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,D3h	ECO_TRIM				ECO_XGM[2:0]			ECO_LP[1:0]		RW : 11

The ECO TRIM Register (ECO_TRIM) controls gain and power settings for the 32-kHz crystal oscillator.

These settings should not be changed from their default state. The default value is 14h.

Bits 4 to 2: ECO_XGM[2:0]. These bits set the amplifier gain. In high-power mode (ECO_LPM=0), the step size of the current source is approximately 400 nA, and the highest source current is with the '000' setting. In low-power mode (ECO_LPM=1), the overall power is approximately 5% lower with the '000' setting than with the '111' setting.

’000’ is the highest gain setting, and has the lowest power in low-power mode (5% power reduction).

’111’ is the lowest gain setting.

This value is factory trimmed; the typical value is '101'.

Bits 1 to 0: ECO_LP[1:0]. These bits set the gain mode.

’00’ is the highest power setting.

’11’ is the lowest power setting. (30% power reduction).

The default value is '00'.

For additional information, refer to the ECO_TRIM register on page 222.

10.3.3 ECO_CFG Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,E1h	ECO_CFG						ECO_LPM	ECO_EXW	ECO_EX	RW : 00

The ECO Configuration Register provides status and control for the ECO.

Bit 2 ECO_LPM. This bit enables the ECO low-power mode when high. This is recommended for use only during sleep mode.

Bit 1 ECO_EXW. The ECO Exists Written bit is used as a status bit to indicate that the ECO EX bit was previously written to. It is read only. When this bit is a '1' indicates that the ECO_CFG register was written to and is now locked.

Bit 0 ECO_EX. The ECO Exists bit serves as a flag to the hardware, to indicate that an external crystal oscillator exists in the system. Just after boot, it may be written only once to a value of '1' (crystal exists) or '0' (crystal does not exist).

If the bit is '0', a switch-over to the ECO is locked out by hardware. If the bit is '1', hardware allows the firmware to

freely switch between the ECO and ILO. It should be written as early as possible after a POR or XRES event.

For additional information, refer to the ECO_CFG register on page 229

10.3.4 Related Registers

- OSC_CR0 Register, on page 101.
- PRTxDR Registers register on page 56.
- PRTxIE Registers register on page 56.

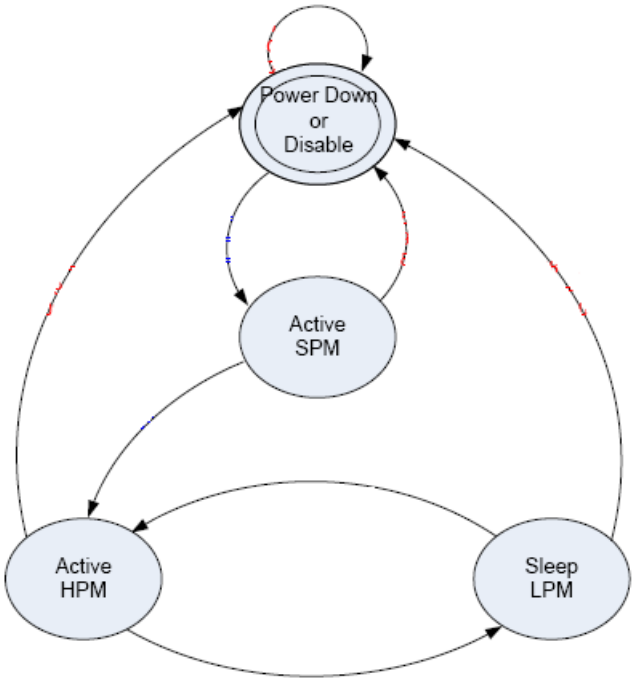
10.4 Usage Modes and Guidelines

This block operates at extremely low current levels, making it vulnerable to coupled noise. Take care to avoid coupling noise from neighboring signal pins. The block is designed to accept a wide variety of 6- or 12-pF crystals. There are four distinct operating power modes:

- Power-down (PDM)
- Startup mode (SPM)
- High-power mode (HPM)
- Low-power mode (LPM)

There are specific sequences that are used to switch between each mode; typically a small delay is required after each instruction.

Figure 10-3. Power Mode State Machine



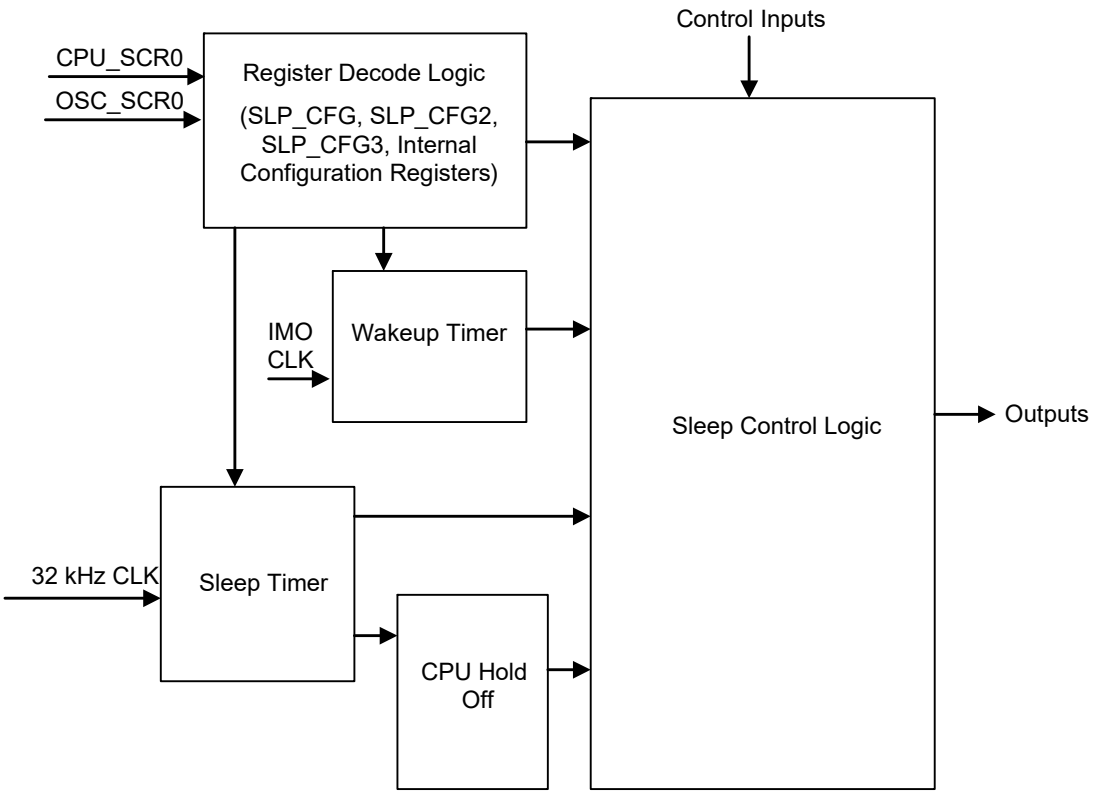
11. Sleep and Watchdog

This chapter discusses the Sleep and Watchdog operations and their associated registers. For a complete table of the Sleep and Watchdog registers, refer to the Summary Table of the Core Registers on page 24. For a quick reference of all enCoRe V registers in address order, refer to the Register Reference chapter on page 163.

11.1 Architectural Description

Device components that are involved in Sleep and Watchdog operation are the selected 32-kHz clock, the wakeup timer, the Sleep bit in the CPU_SCR0 register, the sleep circuit (to sequence going into and coming out of sleep), the bandgap refresh circuit (to periodically refresh the reference voltage during sleep), and the ***watchdog timer***.

Figure 11-1. Sleep Controller Architecture



11.1.1 Sleep Control Implementation Logic

This section details the sleep mode logic implementation.

Conditions for entering the sleep modes:

- Standby Mode: Set the SLEEP bit in the CPU_SCR0 register. This asserts the “sleep” signal for the sleep controller.
- I2C_USB Mode: Set the I2C_ON bit in the SLP_CFG2 register and then set the SLEEP bit in the CPU_SCR0 register. Another way to enter I2C_USB sleep mode is to set the USB Enable bit in the USB_CR0 register and then set the SLEEP bit in the CPU_SCR0 register. This asserts the sleep signal for the sleep controller and also the I2CEnable signal to the power system.

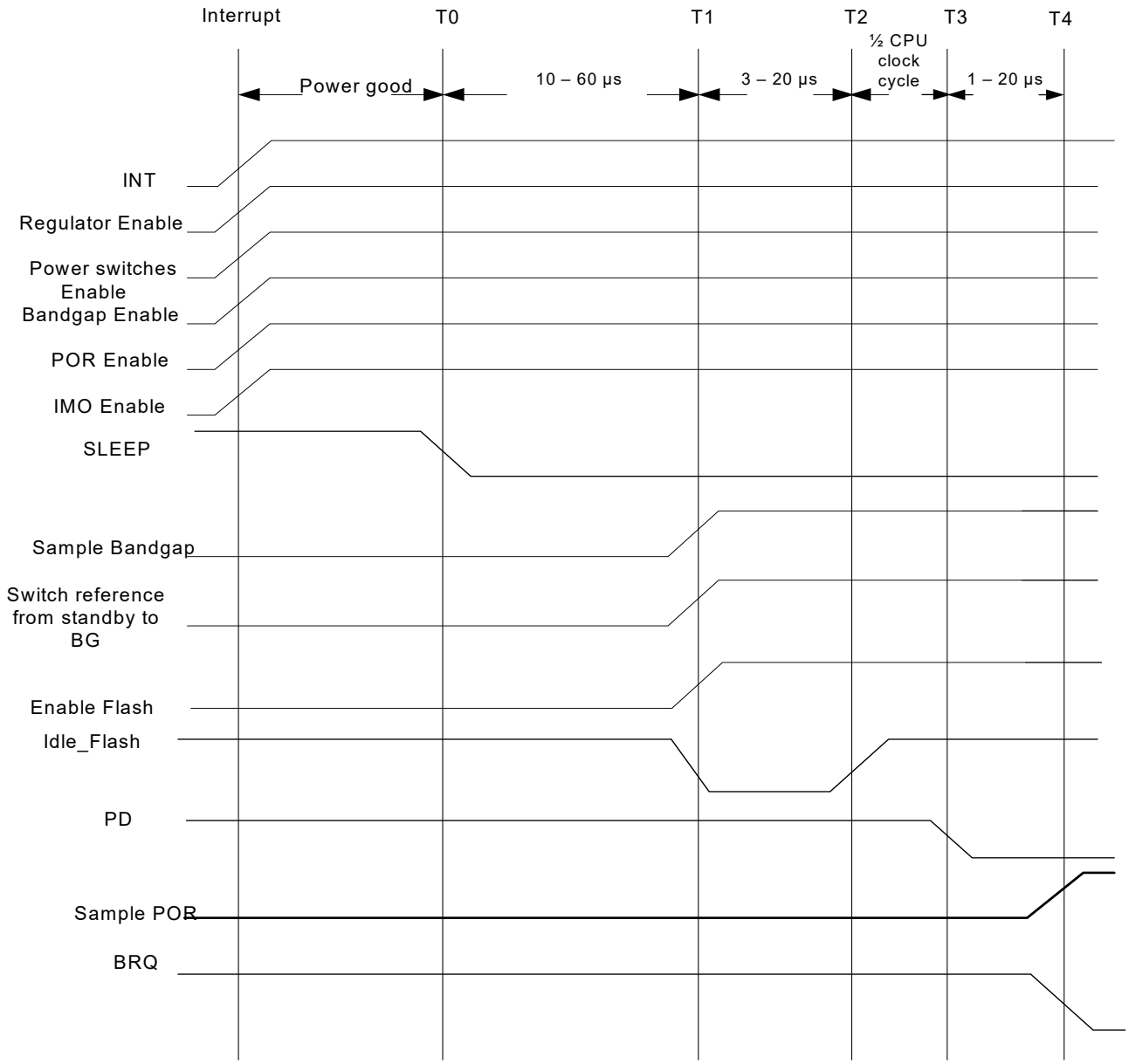
  The I2C block works in I2C_USB sleep mode only to wake up the system. That is, when the device is in sleep, I2C can detect a start condition and receive an address. If the address matches, I2C generates an interrupt and wakes the system (refer to Power Modes on page 120). If you put the device to sleep again while these transactions are occurring (i.e., when you are in the middle of I2C transactions), I2C does not work and will send NAKs. I2C can only detect a start condition and collect an 8-bit address then wake the system through an interrupt during I2C sleep mode. Therefore, it is recommended to check the bus status in the I2C_XSTAT register before putting the device to sleep if there is any I2C data transfer. To ensure data retention in the 32-byte I2C buffer during sleep, the I2C_ON bit in SLP_CFG2 register should be set before entering sleep state.
- Deep Sleep Mode: Configure the I2C_ON bit in the SLP_CFG2 register to 0, then USB Enable bit in the USB_CR0 register to '0' and the X32ON bit in OSC_CR0 to '0'. Set the LSO_OFF bit in the SLP_CFG2 register and then set the “SLEEP” bit in the CPU_SCR0 register. This enables the LSO_OFF signal to power down the LSO. The system enters into deep sleep mode. One point to note here is to not set the X32ON bit to '1' without setting the ECO_EX (ECO exists) bit in the ECO_CFG (1,E1h) register to a '1'. If you do so, the deep sleep mode is not entered, but clk32K is also not running. This implies that the sleep timer interrupt or the programmable timer interrupt cannot occur.

11.1.1.1 Wakeup Logic

- Waking up from standby mode is by an interrupt, which can be a sleep timer interrupt, a GPIO interrupt, a 16-bit programmable timer 0 interrupt, or a USB interrupt.
- For the device, the wakeup from I2C_USB sleep mode can be by an I2C interrupt in addition to a sleep timer interrupt, a programmable timer 0 interrupt, a GPIO interrupt, or a USB interrupt.
- For the device, the wakeup from deep sleep mode can be by either a GPIO interrupt or a USB interrupt.
- In standby mode during buzz, if the external supply falls below the LVD limit, an LVD interrupt occurs and initiates the wakeup sequence.
- In standby mode, if watchdog reset occurs, it first initiates the wakeup sequence. After the wakeup is done, it resets the system.

As shown in Figure 11-2, when the SLEEP bit is deasserted, the wakeup is initiated. The sequence is shown in the following timing diagram. The taps used in this wakeup sequence are generated based upon user configuration settings in the SLP_CFG3 register.

Figure 11-2. Wakeup Sequence for the Device[1, 2, 3]



1. The duration of Power Good is 3 ILO Cycles.
2. The timing of T0 – T4 is based on the IMO frequency and the settings in the SLP_CFG3 register. For additional information, refer to the SLP_CFG3 Register on page 83.
3. The maximum worst-case duration of the wakeup sequence is 263 µs, based on the minimum specified ILO frequency of 19 kHz, the minimum specified IMO frequency, and the default settings of the SLP_CFG3 register.

Note The T0, T1, and T2 mentioned in the SLP_CFG3 register with respect to Figure 11-2 on page 80 are defined as follows:

- T0: Time duration between T0 and T1 in the timing diagram.
- T1: Time duration between T1 and T2 in the timing diagram.
- T2: Time duration between T3 and T4 in the timing diagram.

11.1.2 Sleep Timer

The Sleep Timer is a 15-bit up counter clocked by the 32-kHz clock source. This timer is always enabled except in deep sleep mode. The exception to this is within an ***ICE*** (**in-circuit *emulator***) in ***debugger*** mode and when the Stop bit in the CPU_SCR0 is set; the sleep timer is disabled, so that the user does not get continual watchdog resets when a breakpoint is hit in the debugger environment.

If the associated sleep timer interrupt is enabled, a periodic interrupt to the CPU is generated based upon the sleep interval selected from the OSC_CR0 register. The sleep timer functionality does not need to directly associate with the sleep state. It can be used as a general-purpose timer interrupt regardless of sleep state.

The reset state of the sleep timer is a count value of all zeros. There are two ways to reset the sleep timer. Any hardware reset, (that is, POR, XRES, or Watchdog Reset (WDR)) resets the sleep timer. There is also a method that allows the user to reset the sleep timer in firmware. A write of 38h to the RES_WDT register clears the sleep timer.

Note Any write to the RES_WDT register also clears the watchdog timer.

Clearing the sleep timer is done at anytime to synchronize the sleep timer operation to CPU processing. A good example of this is after POR. The CPU hold off, due to voltage ramp and others, may be significant. In addition, a significant amount of program initialization may be required. However, the sleep timer starts counting immediately after POR and is at an arbitrary count when user code begins execution. In this case, it is desirable to clear the sleep timer before enabling the sleep interrupt initially to ensure that the first sleep period is a full interval.

11.2 Application Overview

The following are notes regarding sleep related to firmware and application issues.

Note 1 If an interrupt is pending, enabled, and scheduled to be taken at the instruction boundary after the write to the SLEEP bit, the system does not go to sleep. The instruction still executes, but it cannot set the SLEEP bit in the CPU_SCR0 register. Instead, the interrupt is taken and the effect of the sleep instruction ignored.

Note 2 There is no need to enable the Global Interrupt Enable (CPU_F register) to wake the system out of sleep state. Individual interrupt enables, as set in the interrupt mask registers, are sufficient. If the Global Interrupt Enable is not set, the CPU does not service the ISR associated with that interrupt. However, the system wakes up and continues executing instructions from the point at which it went to sleep. In this case, the user must manually clear the pending interrupt or subsequently enable the Global Interrupt Enable bit and let the CPU take the ISR. If a pending interrupt is not cleared, it is continuously asserted. Although the SLEEP bit may be written and the sleep sequence executed as soon as the device enters sleep mode, the SLEEP bit is cleared by the pending interrupt and sleep mode is exited immediately.

Note 3 Upon wakeup, the instruction immediately after the sleep instruction is executed before the interrupt service routine (if enabled). The instruction after the sleep instruction is prefetched before the system actually goes to sleep. Thus, when an interrupt occurs to wake the system up, the prefetched instruction executes and the interrupt service routine is executed. (If the Global Interrupt Enable is not set, instruction execution continues where it left off before sleep.)

Note 4 If the Global Interrupt Enable bit is disabled, it is safely enabled just before the instruction that writes the SLEEP bit. It is usually undesirable to get an interrupt on the instruction boundary just before writing the SLEEP bit. This means that upon return from the interrupt, the sleep command is executed, possibly bypassing any firmware preparations that are necessary to go to sleep. To prevent this, disable interrupts before making preparations. After sleep preparations, enable global interrupts and write the SLEEP bit with the two consecutive instructions as follows.

```
and f,~01h // disable global interrupts
// (prepare for sleep, could
// be many instructions)
or f,01h // enable global interrupts
mov reg[ffh],08h // Set the sleep bit
```

Because of the timing of the Global Interrupt Enable instruction, it is not possible for an interrupt to occur immediately after that instruction. The earliest for the interrupt to occur is after the next instruction (write to the SLEEP bit) is executed. If an interrupt is pending, the sleep instruction is executed; but as described in Note 1, the sleep instruction is ignored. The first instruction executed after the ISR is the instruction after sleep.

Note 5 When an interrupt occurs after the sleep bit is written and before “PD” (power down signal) is asserted, the interrupt is ignored. The time from the SLEEP bit being written to PD being asserted is 2.5 CPU cycles. Thus, if the interrupt occurs within this period, it is ignored. Refer to Figure 11-3 on page 84 for sleep. The interrupts that occur after this period, that is, when the device is in sleep, are considered and the device wakes up.

11.3 Register Definitions

The following registers are associated with Sleep and Watchdog operations and are listed in address order. Each register description has an associated register table showing the bit structure for that register. The bits that are grayed out in the the following tables are reserved bits and are not detailed in the register descriptions. Always write reserved bits with a value of '0'. For a complete table of the Sleep and Watchdog registers, refer to the Summary Table of the Core Registers on page 24.

11.3.1 RES_WDT Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,E3h	RES_WDT	WDSL_Clear[7:0]								W : 00

The Reset Watchdog Timer Register (RES_WDT) clears the watchdog timer (a write of any value) and clears both the watchdog timer and the sleep timer (a write of 38h).

Bits 7 to 0: WDSL_Clear[7:0]. The Watchdog Timer (WDT) write-only register is designed to timeout at three sleep timer rollover events. If only the WDT is cleared, the next Watchdog Reset (WDR) occurs anywhere from two to three times the current sleep interval setting. If the sleep timer is near the beginning of its count, the watchdog time-out is closer to three times.

However, if the sleep timer is very close to its ***terminal count***, the watchdog timeout is closer to two times. To ensure a full three times timeout, clear both the WDT and the sleep timer. In applications that need a realtime clock and cannot reset the sleep timer when clearing the WDT, the duty cycle at which the WDT must be cleared is no greater than two times the sleep interval.

For additional information, refer to the RES_WDT register on page 207.

11.3.2 SLP_CFG Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,EBh	SLP_CFG	PSSDC[1:0]								RW : 0

The Sleep Configuration Register (SLP_CFG) sets the sleep duty cycle.

The value placed in this register is based upon factory testing.

Bits 7 and 6: PSSDC[1:0]. The Power System Sleep Duty Cycle bits set the sleep duty cycle.

For additional information, refer to the SLP_CFG register on page 235.

11.3.3 SLP_CFG2 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,ECh	SLP_CFG2					ALT_Buzz [1:0]		I2C_ON	LSO_OFF	RW : 00

The Sleep Configuration Register (SLP_CFG2) holds the configuration for I2C sleep, deep sleep, and buzz.

Bits 3 and 2: ALT_Buzz[1:0]. These bits control additional selections for POR/LVD buzz rates. These are lower rates than the compatibility mode to provide for lower average power.
‘00’ - Compatibility mode, buzz rate determined by PSSDC bits.
‘01’ - Duty cycle is 1/32768.
‘10’ - Duty cycle is 1/8192.
‘11’ - Reserved.

Bit 1: I2C_ON. This bit enables the standby regulator in sleep at a level sufficient to supply the I2C circuitry. It is independent of the LSO_OFF bit.

To ensure data retention in the 32-byte I2C buffer during sleep, the I2C_ON bit should be set before entering sleep state.

Bit 0: LSO_OFF: This bit disables the LSO oscillator when in sleep state. By default, the LSO oscillator runs in sleep. When this bit is ‘0’, the standby regulator is active at a power level to supply the LSO and Sleep timer circuitry and the LSO is enabled. When this bit is ‘1’, the LSO is disabled in sleep, which in turn, disables the Sleep Timer, Watchdog Timer, and POR/LVD buzzing activity in sleep. If I2C_ON is not enabled and this bit is set, the device is in the lowest power deep sleep mode. Only a GPIO interrupt awakens the device from deep sleep mode.

For additional information, refer to the SLP_CFG2 register on page 236.

11.3.4 SLP_CFG3 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,EDh	SLP_CFG3		DBL_TAPS	T2TAP [1:0]		T1TAP [1:0]		T0TAP [1:0]		RW : 0x7F

The Sleep Configuration Register (SLP_CFG3) holds the configuration of the wakeup sequence taps.

It is strongly recommended to not alter this register setting.

Bit 6: DBL_TAPS. When this bit is set, all the tap values (T0, T1, and T2) are doubled for the wakeup sequence.

Bits 5 and 4: T2TAP[1:0]. These bits control the duration of the T2-T4 sequence (see Figure 11-2 on page 80) by selecting a tap from the Wakeup Timer. Note The T2 delay is only valid for the wakeup sequence. It is not used for the buzz sequence.

‘00’ - 1 µs
‘01’ - 2 µs
‘10’ - 5 µs
‘11’ - 10 µs

Bits 3 and 2: T1TAP[1:0]. These bits control the duration of the T1-T2 sequence (see Figure 11-2 on page 80) by selecting a tap from the Wakeup Timer.

‘00’ - 3 µs
01’ - 4 µs
‘10’ - 5 µs
‘11’ - 10 µs

Bits 1 and 0: T0TAP[1:0]. These bits control the duration of the T0-T1 sequence (see Figure 11-2 on page 80) by selecting a tap from the Wakeup Timer.
‘00’ - 10 µs
‘01’ - 14 µs
‘10’ - 20 µs
‘11’ - 30 µs

For additional information, refer to the SLP_CFG3 register on page 237.

11.3.5 Related Registers

- INT_MSK0 Register on page 49.
- OSC_CR0 Register on page 101.
- ILO_TR Register on page 73.
- CPU_SCR0 Register on page 117.
- CPU_SCR1 Register on page 116.

11.4 Timing Diagrams

11.4.1 Sleep Sequence

The SLEEP bit in the CPU_SCR0 register, is an input into the sleep logic circuit. This circuit is designed to sequence the device into and out of the hardware sleep state. Figure 11-3 shows the hardware sequence to put the device to sleep, defined as follows.

1. Firmware sets the SLEEP bit in the CPU_SCR0 register. The Bus Request (BRQ) signal to the CPU is immediately asserted: This is a request by the system to halt CPU operation at an instruction boundary.
2. The CPU issues a Bus Request Acknowledge (BRA) on the following ***positive edge*** of the CPU clock.
3. The sleep logic waits for the following ***negative edge*** of the CPU clock and then asserts a system wide Power Down (PD) signal. In Figure 11-3, the CPU is halted and the system wide PD signal is asserted.

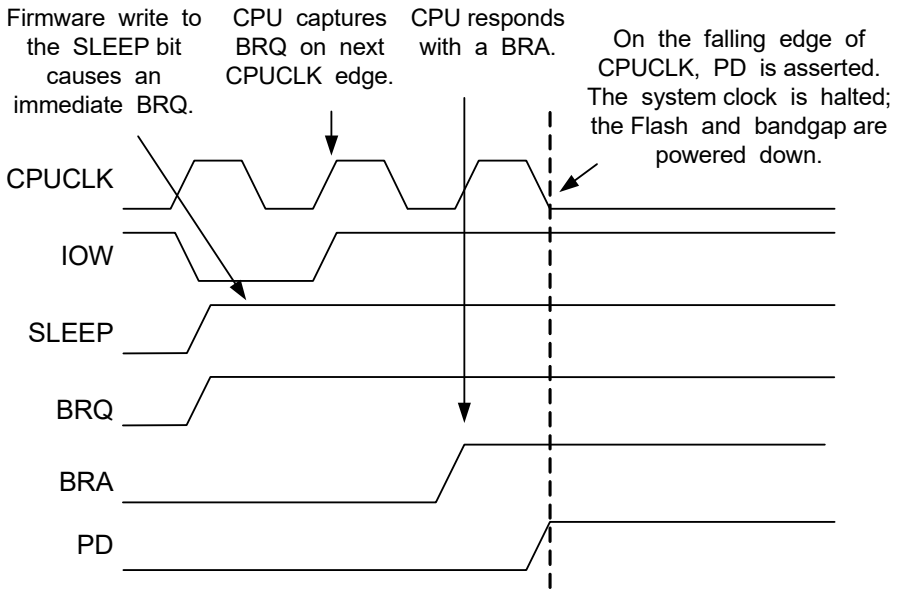
The system-wide PD signal controls three major circuit blocks: the flash memory module, the Internal Main Oscillator (6-/12-MHz oscillator), and the bandgap voltage reference. These circuits transition into a zero power state.

The only operational circuits on the enCoRe V device in standby sleep mode are the ILO, the bandgap refresh circuit, and the supply voltage monitor circuit. In standby sleep mode, the supply voltage monitor circuit is active only during the buzz interval. To properly detect and recover from a VDD brown out condition, the configurable buzz rate must be frequent enough to capture the falling edge of VDD. If the falling edge of VDD is too sharp to be captured by the buzz rate, any of the following actions must be taken to ensure that the device properly responds to a brown out condition.

- Bring the device out of sleep before powering down. This can be accomplished in firmware, or by asserting XRES before powering down.
- Assure that VDD falls below 100 mV before powering back up.
- Set the No Buzz bit in the OSC_CR0 register to keep the voltage monitoring circuit powered during sleep.
- Increase the buzz rate to assure that the falling edge of VDD will be captured. The rate is configured through the PSSDC bits in the SLP_CFG register.

In deep sleep mode, the ILO, bandgap refresh circuit, and supply voltage monitor circuit are all powered down. However, additional low-power voltage monitoring circuitry gets enabled when entering deep sleep. This additional low-power voltage monitoring circuitry allows VDD brown out conditions to be detected for edge rates slower than 1V/ms.

Figure 11-3. Sleep Sequence



11.4.2 Wakeup Sequence

When asleep, the only event that wakes the system is an interrupt. The Global Interrupt Enable of the CPU Flag register does not need to be set. Any unmasked interrupt wakes the system up. It is optional for the CPU to actually take the interrupt after the wakeup sequence.

The wakeup sequence is synchronized to the taps from the wakeup timer (running on IMO clock). This allows the flash memory module enough time to power up before the CPU asserts the first read access. Another reason for the delay is to allow the IMO, bandgap, and LVD/POR circuits time to settle before actually being used in the system. As shown in Figure 11-2, the wakeup sequence is as follows.

1. The wakeup interrupt occurs and the sequence is initiated at INT (shown in Figure 11-2 on page 80). The interrupt asynchronously enables the regulator, the bandgap circuit, LSO, POR, and the IMO. As the core power ramps, the IMO starts to oscillate and the remainder of the sequence is timed with configurable durations from the wakeup timer.
2. At T1, the bandgap is sampled and the flash is enabled.
3. At T2, the flash is put in power saving mode (idle).
4. At T3, the POR/LVD comparators are sampled and the CPU restarts.

There is no difference in wakeup from deep sleep or buzzed sleep because in all cases, to achieve the power specification, the regulator, references, and core blocks must be shut.

11.4.3 Bandgap Refresh

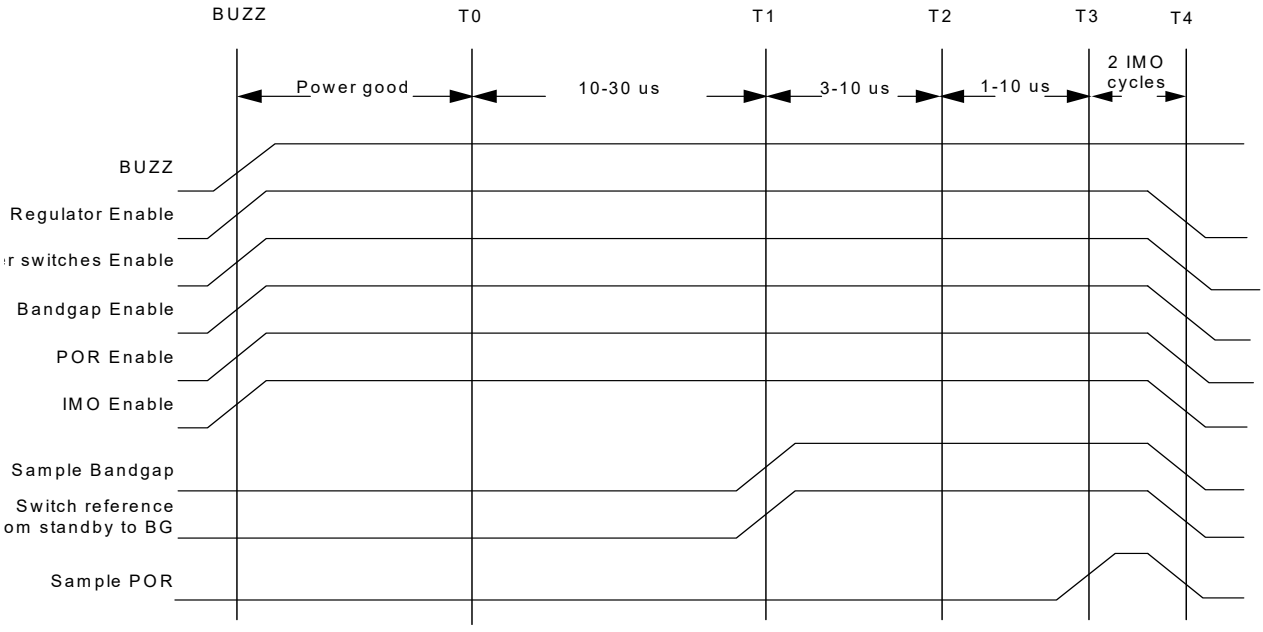
During normal operation the bandgap circuit provides a voltage reference (VRef) to the system for use in the analog blocks, flash, and ***low-voltage detect (LVD)*** circuitry. Normally, the bandgap output is connected directly to the VRef signal. However, during sleep, the ***bandgap reference*** generator block and LVD circuits are completely powered down. The bandgap and LVD blocks are periodically re-enabled during sleep to monitor for low-voltage conditions. This is accomplished by periodically turning on the bandgap.

The rate at which the refresh occurs is related to the 32-kHz clock and controlled by the Power System Sleep Duty Cycle. Table 11-1 lists the available selections.

Table 11-1. Power System Sleep Duty Cycle Selections

PSSDC	Sleep Timer Counts	Period (Nominal)
00b (default)	256	8 ms
01b	1024	31.2 ms
10b	64	2 ms
11b	16	500 µs
Note Valid when ALT_Buzz[1:0] of the SLP_CFG2 register is 00b.		

Figure 11-4. Buzz Sequence Timing



The buzz sequence after the Buzz signal comes. This is shown in Figure 11-4, “Buzz Sequence Timing,” on page 85.

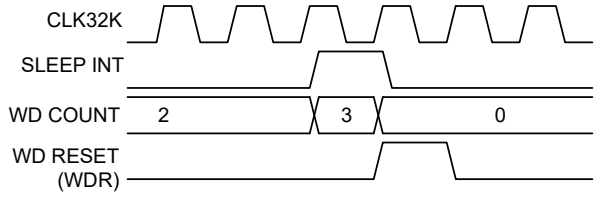
11.4.4 Watchdog Timer

On device boot up, the Watchdog Timer (WDT) is initially disabled. The PORS bit in the System Control register controls the enabling of the WDT. Upon boot, the PORS bit is initially set to '1', indicating that either a POR or XRES event occurred. The WDT is enabled by clearing the PORS bit. After this bit is cleared and the WDT enabled, it cannot be disabled. (The PORS bit cannot be set to '1' in firmware; only cleared.)

The only way to disable the watchdog function after it is enabled is through a subsequent POR or XRES. Even though the WDT is disabled during the first time through initialization code after a POR or XRES, write all code as if it is enabled (that is, periodically review the WDT). This is because in the initialization code after a WDR event, the watchdog timer is enabled so all code must be aware of this.

The watchdog timer is three counts of the sleep timer interrupt output. The watchdog interval is three times the selected sleep timer interval. The available selections for the watchdog interval are shown in Table 11-1. When the sleep timer interrupt is asserted, the watchdog timer increments. When the counter reaches three, a terminal count is asserted. This terminal count is registered by the 32-kHz clock. Therefore, the WDR (Watchdog Reset) signal goes high after the falling edge of the 32-kHz clock and held asserted for one cycle (30 µs nominal). The ***flip-flop*** that registers the WDT terminal count is not reset by the WDR signal when it is asserted, but is reset by all other resets. This timing is shown in Figure 11-5.

Figure 11-5. Watchdog Reset



When enabled, periodically clear the WDT in firmware. Do this with a write to the RES_WDT register. This write is data independent, so any write clears the watchdog timer. (Note that a write of 38h also clears the sleep timer.) If for any reason the firmware fails to clear the WDT within the selected interval, the circuit asserts WDR to the device. WDR is equivalent in effect to any other reset. All internal registers are set to their reset state. (See the table titled Reset Functionality on page 120.) An important aspect to remember about WDT resets is that RAM initialization can be disabled (IRAMDIS is in the CPU_SCR1 register). In this case, the SRAM contents are unaffected; so that when a WDR occurs, program variables are persistent through this reset.

In practical application, it is important to know that the watchdog timer interval can be anywhere between two and three times the sleep timer interval. The only way to guarantee that the WDT interval is a full three times that of the sleep interval is to clear the sleep timer (write 38h) when clearing the WDT register. However, this is not possible in applications that use the sleep timer as a realtime clock. In the case where firmware clears the WDT register without clearing the sleep timer, this occurs at any point in a given sleep timer interval. If it occurs just before the terminal count of a sleep timer interval, the resulting WDT interval is just over two times that of the sleep timer interval.

11.5

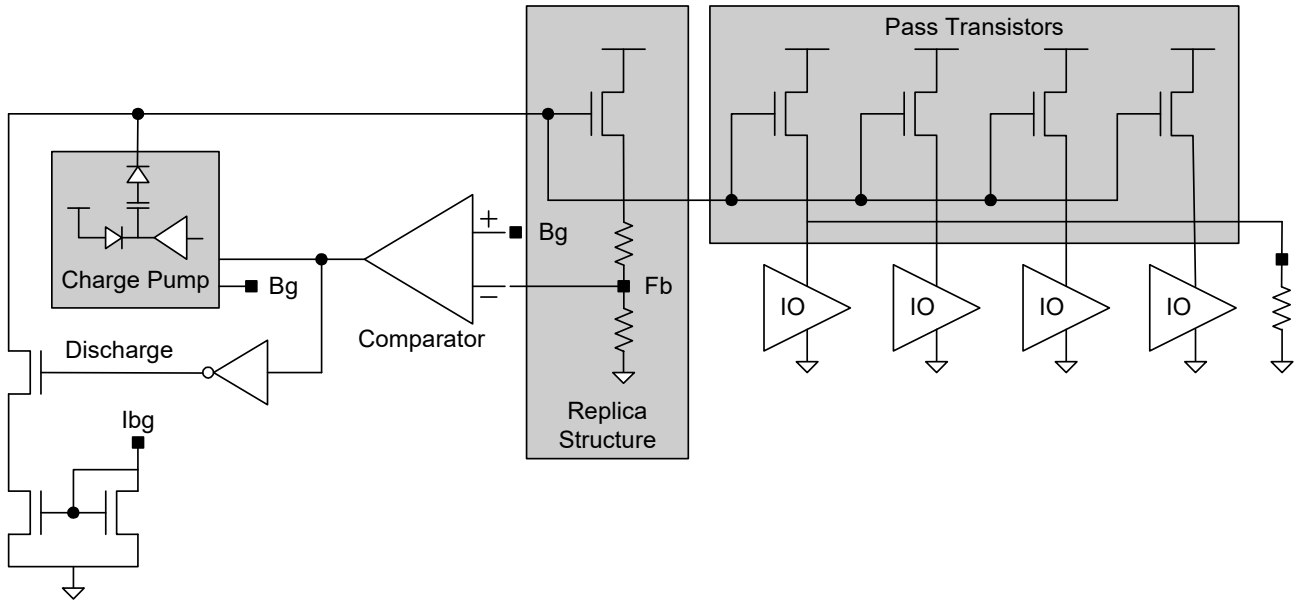
12. Regulated I/O

This chapter presents the architecture of the Regulated I/O and its functionality, along with voltage regulator information. There are no registers associated with the regulated I/O. For a quick reference of all enCoRe V registers in address order, refer to the Register Reference chapter on page 163.

12.1 Architectural Description

The Regulated I/O is an NMOS replica bias voltage regulator. This I/O regulator has two operating ranges. For a chip supply between 3.1V and 5.5V it can be configured to regulate the I/O output voltage to 3.0V, 2.4V, or 1.8V. For a chip Vdd between 2.4V to 3.0V it can provide regulated output voltage of 1.8V.

Figure 12-1. Regulated I/O Block Diagram



12.1.1 Bias Generator

The bias generator generates gate bias to all pass transistors. This block mainly contains a charge pump an opamp along with a NMOS diode in a forward biased condition.

12.1.2 Charge Pump

The Charge Pump gets clock information from the internal main oscillator and pumps the charge to forward bias the NMOS diode. The NMOS diode is always in a slightly forward biased condition. There are control bits that control the current pumped into the charge pump for different frequencies of the charge pump clock input.

12.1.3 Comparator

The comparator expects an input of 1.2V reference and is used in negative feedback to generate a regulation voltage at the output. he NMOS replica structure is an NMOS diode configuration and is stacked over the output of the opamp. Because of the charge pump forward bias in the NMOS diode, the drain voltage is always the sum of the comparator output voltage and the transistor threshold voltage. This voltage is given as bias voltage to the gates of all pass transistors.

12.1.4 Replica Structure

The Replica Structure has a NMOS in series with the resistive divider structure. The resistive divider network provides feedback to the comparator in such a way that the source of the replica NMOS is always set to 3V.

12.1.5 Pass Transistors

The Pass Transistors are NMOS pass transistors that sit with the I/O driver. These transistors ensure that the I/O output voltage does not exceed the regulated voltage and support5 mA drive strength. In order to have acceptable control of the I/O output voltage, the NMOS pass transistors provide the same operating conditions as that of the replica structure. This is archived by connecting a resistance of one fourth of the replica resistor to the source of the NMOS pass transistor. This is a common resistor for all four I/O.

12.2 Application Overview

The I/O voltage regulator regulates the output of the I/O to 3.0, 2.5, 1.8 volts. The pass transistor of the regulator sits with the driver of the I/O. Bias for the pass transistor is routed along the supply ring.

12.3 Register Definitions

The following registers are associated with the Regulated I/O and are listed in address order. The register descriptions have an associated register table showing the bit structure for that register. The bits in the tables that are grayed out are reserved bits and are not detailed in the register descriptions that follow. Always write reserved bits with a value of '0'. For a complete table of Regulated I/O registers, refer to the Core Register Summary on page 24.

12.3.1 IO_CFG1 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,DCh	IO_CFG1	StrongP		Range[1:0]		P1_LOW_THRS	SPICLK_ON_P10	REG_EN	IOINT	RW : 00

The Input/Output Configuration Register 1 (IO_CFG1) configures the Port 1 output regulator and set the Interrupt mode for all GPIO.

Bit 7: StrongP. Setting this bit increases the drive strength and edge ratio for high outputs.

Bit 5 and 4: Range[1:0]. These bits select the regulator output level for Port 1. Available levels are 3.0V, 1.8V, and 2.5V.

Bit 3 P1_LOW_THRS. This bit reduces the threshold voltage of the P1 port input buffers so that there are no compatibility issues when Port 1 is communicating at regulated voltage levels.

'0' is standard threshold of VIH, VIL. '1' is reduce threshold of VIH, VIL.

Bit 2: SPICLK_ON_P10. When this bit is set to '1', the SPI clock is mapped to Port 1 pin 0. Otherwise, it is mapped to Port 1 pin 3.

Bit 1: REG_EN. The Register Enable bit (REG_EN) controls the regulator on Port 1 outputs.

Bit 0: IO INT. This bit sets the GPIO Interrupt mode for all pins in the CY7C643xx and CY7C604xx enCoRe V devices. GPIO interrupts are controlled at each pin by the PRTxIE registers, and also by the global GPIO bit in the INT_MSK0 register.

For additional information, refer to the IO_CFG1 register on page 224.

12.3.2 IO_CFG2 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,DEh	IO_CFG2			REG_LEVEL[2:0]				REG_CLOCK[1:0]		RW : 00

The Input/Output Configuration Register 2 (IO_CFG2) selects output regulated supply and clock rates.

Bits 5 to 3: REG_LEVEL[2:0]. These bits select output regulated supply.

REG_LEVEL[2:0]	Approx. Regulated Supply (V)		
000	3	2.5	1.8
001	3.1	2.6	1.9
010	3.2	2.7	2.0
011	3.3	2.8	2.1
100	3.4	2.9	2.2
101	3.5	3.0	2.3
110	3.6	3.1	2.4
111	3.7	3.2	2.5

Bits 1 to 0: REG_CLOCK[1:0]. The Regulated I/O charge pump can operate with a maximum clock speed of 12 MHZ. The REG_CLOCK[1:0] bits select clocking options for the regulator. Setting REG_CLOCK[1:0] to '10' should be used with 24 MHz SYSCLK and '01' should be used with 6/12 MHz SYSCLK.

REG_CLOCK[1:0]	SYSCLK Clock Rate
10	24 MHz
01	6/12 MHz

For additional information, refer to the IO_CFG2 register on page 227.

13. I/O Analog Multiplexer

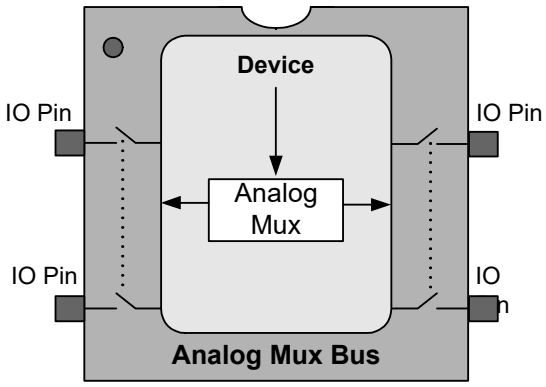
This chapter explains the device-wide I/O Analog Multiplexer for the CY7C643xx and CY7C604xx enCoRe V devices and their associated registers. For a quick reference of all enCoRe V registers in address order, refer to the Register Reference chapter on page 163.

13.1 Architectural Description

The CY7C643xx and CY7C604xx enCoRe V devices contain an enhanced analog multiplexer (mux) capability. This function allows many I/O pins to connect to a common internal analog global bus.

You can connect any number of pins simultaneously, and dedicated support circuitry allows selected pins to be alternately charged high or connected to the bus. The analog global bus can be connected as a comparator input. Figure 13-1 shows a block diagram of the I/O analog mux system.

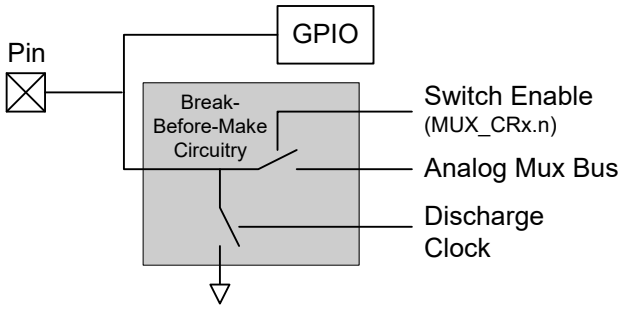
Figure 13-1. I/O Analog Mux System



For each pin, the mux capability exists in parallel with the normal GPIO cell, shown in Figure 13-2. Normally, the associated GPIO pin is put into a high-impedance state for these applications, although there are cases where the GPIO cell is configured by the user to briefly drive pin initialization states as described ahead.

Pins are individually connected to the internal bus by setting the corresponding bits in the MUX_CRx registers. Any number of pins can be enabled at the same time. At reset, all of these mux connections are open (disconnected).

Figure 13-2. I/O Pin Configuration



13.2 Register Definitions

The following registers are only associated with the Analog Bus Mux in the CY7C643xx and CY7C604xx enCoRe V devices and are listed in address order. Each register description has an associated register table showing the bit structure for that register. Register bits that are grayed out throughout this document are reserved bits and are not detailed in the register descriptions that follow. Always write reserved bits with a value of '0'.

13.2.1 MUX_CRx Registers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,D8h	MUX_CR0	ENABLE[7:0]								RW : 00
1,D9h	MUX_CR1	ENABLE[7:0]								RW : 00
1,DAh	MUX_CR2	ENABLE[7:0]								RW : 00
1,DBh	MUX_CR3	ENABLE[7:0]								RW : 00
1,DFh	MUX_CR4					ENABLE[3:0]				RW : 0

The Analog Mux Port Bit Enable Registers (MUX_CR0, MUX_CR1, MUX_CR2, MUX_CR3, and MUX_CR4) control the connection between the analog mux bus and the corresponding pin.

Bits 7 to 0: ENABLE[7:0]. The bits in these registers enable connection of individual pins to the analog mux bus. Each I/O port has a corresponding MUX_CRx register.

Setting a bit high connects the corresponding pin to the analog bus.

For additional information, refer to the MUX_CRx register on page 223.

Section C: System Resources

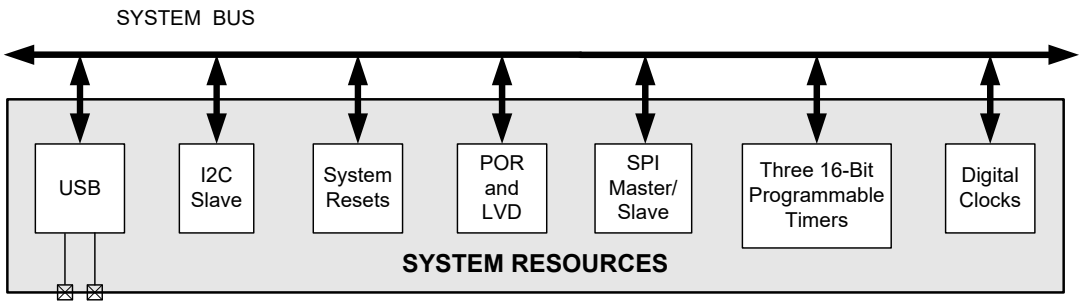
This section discusses the system resources that are available for the enCoRe V devices and the registers associated with those resources. This section includes the following chapters:

- Digital Clocks on page 96.
- I²C Slave on page 103.
- System Resets on page 114.
- POR and LVD on page 121.
- SPI on page 123.
- Programmable Timer on page 137.
- Full-Speed USB on page 141.

Top-Level System Resources Architecture

The following figure displays the top-level architecture of the enCoRe V system resources. Each component of the figure is discussed at length in the chapters that follow.

enCoRe V System Resources



System Resources Register Summary

The following table lists all the enCoRe V registers for the system resources, in address order, within their system resource configuration. The bits that are grayed out are reserved bits. If you write these bits, always write them with a value of '0'.

Summary Table of the System Resource Registers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
DIGITAL CLOCK REGISTERS (page 99)										
1,BDh	USB_MISC_CR						USB_SE_EN	USB_ON	USB_CLK_ON	RW : 0
1,DDh	OUT_P1	P16D	P16EN						P10EN	RW : 00
1,E0h	OSC_CR0	X32ON	Disable Buzz	No Buzz	Sleep[1:0]		CPU Speed[2:0]			RW : 01
1,E2h	OSC_CR2				CLK48MEN		EXT-CLKEN	RSVD		RW : 0
I2C SLAVE REGISTERS (page 106)										
0,C8h	I2C_XCFG								HW Addr En	RW : 0
0,CAh	I2C_ADDR		Slave Address[6:0]							RW : 00
0,D6h	I2C_CFG		PSelect		Stop IE	Clock Rate[1:0]			Enable	RW : 00
0,D7h	I2C_SCR	Bus Error		Stop Status	ACK	Address	Transmit	LRB	Byte Complete	# : 00
0,D8h	I2C_DR	Data[7:0]								RW : 00
SYSTEM RESET REGISTERS (page 116)										
x,FEh	CPU_SCR1	IRESS			SLIMO[1:0]				IRAMDIS	# : 0
x,FFh	CPU_SCR0	GIES		WDRS	PORS	Sleep			STOP	# : XX
POR REGISTERS (page 121)										
1,E3h	VLT_CR	HPOR		PORLEV[1:0]		LVDTBEN	VM[2:0]			RW : 00
1,E4h	VLT_CMP				NoWrite		POR_EXT	LVD		R : #
SPI REGISTERS (page 125)										
0,29h	SPI_TXR	Data[7:0]								W : 00
0,2Ah	SPI_RXR	Data[7:0]								R : 00
0,2Bh	SPI_CR	LSb First	Overrun	SPI Complete	TX Reg Empty	RX Reg Full	Clock Phase	Clock Polarity	Enable	# : 00
1,29h	SPI_CFG	Clock Sel [2:0]			Bypass	SS_	SS_EN_	Int Sel	Slave	RW : 00
PROGRAMMABLE TIMER REGISTERS (page 139)										
0,B0h	PT0_CFG						CLKSEL	One Shot	START	RW : 0
0,B1h	PT0_DATA1	DATA[7:0]								RW : 00
0,B2h	PT0_DATA0	DATA[7:0]								RW : 00
0,B3h	PT1_CFG						CLKSEL	One Shot	START	RW : 0
0,B4h	PT1_DATA1	DATA[7:0]								RW : 00
0,B5h	PT1_DATA0	DATA[7:0]								RW : 00
0,B6h	PT2_CFG						CLKSEL	One Shot	START	RW : 0
0,B7h	PT2_DATA1	DATA[7:0]								RW : 00
0,B8h	PT2_DATA0	DATA[7:0]								RW : 00
USB REGISTERS (page 147)										
0,58h	PMAx_DR	Data Byte[7:0]								RW : 00
0,59h	PMAx_DR	Data Byte[7:0]								RW : 00
0,5Ah	PMAx_DR	Data Byte[7:0]								RW : 00
0,5Bh	PMAx_DR	Data Byte[7:0]								RW : 00
0,5Ch	PMAx_DR	Data Byte[7:0]								RW : 00
0,5Dh	PMAx_DR	Data Byte[7:0]								RW : 00
0,5Eh	PMAx_DR	Data Byte[7:0]								RW : 00
0,5Fh	PMAx_DR	Data Byte[7:0]								RW : 00

Summary Table of the System Resource Registers *(continued)*

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,64h	PMAx_DR	Data Byte[7:0]								RW : 00
0,65h	PMAx_DR	Data Byte[7:0]								RW : 00
0,66h	PMAx_DR	Data Byte[7:0]								RW : 00
0,67h	PMAx_DR	Data Byte[7:0]								RW : 00
0,68h	PMAx_DR	Data Byte[7:0]								RW : 00
0,69h	PMAx_DR	Data Byte[7:0]								RW : 00
0,6Ah	PMAx_DR	Data Byte[7:0]								RW : 00
0,6Bh	PMAx_DR	Data Byte[7:0]								RW : 00
0,31h	USB_SOF0	Frame Number[7:0]								R : 00
0,32h	USB_SOF1						Frame Number[10:8]			R : 0
0,33h	USB_CR0	USB Enable	Device Address[6:0]							RW : 00
0,34h	USBIO_CR0	TEN	TSE0	TD					RD	# : 0
0,35h	USBIO_CR1	IOMode	Drive Mode	DPI	DMI	PS2PUEN	USB-PUEN	DPO	DMO	RW : 00
0,41h	EPx_CNT1	Data Count[7:0]								RW : 00
0,43h	EPx_CNT1	Data Count[7:0]								RW : 00
0,45h	EPx_CNT1	Data Count[7:0]								RW : 00
0,47h	EPx_CNT1	Data Count[7:0]								RW : 00
0,49h	EPx_CNT1	Data Count[7:0]								RW : 00
0,4Bh	EPx_CNT1	Data Count[7:0]								RW : 00
0,4Dh	EPx_CNT1	Data Count[7:0]								RW : 00
0,4Fh	EPx_CNT1	Data Count[7:0]								RW : 00
0,31h	USB_SOF0	Frame Number[7:0]								R : 00
0,32h	USB_SOF1						Frame Number[10:8]			R : 0
0,33h	USB_CR0	USB Enable	Device Address[6:0]							RW : 00
0,34h	USBIO_CR0	TEN	TSE0	TD					RD	# : 00
0,35h	USBIO_CR1	IOMode	Drive Mode	DPI	DMI	PSPUEN	USB-PUEN	DPO	DMO	RW : 00
0,36h	EP0_CR	Setup Received	IN Received	OUT Received	ACK'd Transaction	Mode[3:0]				RW : 00
0,37h	EP0_CNT	Data Toggle	Data Valid			Byte Count[3:0]				# : 00
0,38h	EP0_DRx	Data Byte[7:0]								RW : 00
0,39h	EP0_DRx	Data Byte[7:0]								RW : 00
0,3Ah	EP0_DRx	Data Byte[7:0]								RW : 00
0,3Bh	EP0_DRx	Data Byte[7:0]								RW : 00
0,3Ch	EP0_DRx	Data Byte[7:0]								RW : 00
0,3Dh	EP0_DRx	Data Byte[7:0]								RW : 00
0,3Eh	EP0_DRx	Data Byte[7:0]								RW : 00
0,3Fh	EP0_DRx	Data Byte[7:0]								RW : 00
0,40h	EPx_CNT0	Data Toggle	Data Valid						Count MSB	# : 0
0,42h	EPx_CNT0	Data Toggle	Data Valid						Count MSB	# : 0
0,44h	EPx_CNT0	Data Toggle	Data Valid						Count MSB	# : 0
0,46h	EPx_CNT0	Data Toggle	Data Valid						Count MSB	# : 0
0,48h	EPx_CNT0	Data Toggle	Data Valid						Count MSB	# : 0
0,4Ah	EPx_CNT0	Data Toggle	Data Valid						Count MSB	# : 0
0,4Ch	EPx_CNT0	Data Toggle	Data Valid						Count MSB	# : 0

Summary Table of the System Resource Registers *(continued)*

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,4Eh	EPx_CNT0	Data Toggle	Data Valid						Count MSB	# : 0
1,30h	USB_CR1						BusActiv-ity	Ena-bleLock	RegEnable	RW : 0
1,34h	PMAx_WA	Write Address[7:0]								RW : 00
1,35h	PMAx_WA	Write Address[7:0]								RW : 00
1,36h	PMAx_WA	Write Address[7:0]								RW : 00
1,37h	PMAx_WA	Write Address[7:0]								RW : 00
1,38h	PMAx_WA	Write Address[7:0]								RW : 00
1,39h	PMAx_WA	Write Address[7:0]								RW : 00
1,3Ah	PMAx_WA	Write Address[7:0]								RW : 00
1,3Bh	PMAx_WA	Write Address[7:0]								RW : 00
1,44h	PMAx_WA	Write Address[7:0]								RW : 00
1,45h	PMAx_WA	Write Address[7:0]								RW : 00
1,46h	PMAx_WA	Write Address[7:0]								RW : 00
1,47h	PMAx_WA	Write Address[7:0]								RW : 00
1,48h	PMAx_WA	Write Address[7:0]								RW : 00
1,49h	PMAx_WA	Write Address[7:0]								RW : 00
1,4Ah	PMAx_WA	Write Address[7:0]								RW : 00
1,4Bh	PMAx_WA	Write Address[7:0]								RW : 00
1,3Ch	PMAx_RA	Read Address[7:0]								RW : 00
1,3Dh	PMAx_RA	Read Address[7:0]								RW : 00
1,3Eh	PMAx_RA	Read Address[7:0]								RW : 00
1,3Fh	PMAx_RA	Read Address[7:0]								RW : 00
1,40h	PMAx_RA	Read Address[7:0]								RW : 00
1,41h	PMAx_RA	Read Address[7:0]								RW : 00
1,42h	PMAx_RA	Read Address[7:0]								RW : 00
1,43h	PMAx_RA	Read Address[7:0]								RW : 00
1,4Ch	PMAx_RA	Read Address[7:0]								RW : 00
1,4Dh	PMAx_RA	Read Address[7:0]								RW : 00
1,4Eh	PMAx_RA	Read Address[7:0]								RW : 00
1,4Fh	PMAx_RA	Read Address[7:0]								RW : 00
1,50h	PMAx_RA	Read Address[7:0]								RW : 00
1,51h	PMAx_RA	Read Address[7:0]								RW : 00
1,52h	PMAx_RA	Read Address[7:0]								RW : 00
1,53h	PMAx_RA	Read Address[7:0]								RW : 00
1,54h	EPx_CR0	Stall		NAK_INT_EN	ACKed Tx	Mode[3:0]				# : 00
1,55h	EPx_CR0	Stall		NAK_INT_EN	ACKed Tx	Mode[3:0]				# : 00
1,56h	EPx_CR0	Stall		NAK_INT_EN	ACKed Tx	Mode[3:0]				# : 00
1,57h	EPx_CR0	Stall		NAK_INT_EN	ACKed Tx	Mode[3:0]				# : 00
1,58h	EPx_CR0	Stall		NAK_INT_EN	ACKed Tx	Mode[3:0]				# : 00
1,59h	EPx_CR0	Stall		NAK_INT_EN	ACKed Tx	Mode[3:0]				# : 00
1,5Ah	EPx_CR0	Stall		NAK_INT_EN	ACKed Tx	Mode[3:0]				# : 00
1,5Bh	EPx_CR0	Stall		NAK_INT_EN	ACKed Tx	Mode[3:0]				# : 00

Legend
Access is bit specific. Refer to the Register Reference chapter on page 163 for additional information.
R Read register or bit(s).
W Write register or bit(s).

14. Digital Clocks

This chapter discusses the Digital Clocks and their associated registers. It serves as an overview of the clocking options available in the enCoRe V devices. For detailed information on specific oscillators, see the individual oscillator chapters in the section called enCoRe V Core on page 23. For a complete table of the digital clock registers, refer to the System Resources Register Summary on page 93. For a quick reference of all enCoRe V registers in address order, refer to the Register Reference chapter on page 163.

14.1 Architectural Description

The enCoRe V M8C core has a large number of clock sources that increase the flexibility of the enCoRe V device, as listed in Table 14-1 and illustrated in Figure 14-1.

Table 14-1. System Clocking Signals and Definitions

Signal	Definition
SYSCLK	Either the direct output of the Internal Main Oscillator or the direct input of the EXTCLK pin while in external clocking mode.
CPUCLK	SYSCLK is divided down to one of eight possible frequencies to create CPUCLK, which determines the speed of the M8C. See the OSC_CR0 Register on page 101.
CLK32K	The Internal Low-speed Oscillators output. See the OSC_CR0 Register on page 101.
CLKIM0	The internally generated clock from the IMO. By default, this clock drives SYSCLK; however, an external clock may be used by enabling EXTCLK mode. The IMO can be set to various frequencies; the default is 12 MHz.
SLEEP	One of four sleep intervals may be selected from 1.95 ms to 1 second. See the OSC_CR0 Register on page 101.

14.1.1 Internal Main Oscillator

The Internal Main Oscillator (IMO) is the foundation upon which almost all other clock sources in the enCoRe V device are based. The default mode of the IMO creates a 12 MHz reference clock that is used by many other circuits in the device. The enCoRe V device has an option to replace the IMO with an externally supplied clock that becomes the base for all of the clocks the IMO normally serves. The internal base clock net is called SYSCLK and is driven by either the IMO or an external clock (EXTCLK).

Whether the external clock or the internal main oscillator is selected, all device functions are clocked from a derivative of SYSCLK or are resynchronized to SYSCLK. All external asynchronous signals and the internal low-speed oscillator are resynchronized to SYSCLK for use in the digital blocks.

The IMO frequency can be adjusted to other frequencies besides 12 MHz. See the Architectural Description on page 67 for more information.

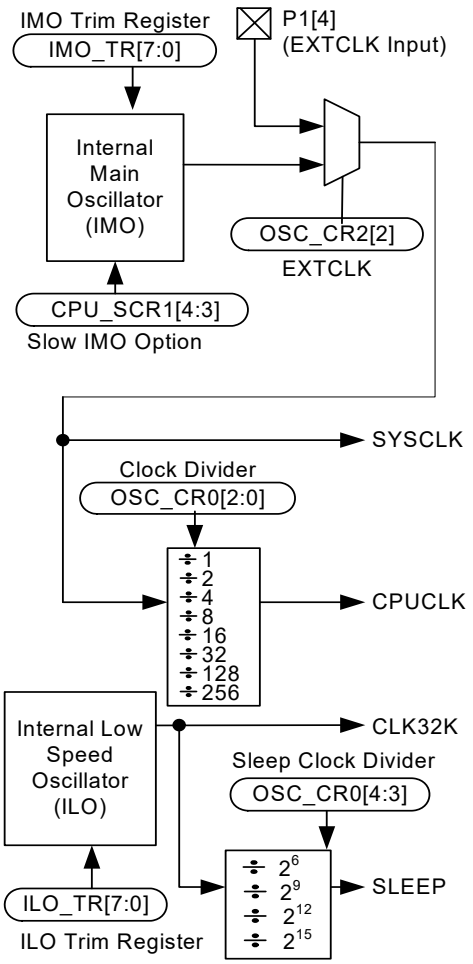
The IMO is discussed in detail in the Internal Main Oscillator (IMO) chapter on page 67.

14.1.2 Internal Low-speed Oscillator

The Internal Low-speed Oscillator (ILO) is available as a general clock, but is also the clock source for the sleep and watchdog timers. The ILO can be disabled in deep sleep mode, or in other sleep modes when the ECO is enabled.

The ILO is discussed in detail in the Internal Low-speed Oscillator (ILO) chapter on page 72.

Figure 14-1. Overview of enCoRe V Clock Sources



14.1.3 External Clock

In addition to the IMO clock source, an externally supplied clock may be selected as the device master clock (see Figure 14-1).

Pin P1[4] is the input pin for the external clock. If P1[4] is selected as the external clock source, the drive mode of the pin must be set to High-Z (not High-Z Analog).

An external clock with a frequency between 1 MHz and 24 MHz can be supplied. The reset state of the EXTCLKEN bit is ‘0’. With this setting, the device always boots up under the control of the IMO. The system cannot be started from a reset with the external clock.

When the EXTCLKEN bit is set, the external clock becomes the source for the internal clock tree, SYSCLK, which drives most enCoRe V device clocking functions. All external and internal signals, including the ILO or ECO low frequency clock, are synchronized to this clock source. Note that there is no glitch protection in the device for an external clock. User should ensure that the external clock is glitch free. See device datasheet for the clock specifications.

In applications where XRES is used when in external clock mode, care must be taken to switch the clock source to IMO before entering the low-power modes. The clock source can be switched back to external clock upon completion of wake up either in the interrupt routine or in the main code. Failure to do this will cause the device to hang up.

An example implementation is shown here:

```
OSC_CR2 &= ~0x04; /* Disconnect External
Clock and connect IMO to SYSCLK*/
M8C_Sleep; /* Entering sleep */
asm("nop");
OSC_CR2 |= 0x04; /*Connect External Clock to
SYSCLK */
```

14.1.3.1 Switch Operation

Switching between the IMO and the external clock is done in firmware at any time and is transparent to the user.

Switch timing depends upon whether the CPU clock divider is set for divide by 1, or divide by 2 or greater. If the CPU clock divider is set for divide by 2 or greater, as shown in Figure 14-2, the setting of the EXTCLKEN bit occurs shortly after the rising edge of SYSCLK. The SYSCLK output is then disabled after the next falling edge of SYSCLK, but before the next rising edge. This ensures a glitch free transition and provides a full cycle of setup time from SYSCLK to output disable. After the current clock selection is disabled, the enable of the newly selected clock is double synchronized to that clock. After synchronization, on the subsequent negative edge, SYSCLK is enabled to output the newly selected clock.

In the 12 MHz case, as shown in Figure 14-3, the assertion of IOW_ and thus the setting of the EXTCLKEN bit occurs on the falling edge of SYSCLK. Because SYSCLK is already low, the output is immediately disabled. Therefore, the setup time from SYSCLK to disable is one-half SYSCLK.

Figure 14-2. Switch from IMO to the External Clock with a CPU Clock Divider of Two or Greater

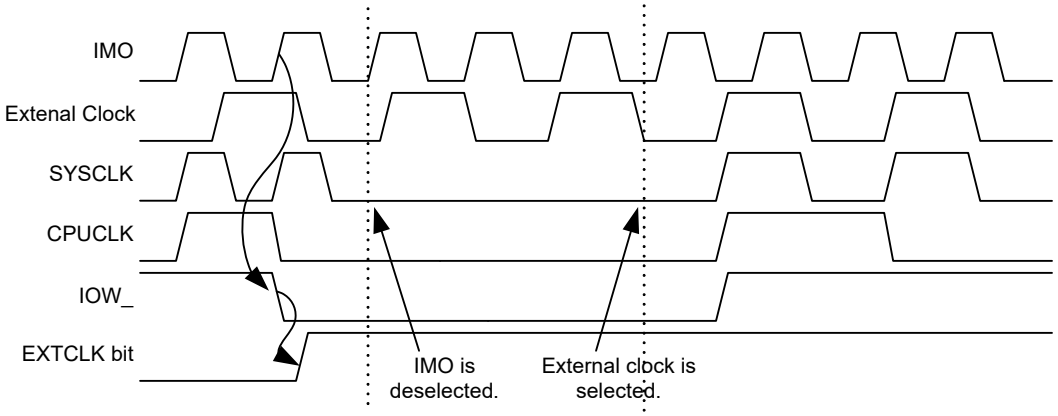
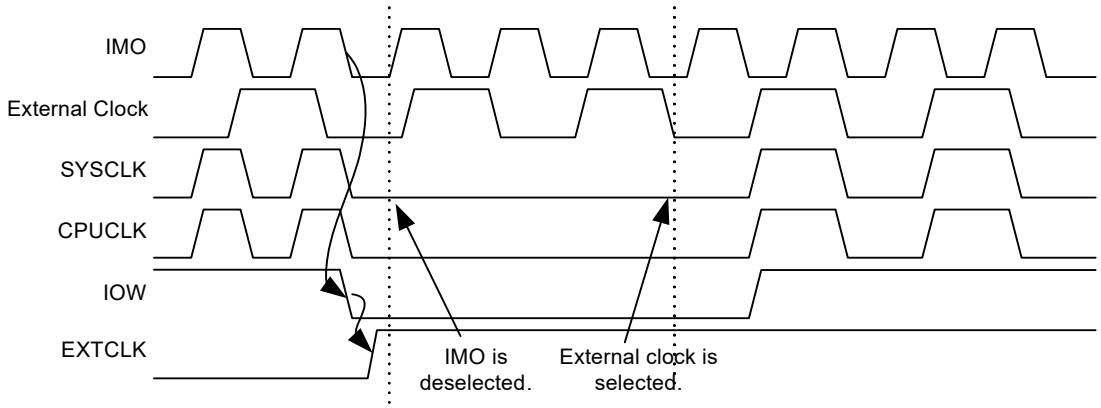


Figure 14-3. Switch from IMO to External Clock with the CPU Running with a CPU Clock Divider of One



14.2 Register Definitions

The following registers are associated with the Digital Clocks and are listed in address order. Each register description has an associated register table showing the bit structure for that register. The bits in the tables that are grayed out throughout this manual are reserved bits and are not detailed in the register descriptions that follow. Always write reserved bits with a value of ‘0’. For a complete table of digital clock registers, refer to the “System Resources Register Summary” on page 93.

14.2.1 USB_MISC_CR Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,BDh	USB_MISC_CR						USB_SE_EN	USB_ON	USB_CLK_ON	RW : 0

The USB Miscellaneous Control Register controls the clocks to the USB block, to make the IMO work with better accuracy for the USB part and to disable the single-ended input of the USBIO in the case of a non-USB part.

Bit 2: USB_SE_EN. The single-ended outputs of USBIO is enabled or disabled based upon this bit setting. Set this bit to '1' when using this part as a USB part for USB transactions to occur. Set this bit to '0' to disable single-ended outputs of USBIO. The DPO and DMO are held at logic high state and RSE0 is held at a low state.

Note Bit [1:0] of the USBIO_CR1 register is also affected depending on this register setting. When this bit is '0' (default), regardless of the DP and DM state, the DPO and DMO bits of USBIO_CR1 are '11b'.

Bit 1: USB_ON. This bit is used by the IMO DAC block to either work with better DNL consuming higher power, or with sacrificed DNL consuming lower power. Set this bit to '1' when the part is used as a USB part. A '0' runs the IMO with sacrificed DNL by consuming less power. A '1' runs the IMO with better DNL by consuming more power.

Bit 0: USB_CLK_ON. This bit either enables or disables the clocks to the USB block. It is used to save power in cases when the device need not respond to USB traffic. Set this bit to '1' when the device is used as a USB part.

When this bit is a ‘0’, all clocks to the USB block are driven. The device does not respond to USB traffic and none of the USB registers, except IMO_TR, IMO_TR1 and USBIO_CR1, listed in the Register Definitions on page 147 are writable.

When this bit is a ‘1’, clocks are not blocked to the USB block. The device responds to USB traffic depending on the other register settings mentioned under Register Definitions in the Full-Speed USB chapter on page 141.

For additional information, refer to the USB_MISC_CR on page 220.

14.2.2 OUT_P1 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,DDh	OUT_P1	P16D	P16EN	RSVD	RSVD	RSVD	P12EN	RSVD	P10EN	RW : 00

The Output Override to Port 1 Register (OUT_P1) enables specific internal signals to output to Port 1 pins. If any other function, such as I2C, is enabled for output on these pins, that function has higher priority than the OUT_P1 signals.Reserved bits must always be written with a value of ‘0’.

Bit 7: P16D. Bit selects the data output to P1[6] when P16EN is high.

0 - Select Timer output (TIMEROUT)

1 - Select CLK32

Bit 6: P16EN. This bit enables pin P1[6] for signal output selected by the P16D bit.

0 - No internal signal output to P1[6]

1 - Output the signal selected by P16D to P1[6]

Bit 2: P12EN. This bit enables P1[2] to output the main system clock (SYSCLK).

Bit 0: P10EN Bit enables pin P1[0] to output the sleep interrupt (SLPINT).

0 - No internal signal output to P1[0]

1 - Output SLPINT to P1[0]

For additional information, refer to the OUT_P1 on page 225.

14.2.3 OSC_CR0 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,E0h	OSC_CR0	X32ON	Disable Buzz	No Buzz	Sleep[1:0]		CPU Speed[2:0]			RW : 01

The Oscillator Control Register 0 (OSC_CR0) configures various features of internal clock sources and clock nets.

Bit 7: X32ON. This bit enables the 32-kHz external crystal oscillator (ECO) when set high. See the Application Overview on page 75 for the proper sequence to enable the ECO.

Bit 6: Disable Buzz. Setting this bit high causes the bandgap and POR/LVD systems to remain powered off continuously during sleep. In this case, there is no periodic “buzz” (brief wakeup) of these functions during sleep. This bit has no effect when the No Buzz bit is set high.

Bit 5: No Buzz. Normally, when the SLEEP bit is set in the CPU_SCR register, all enCoRe V device systems are powered down, including the bandgap reference. However, to facilitate the detection of POR and LVD events at a rate higher than the sleep interval, the bandgap circuit is powered up periodically (for about 60 μs) at the Sleep System Duty Cycle, which is independent of the sleep interval and typically higher. When the No Buzz bit is set, the Sleep System Duty Cycle value is overwritten and the bandgap circuit is forced to be on during sleep. This results in faster response to an LVD or POR event (continuous detection as opposed to periodic), at the expense of higher average sleep current.

Bits 4 and 3: Sleep[1:0]. The available sleep interval selections are shown in the following table. Sleep intervals are approximate based upon the accuracy of the internal low-speed oscillator.

Sleep Interval OSC_CR[4:3]	Sleep Timer Clocks	Sleep Period (32-kHz ILO)	Sleep Period (1-kHz ILO)	Watchdog Period (Nominal)
00b (Default)	64	1.95 ms	64 ms	6 ms
01b	512	15.6 ms	512 ms	47 ms
10b	4096	125 ms	4 s	375 ms
11b	32,768	1 s	32 s	3 s

Bits 2 to 0: CPU Speed[2:0]. The enCoRe V M8C operates over a range of CPU clock speeds, allowing you to tailor the M8C’s performance and power requirements to the application.

The reset value for the CPU speed bits is 010b. Therefore, the default CPU speed is one-half of the clock source. The internal main oscillator is the default clock source for the CPU speed circuit; therefore, the default CPU speed is 6.0 MHz. See External Clock on page 97 for more information on the supported frequencies for externally supplied clocks.

The CPU frequency is changed with a write to the OSC_CR0 register. There are eight frequencies generated from a power-of-two divide circuit that is selected by a 3-bit code. At any given time, the CPU 8-to-1 clock mux is selecting one of the available frequencies, which is resynchronized to the 24-MHz master clock at the output. The IMO frequency is also selectable, as discussed in Architectural Description on page 67. This offers an option to lower both system and CPU clock speed to save power. The selections are shown in the following table (reset state is 010b).

Bits	6 MHz Internal Main Oscillator	12 MHz Internal Main Oscillator	24 MHz Internal Main Oscillator	External Clock
000b	750 kHz	1.5 MHz	3 MHz	EXTCLK/ 8
001b	1.5 MHz	3.0 MHz	6 MHz	EXTCLK/ 4
010b	3 MHz	6.0 MHz	12 MHz	EXTCLK/ 2
011b	6 MHz	12.0 MHz	24 MHz	EXTCLK/ 1
100b	375 kHz	750 Hz	1.5 MHz	EXTCLK/ 16
101b	187.5 kHz	375 kHz	750 kHz	EXTCLK/ 32
110b	46.8 kHz	93.7 kHz	187.5 kHz	EXTCLK/ 128
111b	23.4 kHz	46.8 kHz	93.7 kHz	EXTCLK/ 256

An automatic protection mechanism is available for systems that need to run at peak CPU clock speed but cannot guarantee a high enough supply voltage for that clock speed. See the LVDTBEN bit in the VLT_CMP Register on page 122 for more information.

Note During USB operation, the CPU speed can be set to any setting. Be aware that USB throughput decreases with a decrease in CPU speed. For maximum throughput, the CPU clock should be made equal to the system clock. The system clock must be 24 MHz for USB operation.

For additional information, refer to the OSC_CR0 on page 228.

14.2.4 OSC_CR2 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,E2h	OSC_CR2				CLK48MEN		EXTCLKEN	RSVD		RW : 00

The Oscillator Control Register 2 (OSC_CR2) configures various features of internal clock sources and clock nets.

Bit 4: CLK48MEN

This is the 48-MHz clock enable bit.

'0' disables the bit and '1' enables the bit. This register setting applies only when the device is **not** in OCD mode. When in OCD mode, the 48-MHz clock is always active.

Bit 2: EXTCLKEN.

When the EXTCLKEN bit is set, the external clock becomes the source for the internal clock tree, SYSCLK, which drives most device clocking functions. All external and internal signals, including the low-speed oscillator, are synchronized to this clock source. The external clock input operates from the clock supplied at P1[4]. When using this input, the pin drive mode must be set to High-Z (not High-Z Analog), such as drive mode 11b with the PRT1DR bit 4 set high.

Bit 1: RSVD

This is a reserved bit. It should always be 0.

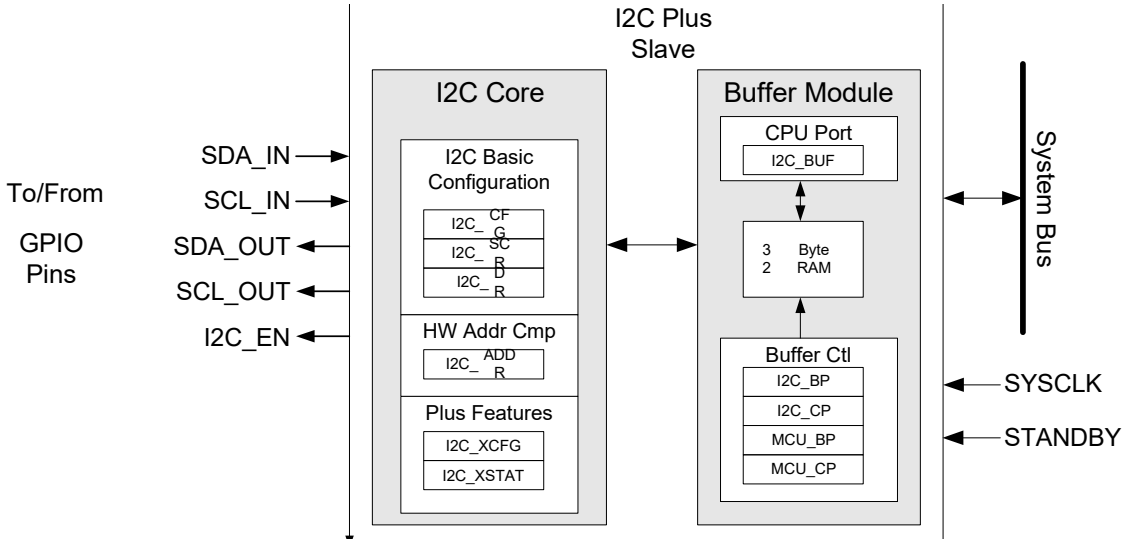
For additional information, refer to the OSC_CR2 on page 230.

15. I²C Slave

This chapter explains the I²C Slave block and its associated registers. The I²C communications block is a serial processor designed to implement a complete I²C slave. For a complete table of the I²C registers, refer to the Summary Table of the System Resource Registers on page 93. For a quick reference of all enCoRe V registers in address order, refer to the Register Reference chapter on page 163.

15.1 Architectural Description

Figure 15-1. The I²C slave enhanced communications block is a serial-to-parallel processor, designed to interface the enCoRe V device to a two-wire I²C serial communications bus. To eliminate the need for excessive M8C microcontroller intervention and overhead, the block provides I²C-specific support for status detection and generation of framing bits. I²C Block Diagram



Basic I²C features include:

- Slave, transmitter, and receiver operation
- Byte processing for low CPU overhead
- Interrupt or polling CPU interface
- Support for clock rates of up to 400 kHz
- 7- or 10-bit addressing (through firmware support)
- SMBus operation (through firmware support)

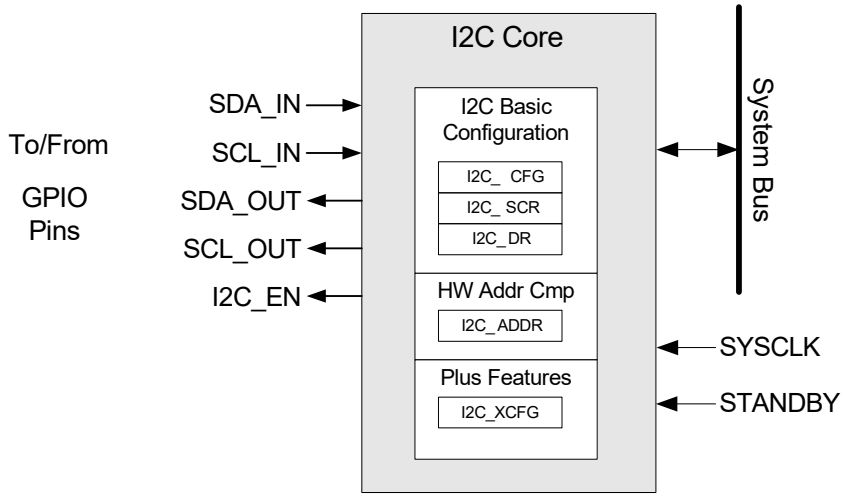
Enhanced features of the I²C Slave Enhanced module include:

- Support for 7-bit hardware address compare
- Flexible data buffering schemes

A “no bus stalling” operating mode

This block has a low-power bus monitoring mode.

Figure 15-2. I²C Slave Block Diagram



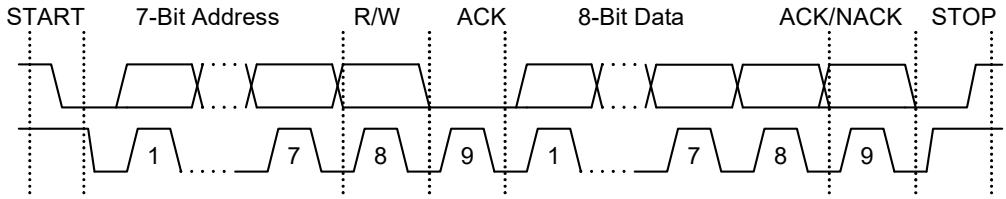
enCoRe V

15.1.1 Basic I²C Data Transfer

Figure 15-3 shows the basic form of data transfers on the I²C bus with a 7-bit address format. For a detailed description, see the Philips Semiconductors (now NXP Semiconductors) I²C-Bus Specification, version 2.1.

A Start condition (generated by the master) is followed by a data byte, consisting of a 7-bit slave address (there is also a 10-bit address mode) and a read/write (RW) bit. The RW bit sets the direction of data transfer. The addressed slave is required to acknowledge (ACK) the bus by pulling the data line low during the ninth bit time. If the ACK is received, the transfer proceeds and the master transmits or receives an indeterminate number of bytes, depending upon the RW direction. If, for any reason, the slave does not respond with an ACK, a Stop condition is generated by the master to terminate the transfer or a Restart condition is generated for a retry attempt.

Figure 15-3. Basic I²C Data Transfer with 7-Bit Address Format



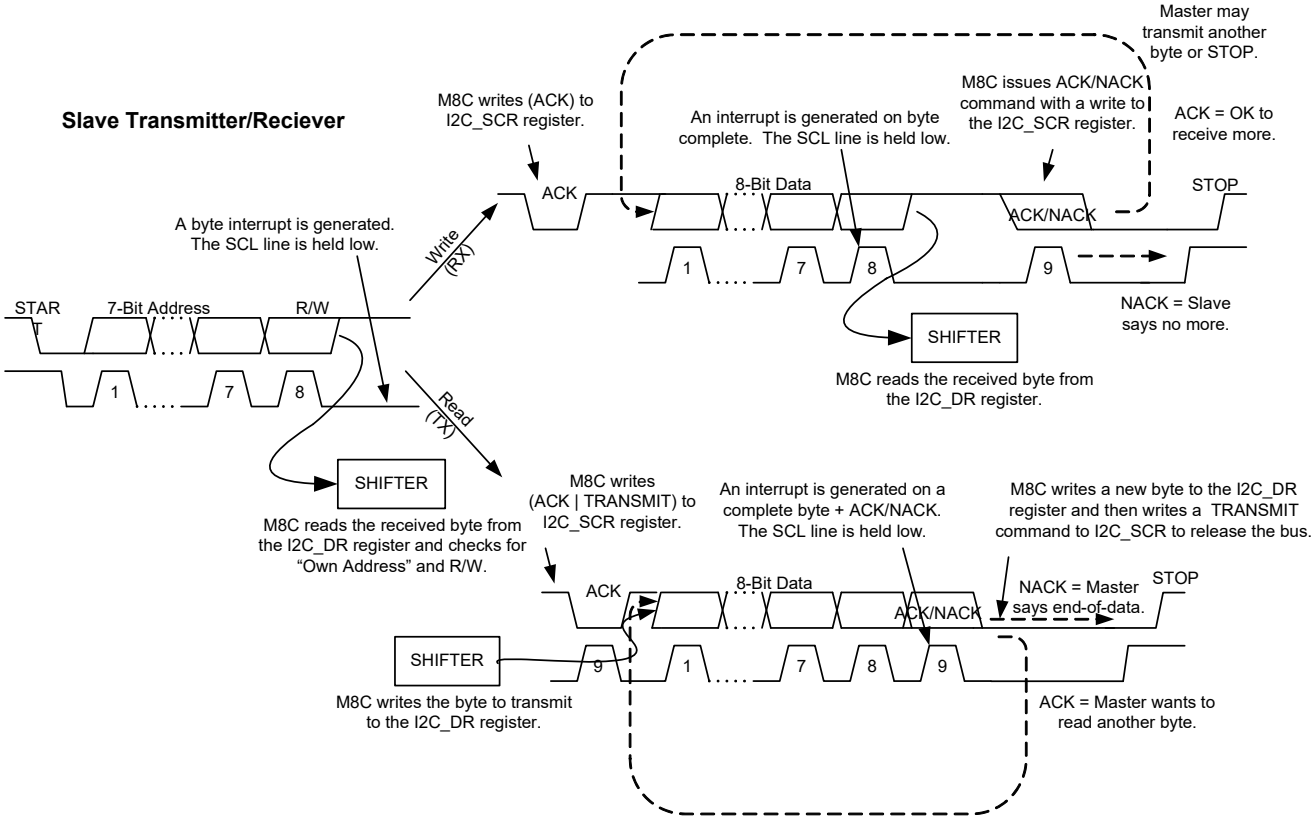
15.2 Application Overview

15.2.1 Slave Operation

When Slave mode is enabled, it is continually listening on the bus for a Start condition. When detected, the transmitted address/RW byte is received and read from the I²C block by firmware. At the point where eight bits of the address/RW byte are received, a byte complete interrupt is generated. On the following low of the clock, the bus is stalled by holding the SCL line low until the enCoRe V device has had a chance to read the address byte and compare it to its own address. It Issues an ACK or NACK command based upon that comparison.

If there is an address match, the RW bit determines how the enCoRe V device sequences the data transfer in Slave mode, as shown in the two branches of Figure 15-4. I²C handshaking methodology (slave holds the SCL line low to "stall" the bus) is used, as necessary, to give the enCoRe V device time to respond to the events and conditions on the bus. Figure 15-4 is a graphical representation of a typical data transfer from the slave perspective.

Figure 15-4. Slave Operation



15.3 Register Definitions

The registers shown here are associated with I2C slave and are listed in address order. Each register description has an associated register table showing the bit structure for that register. The grayed out bits in the tables are reserved bits and are not detailed in the register descriptions that follow. Always write reserved bits with a value of '0'. For a complete table of I2C registers, refer to the "Summary Table of the System Resource Registers" on page 93.

15.3.1 I2C_XCFG Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,C8h	I2C_XCFG								HW Addr EN	RW : 0

The I2C Extended Control Register (I2C_XCFG) is used to enable hardware I2C address detection block. The Enable bit (bit 0) of the I2C_CFG (0,D6h) register should be set to 1 for the I2C enhanced features to work.

Bit 0: HW Addr En. When this bit is set to a '1', hardware address compare is enabled. Upon a compare, the address is automatically ACKed, and upon a mismatch, the address is automatically NAKed and the hardware reverts to an idle state waiting for the next Start detection. You must configure the compare address in the I2C_ADDR register. When this bit is a '0', bit 3 of the I2C_SCR register is set and the bus stalls, and the received address is available in the I2C_DR register to enable the CPU to do a firmware address compare. The functionality of this bit is independent of the data buffering mode.

For additional information, refer to the I2C_XCFG register on page 186.

15.3.2 I2C_ADDR Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,CAh	I2C_ADDR		Slave Address[6:0]							RW : 00

The I2C Slave Address Register (I2C_ADDR) holds the slave's 7-bit address. All bits are RW.

Note When hardware address compare mode is not enabled in the I2C_XCFG register, this register is not in use.

Bits 6 to 0: Slave Address[6:0]. These 7 bits hold the slave's own device address.

For additional information, refer to the I2C_ADDR register on page 187.

15.3.3 I2C_CFG Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,D6h	I2C_CFG		PSelect		Stop IE	Clock Rate[1:0]			Enable	RW : 00

The I²C Configuration Register (I2C_CFG) is used to set the basic operating modes, baud rate, and selection of interrupts. The bits in this register control baud rate selection and optional interrupts. The values are typically set once for a given configuration. The bits in this register are all RW.

Table 15-1. I2C_CFG Configuration Register

Bit	Access	Description	Mode
6	RW	I2C Pin Select 0 = P1[7], P1[5] 1 = P1[1], P1[0]	Slave
4	RW	Stop IE Stop interrupt enable. 0 = Disabled. 1 = Enabled. An interrupt is generated upon the detection of a Stop condition.	Slave
3:2	RW	Clock Rate 00 = 100K Standard Mode 01 = 400K Fast Mode 10 = 50K Standard Mode 11 = Reserved	Slave

Bit 6: PSelect. Pin Select. With the default value of zero, the I²C pins are P1[7] for clock and P1[5] for data. When this bit is set, the pins for I²C switch to P1[1] for clock and P1[0] for data. You cannot change this bit while the Enable bit is set. However, the PSelect bit may be set at the same time as the enable bits. The two sets of pins used on I²C are not equivalent. The default set, P1[7] and P1[5], are the preferred set. The alternate set, P1[1] and P1[0], are provided so that I²C may be used with 8-pin enCoRe V devices.

If In-System Serial Programming (ISSP) is used and the alternate I²C pin set is also used, you must consider the interaction between the enCoRe V Test Controller and the I²C bus. The interface requirements for ISSP must be reviewed to ensure that they are not violated.

Even if ISSP is not used, pins P1[1] and P1[0] respond differently than other I/O pins to a POR or XRES event. After an XRES event, both pins are pulled down to ground by going into the resistive zero drive mode before reaching the High-Z drive mode. After a POR event, P1[0] drives out a one, then goes to the resistive zero state for some time, and finally reaches the High-Z drive mode state. After POR, P1[1] goes into a resistive zero state for a while before going to the High-Z drive mode.

Bit 4: Stop IE. Stop Interrupt Enable. When this bit is set, a slave can interrupt upon Stop detection. The status bit associated with this interrupt is the Stop Status bit in the I2C_SCR Register. When the Stop Status bit transitions from ‘0’ to ‘1’, the interrupt is generated. It is important to note that the Stop Status bit is not automatically cleared. Therefore, if it is already set, no new interrupts are generated until it is cleared by firmware.

Bits 3 and 2: Clock Rate[1:0]. These bits offer a selection of three sampling and bit rates. All block clocking is based on the SYSCLK input, which is nominally12 MHz or 6 MHz. The sampling rate and the baud rate are determined as follows:

- Sample Rate = SYSCLK/Prescale Factor
- Baud Rate = 1/(Sample Rate x Samples per Bit)

When clocking the input with a frequency other than 6-/12-MHz (for example, clocking the device with an external clock), the baud rates and sampling rates scale accordingly. Whether the block works in a Standard Mode or Fast Mode system depends upon the sample rate.

The sample rate must be sufficient to resolve bus events, such as Start and Stop conditions. See the Philips Semiconductors (now NXP Semiconductors) I²C-Bus Specification, version 2.1, for minimum start and stop hold times.

Bit 0: Enable. When the slave is enabled, the block generates an interrupt upon any Start condition and an address byte that it receives indicating the beginning of an I²C transfer. When operating as a slave, the block is clocked from an external master. Therefore, the block works at any frequency up to the maximum defined by the currently selected clock rate. The internal clock is only used in slave mode, to ensure that there is adequate setup time from data output to the next clock upon the release of a slave stall. When the Enable bit is ‘0’, the block is held in reset and all status is cleared. Block enable is synchronized to the SYSCLK clock input (see “Timing Diagrams” on page 111). If the block is enabled when there is an I²C transaction already on the I²C bus, an erroneous start is detected if SCL is high and SDA is low. (SDA in this case is low because of a data/ack bit.) Thus, to avoid this erroneous start detection, enable the block only when the bus is idle (SCL and SDA are high when idle) or only when both SCL and SDA are low.

The other option is to change the drive modes of the I2C pins to be other than open drain mode and then enable the I2C block. After enabling the I2C block, wait for three I2C sample clocks, then configure the drive modes of the I2C pins to be in open drain mode.

Table 15-2. Enable Operation in I2C_CFG

Enable	Block Operation
No	Disabled The block is disconnected from the GPIO pins, P1[5] and P1[7]. (The pins may be used as general-purpose I/O.) When the slave is enabled, the GPIO pins are under control of the I2C hardware and are unavailable. All internal registers (except I2C_CFG) are held in reset.
Yes	Slave Mode Any external Start condition causes the block to start receiving an address byte. Regardless of the current state, any Start resets the interface and initiates a Receive operation. Any Stop causes the block to revert to an idle state

For additional information, refer to the I2C_CFG register on page 193.

15.3.4 I2C_SCR Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,D7h	I2C_SCR	Bus Error		Stop Status	ACK	Address	Transmit	LRB	Byte Complete	# : 00

Legend
Access is bit specific.

The I²C Status and Control Register (I2C_SCR) is used by the slave to control the flow of data bytes and to keep track of the bus state during a transfer.

This register contains status bits to determine the state of the current I²C transfer, and control bits to determine the actions for the next byte transfer. At the end of each byte transfer, the I²C hardware interrupts the M8C microcontroller and stalls the I²C bus on the subsequent low of the clock, until the enCoRe V device intervenes with the next command. This register may be read as many times as necessary; but on a subsequent write to this register, the bus stall is released and the current transfer continues.

There are five status bits: Byte Complete, LRB, Address, Stop Status, and Bus Error. These bits have Read/Clear (RC) access, which means that they are set by hardware but may be cleared by a write of '0' to the bit position. Under certain conditions, status is cleared automatically by the hardware.

There are two control bits: Transmit and ACK. These bits have RW access and may be cleared by hardware.

Bit 7: Bus Error. The Bus Error status detects misplaced Start or Stop conditions on the bus. These may be due to noise, rogue devices, or other devices that are not yet synchronized with the I²C bus traffic. According to the I²C specification, all compatible devices must reset their interface upon a received Start or Stop. This is a natural thing to do in slave mode because a Start initiates an address reception and a Stop idles the slave.

A bus error is defined as follows. A Start is only valid if the block is idle or a slave receiver is ready to receive the first bit of a new byte after an ACK. Any other timing for a Start condition sets the Bus Error bit. A Stop is only valid if the block is idle or a slave receiver is ready to receive the first bit of a new byte after an ACK. Any other timing for a Stop condition sets the Bus Error bit.

Bit 5: Stop Status. Stop status is set upon detection of an I²C Stop condition. This bit is sticky, which means that it remains set until a '0' is written back to it by the firmware. This bit may only be cleared if the Byte Complete status bit is set. If the Stop Interrupt Enable bit is set, an interrupt is also generated upon Stop detection. It is never automatically cleared. Using this bit, a slave can distinguish between a previous Stop or Restart upon a given address byte interrupt.

The selections are shown in the following table:

Bit	Access	Description
7	RC	Bus Error 1 = A misplaced Start or Stop condition was detected. This status bit must be cleared by firmware with a write of '0' to the bit position. It is never cleared by the hardware.
5	RC	Stop Status 1 = A Stop condition was detected. This status bit must be cleared by firmware with a write of '0' to the bit position. It is never cleared by the hardware.
4	RW	ACK: Acknowledge Out 0 = NACK the last received byte. 1 = ACK the last received byte. This bit is automatically cleared by hardware upon the following byte complete event.
3	RC	Address 1 = The transmitted or received byte is an address. This status bit must be cleared by firmware with a write of '0' to the bit position.
2	RW	Transmit 0 = Receive Mode. 1 = Transmit Mode. This bit is set by firmware to define the direction of the byte transfer. Any Start detect automatically clears this bit.
1	RC	LRB: Last Received Bit The value of the ninth bit in a transmit sequence, which is the acknowledge bit from the receiver. 0 = Last transmitted byte was ACKed by the receiver. 1 = Last transmitted byte was NAKed by the receiver. Any Start detect automatically clears this bit.
0	RC	Byte Complete Transmit Mode: 1 = 8 bits of data have been transmitted and an ACK or NACK is received. Receive Mode: 1 = 8 bits of data have been received. Any Start detect automatically clears this bit.

Bit 4: ACK.

This control bit defines the acknowledge data bit that is transmitted out in response to a received byte. When receiving, a byte complete interrupt is generated after the eighth data bit is received. Upon the subsequent write to this register to continue (or terminate) the transfer, the state of this bit determines the next transmitted data bit. It is ***active high***. A '1' sends an ACK and a '0' sends a NACK. A slave receiver sends a NACK to inform the master that it cannot receive any more bytes.

Bit 3: Address.

This bit is set when an address is received. This consists of a Start or Restart, and an address byte.

In slave mode when this status is set, firmware reads the received address from the data register and compares it with its own address. If the address does not match, the firmware writes a NACK indication to this register. No further interrupts occur until the next address is received. If the address does match, firmware must ACK the received byte, then byte complete interrupts are generated on subsequent bytes of the transfer.

When in I²C compatibility mode the Address bit should not be cleared between a start and the next byte complete. If it is cleared during this time the start status is cleared internally.

Bit 2: Transmit. This bit sets the direction of the shifter for a subsequent byte transfer. The shifter is always shifting in data from the I²C bus, but a write of '1' enables the output of the shifter to drive the SDA output line. Because a write to this register initiates the next transfer, data must be written to the data register before writing this bit. In receive mode, the previously received data must have been read from the data register before this write. Firmware derives this direction from the RW bit in the received slave address.

This direction control is only valid for data transfers. The direction of address bytes is determined by the hardware.

Bit 1: LRB. Last Received Bit. This is the last received bit in response to a previously transmitted byte. In transmit mode, the hardware sends a byte from the data register and clock in an acknowledge bit from the receiver. Upon the subsequent byte complete interrupt, firmware checks the value of this bit. A '0' is the ACK value and a '1' is a NACK value.

The meaning of LRB depends upon the current operating mode.

'0': ACK. The master wants to read another byte. The slave loads the next byte into the I2C_DR Register and sets the Transmit bit in the I2C_SCR register to continue the transfer.

'1': NACK. The master is done reading bytes. The slave reverts to IDLE state on the subsequent I2C_SCR write (regardless of the value written).

Bit 0: Byte Complete. The I²C hardware operates on a byte basis. In transmit mode, this bit is set and an interrupt is generated at the end of nine bits (the transmitted byte + the received ACK). In receive mode, the bit is set after the eight bits of data are received. When this bit is set, an interrupt is generated at these data sampling points, which are associated with the SCL input clock rising (see details in Timing Diagrams on page 111). If the enCoRe V device responds with a write back to this register before the subsequent falling edge of SCL (which is approximately one-half bit time), the transfer continues without interruption. However, if the device cannot respond within that time, the hardware holds the SCL line low, stalling the I²C bus. A subsequent write to the I2C_SCR register releases the stall.

For additional information, refer to the I2C_SCR register on page 194.

15.3.5 I2C_DR Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,D8h	I2C_DR	Data[7:0]								RW : 00

The I²C Data Register (I2C_DR) provides read/write access to the Shift register.

Bits 7 to 0: Data[7:0]. This register is not buffered; therefore, writes and valid data reads occur at specific points in the transfer. These cases are outlined as follows:

- **Slave Receiver** – Data in the I2C_DR register is only valid for reading when the Byte Complete status bit is set. Data bytes must be read from the I2C_DR register before writing to the I2C_SCR Register, which continues the transfer.
- **Slave Transmitter** – Data bytes must be written to the I2C_DR register before the Transmit bit is set in the I2C_SCR Register, which continues the transfer.

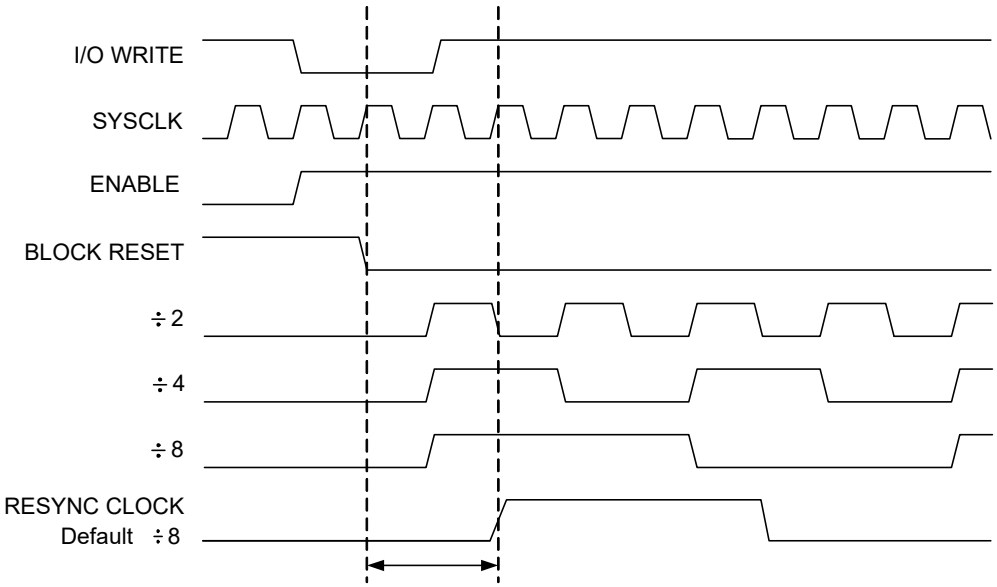
For additional information, refer to the I2C_DR register on page 195.

15.4 Timing Diagrams

15.4.1 Clock Generation

Figure 15-5 illustrates the I²C input clocking scheme. The SYSCLK pin is an input into a three-stage ripple divider that provides the baud rate selections. When the block is disabled, all internal state is held in a reset state. When the Enable bit in the I2C_CFG Register is set, the reset is synchronously released and the clock generation is enabled. All three taps from the ***ripple divider*** are selectable (/2, /4, /8) from the clock rate bits in the I2C_CFG Register. If any of the three divider taps is selected, that clock is resynchronized to SYSCLK. The resulting clock is routed to all of the synchronous elements in the design.

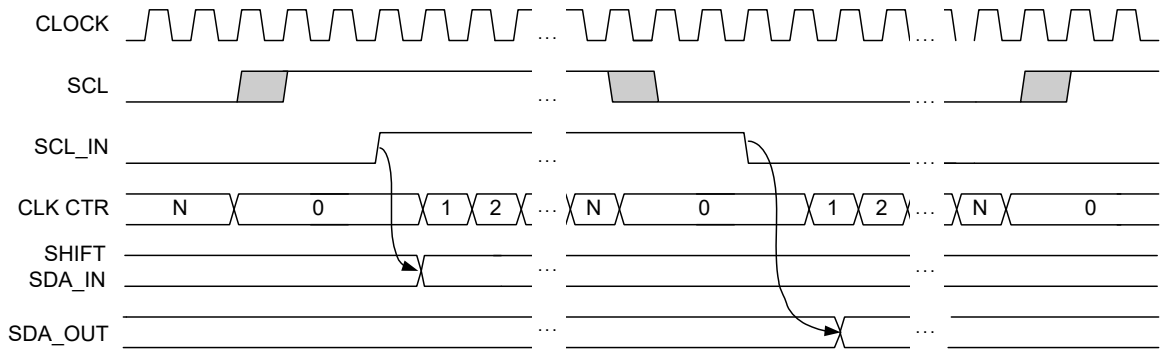
Figure 15-5. I²C Input Clocking



15.4.2 Basic I/O Timing

Figure 15-6 illustrates basic input/output timing that is valid for both 16 times sampling and 32 times sampling. For 16 times sampling, N=4; for 32 times sampling, N=12. N is derived from the half-bit rate sampling of eight and 16 clocks, respectively, minus the input latency of three (count of 4 and 12 correspond to 5 and 13 clocks).

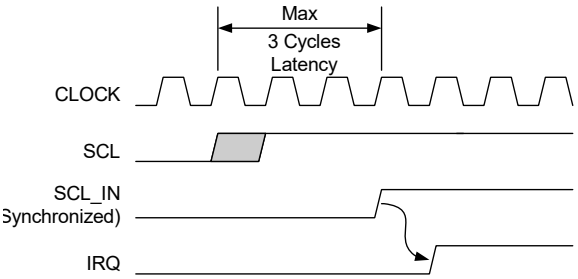
Figure 15-6. Basic Input/Output Timing



15.4.3 Status Timing

Figure 15-7 illustrates the interrupt timing for byte complete, which occurs on the positive edge of the ninth clock (byte + ACK/NACK) in transmit mode and on the positive edge of the eighth clock in receive mode. There is a maximum of three cycles of latency due to the input synchronizer/filter circuit. As shown, the interrupt occurs on the clock following a valid SCL positive edge input transition (after the synchronizers). The Address bit is set with the same timing but only after a slave address is received. The LRB (Last Received Bit) status is also set with the same timing but only on the ninth bit after a transmitted byte.

Figure 15-7. Byte Complete, Address, LRB Timing



Transmit: Ninth positive edge SCL
Receive: Eighth positive edge SCL

Figure 15-8 shows the timing for Stop status. This bit is set (and the interrupt occurs) two clocks after the synchronized and filtered SDA line transitions to a '1', when the SCL line is high.

Figure 15-8. Stop Status and Interrupt Timing

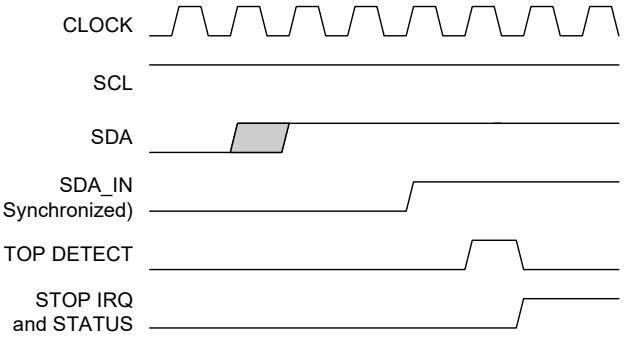
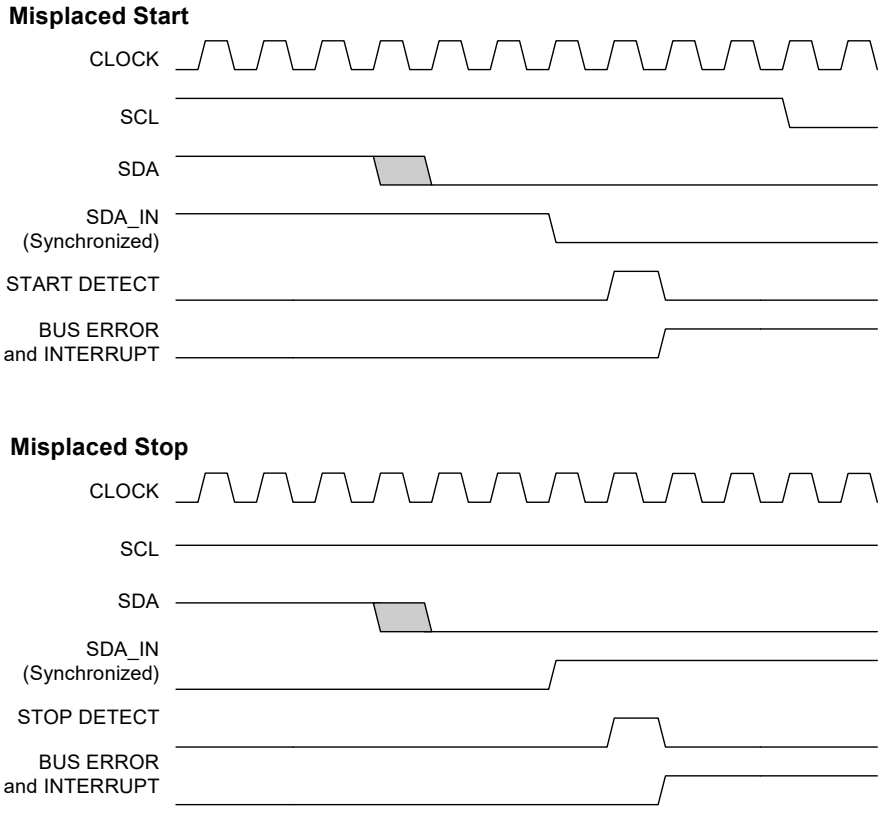


Figure 15-9 illustrates the timing for bus error interrupts. Bus Error status (and interrupt) occurs one cycle after the internal Start or Stop detect (two cycles after the filtered and synchronized SDA input transition).

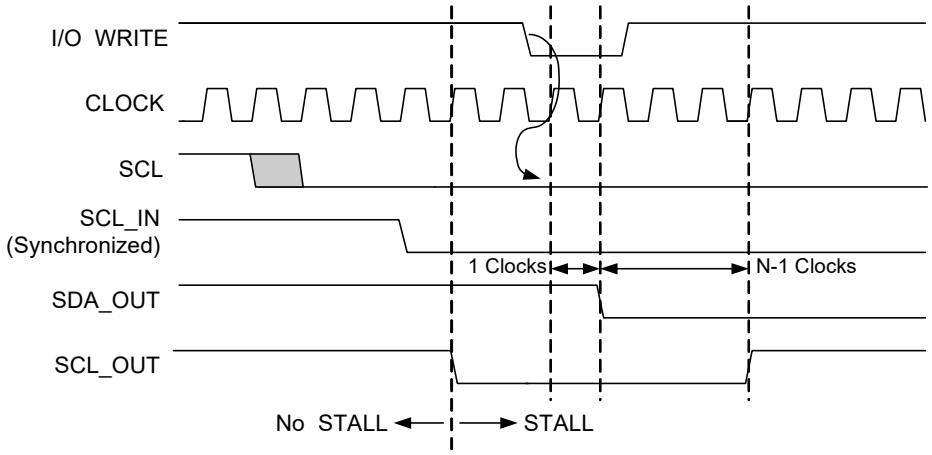
Figure 15-9. Bus Error Interrupt Timing



15.4.4 Slave Stall Timing

When a byte complete interrupt occurs, the enCoRe V device firmware must respond with a write to the I2C_SCR Register to continue the transfer (or terminate the transfer). The interrupt occurs two clocks after the rising edge of SCL_IN (see Status Timing on page 112). As illustrated in Figure 15-10, firmware has until one clock after the falling edge of SCL_IN to write to the I2C_SCR Register; otherwise, a stall occurs. After stalled, the I/O write releases the stall. The setup time between data output and the next rising edge of SCL is always N-1 clocks.

Figure 15-10. Slave Stall Timing



16. System Resets

This chapter discusses the System Resets and their associated registers. enCoRe V devices support several types of resets. The various resets are designed to provide error-free operation during power up for any voltage ramping profile, to allow user-supplied external reset, and to provide recovery from errant code operation. For a complete table of the System Reset registers, refer to the Summary Table of the System Resource Registers on page 93. For a quick reference of all enCoRe V registers in address order, refer to the Register Reference chapter on page 163.

16.1 Architectural Description

When reset is initiated, all registers are restored to their default states. In the Register Reference chapter on page 163, this is indicated by the POR row in the register tables and elsewhere it is indicated in the Access column values on the right side of the colon, in the register tables. Minor exceptions are explained ahead.

The following types of resets occur in the enCoRe V device:

- **Power-on-Reset (POR).** This occurs at low supply voltage and is comprised of multiple sources.
- **External Reset (XRES).** This active high reset is driven into the enCoRe V device on parts that contain an XRES pin.
- **Watchdog Reset (WDR).** This optional reset occurs when the watchdog timer expires before being cleared by user firmware. Watchdog resets default to off.
- **Internal Reset (IRES).** This occurs during the boot sequence if the SROM code determines that flash reads are invalid.

The occurrence of a reset is recorded in the Status and Control registers (CPU_SCR0 for POR, XRES, and WDR) or in the System Status and Control Register 1 (CPU_SCR1 for IRESS). Firmware can interrogate these registers to determine the cause of a reset.

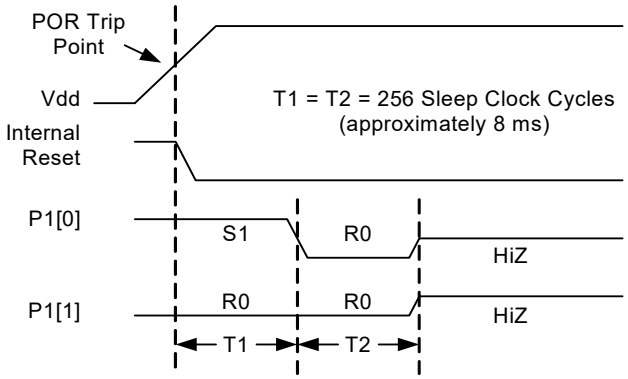
16.2 Pin Behavior During Reset

POR and XRES cause toggling on two GPIO pins, P1[0] and P1[1], as described ahead and illustrated in Figure 16-1 and Figure 16-2. This allows programmers to synchronize with the enCoRe V device. All other GPIO pins are placed in a high-impedance state during and immediately following reset.

16.2.1 GPIO Behavior on Power Up

At power-up, the internal POR causes P1[0] to initially drive a strong high (1) while P1[1] drives a resistive low (0). After 256 sleep oscillator cycles (approximately 8 ms), the P1[0] signal transitions to a resistive low state. After an additional 256 sleep oscillator clocks, both pins transition to a high-impedance state and normal CPU operation begins. This is illustrated in the following figure.

Figure 16-1. P1[1:0] Behavior on Power Up



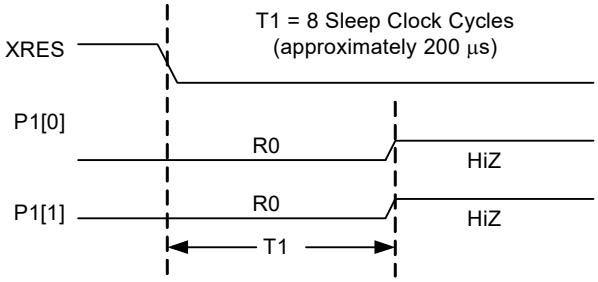
16.2.2 Powerup External Reset Behavior

The device's core runs on chip regulated supply, so there is a time delay in powering up the core. A short XRES pulse at power up causes an external reset startup behavior. However, the event is latched and applied only after the core has powered up (a delay of about 1 ms).

16.2.3 GPIO Behavior on External Reset

During external reset (XRES=1), both P1[0] and P1[1] drive resistive low (0). After XRES deasserts, these pins continue to drive resistive low for another eight sleep clock cycles (approximately 200 μs). After this time, both pins transition to a high-impedance state and normal CPU operation begins. This is illustrated in Figure 16-2.

Figure 16-2. P1[1:0] Behavior on External Reset (XRES)



16.3 Register Definitions

The following registers are associated with the enCoRe V System Resets and are listed in address order. Each register description has an associated register table showing the bit structure for that register. The bits in the tables that are grayed out are reserved bits and are not detailed in the register descriptions that follow. Always write reserved bits with a value of 0. For a complete table of system reset registers, refer to the "Summary Table of the System Resource Registers" on page 93.

16.3.1 CPU_SCR1 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
x,FEh	CPU_SCR1	IRESS			SLIMO[1:0]				IRAMDIS	# : 0

Legend
x An "x" before the comma in the address field indicates that this register can be read or written to no matter what bank is used.
Access is bit specific. Refer to the Register Reference chapter on page 163 for additional information.

The System Status and Control Register 1 (CPU_SCR1) conveys the status and control of events related to internal resets and watchdog reset.

Bit 7: IRESS. Internal Reset Status. This bit is a read-only bit that determines if the booting process occurred more than once.

When this bit is set, it indicates that the SROM SWBootReset code executed more than once. If this bit is not set, the SWBootReset executed only once. In either case, the SWBootReset code does not allow execution from code stored in flash until the M8C core is in a safe operating mode with respect to supply voltage and flash operation. There is no need for concern when this bit is set. It is provided for systems that may be sensitive to boot time, so that they can determine if the normal one-pass boot time was exceeded.

For more information on the SWBootReset code see the Supervisory ROM (SROM) on page 32.

Bit 4:3 SLIMO[1:0]. These bits set the IMO frequency range. See the table ahead for more information. These changes allow optimization of speed and power. The IMO trim value must also be changed when SLIMO is changed (see Engaging Slow IMO on page 67).

When not in external clocking mode, the IMO is the source for SYSCLK; therefore, when the speed of the IMO changes so does SYSCLK.

SLIMO	CY7C6xxxx
00	12
01	6
10	24
11	Reserved

Bit 0: IRAMDIS. Initialize RAM Disable. This bit is a control bit that is readable and writeable. The ***default value*** for this bit is '0', which indicates that the maximum amount of SRAM must be initialized on watchdog reset to a value of 00h. When the bit is '1', the minimum amount of SRAM is initialized after a watchdog reset.

For additional information, refer to the CPU_SCR1 register on page 210.

16.3.2 CPU_SCR0 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
x,FFh	CPU_SCR0	GIES		WDRS	PORS	Sleep			STOP	# : XX

Legend
Access is bit specific. Refer to register detail for additional information.
XX The reset value is 10h after POR/XRES and 20h after a watchdog reset.

The System Status and Control Register 0 (CPU_SCR0) is used to convey the status and control of events for various functions of a enCoRe V device.

Bit 7: GIES. Global Interrupt Enable Status. This bit is a read-only status bit and its use is discouraged. The GIES bit is a legacy bit that was used to provide the ability to read the GIE bit of the CPU_F register. However, the CPU_F register is now readable. When this bit is set, it indicates that the GIE bit in the CPU_F register is also set, which in turn, indicates that the microprocessor services interrupts.

Bit 5: WDRS. WatchDog Reset Status. This bit may not be set. It is normally '0' and automatically set whenever a watchdog reset occurs. The bit is readable and clearable by writing a '0' to its bit position in the CPU_SCR0 register.

Bit 4: PORS. Power-on-reset Status. This bit, which is the watchdog enable bit, is set automatically by a POR or XRES. If the bit is cleared by user code, the watchdog timer is enabled. After cleared, the only way to reset the PORS bit is to go through a POR or XRES. Thus, there is no way to disable the watchdog timer other than to go through a POR or XRES.

Bit 3: Sleep. This bit is used to enter Low-power Sleep mode when set. To wake up the system, this register bit is cleared asynchronously by any enabled interrupt. Two special features of this bit ensure proper sleep operation. First, the write to set the register bit is blocked if an interrupt is about to be taken on that instruction boundary (immediately after the write). Second, there is a hardware interlock to ensure that, when set, the SLEEP bit may not be cleared by an incoming interrupt until the sleep circuit has finished performing the sleep sequence and the system-wide power down signal is asserted. This prevents the sleep circuit from being interrupted in the middle of the process of system power down, possibly leaving the system in an indeterminate state.

Bit 0: STOP. This bit is readable and writeable. When set, the enCoRe V M8C stops executing code until a reset event occurs. This can be either a POR, WDR, or XRES. If an application wants to stop code execution until a reset, the preferred method is to use the HALT instruction rather than a register write to this bit.

For additional information, refer to the CPU_SCR0 register on page 211.

16.4 Timing Diagrams

16.4.1 Power-On-Reset

A power-on-reset (POR) is triggered whenever the supply voltage is below the POR trip point. POR ends when the supply voltage rises above this voltage. Refer to the POR and LVD chapter on page 121 for more information on the operation of the POR block.

POR consists of two pieces: an Imprecise POR (IPOR) and a Precision POR (PPOR). POR refers to the OR of these two functions. IPOR has coarser accuracy and its trip point is typically lower than PPOR's trip point. PPOR is derived from a circuit that is calibrated (during boot) for a very accurate location of the POR trip point.

During POR (POR=1), the IMO is powered off for low power during startup. After POR deasserts, the IMO is started (see Figure 16-4).

POR configures register reset status bits, as shown in 16.4.4 Reset Details on page 120.

16.4.2 External Reset

An external reset (XRES) is caused by pulling the XRES pin high. The XRES pin has an always-on, pull-down resistor, so it does not require an external pull down for operation and can be tied directly to ground or left open. Behavior after XRES is similar to POR. In reset, shifting key AC52h (with MSB shifted in first) onto ISSP bus lines can lead to permanent chip damage. This is possible because, upon reception of key AC52h, the chip enters a test mode in which the internal regulator is bypassed and the core powered with a higher voltage than specified. For this reason, avoid shifting key AC52h while in reset.

During XRES (XRES=1), the IMO is powered off for low power during startup. After XRES deasserts, the IMO is started (see Figure 16-4).

16.4.4 Reset Details on page 120 shows how the XRES configures register reset status bits.

16.4.3 Watchdog Timer Reset

You can enable the Watchdog Timer Reset (WDR), by clearing the PORS bit in the CPU_SCR0 register. After the PORS bit is cleared, the watchdog timer cannot be disabled. The only exception to this is if a POR/XRES event takes place, which disables the WDR. Note that a WDR does not clear the watchdog timer. See Watchdog Timer on page 86 for details of the watchdog operation.

When the watchdog timer expires, a watchdog event occurs, resulting in the reset sequence. Some characteristics unique to the WDR are as follows.

- enCoRe V device reset asserts for one cycle of the CLK32K clock (at its reset state).
- The IMO is not halted during or after WDR (that is, the part does not go through a low-power phase).
- CPU operation re-starts one CLK32K cycle after the internal reset deasserts (see Figure 16-3).

16.4.4 Reset Details on page 120 shows how the WDR configures register reset status bits.

Figure 16-3. Key Signals During WDR

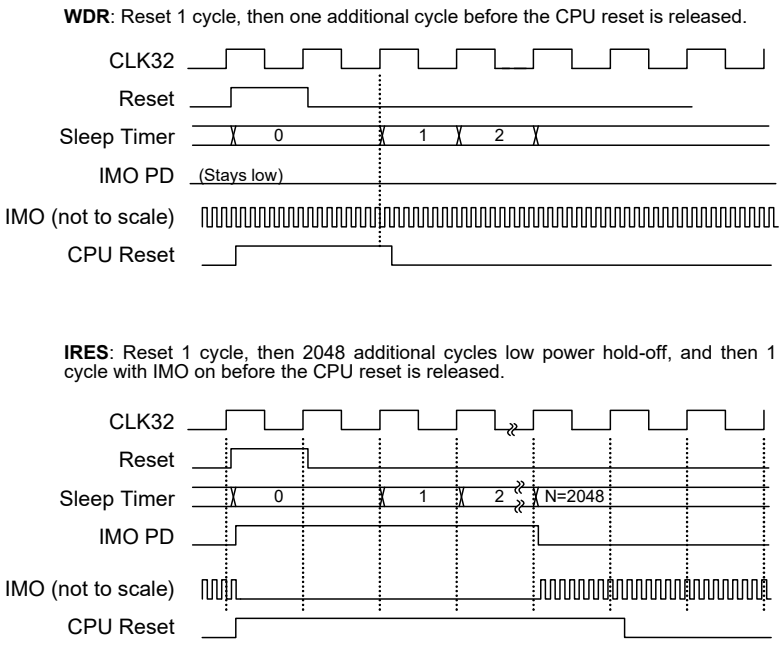
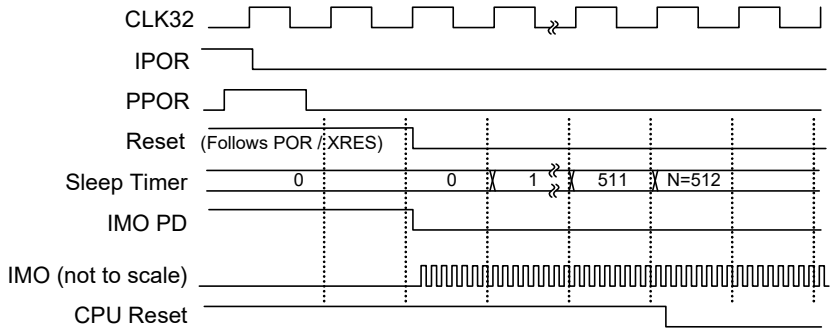
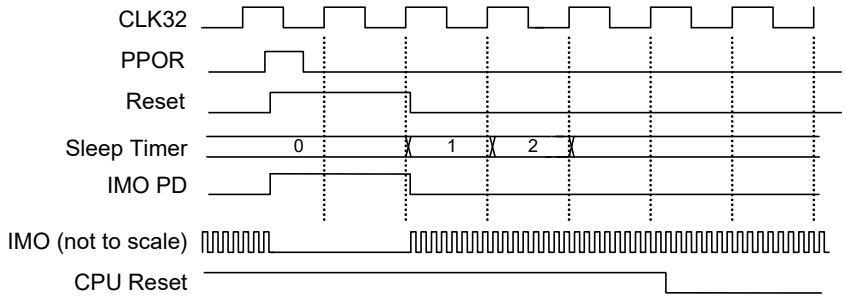


Figure 16-4. Key Signals During POR and XRES

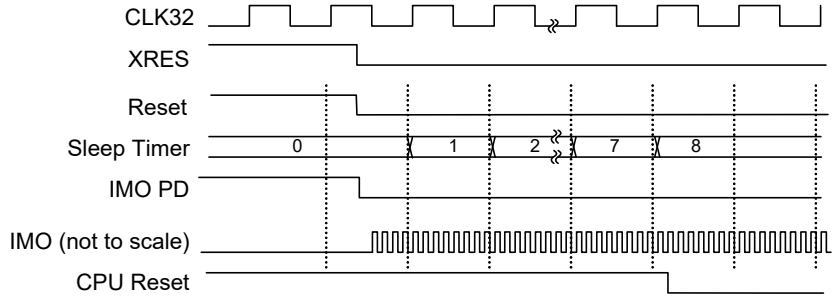
POR (IPOR followed by PPOR): Reset while POR is high (IMO off), then 511(+) cycles (IMO on), and then the CPU reset is released. **XRES** is the same, with N=8.



PPOR (with no IPOR): Reset while PPOR is high and to the end of the next 32K cycle (IMO off); 1 cycle IMO on before the CPU reset is released. Note that at the 3V level, PPOR tends to be brief because the reset clears the POR range register (VLT_CR) back to the default 2.4V setting.



XRES: Reset while XRES is high (IMO off), then 7(+) cycles (IMO on), and then the CPU reset is released.



16.4.4 Reset Details

Timing and functionality details are summarized in Table 16-1. Figure 16-4 on page 119 shows some of the relevant signals for IPOR, PPOR, XRES, and WDR.

Table 16-1. Reset Functionality

Item	IPOR (Part of POR)	PPOR (Part of POR)	XRES	WDR
Reset Length	While POR=1	While PPOR=1, plus 30-60 µs (1-2 clocks)	While XRES=1	30 µs (1 clock)
Low-power (IMO Off) During Reset?	Yes	Yes	Yes	No
Low-power Wait Following Reset?	No	No	No	No
CLK32K Cycles from End of Reset to CPU Reset Deasserts[a]	512	1	8	1
Register Reset (See Next Line for CPU_SCR0, CPU_SCR1)	All	All, except PPOR does not reset Bandgap Trim register	All	All
Reset Status Bits in CPU_SCR0, CPU_SCR1	Set PORS, Clear WDRS, Clear IRAMDIS	Set PORS, Clear WDRS, Clear IRAMDIS	Set PORS, Clear WDRS, Clear IRAMDIS	Clear PORS, Set WDRS, IRAMDIS unchanged
Bandgap Power	On	On	On	On
Boot Time[b]	2.2 ms	2.2 ms	2.2 ms	2.2 ms

a. CPU reset is released after synchronization with the CPU clock.
b. Measured from CPU reset release to execution of the code at flash address 0x0000.

16.5 Power Modes

- The ILO block drives the CLK32K clock used to time most events during the reset sequence. This clock is powered down by IPOR but not by any other reset. The sleep timer provides interval timing.
- While POR or XRES assert, the IMO is powered off to reduce startup power consumption.
- During and following IRES (for 64 ms nominally), the IMO is powered off for low average power during slow supply ramps.
- During and after POR or XRES, the bandgap circuit is powered up.
- The IMO is always on for at least one CLK32K cycle before CPU reset is deasserted.

17. POR and LVD

This chapter briefly discusses the power-on-reset (POR) and low-voltage detect (LVD) circuits and their associated registers. For a complete table of the POR registers, refer to the Summary Table of the System Resource Registers on page 93. For a quick reference of all enCoRe V registers in address order, refer to the Register Reference chapter on page 163.

17.1 Architectural Description

The power-on-reset (POR) and low-voltage detect (LVD) circuits provide protection against low voltage conditions. The POR function senses Vcc and Vcore (regulated voltage) holding the system in reset until the magnitude of Vcc and Vcore supports operation to specification. The LVD function senses Vcc and provides an interrupt to the system when Vcc falls below a selected threshold. Other outputs and status bits are provided to indicate important voltage trip levels. Refer to Section 16.2 Pin Behavior During Reset for a description of GPIO pin behavior during power up.

17.2 Register Definitions

The following registers are associated with the POR and LVD, and are listed in address order. The following register descriptions have an associated register table showing the bit structure. The bits that are grayed out in the register tables are reserved bits and are not detailed in the register descriptions that follow. Reserved bits must always be written with a value of '0'. For a complete table of the POR registers, refer to the Summary Table of the System Resource Registers on page 93.

17.2.1 VLT_CR Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,E3h	VLT_CR	HPOR		PORLEV[1:0]		LVDTBEN	VM[2:0]			RW : 00

The Voltage Monitor Control Register (VLT_CR) sets the trip points for POR, MON1 and LVD.

The VLT_CR register is cleared by all resets. This can cause reset cycling during very slow supply ramps to 5 V when the MON1 range is set for the 5-V range. This is because the reset clears the MON1 range setting back to 1.8 V and a new boot or startup occurs (possibly many times). You can manage this with sleep mode or reading voltage status bits if such cycling is an issue.

Bit 7: HPOR. This bit along with PORLEV sets one of the four values for the PPOR trip voltage. See Table 21-1 on page 231.

Bits 5 and 4: PORLEV[1:0]. These bits along with HPOR sets one of the four values for the PPOR trip voltage. See Table 21-1 on page 231.

See the "DC POR and LVD Specifications" table in the Electrical Specifications section of the enCoRe V device datasheet for voltage tolerances for each setting.

Bit 3: LVDTBEN. This bit is ANDed with LVD to produce a throttle back signal that reduces CPU clock speed when low voltage conditions are detected. When the throttle back signal is asserted, the CPU speed bits in the OSC_CR0 register are reset, forcing the CPU speed to its reset state.

Bits 2 to 0: VM[2:0]. These bits set the Vdd level at which the LVD Comparator switches.

See the "DC POR and LVD Specifications" table in the Electrical Specifications section of the enCoRe V device datasheet for voltage tolerances for each setting.

For additional information, refer to the VLT_CR register on page 231.

17.2.2 VLT_CMP Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,E4h	VLT_CMP							LVD		RW : 0

The Voltage Monitor Comparators Register (VLT_CMP) reads the state of internal supply voltage monitors.

Bit 1: LVD. This bit reads the state of the LVD comparator. Zero Vdd is above the trip point. The trip points for LVD are set by VM[2:0] in the VLT_CR register.

For additional information, refer to the VLT_CMP register on page 232.

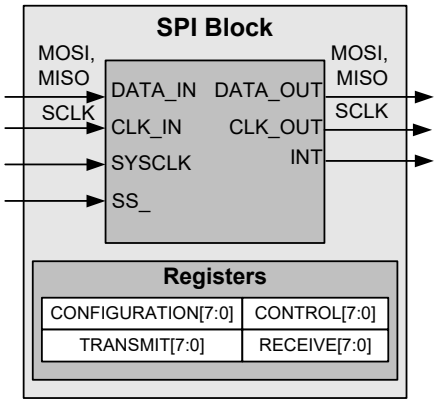
18. SPI

This chapter presents the Serial Peripheral Interconnect (SPI) and its associated registers. For a complete table of the SPI registers, see the Summary Table of the System Resource Registers on page 93. For a quick reference of all enCoRe V registers in address order, refer to the Register Reference chapter on page 163.

18.1 Architectural Description

The Serial Peripheral Interconnect (SPI) block is a dedicated master or slave SPI. The SPI slave function requires three inputs: Clock, Data, and SS_ (unless the SS_ is forced active with the SS_bit in the configuration register).

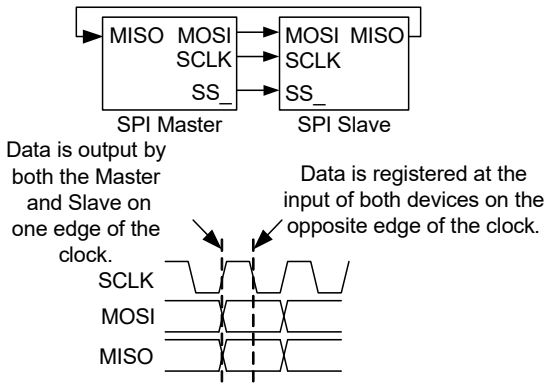
Figure 18-1. SPI Block Diagram



18.1.1 SPI Protocol Function

The SPI is a Motorola™ specification for implementing full-duplex synchronous serial communication between devices. The 3-wire protocol uses both edges of the clock to enable synchronous communication without the need for stringent setup and hold requirements. Figure 18-2 shows the basic signals in a simple connection.

Figure 18-2. Basic SPI Configuration



A device can be a master or slave. A master outputs clock and data to the ***slave device*** and inputs slave data. A slave device inputs clock and data from the master device and outputs data for input to the master. Together, the master and slave are essentially a circular Shift register, where the master generates the clocking and initiates data transfers.

A basic data transfer occurs when the master sends eight bits of data, along with eight clocks. In any transfer, both master and slave transmit and receive simultaneously. If the master only sends data, the received data from the slave is ignored. If the master wishes to receive data from the slave, the master must send dummy bytes to generate the clocking for the slave to send data back.

18.1.1.1 SPI Protocol Signal Definitions

The SPI protocol signal definitions are located in Table 18-1. The use of the SS_ signal varies according to the capability of the slave device.

Table 18-1. SPI Protocol Signal Definitions

Name	Function	Description
MOSI	Master Out Slave In	Master data output.
MISO	Master In Slave Out	Slave data output.
SCLK	Serial Clock	Clock generated by the master.
SS_	Slave Select (Active Low)	This signal is provided to enable multi-slave connections to the MISO pin. The MOSI and SCLK pins can be connected to multiple slaves, and the SS_ input selects which slave receives the input data and drives the MISO line.

18.1.2 SPI Master Function

The SPI Master (SPIM) offers SPI operating modes 0-3. By default, the most significant bit (MSb) of the data byte is shifted out first. An additional option can be set to reverse the direction and shift the data byte out the least significant bit (LSb) first. (Refer to the timing diagrams for this function on page 130.)

When configured for SPIM, DR0 functions as a Shift register with input from the DATA input (MISO) and output to the primary output F1 (MOSI). DR1 is the TX Buffer register and DR2 is the RX Buffer register.

The SPI protocol requires data to be registered at the device input, on the opposite edge of the clock that operates the output shifter. An additional register (RXD), at the input to the DR0 Shift register, has been implemented for this purpose. This register stores received data for one-half cycle before it is clocked into the Shift register.

The SPIM controls data transmission between master and slave because it generates the bit clock for internal clocking and for clocking the SPIS. The bit clock is derived from the CLK input selection.

There are four control bits and four status bits in the Control register (SPI_CR) that provide for enCoRe V device interfacing and synchronization.

The SPIM hardware has no support for driving the Slave Select (SS_) signal. The behavior and use of this signal is application and enCoRe V device dependent and, if required, must be implemented in firmware.

18.1.2.1 Usability Exceptions

The following are usability exceptions for the SPI Protocol function.

- The SPI_RXR (RX Buffer) register is not writeable.
- The SPI_TXR (TX Buffer) register is not readable.

18.1.2.2 Block Interrupt

The SPIM block has a selection of two interrupt sources: interrupt on TX Reg Empty (default) or interrupt on SPI Complete. Mode bit 1 in the Function register controls the selection. These modes are discussed in detail in SPIM Timing on page 130.

If SPI Complete is selected as the block interrupt, the Control register must be read in the interrupt routine so that this status bit is cleared; otherwise, no subsequent interrupts are generated.

18.1.3 SPI Slave Function

The SPI Slave (SPIS) offers SPI operating modes 0-3. By default, the MSb of the data byte is shifted out first. An additional option can be set to reverse the direction and shift the data byte out LSb first. (Refer to the timing diagrams for this function on page 134.)

The SPI protocol requires data to be registered at the device input, on the opposite edge of the clock that operates the output shifter. An additional register (RXD), at the input to the DR0 Shift register, is implemented for this purpose. This register stores received data for one-half cycle before it is clocked into the Shift register.

The SPIS function derives all clocking from the SCLK input (typically an external SPI Master). This means that the master must initiate all transmissions. For example, to read a byte from the SPIS, the master must send a byte.

There are four control bits and four status bits in the Control register (SPI_CR) that provide for enCoRe V device interfacing and synchronization.

There is an additional data input in the SPIS, Slave Select (SS_), which is an active low signal. SS_ must be asserted to enable the SPIS to receive and transmit. SS_ has two high level functions: 1) To allow the selection of a given slave in a multi-slave environment, and 2) To provide additional clocking for TX data queuing in SPI modes 0 and 1.

SS_ may be controlled from an external pin or can be controlled by way of user firmware.

When SS_ is negated, the SPIS ignores any MOSI/SCLK input from the master. In addition, the SPIS state machine is reset and the MISO output is forced to idle at logic 1. This allows for a wired-AND connection in a multi-slave environment. Note that if High-Z output is required when the slave is not selected, this behavior must be implemented in firmware with I/O writes to the Port Drive register.

18.1.3.1 Usability Exceptions

The following are usability exceptions for the SPI Slave function.

- The SPI_RXR (RX Buffer) register is not writeable.
- The SPI_TXR (TX Buffer) register is not readable.

18.1.3.2 Block Interrupt

The SPIS block has a selection of two interrupt sources: Interrupt on TX Reg Empty (default) or interrupt on SPI Complete (same selection as the SPIM). Mode bit 1 in the Function register controls the selection.

If SPI Complete is selected as the block interrupt, the Control register must still be read in the interrupt routine so that this status bit is cleared; otherwise, no subsequent interrupts are generated.

18.1.4 Input Synchronization

All pin inputs are double synchronized to SYSCLK by default. Synchronization can be bypassed by setting the BYPS bit in the SPI_CFG register.

18.2 Register Definitions

The following registers are associated with the SPI and are listed in address order. The register descriptions have an associated register table showing the bit structure for that register. For a complete table of SPI registers, refer to the Summary Table of the System Resource Registers on page 93.

Data Registers

18.2.1 SPI_TXR Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,29h	SPI_TXR	Data[7:0]								W : 00

The SPI Transmit Data Register (SPI_TXR) is the SPI's transmit data register.

Bits 7 to 0: Data[7:0]. These bits encompass the SPI Transmit register. They are discussed by function type in Table 18-2 and Table 18-3.

For additional information, refer to the SPI_TXR register on page 166.

18.2.2 SPI_RXR Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,2Ah	SPI_RXR	Data[7:0]								R : 00

The SPI Receive Data Register (SPI_RXR) is the SPI's receive data register. A write to this register clears the RX Reg Full status bit in the Control register (SPI_CR).

Bits 7 to 0: Data[7:0]. These bits encompass the SPI Receive register. They are discussed by function type in Table 18-2 and Table 18-3.

For additional information, refer to the SPI_RXR register on page 167.

18.2.2.1 SPI Master Data Register Definitions

There are two 8-bit Data registers and one 8-bit Control/Status register. Table 18-2 explains the meaning of the Transmit and Receive registers in the context of SPIM operation.

Table 18-2. SPIM Data Register Descriptions

Name	Function	Description
SPI_TXR	TX Buffer	Write only register. If no transmission is in progress and this register is written to, the data from this register is loaded into the Shift register on the following clock edge, and a transmission is initiated. If a transmission is currently in progress, this register serves as a buffer for TX data. This register must only be written to when TX Reg Empty status is set and the write clears the TX Reg Empty status bit in the Control register. When the data is transferred from this register to the Shift register, then TX Reg Empty status is set.
SPI_RXR	RX Buffer	Read-only register. When a byte transmission/reception is complete, the data in the shifter is transferred into the RX Buffer register and RX Reg Full status is set in the Control register. A read from this register clears the RX Reg Full status bit in the Control register.

18.2.2.2 SPI Slave Data Register Definitions

There are two 8-bit Data registers and one 8-bit Control/Status register. Table 18-3 explains the meaning of the Transmit and Receive registers in the context of SPIS operation.

Table 18-3. SPIS Data Register Descriptions

Name	Function	Description
SPI_TXR	TX Buffer	Write only register. This register must only be written to when TX Reg Empty status is set and the write clears the TX Reg Empty status bit in the Control register. When the data is transferred from this register to the Shift register, then TX Reg Empty status is set.
SPI_RXR	RX Buffer	Read-only register. When a byte transmission/reception is complete, the data in the shifter is transferred into the RX Buffer register and RX Reg Full status is set in the Control register. A read from this register clears the RX Reg Full status bit in the Control register.

Control Register

18.2.3 SPI_CR Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,2Bh	SPI_CR	LSb First	Overrun	SPI Complete	TX Reg Empty	RX Reg Full	Clock Phase	Clock Polarity	Enable	# : 00

Legend
Access is bit specific. Refer to the register detail for additional information.

The SPI Control Register (SPI_CR) is the SPI's control register.

Bit 7: LSb First. This bit determines how the serial data is shifted out, either LSb or MSb first.

Bit 6: Overrun. This status bit indicates whether there was a receive buffer overrun. A read from the receive buffer after each received byte must be performed before the reception of the next byte to avoid an overrun condition.

Bit 5: SPI Complete. This status bit indicates the completion of a transaction. A read from this register clears this bit.

Bit 4: TX Reg Empty. This status bit indicates whether the Transmit register is empty.

Bit 3: RX Reg Full. This status bit indicates a Receive register full condition.

Bit 2: Clock Phase. This bit determines the edge (rising or falling) on which the data changes.

Bit 1: Clock Polarity. This bit determines the logic level the clock codes to in its idle state.

Bit 0: Enable. This bit enables the SPI block.

For additional information, refer to the SPI_CR register on page 168.

18.2.3.1 SPI Control Register Definitions

Table 18-4. SPI Control Register Descriptions

Bit #	Name	Access	Description
7	LSb First	Read/Write	0 = Data shifted out MSb First. 1 = Data shifted out LSb First.
6	Overrun	Read Only	0 = No overrun. 1 = Indicates new byte received before previous one is read.
5	SPI Complete	Read Only	0 = Transaction in progress. 1 = Transaction is complete. Reading SPI_CR clears this bit.
4	TX Reg Empty	Read Only	0 = TX register is full. 1 = TX register is empty. Writing SPI_TXR register clears this bit.
3	RX Reg Full	Read Only	0 = RX register is not full. 1 = RX register is full. Reading SPI_RXR register clears this bit.
2	Clock Phase	Read/Write	0 = Data changes on trailing edge. 1 = Data changes on leading clock edge.
1	Clock Polarity	Read/Write	0 = Non-inverted, clock idles low (modes 0, 2). 1 = Inverted, clock idles high (modes 1, 3).
0	Enable	Read/Write	0 = Disable SPI function. 1 = Enable SPI function.

Configuration Register

The configuration block contains 1 register. This register must not be changed while the block is enabled. Note that the SPI Configuration register is located in bank 1 of the enCoRe V device's memory map.

18.2.4 SPI_CFG Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,29h	SPI_CFG	Clock Sel[2:0]			Bypass	SS_	SS_EN_	Int Sel	Slave	RW : 00

The SPI Configuration Register (SPI_CFG) is used to configure the SPI.

Bits 7 to 5: Clock Sel [2:0]. Clock Selection. These bits determine the operating frequency of the SPI Master.

Bit 4: Bypass. This bit determines whether the inputs are synchronized to SYSCLK.

Bit 3: SS_. Slave Select. This bit determines the logic value of the SS_ signal when the SS_EN_ signal is asserted (SS_EN_ = 0).

Bit 2: SS_EN_. Slave Select Enable. This active low bit determines if the slave select (SS_) signal is driven internally. If it is driven internally, its logic level is determined by the SS_ bit. If it is driven externally, its logic level is determined by the external pin.

Bit 1: Int Sel. Interrupt Select. This bit selects which condition produces an interrupt.

Bit 0: Slave. This bit determines whether the block functions as a master or slave.

For additional information, refer to the SPI_CFG register on page 214.

18.2.4.1 SPI Configuration Register Definitions

Table 18-5. SPI Configuration Register Descriptions

Bit #	Name	Access	Mode	Description
7:5	Clock Sel	Read/Write	Master	SYSCLK 000b / 2 001b / 4 010b / 8 011b / 16 100b / 32 101b / 64 110b / 128 111b / 256
4	Bypass	Read/Write	Master/Slave	0 = All pin inputs are doubled, synchronized. 1 = Input synchronization is bypassed.
3	SS_	Read/Write	Slave	0 = Slave selected. 1 = Slave not selected. Slave selection is determined from external SS_ pin.
2	SS_EN_	Read/Write	Slave	0 = Slave selection determined from SS_ bit. 1 = Slave selection determined from external SS_ pin.
1	Int Sel	Read/Write	Master/Slave	0 = Interrupt on TX Reg Empty. 1 = Interrupt on SPI Complete.
0	Slave	Read/Write	Master/Slave	0 = Operates as a master. 1 = Operates as a slave.

18.2.5 Related Registers

- IO_CFG1 Register on page 224.

18.3 Timing Diagrams

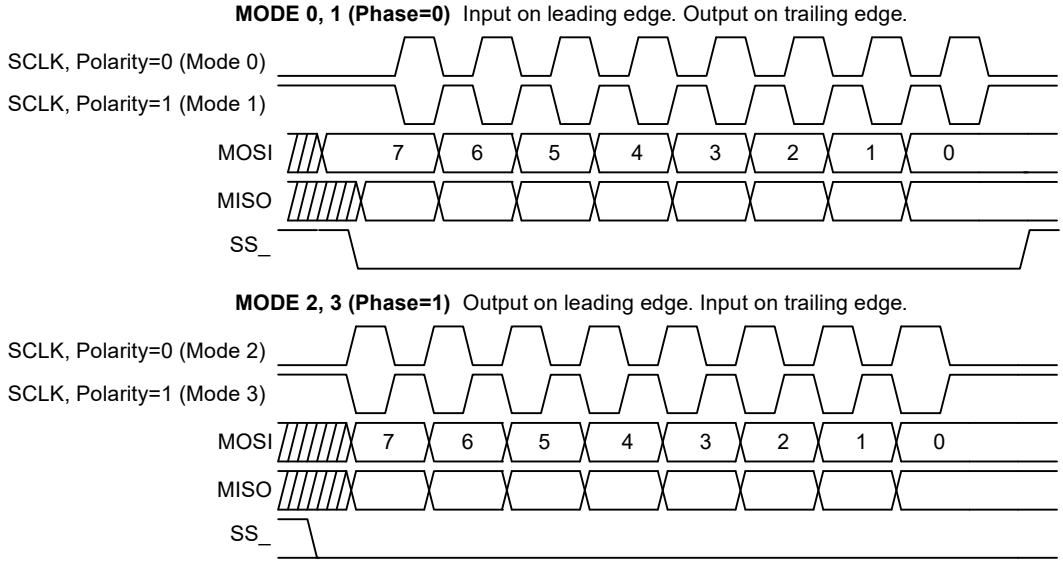
18.3.1 SPI Mode Timing

Figure 18-3 shows the SPI modes that are typically defined as 0, 1, 2, or 3. These mode numbers are an encoding of two control bits: Clock Phase and Clock Polarity.

Clock Phase indicates the relationship of the clock to the data. When the clock phase is '0', it means that the data is registered as an input on the leading edge of the clock and the next data is output on the trailing edge of the clock. When the clock phase is '1', it means that the next data is output on the leading edge of the clock and that data is registered as an input on the trailing edge of the clock.

Clock Polarity controls clock inversion. When clock polarity is set to '1', the clock idle state is high.

Figure 18-3. SPI Mode Timing



18.3.2 SPIM Timing

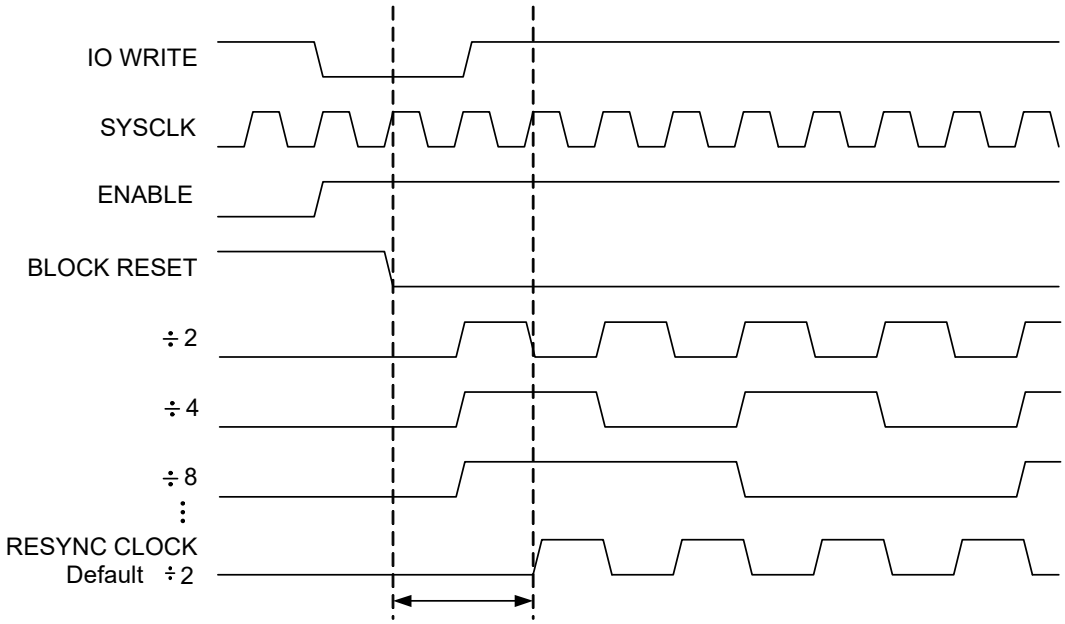
Enable/Disable Operation. As soon as the block is configured for SPIM, the primary output is the MSb or LSb of the Shift register, depending on the LSb First configuration in bit 7 of the Control register. The auxiliary output is '1' or '0', depending on the idle clock state of the SPI mode. This is the idle state.

Clock Generation. Figure 18-4 illustrates the SPIM input clocking scheme. The SYSCLK pin is an input into an eight-stage ripple divider that provides the baud rate selections. When the block is disabled, all internal state is held in a reset state.

When the Enable bit in the SPI_CR register is set, the reset is synchronously released and the clock generation is enabled. All eight taps from the ripple divider are selectable (/2, /4, /8, /16, /32, /64, /128, /256) from the Clock Sel bits in the SPI_CFG register. The selected divider tap is resynchronized to SYSCLK. The resulting clock is routed to all of the synchronous elements in the design.

When the block is disabled, the SCLK and MOSI outputs revert to their idle state. All internal state is reset (including CR0 status) to its configuration-specific reset state, except for DR0, DR1, and DR2, which are unaffected.

Figure 18-4. SPI Input Clocking



Normal Operation. Typical timing for an SPIM transfer is shown in Figure 18-5 and Figure 18-6. The user initially writes a byte to transmit when TX Reg Empty status is true. If no transmission is currently in progress, the data is loaded into the shifter and the transmission is initiated. The TX Reg Empty status is asserted again and the user is allowed to write the next byte to be transmitted to the TX Buffer register.

After the last bit is output, if TX Buffer data is available with one-half clock setup time to the next clock, a new byte transmission is initiated. An SPIM block receives a byte at the same time that it sends one. The SPI Complete or RX Reg Full can be used to determine when the input byte is received.

Figure 18-5. Typical SPIM Timing in Mode 0 and 1

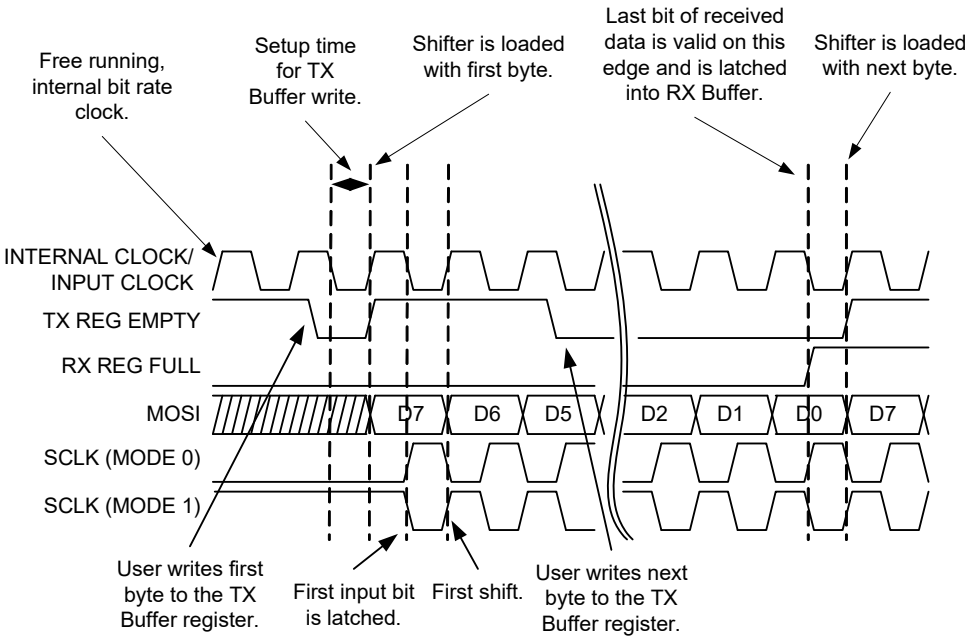
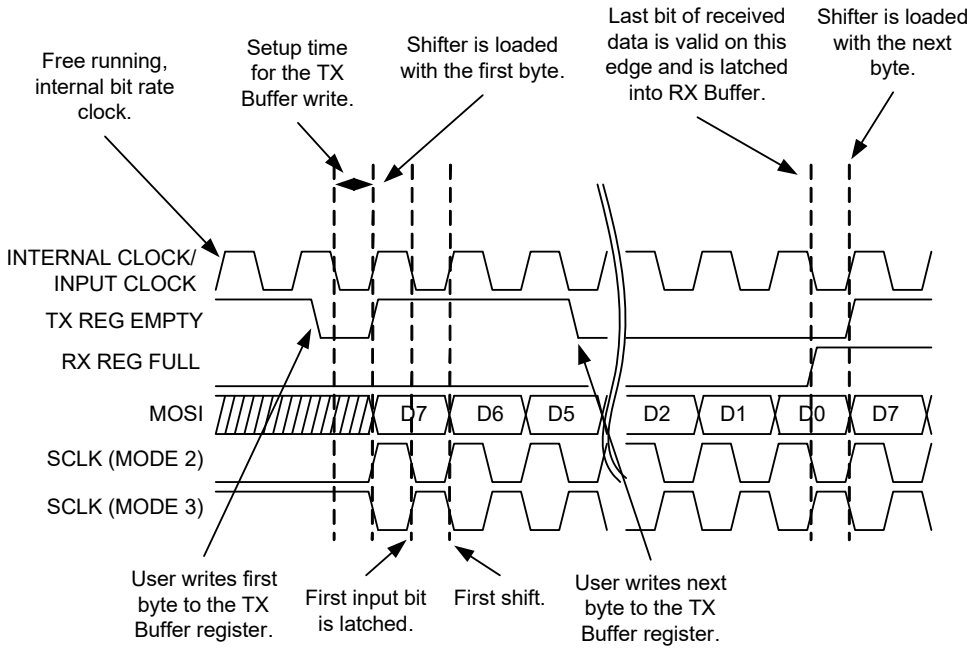


Figure 18-6. Typical SPIM Timing in Mode 2 and 3



Status Generation and Interrupts. There are four status bits in an SPI block: TX Reg Empty, RX Reg Full, SPI Complete, and Overrun.

TX Reg Empty indicates that a new byte can be written to the TX Buffer register. When the block is enabled, this status bit is immediately asserted. This status bit is cleared when the user writes a byte of data to the TX Buffer register. TX Reg Empty is a control input to the state machine and, if a transmission is not already in progress, the assertion of this control signal initiates one. This is the default SPIM block interrupt. However, an initial interrupt is not generated when the block is enabled. The user must write a byte to the TX Buffer register and that byte must be loaded into the shifter before interrupts generated from the TX Reg Empty status bit are enabled.

RX Reg Full is asserted on the edge that captures the eighth bit of receive data. This status bit is cleared when the user reads the RX Buffer register (DR2).

SPI Complete is an optional interrupt and is generated when eight bits of data and clock are sent. In modes 0 and 1, this occurs one-half cycle after RX Reg Full is set; because in these modes, data is latched on the leading edge of the clock and there is an additional one-half cycle remaining to complete that clock. In modes 2 and 3, this occurs at the same edge that the receive data is latched. This signal may be used to read the received byte or it may be used by the SPIM to disable the block after data transmission is complete.

Overrun status is set if RX Reg Full is still asserted from a previous byte when a new byte is about to be loaded into the RX Buffer register. Because the RX Buffer register is implemented as a latch, Overrun status is set one-half bit clock before RX Reg Full status.

See Figure 18-7 and Figure 18-8 for status timing relationships.

Figure 18-7. SPI Status Timing for Modes 0 and 1

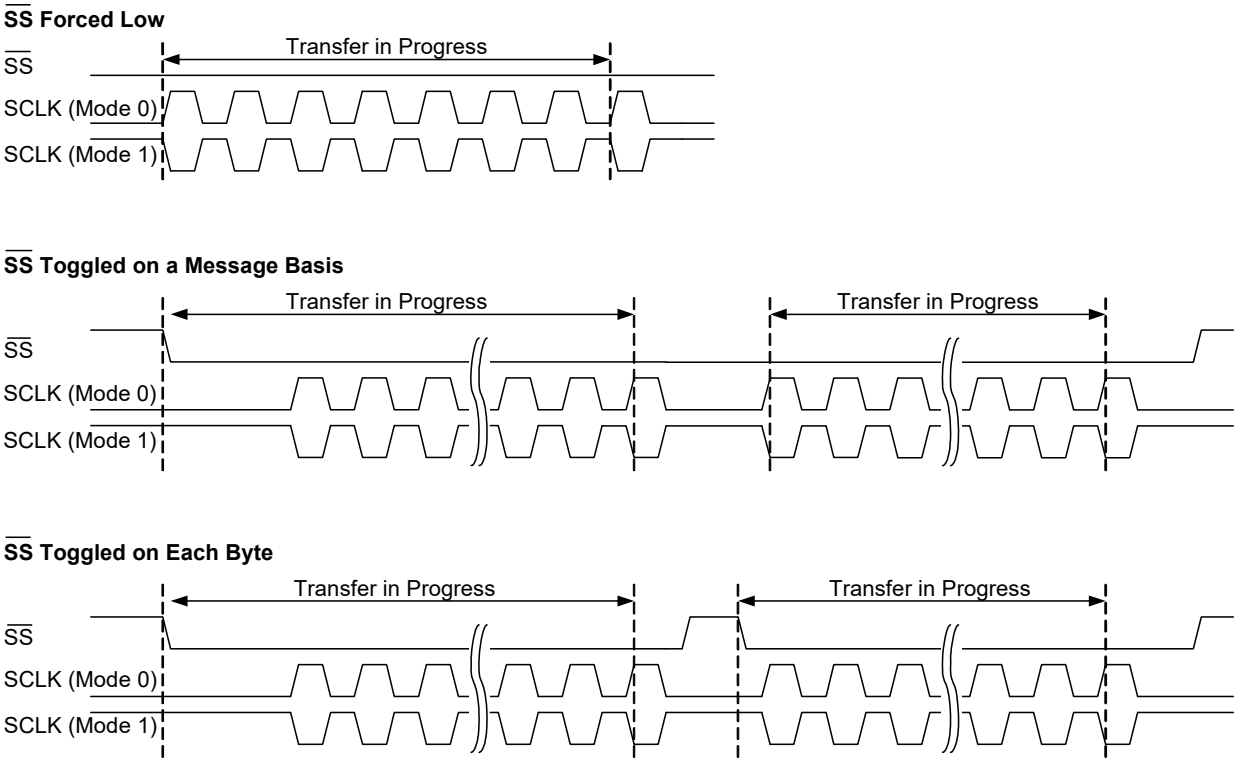
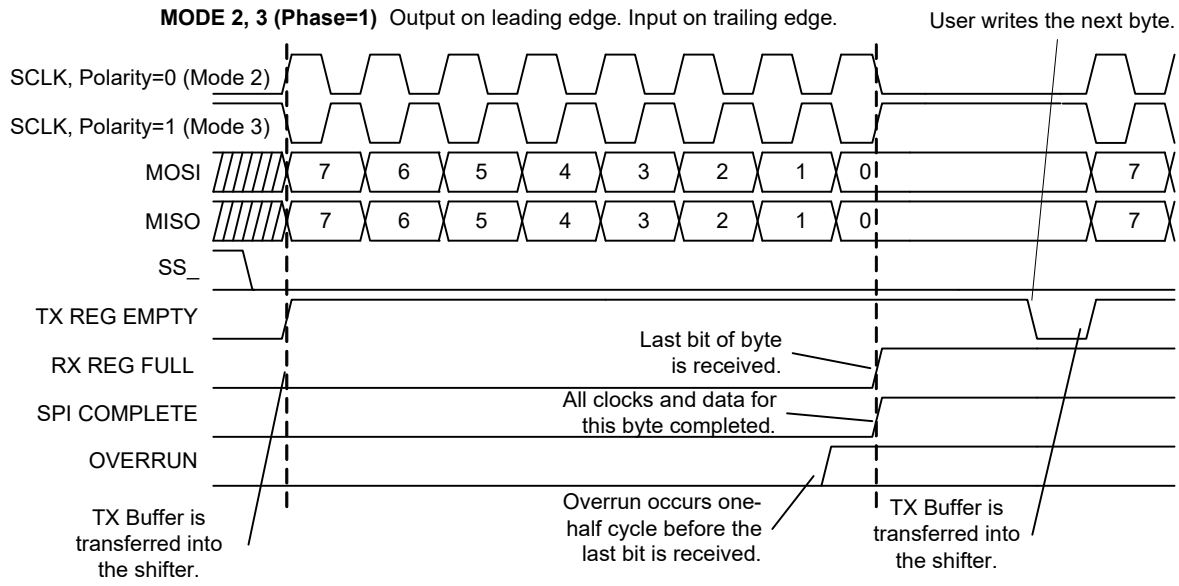


Figure 18-8. SPI Status Timing for Modes 2 and 3



18.3.3 SPIS Timing

Enable/Disable Operation. As soon as the block is configured for SPI Slave and before enabling, the MISO output is set to idle at logic 1. The Enable bit must be set and the SS_ asserted (either driven externally or forced by firmware programming) for the block to output data. When enabled, the primary output is the MSb or LSb of the Shift register, depending on the LSb First configuration in bit 7 of the Control register. The auxiliary output of the SPIS is always forced into tristate.

The SPIS has no internal clock and must be enabled with setup time to any external master supplying the clock. Setup time is also required for a TX Buffer register write before the first edge of the clock or the first falling edge of SS_, depending on the mode. This setup time must be assured through the protocol and an understanding of the timing between the master and slave in a system.

When the block is disabled, the MISO output reverts to its idle 1 state. All internal state is reset (including CR0 status) to its configuration-specific reset state, except for DR0, DR1, and DR2, which are unaffected.

Normal Operation. Typical timing for an SPIS transfer is shown in Figure 18-9 and Figure 18-10. If the SPIS is primarily being used as a receiver, the RX Reg Full (polling only) or SPI Complete (polling or interrupt) status may be used to determine when a byte is received. In this way, the SPIS operates identically with the SPIM. However, there are two main areas in which the SPIS operates differently: 1) SPIS behavior related to the SS_ signal, and 2) TX data queuing (loading the TX Buffer register).

Figure 18-9. Typical SPIS Timing in Modes 0 and 1

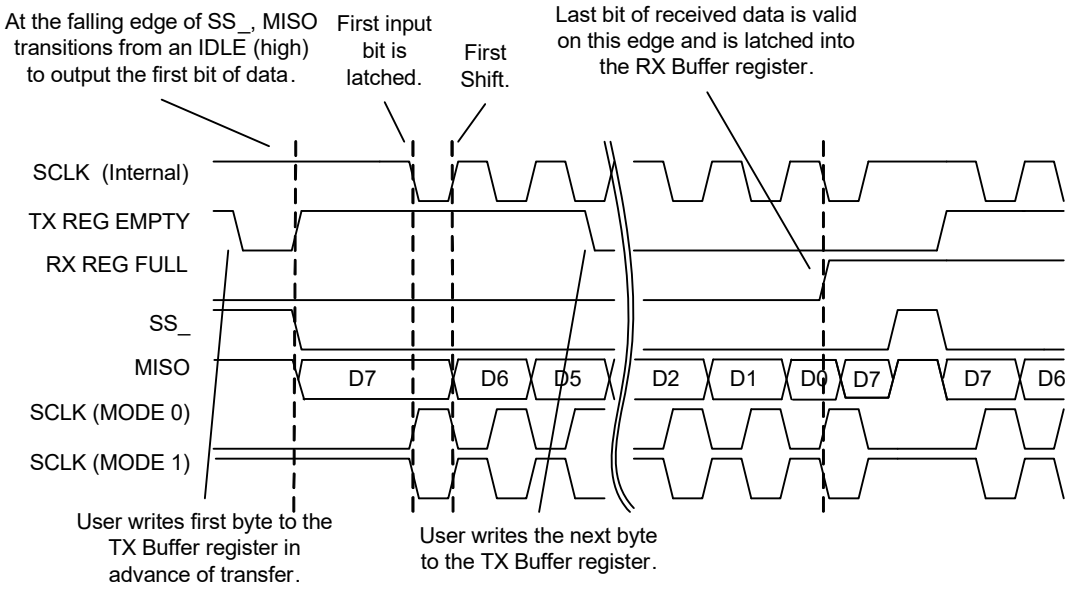
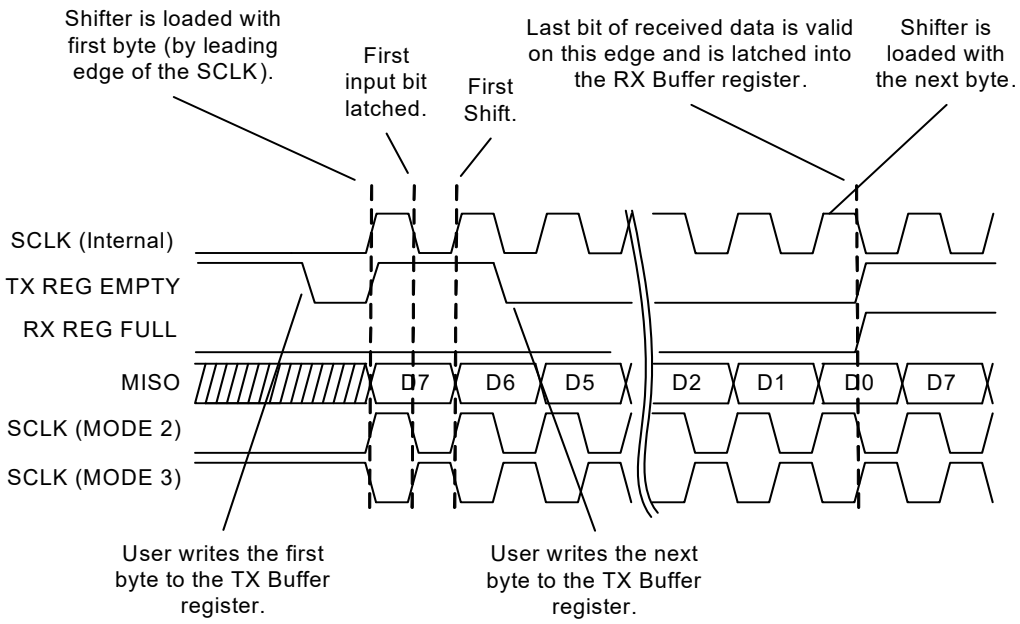


Figure 18-10. Typical SPIS Timing in Modes 2 and 3



Slave Select (SS_, Active Low). Slave Select must be asserted to enable the SPIS for receive and transmit. There are two ways to do this:

- Drive the auxiliary input from a pin (selected by the Aux I/O Select bits in the output register). This gives the SPI master control of the slave selection in a multi-slave environment.
- SS_ may be controlled in firmware with register writes to the output register. When Aux I/O Enable = 1, Aux I/O Select bit 0 becomes the SS_ input. This allows the user to save an input pin in single-slave environments.

When SS_ is negated (whether from an external or internal source), the SPIS state machine is reset and the MISO output is forced to idle at logic 1. In addition, the SPIS ignores any incoming MOSI/SCLK input from the master.

Status Generation and Interrupts. There are four status bits in the SPIS block: TX Reg Empty, RX Reg Full, SPI Complete, and Overrun. The timing of these status bits are identical to the SPIM, with the exception of TX Reg Empty, which is covered in the section on TX data queuing.

Status Clear On Read. Refer to the same subsection in SPIM Timing on page 130.

TX Data Queuing. Most SPI applications call for data to be sent back from the slave to the master. Writing firmware to accomplish this requires an understanding of how the Shift register is loaded from the TX Buffer register.

All modes use the following mechanism: 1) If there is no transfer in progress, 2) if the shifter is empty, and 3) if data is available in the TX Buffer register, the byte is loaded into the shifter.

The only difference between the modes is that the definition of “transfer in progress” is slightly different between modes 0 and 1, and modes 2 and 3.

Figure 18-11 illustrates TX data loading in modes 0 and 1. A transfer in progress is defined to be from the falling edge of SS_ to the point at which the RX Buffer register is loaded with the received byte. This means that to send a byte in the next transfer, it must be loaded into the TX Buffer register before the falling edge of SS_. This ensures a minimum setup time for the first bit, because the leading edge of the first SCLK must latch in the received data. If SS_ is not toggled between each byte or is forced low through the configuration register, the leading edge of SCLK is used to define the start of transfer. However, the user must provide the required setup time (one-half clock minimum before the leading edge) with a knowledge of system latencies and response times.

Figure 18-11. Mode 0 and 1 Transfer in Progress

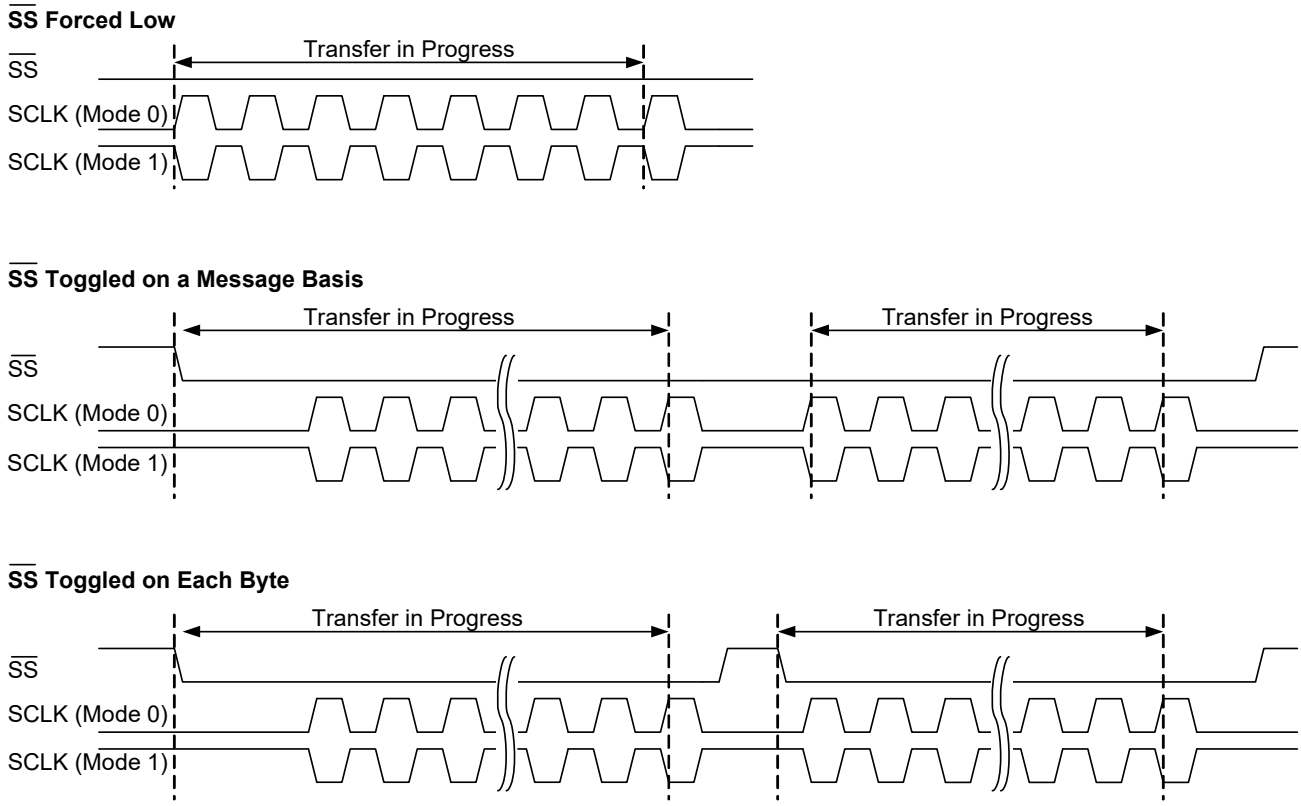
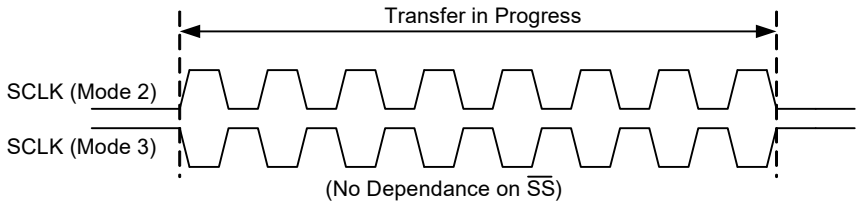


Figure 18-12 illustrates TX data loading in modes 2 and 3. In this case, a transfer in progress is defined to be from the leading edge of the first SCLK to the point at which the RX Buffer register is loaded with the received byte. Loading the shifter by the leading edge of the clock has the effect of providing the required one-half clock setup time, as the data is latched into the receiver on the trailing edge of the SCLK in these modes.

Figure 18-12. Mode 2 and 3 Transfer in Progress



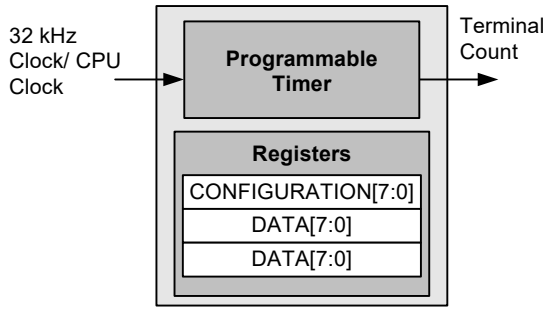
19. Programmable Timer

This chapter presents the Programmable Timer and its associated registers. For a complete table of the programmable timer registers, refer to the Summary Table of the System Resource Registers on page 93. For a quick reference of all enCoRe V registers in address order, refer to the Register Reference chapter on page 163.

19.1 Architectural Description

The device has three programmable timers (TIMER0, TIMER1, TIMER2). All three timers are individually controlled. The programmable timers are 16-bit down counters for the device. TIMER0 has a terminal count output. The timers have one configuration and two data registers associated with them. You start these timers by setting the START bit in their configuration registers (PT0_CFG, PT1_CFG, PT2_CFG). When started, the timers always start counting down from the value loaded into their data registers (PT*_DATA1, PT*_DATA0). The timers have a one-shot mode, in which the timers complete one full count cycle and stop. In one-shot mode the START bit in the configuration register is cleared after completion of one full count cycle. Setting the START bit restarts the timer.

Figure 19-1. Programmable Timer Block Diagram



19.1.1 Operation

When started, the programmable timer loads the value contained in its data registers and counts down to its terminal count of zero. The timers output an active high terminal count pulse for one clock cycle upon reaching the terminal count. The low time of the terminal count pulse is equal to the loaded decimal count value, multiplied by the clock period (TC_{pw} = COUNT $VALUE_{decimal}$ * CLK_{period}). The period of the terminal count output is the pulse width of the terminal count, plus one clock period (TC_{period} = TC_{pw} + CLK_{period}). Refer to Figure 19-2 and Figure 19-3.

Only TIMER0 outputs this terminal count output. TIMER1 and TIMER2 do not have a terminal count output.

The timers work on either the 32-kHz clock or CPU clock. This clock selection is done using the CLKSEL bits in the respective configuration registers (PT0_CFG, PT1_CFG, PT2_CFG). Make clock selections before setting the START bit so that the timing is not affected and clock frequency does not change while the timer is running.

Figure 19-2. Continuous Operation Example

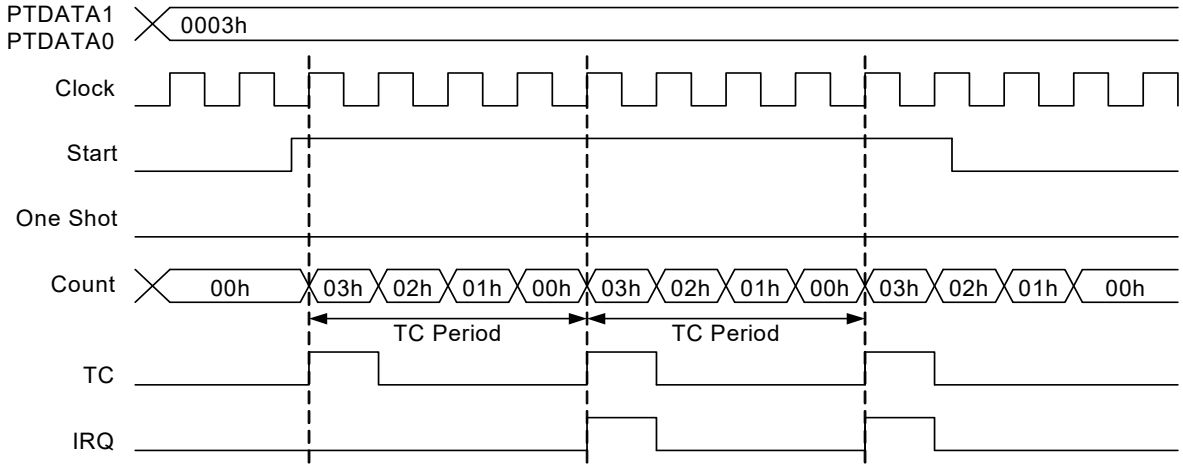
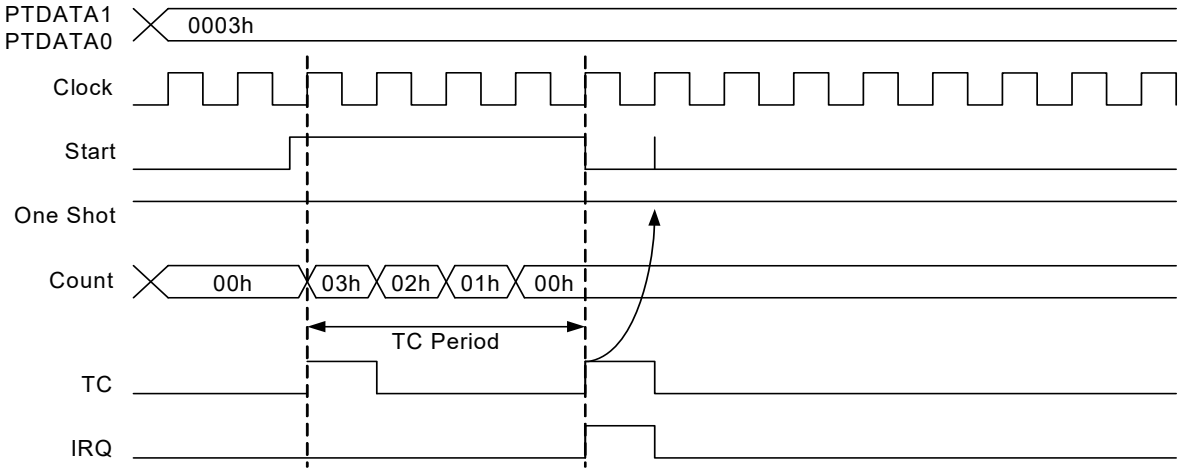


Figure 19-3. One-Shot Operation Example



19.2 Register Definitions

The following registers are associated with the Programmable Timer and are listed in address order. The register descriptions have an associated register table showing the bit structure for that register. The bits in the tables that are grayed out are reserved bits and are not detailed in the register descriptions that follow. Reserved bits must always be written with a value of '0'. For a complete table of programmable timer registers, refer to the Summary Table of the System Resource Registers on page 93.

19.2.1 PT0_CFG Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,B0h	PT0_CFG						CLKSEL	One Shot	START	RW : 0

The Programmable Timer Configuration Register (PT0_CFG) configures the enCoRe V's programmable timer.

Bit 2: CLKSEL. This bit determines if the timer runs on the 32-kHz clock or CPU clock. If the bit is set to 1, the timer runs on the CPU clock, otherwise, the timer runs on the 32-kHz clock.

Bit 1: One Shot. This bit determines if the timer runs in one-shot mode or continuous mode. In one-shot mode the timer completes one full count cycle and terminates. Upon termination, the START bit in this register is cleared. In continuous mode, the timer reloads the count value each time upon completion of its count cycle and repeats.

Bit 0: START. This bit starts the timer counting from a full count. The full count is determined by the value loaded into the data registers. This bit is cleared when the timer is running in one-shot mode upon completion of a full count cycle.

For additional information, refer to the PT0_CFG register on page 181.

19.2.2 PT1_CFG Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,B3h	PT1_CFG						CLKSEL	One Shot	START	RW : 0

The Programmable Timer Configuration Register (PT1_CFG) configures the enCoRe V's programmable timer.

Bit 2: CLKSEL. This bit determines if the timer runs on the 32-kHz clock or CPU clock. If the bit is set to '1', the timer runs on the CPU clock, otherwise, the timer runs on the 32-kHz clock.

Bit 1: One Shot. This bit determines if the timer runs in one-shot mode or continuous mode. In one-shot mode the timer completes one full count cycle and terminates. Upon termination, the START bit in this register is cleared. In continuous mode, the timer reloads the count value each time upon completion of its count cycle and repeats.

Bit 0: START. This bit starts the timer counting from a full count. The full count is determined by the value loaded into the data registers. This bit is cleared when the timer is running in one-shot mode upon completion of a full count cycle.

For additional information, refer to the PT1_CFG register on page 184.

19.2.3 PT2_CFG Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,B6h	PT2_CFG						CLKSEL	One Shot	START	RW : 0

The Programmable Timer Configuration Register (PT2_CFG) configures the enCoRe V's programmable timer.

Bit 2: CLKSEL. This bit determines if the timer runs on the 32-kHz clock or CPU clock. If the bit is set to 1, the timer runs on the CPU clock, otherwise, the timer runs on the 32-kHz clock.

Bit 1: One Shot. This bit determines if the timer runs in one-shot mode or continuous mode. In one-shot mode the timer completes one full count cycle and terminates. Upon termination, the START bit in this register is cleared. In continuous mode, the timer reloads the count value each time upon completion of its count cycle and repeats.

Bit 0: START. This bit starts the timer counting from a full count. The full count is determined by the value loaded into the data registers. This bit is cleared when the timer is running in one-shot mode upon completion of a full count cycle.

For additional information, refer to the PT2_CFG register on page 185.

19.2.4 PTx_DATA0 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,B2h	PT0_DATA0	DATA[7:0]								RW : 00
0,B5h	PT1_DATA0	DATA[7:0]								RW : 00
0,B8h	PT2_DATA0	DATA[7:0]								RW : 00

The Programmable Timer Data Register 0 (PT0_DATA0, PT1_DATA0, PT2_DATA0) holds the lower 8 bits of the programmable timer's count value.

Bits 7 to 0: DATA[7:0]. This is the lower byte of a 16-bit timer. The upper byte is in the corresponding PTxDATA1 register.

For additional information, refer to the PT0_DATA0 register on page 183, PT1_DATA0 register on page 183 and PT2_DATA0 register on page 183.

19.2.5 PTx_DATA1 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,B1h	PT0_DATA1	DATA[7:0]								RW : 00
0,B4h	PT1_DATA1	DATA[7:0]								RW : 00
0,B7h	PT2_DATA1	DATA[7:0]								RW : 00

The Programmable Timer Data Register 1 (PT0_DATA1, PT1_DATA1, PT2_DATA1) holds the 8 bits of the programmable timer's count value for the device

Bits 7 to 0: DATA[7:0]. This is the upper byte of a 16-bit timer. The lower byte is in the corresponding PTx_DATA0 register.

For additional information, refer to the PT0_DATA1 register on page 182, PT1_DATA1 register on page 182, and PT2_DATA1 register on page 182.

20. Full-Speed USB

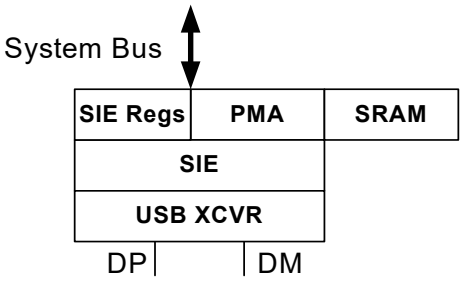
This chapter explains the Full-Speed USB (Universal Serial Bus) resource and its associated registers. For a quick reference of all enCoRe V registers in address order, refer to the Register Reference chapter on page 163. USB is only available on certain devices.

20.1 Architectural Description

The enCoRe V USB system resource adheres to the USB 2.0 Specification for full-speed devices operating at 12 Mbps with one upstream port and one USB address. enCoRe V USB consists of these components:

- Serial Interface Engine (SIE) block
- PSoC Memory Arbiter (PMA) block
- 512 bytes of dedicated SRAM
- A Full-Speed USB Transceiver with internal regulator and two dedicated USB pins

Figure 20-1. USB Block Diagram



At the enCoRe V system level, the full-speed USB system resource interfaces to the rest of the enCoRe V by way of the M8C's register access instructions and to the outside world by way of the two USB pins. The SIE supports nine endpoints including a bidirectional control endpoint (endpoint 0) and eight uni-directional data endpoints (endpoints 1 to 8). The uni-directional data endpoints are individually configurable as either IN or OUT.

20.2 Application Description

The individual components and issues of the USB system are described in detail in the following sections.

20.2.1 USB SIE

The USB Serial Interface Engine (SIE) allows the enCoRe V device to communicate with the USB host at full-speed data rates (12 Mbps). The SIE simplifies the interface to USB traffic by automatically handling the following USB processing tasks without firmware intervention:

- Translates the encoded received data and formats the data to be transmitted on the bus.
- Generates and checks CRCs. Incoming packets failing checksum verification are ignored.
- Checks addresses. Ignores all transactions not addressed to the device.
- Sends appropriate ACK/NAK/Stall handshakes.
- Identifies token type (SETUP, IN, OUT) and sets the appropriate token bit when a valid token in received.
- Identifies Start-of-Frame (SOF) and saves the frame count.
- Sends data to or retrieves data from the USB SRAM, by way of the PSoC Memory Arbiter (PMA).

Firmware is required to handle various parts of the USB interface. The SIE issues interrupts after key USB events to direct firmware to appropriate tasks:

- Fill and empty the USB data buffers in USB SRAM.
- Enable PMA channels appropriately.
- Coordinate enumeration by decoding USB device requests.
- Suspend and resume coordination.
- Verify and select data toggle values.

Table 20-1. Mode Encoding for Control and Non-Control Endpoints

Mode	Encoding	SETUP	IN	OUT	Comments
Disable	0000	Ignore	Ignore	Ignore	Ignore all USB traffic to this endpoint.
NAK IN/OUT	0001	Accept	NAK	NAK	NAK IN and OUT token.
Status OUT Only	0010	Accept	STALL	Check	For control endpoint, STALL IN and ACK zero byte OUT.
STALL IN/OUT	0011	Accept	STALL	STALL	For control endpoint, STALL IN and OUT token.
Reserved	0100	Ignore	Ignore	Ignore	
ISO OUT	0101	Ignore	Ignore	Always	Isochronous OUT.
Status IN Only	0110	Accept	TX 0 Byte	STALL	For control endpoint, STALL OUT and send zero byte data for IN token.
ISO IN	0111	Ignore	TX Count	Ignore	Isochronous IN.
NAK OUT	1000	Ignore	Ignore	NAK	Send NAK handshake to OUT token.
ACK OUT (STALL = 0)	1001	Ignore	Ignore	ACK	This mode is changed by the SIE to mode 1000 on issuance of ACK handshake to an OUT.
ACK OUT (STALL = 1)	1001	Ignore	Ignore	STALL	STALL the OUT transfer.
Reserved	1010	Ignore	Ignore	Ignore	
ACK OUT – STATUS IN	1011	Accept	TX0 Byte	ACK	ACK the OUT token or send zero byte data for IN token.
NAK IN	1100	Ignore	NAK	Ignore	Send NAK handshake for IN token.
ACK IN (STALL = 0)	1101	Ignore	TX Count	Ignore	This mode is changed by the SIE to mode 1100 after receiving ACK handshake to an IN data.
ACK IN (STALL = 1)	1101	Ignore	STALL	Ignore	STALL the IN transfer.
Reserved	1110	Ignore	Ignore	Ignore	
ACK IN – Status OUT	1111	Accept	TX Count	Check	Respond to IN data or Status OUT.

20.2.2 USB SRAM

The enCoRe V USB System Resource contains dedicated 512 bytes SRAM. This SRAM is identical to two SRAM pages used in the enCoRe V core; however, it is not accessible by way of the M8C memory access instructions. The USB's dedicated SRAM may only be accessed by way of the PMA registers. For more information on how to use the USB's dedicated SRAM, see the next section, PSoC Memory Arbiter (PMA).

The USB SRAM contents are not directly affected by any reset, but must be treated as unknown after any POR, WDR, and XRES.

20.2.2.1 PSoC Memory Arbiter

The PSoC Memory Arbiter (PMA) is the interface between the USB's dedicated SRAM and the two blocks that access the SRAM: the M8C and the USB SIE. The PMA provides 16 channels to manage data with four Endpoints/8 channels mapped to Page 0 of USB dedicated SRAM and other four Endpoints/8 channels mapped to Page 1 of USB dedicated SRAM. All the channel registers may be used by the M8C, but the eight non-control USB endpoints are each allocated to a specific set of PMA channel registers. It is the responsibility of the firmware to ensure that the M8C is not accessing a set of channel registers that are in use by the USB SIE.

If the M8C wants to access the same data that an SIE channel is using, two channels must be configured to access the same SRAM address ranges. Table 20-2 shows the mapping between PMA channels and the blocks that can use them.

Table 20-2. PMA Channel Assignments

PMA Channel	USB SIE	M8C	Channel Registers (PMAx_xx)
0		Page 0	PMA0_DR, PMA0_RA, PMA0_WA
1	EP1	Page 0	PMA1_DR, PMA1_RA, PMA1_WA
2	EP2	Page 0	PMA2_DR, PMA2_RA, PMA2_WA
3	EP3	Page 0	PMA3_DR, PMA3_RA, PMA3_WA
4	EP4	Page 0	PMA4_DR, PMA4_RA, PMA4_WA
5		Page 0	PMA5_DR, PMA5_RA, PMA5_WA
6		Page 0	PMA6_DR, PMA6_RA, PMA6_WA
7		Page 0	PMA7_DR, PMA7_RA, PMA7_WA
8		Page 1	PMA8_DR, PMA8_RA, PMA8_WA
9	EP5	Page 1	PMA9_DR, PMA9_RA, PMA9_WA
10	EP6	Page 1	PMA10_DR, PMA10_RA, PMA10_WA
11	EP7	Page 1	PMA11_DR, PMA11_RA, PMA11_WA
12	EP8	Page 1	PMA12_DR, PMA12_RA, PMA12_WA
13		Page 1	PMA13_DR, PMA13_RA, PMA13_WA
14		Page 1	PMA14_DR, PMA14_RA, PMA14_WA
15		Page 1	PMA15_DR, PMA15_RA, PMA15_WA

The PMA's purpose is to manage the potentially conflicting SRAM access requests from the M8C and the USB SIE. From a performance standpoint, the PMA guarantees that a continuous stream of move instructions (see ahead), are serviced by the PMA without delay even while the USB SIE is transferring data at its maximum rate in to or out of the dedicated USB SRAM.

When servicing a request, the PMA is in one of two addressing modes. For M8C access the PMA always uses Post-Increment Addressing. After a read or write request is made to the channel's PMAx_DR register, the PMA automatically increments the pointer into SRAM. For a read access, the next value is also automatically prefetched. For USB SIE accesses, the PMA uses an offset addressing mode. In this mode, the channel's base address, as stored in the PMAx_WA and PMAx_RA registers, is added to the byte count value provided by the USB SIE.

A PMA channel does not have a defined upper limit. It is the responsibility of the firmware to ensure that channels do not access memory outside of the range defined by the application.

During SIE writes to the USB SRAM, the maximum number of bytes written is limited to the count value in the respective endpoint's count registers. This value must be loaded by firmware before data is received.

The rest of the description of the PMA is broken into two parts: the M8C interface and the USB SIE interface, which are described as follows.

PMA to M8C Interface

The M8C accesses the PMA, and thus the USB's dedicated SRAM, by way of a register interface. Each PMA channel has three registers associated with it, as shown in Table 20-2 on page 142. Only the following basic M8C register access instructions may be used with these registers.

```
MOV A, reg[expr]
MOV A, reg[X+expr]
MOV [expr], [expr]
MOV reg[expr], A
MOV reg[X+expr], A
MOV reg[expr], expr
MOV reg[X+expr], expr
```

When the M8C uses a PMA channel to write data into SRAM, follow these steps:

1. Choose a PMA channel that is not allocated to a USB endpoint, or choose a channel where the endpoint is inactive.
2. Write the channel's PMAx_WA register with the first address in SRAM that must be used by this channel.
3. Write data to the channel's PMAx_DR register. The PMA logic automatically increments the PMAx_WA address after each write.

While these steps are executed by the M8C, the USB SIE may be fully active on any other PMA channel.

The M8C may also service another channel and come back to the channel being serviced by the previous steps. To determine the next address that is used when data is written to the channel's PMAx_DR register, the PMAx_WA register may be read.

When the M8C uses a PMA channel to read data from SRAM, follow these steps:

1. Choose a PMA channel that is not allocated to a USB endpoint, or choose a channel where the endpoint is inactive.
2. Write the channel's PMAx_RA register with the first address in SRAM that must be read by this channel.
3. Read data from the channel's PMAx_DR register. The PMA logic automatically increments the PMAx_RA address after each read.

When data is read from a PMA channel the data is prefetched; therefore, the channel must be pre-loaded prior to the first M8C read that expects to get actual data. This pre-loading is taken care of automatically when the PMAx_RA register is written. This pre-loading mechanism is actually the only difference between the PMAx_RA and PMAx_WA registers.

PMA to USB SIE Interface

The USB SIE accesses the PMA, and thus the dedicated USB SRAM, by way of a private interface and does not affect the enCoRe V Core address or data bus. The only area of contention that is not automatically arbitrated between the M8C, PMA, and USB SIE are the PMAx_xx registers. When the USB SIE is actively using a PMA channel, the M8C must not attempt to access that channel's PMA registers. If the M8C wants to access the same data as an active USB endpoint, the M8C must use a PMA channel separate from the PMA channel that is permanently allocated to that endpoint.

Just as the M8C has two uses for PMA channels, read or write, the USB SIE has two uses for a PMA channel. The USB SIEs use of a channel may be thought of as read or write; but, in USB terms the USB SIEs need to read data is associated with an IN transaction and the need to write data with an OUT transaction.

For a USB IN transaction, the USB SIE is reading data from the PMA and sending the data to the USB host. The following steps must be used to set up a PMA channel for a USB IN transaction. These steps assume that the data has already been written by the M8C into the dedicated USB SRAM.

1. Select the PMA channel whose number matches the endpoint number that is handling the IN transaction.
2. Write the PMA channel's PMAx_RA register with the address of the first byte in SRAM that is used for the IN transaction.
3. Configure the USB endpoint registers with the proper byte count and enable the endpoint to send data when the IN transaction occurs.

Because the PMA prefetches data for M8C and USB SIE reads, step two above is very important. This step not only sets the first address from which data is read by the USB SIE; but, it also triggers a read operation on the dedicated USB SRAM and stores the result of that read in the PMAx_DR ready for the USB SIE to read. When the USB SIE begins the IN transaction for the endpoint, it uses its byte counter to tell the PMA which byte is needed next. Therefore, when the first byte of the transaction is read by the SIE, the PMA automatically fetches the next byte in preparation for the USB SIEs next byte request.

For a USB OUT transaction, the USB SIE is writing data to the PMA that was received from the USB host. The following steps must be used to set up a PMA channel for a USB OUT transaction.

1. Select the PMA channel whose number matches the endpoint number that is handling the OUT transaction.
2. Write the PMA channel's PMAx_WA register with the address of the first byte in SRAM that is used for the OUT transaction.
3. Configure the USB endpoint with the proper maximum receive byte count and enable the endpoint to receive the OUT transaction.

As with the IN transaction, the PMA uses the byte counter from the SIE as an offset to the value of the PMAx_WA register. As the USB SIE sends bytes to the PMA, the counter is added to the base address and the data byte is written into the dedicated USB SRAM. Should an error occur in the OUT transaction and the packet be resent by the USB host, the byte count is reset to zero and the PMA writes the new data over top of the potentially corrupt data from the previous failed transaction.

If the number of bytes received exceeds the count in the endpoint's count register, the extra bytes are not written into the USB SRAM, but the received byte count reported in the endpoint count registers includes the ignored bytes.

20.2.3 Oscillator Lock

The enCoRe V device can operate without using any external components, such as a crystal, and still achieve the clock accuracy required for full-speed USB. It does this by locking its internal oscillator to the incoming USB traffic. Therefore, the initial accuracy of the oscillator may not meet the required accuracy (± 0.25%), but it self-tunes to this precision before the device needs to transmit USB data.

This oscillator locking feature is disabled by default and must be enabled by firmware. In USB systems, this feature must always be enabled unless the device is being used with an accurate external clock. The EnableLock bit in the USB_CR1 register is used to turn on the locking feature.

20.2.4 Transceiver

The internal USB transceiver interfaces to the external USB bus to transmit and receive signals according to the USB 2.0 Specification. In normal USB operation, the transceiver interfaces directly to the SIE and no user interaction is needed after initialization. The USB Enable bit in the USB_CR0 register must be set to enable the transceiver for USB operation. The I/O Mode bit in USBIO_CR1 must not be set during normal USB mode of operation. The transceiver can also be used in non-USB modes, because the D+ and D– pins can be read and written through register control bits. This enables multi-purpose use of these pins (for example, in a system that supports both USB and PS/2 signaling). Clearing the USB Enable bit of USB_CR0 powers down the USB differential receiver and disables USB communication.

For USB operation, the transceiver contains an internal 1.5-kΩ pull-up resistor on the D+ line. This resistor is isolated from the D+ pin at reset and is attached under firmware control through the USBPUEN bit in the USBIO_CR1 register. After the D+ pull-up resistor is connected to the D+ line, the system normally detects that as an attach and begins the USB enumeration process. No additional external pull-up resistor must be added to the D+ line, because the transceiver signaling is optimized for use with the internal D+ pull-up resistor. However, low-value series resistors (22 Ω) must be added externally to meet the driving impedance requirement for full-speed USB.

The transceiver also includes 5-kΩ pull-up resistors on both the D+ and D– pins for communication at PS/2 or similar signaling levels. These resistors are disconnected at reset and can be connected with the PS2PUEN bit in the USBIO_CR1 register. The D+ and D– pins can also be driven individually high and low in both USB and non-USB modes. The state of those pins can be read in any mode. Refer to the description of the USBIO_CR0 and USBIO_CR1 registers for more detail.

20.2.5 USB Suspend

Loss of USB activity, while the USB VBus is still asserted, indicates that the device must enter USB Suspend mode. (Self-powered devices do not need to go into Suspend mode.) This condition is detected by monitoring the Bus Activity bit in the USB_CR1 register. This bit must be polled periodically. If it reads high (bus activity present), it must be cleared by firmware. If no activity is detected for the desired time (for example, 3 ms), then the device must enter Suspend mode.

Not all sleep modes preserve the USB configuration register states during sleep. The Standby I2C-USB mode is the preferred sleep mode for USB operation because the state of all USB registers is maintained during sleep. The other sleep modes do not preserve all of these registers to save sleep power.

The USB regulator settings must not be changed when entering sleep state, because the regulator automatically enters a low-power state for the given regulator mode (pass-through or regulating).

20.2.5.1 Using Standby I2C-USB Sleep Mode for USB Suspend

To enter Standby I2C-USB mode, firmware powers down the desired functions, as it would to enter a Standby I2C-USB mode sleep state as documented in 11.1.1 Sleep Control Implementation Logic on page 79, including writing the SLEEP bit of the CPU_SCR0 register. In this mode, USB configuration registers and data are preserved during sleep.

20.2.5.2 Using Standby or Deep Sleep Modes for USB Suspend

The Standby and Deep Sleep modes do not have hardware supported for USB suspend operation because not all USB registers are preserved in these modes. In USB parts, if there is a need to enter Standby or Deep Sleep mode, then user firmware must save the non-retained registers to SRAM before entering into sleep and restore these registers after the device wakes up. Some configuration registers retain their values in all sleep modes: USB_CR0, USB_CR1, USBIO_CR0, USBIO_CR1, IMO_TR, IMO_TR1. In addition, the USB SRAM contents are preserved in all sleep modes.

The USB registers that retain state in Standby I2C-USB mode but not in other sleep modes are: the endpoint control registers (EPx_CRx), endpoint PMA write address (PMAx_WA)/read address (PMAx_RA) registers, PMA data registers (PMAx_DR), endpoint count registers (EPx_CNTx), endpoint 0 data register (EP0_DRx) and start of frame registers (USB_SOFx). These registers are reset after the device comes out of sleep.

An alternative is to simply disconnect from the USB bus before going into one of these sleep modes, and then re-connect to re-initialize the USB system after waking up.

20.2.5.3 Wakeup from Suspend

The USB wake interrupt must be enabled to allow the device to exit the sleep state when there is activity on the USB bus. This interrupt can be enabled at any time because it only asserts when the device is in the sleep state. Other interrupts may be optionally enabled, such as the sleep interrupt, GPIO, I2C, to periodically wake the device while in USB suspend state. The USB wake interrupt can wake up the device from all the three sleep states (standby sleep, I2C-USB sleep and deep sleep). If D+ is low when the SLEEP bit is being set, the device briefly enters sleep state, and then exits sleep due to the USB wake interrupt.

By carefully using a sleep timer interrupt, the device can wake periodically, monitor the environment, and return to sleep while maintaining a low average current that meets the USB suspend current specification.

If the device needs to issue a resume signal to the USB system, firmware can write to the TEN and TD bits in the USBIO_CR0 register to manually force a K state on the bus. Using these bits produces signaling that meets the USB timing specifications.

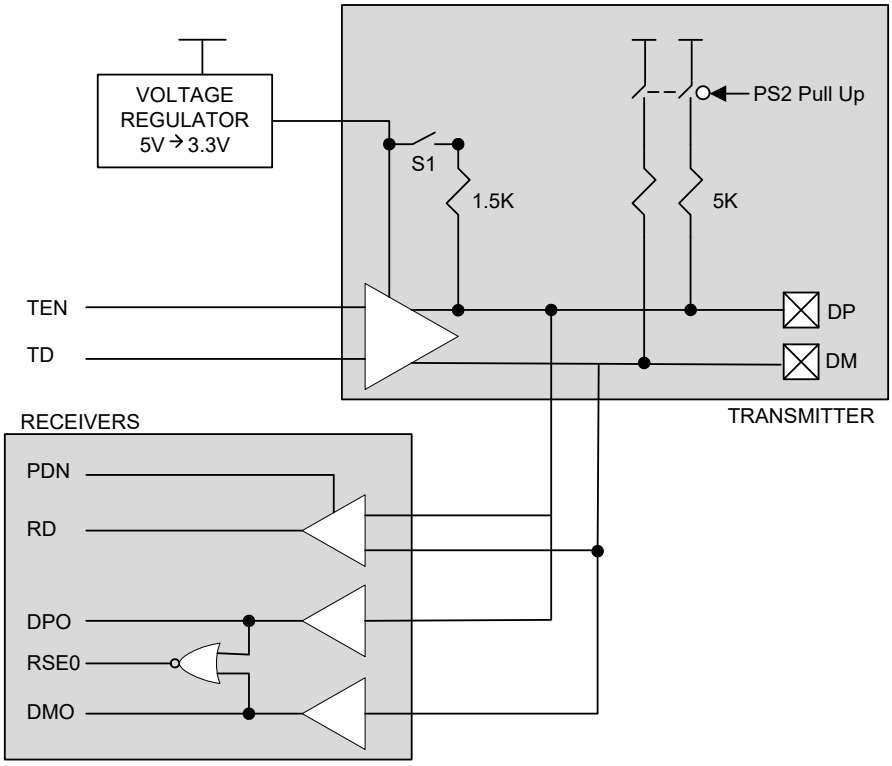
When driving a resume, the J state (TD=1) must be driven briefly before driving the K state (TD=0). The steps are summarized as follows:

1. Drive the J state (TEN=1, TD=1) for one instruction.
2. Drive the resume, or K state (TEN=1, TD=0) for the proper time (1 ms to 15 ms).
3. Stop driving the USB bus manually (TEN=0).

20.2.6 Regulator

The transceiver contains a built-in regulator that can be used to power the transceiver from the USB bus voltage or other supply around 5 V. The regulator supplies the proper levels for USB signals, which switch between 0 V and 3.3 V nominally. If the enCoRe V device is operating with a Vdd supply near 3.3 V, then the regulator must be placed into a pass-through mode so that the Vdd voltage is directly supplied to the transceiver, without regulation. The RegEnable bit (bit 0 in the USB_CR1 register) is used to pick between the regulating mode (5-V supply) or the passthrough mode (3.3-V supply). At power-up, the regulator is automatically held in pass-through mode, but the USB transceiver pins are tristated.

Figure 20-2. Transceiver and Regulator Block Diagram



20.3 Register Definitions

The following registers are related to Full-Speed USB in the enCoRe V device. For a complete table of the Full-Speed USB registers, refer to the Registers table Summary Table of the System Resource Registers on page 93. Register bits that are grayed out in this document are reserved bits and are not detailed in the register descriptions that follow. Always write reserved bits with a value of '0'.

20.3.1 USB_SOF0 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,31h	USB_SOF0	Frame Number[7:0]								R : 00
0,32h	USB_SOF1						Frame Number[10:8]			R : 0

The USB Start of Frame Registers (USB_SOF0 and USB_SOF1) provide access to the 11-bit SOF frame number. Start of frame packets are sent from the host (for example, the PC) every 1 ms. For more information, see the *Universal Serial Bus Specification, revision 2.0*.

Bits 7 to 0: Frame Number. The USB_SOF0 register has the lower 8 bits [7:0] and the USB_SOF1 register has the upper 3 bits [10:8] of the SOF frame number.

For additional information, refer to the USB_SOF0 register on page 169 and the USB_SOF1 register on page 170.

20.3.2 USB_CR0 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,33h	USB_CR0	USB Enable	Device Address[6:0]							RW : 00

The USB Control Register 0 (USB_CR0) is used to set the enCoRe V's USB address and enable the USB system resource. The USB_MISC_CR register on page 220 must be set correctly for the bits in this register to function as described here.

All bits in this register are reset to zero when a USB bus reset interrupt occurs.

Note Set the IMO frequency to 24 MHz and enable the 48-MHz clock in the OSC_CR2 register before USB is enabled. See IMO_TR and CPU_SCR1 registers for selecting IMO frequency as 24 MHz.

Bit 7: USB Enable. This bit enables the enCoRe V device to respond to USB traffic. '0' is USB disabled. The device does not respond to USB traffic. '1' is USB enabled.

Bits 6 to 0: Device Address[6:0]. These bits specify the USB device address to which the SIE responds. This address must be set by firmware and is specified by the USB host with a SET ADDRESS command during USB enumeration. This value must be programmed by firmware when assigned during enumeration. It is not set automatically by the hardware.

For additional information, refer to the USB_CR0 register on page 171.

20.3.3 USBIO_CR0 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,34h	USBIO_CR0	TEN	TSE0	TD					RD	# : 0

The USB I/O Control Register 0 (USBIO_CR0) is used for manually transmitting on the USB D+ and D– pins, or reading the differential receiver.

Bit 7: TEN. This is used to manually transmit on the D+ and D– pins. Normally, this bit must be cleared to allow the internal SIE to drive the pins. The most common reason for manually transmitting is to force a resume state on the bus. '0' is manual transmission off (TSE0 and TD have no effect). '1' is manual transmission enabled (TSE0 and TD determine the state of the D+ and D– pins).

Bit 6: TSE0. Transmit Single-Ended Zero. SE0: both D+ and D– low. No effect if TEN=0. '0' is do not force SE0. '1' is force SE0 on D+ and D–.

Bit 5: TD. Transmit a USB J or K state on the USB bus. No effect if TEN=0 or TSE0=1. '0' forces USB K state (D+ is low, D– is high). '1' forces USB J state (D+ is high, D– is low).

Bit 0: RD. This read only bit gives the state of the USB differential receiver. '0' is D+ < D– or D+ = D– = 0. '1' is D+ > D–.

For additional information, refer to the USBIO_CR0 register on page 172.

20.3.4 USBIO_CR1 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,35h	USBIO_CR1	IOMode	Drive Mode	DPI	DMI	PS2PUEN	USBPUEN	DPO	DMO	# : 03

The USB I/O Control Register 1 (USBIO_CR1) is used to manually read or write the D+ and D– pins, and to configure internal pull-up resistors on those pins.

Bit 7: IOMode. This bit allows the D+ and D– pins to be configured for either USB mode or bit banged modes. If this bit is set, the DMI and DPI bits are used to drive the D– and D+ pins. '0' is USB mode. Drive mode has no effect. '1' is drive mode, DMI and DPI determine state of the D+ and D– pins.

Bit 6: Drive Mode. If the IOMode bit is set, this bit configures the D– and D+ pins for either CMOS drive or open drain drive. If IOMode is cleared, this bit has no effect. Note that in open drain mode, 5-kΩ pull-up resistors can be connected internally with the PS2PUEN bit. '0' is D+ and D– are in open drain mode. If the DPI or DMI bits are set high, the corresponding D+ or D– pad is high-impedance. '1' is D+ and D– are in CMOS drive mode. D+ follows DPI and D– follows DMI.

Bit 5: DPI. This bit is used to drive the D+ pin if IOMode=1. Refer to the Drive Mode bit for drive state of pad. '0' is drive D+ pad low. '1' is drive D+ pad high (unless Drive Mode=0).

Bit 4: DMI. This bit is used to drive the D– pin if IOMode=1. Refer to the Drive Mode bit for drive state of pad. '0' is drive D– pad low. '1' is drive D– pad high (unless Drive Mode=0).

Bit 3: PS2PUEN. This bit controls the connection of the two internal 5-kΩ pull-up resistors to the D+ and D– pins. '0' is no effect. '1' is apply 5k pull ups between Vdd and both D+ and D– pads, independent of the IOMode and Drive Mode bits.

Bit 2: USBPUEN. This bit controls the connection of the internal 1.5-kΩ pull-up resistor on the D+ pin. '0' is no effect. '1' is apply internal USB pull-up resistor to D+ pad.

Bit 1: DPO. This read-only bit gives the state of the D+ pin.

Bit 0: DMO. This read-only bit gives the state of the D– pin.

For additional information, refer to the USBIO_CR1 register on page 173.

20.3.5 EP0_CR Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,36h	EP0_CR	Setup Received	IN Received	OUT Received	ACKed Transaction	Mode[3:0]				# : 00

The Endpoint Control Register (EP0_CR) is used to configure endpoint 0.

Because both firmware and the SIE are allowed to write to the Endpoint 0 Control and Count registers, the SIE provides an interlocking mechanism to prevent accidental overwriting of data. When the SIE writes to these registers they are locked and the processor cannot write to them until after reading the EP0_CR register. Writing to this register clears the upper four bits regardless of the value written.

Note The register lock is removed when the following M8C instructions are used to write the EP0_CR register as these instructions generate a read (IOR_) signal, which causes the lock to be removed even without the firmware actually reading the EP0_CR register.

```
mov reg[expr], expr
mov reg[X+expr], expr
```

Bit 7: Setup Received. When set, this bit indicates a valid setup packet was received and ACKed. This bit is forced high from the start of the data packet phase of the setup transaction, until the start of the ACK packet returned by the SIE. The CPU is prevented from clearing this bit during this interval. After this interval, the bit remains set until cleared by firmware. While this bit is set to '1', the CPU cannot write to the EP0_DRx registers. This prevents firmware from overwriting an incoming setup transaction before firmware has a chance to read the setup data. This bit is cleared by any non-locked writes to the register.

Bit 6: IN Received. When set, this bit indicates a valid IN packet has been received. This bit is updated to '1' after the host acknowledges an IN data packet. When clear, this bit indicates either no IN has been received or that the host did not acknowledge the IN data by sending an ACK handshake. It is cleared by any non-locked writes to the register.

Bit 5: OUT Received. When set, this bit indicates a valid OUT packet has been received and ACKed. This bit is updated to '1' after the last received packet in an OUT transaction. When clear, this bit indicates no OUT has been received. It is cleared by any non-locked writes to the register.

Bit 4: ACKed Transaction. This bit is set whenever the SIE engages in a transaction to the register's endpoint that completes with an ACK packet. This bit is cleared by any non-locked writes to the register.

Bits 3 to 0: Mode[3:0]. The mode bits control how the USB SIE responds to traffic and how the USB SIE changes the mode of that endpoint as a result of host packets to the endpoint.

For additional information, refer to the EP0_CR register on page 174.

20.3.6 EP0_CNT Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,37h	EP0_CNT	Data Toggle	Data Valid			Byte Count[3:0]				# : 00

The Endpoint 0 Count Register (EP0_CNT) is used to configure endpoint 0.

Whenever the count updates from a setup or OUT transaction, this register locks and cannot be written by the CPU. Reading the EP0_CR register unlocks this register. This prevents firmware from overwriting a status update on incoming setup or OUT transactions, before firmware has a chance to read the data.

Bit 7: Data Toggle. This bit selects the data packet's toggle state. For IN transactions, firmware must set this bit. For OUT or setup transactions, the SIE hardware sets this bit to the state of the received Data Toggle bit. ‘0’ is DATA0. ‘1’ is DATA1.

Bit 6: Data Valid. This bit indicates whether there were errors in OUT or setup transactions. It is cleared to '0' if CRC, bit stuff, or PID errors have occurred. This bit does not update for some endpoint mode settings. This bit may be cleared by writing a zero to it when the register is not locked. ‘0’ is error in data received. ‘1’ is no error.

Bits 3 to 0: Byte Count[3:0]. These bits indicate the number of data bytes in a transaction. For IN transactions, firmware loads the count with the number of bytes to be transmitted to the host from the endpoint FIFO. Valid values are 0 to 8. For OUT or setup transactions, the count is updated by hardware to the number of data bytes received, plus two for the CRC bytes. Valid values are 2 to 10.

For additional information, refer to the EP0_CNT register on page 175.

20.3.7 EP0_DRx Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,38h	EP0_DR0	Data Byte[7:0]								RW : 00
0,39h	EP0_DR1	Data Byte[7:0]								RW : 00
0,3Ah	EP0_DR2	Data Byte[7:0]								RW : 00
0,3Bh	EP0_DR3	Data Byte[7:0]								RW : 00
0,3Ch	EP0_DR4	Data Byte[7:0]								RW : 00
0,3Dh	EP0_DR5	Data Byte[7:0]								RW : 00
0,3Eh	EP0_DR6	Data Byte[7:0]								RW : 00
0,3Fh	EP0_DR7	Data Byte[7:0]								RW : 00

The Endpoint 0 Data Register (EP0_DRx) is used to read and write data to the USB control endpoint.

The EP0_DRx registers have a hardware-locking feature that prevents the CPU write when setup is active. The registers are locked as soon as the setup token is decoded and remain locked throughout the setup transaction and until the EP0_CR register has been read. This is to prevent overwriting new setup data before firmware knows it has arrived.

All other endpoint data buffers do not have this locking feature.

Bits 7 to 0: Data Byte[7:0]. These registers are shared for both transmit and receive. The count in the EP0_CNT register determines the number of bytes received or to be transferred.

For additional information, refer to the EP0_DRx register on page 176.

20.3.8 EPx_CNT1 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,41h	EP1_CNT1	Data Count[7:0]								RW : 00
0,43h	EP2_CNT1	Data Count[7:0]								RW : 00
0,45h	EP3_CNT1	Data Count[7:0]								RW : 00
0,47h	EP4_CNT1	Data Count[7:0]								RW : 00
0,49h	EP5_CNT1	Data Count[7:0]								RW : 00
0,4Bh	EP6_CNT1	Data Count[7:0]								RW : 00
0,4Dh	EP7_CNT1	Data Count[7:0]								RW : 00
0,4Fh	EP8_CNT1	Data Count[7:0]								RW : 00

The Endpoint Count Register 1 (EPx_CNT1) sets or reports the number of bytes in a USB data transfer to the non-control endpoints.

Bit 7 to 0: Data Count. These bits are the eight LSb of a 9-bit counter. The MSb is the Count MSb bit of the EPx_CNT0 register. The 9-bit count indicates the number of data bytes in a transaction. For IN transactions, firmware loads the count with the number of data bytes to be transmitted to the host. Valid values are 0 to 256. The 9-bit count also sets the limit for the number of bytes that are received for an OUT transaction. Before an OUT transaction is received for an endpoint, this count value must be set to the maximum number of data bytes to receive. If this count value is set to a value greater than the number of bytes (Data + CRC) received, both the data from the USB packet and the two-byte CRC are written to the USB's dedicated SRAM.

If the number of data bytes received is exactly the same as the 9-bit count, then only the data is updated into the USB SRAM and the CRC is discarded but the OUT transaction is completed according to the Mode bits of the EPx Control Register. If the number of data bytes received is more than the 9-bit count, then the OUT transaction is ignored.

After the OUT transaction is complete, the full 9-bit count is updated by the SIE to the actual number of data bytes received by the SIE plus two for the packet's CRC. Valid values are 2 to 258.

To get the actual number of bytes received, firmware must decrement the 9-bit count by two.

For additional information, refer to the EPx_CNT1 register on page 178.

20.3.9 EPx_CNT0 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,40h	EP1_CNT0	Data Toggle	Data Valid						Count MSB	# : 0
0,42h	EP2_CNT0	Data Toggle	Data Valid						Count MSB	# : 0
0,44h	EP3_CNT0	Data Toggle	Data Valid						Count MSB	# : 0
0,46h	EP4_CNT0	Data Toggle	Data Valid						Count MSB	# : 0
0,48h	EP5_CNT0	Data Toggle	Data Valid						Count MSB	# : 0
0,4Ah	EP6_CNT0	Data Toggle	Data Valid						Count MSB	# : 0
0,4Ch	EP7_CNT0	Data Toggle	Data Valid						Count MSB	# : 0
0,4Eh	EP8_CNT0	Data Toggle	Data Valid						Count MSB	# : 0

The Endpoint Count Register 0 (EPx_CNT0) is used for configuring endpoints 1 through 8.

Bit 7: Data Toggle. This bit selects the data packet's toggle state. For IN transactions, firmware must set this bit to the expected state. For OUT transactions, the hardware sets this bit to the state of the received Data Toggle bit. ‘0’ is DATA0. ‘1’ is DATA1.

Bit 6: Data Valid. This bit is used for OUT transactions only and is read only. It is cleared to '0' if CRC, bit stuffing errors, or PID errors occur. This bit does not update for some endpoint mode settings. ‘0’ is error in data received. ‘1’ is no error.

Bit 0: Count MSB. This bit is the one MSb of a 9-bit counter. The LSb are the Data Count[7:0] bits of the EPx_CNT1 register. Refer to the EPx_CNT1 Register for more information.

For additional information, refer to the EPx_CNT0 register on page 177.

20.3.10 EPx_CR0 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,54h	EP1_CR0	Stall		NAK_INT_EN	ACKed Tx	Mode[3:0]				# : 00
1,55h	EP2_CR0	Stall		NAK_INT_EN	ACKed Tx	Mode[3:0]				# : 00
1,56h	EP3_CR0	Stall		NAK_INT_EN	ACKed Tx	Mode[3:0]				# : 00
1,57h	EP4_CR0	Stall		NAK_INT_EN	ACKed Tx	Mode[3:0]				# : 00
1,58h	EP5_CR0	Stall		NAK_INT_EN	ACKed Tx	Mode[3:0]				# : 00
1,59h	EP6_CR0	Stall		NAK_INT_EN	ACKed Tx	Mode[3:0]				# : 00
1,5Ah	EP7_CR0	Stall		NAK_INT_EN	ACKed Tx	Mode[3:0]				# : 00
1,5Bh	EP8_CR0	Stall		NAK_INT_EN	ACKed Tx	Mode[3:0]				# : 00

The Endpoint Control Register 0 (EPx_CR0) is used for status and configuration of the non-control endpoints 1 to 8.

Bit 7: Stall. When this bit is set, the SIE stalls an OUT packet if the mode bits are set to ACK-OUT. The SIE stalls an IN packet if the mode bits are set to ACK-IN. This bit must be cleared for all other modes. ‘0’ is do not issue a stall. ‘1’ is stall an OUT packet if mode bits are set to ACK-OUT, or stall an IN packet if mode bits are set to ACK-IN.

Bit 5: NAK Int Enable. When set, this bit causes an endpoint interrupt to be generated even when a transfer completes with a NAK. ‘0’ is do not issue an interrupt after completing the transaction by sending NAK. ‘1’ is interrupt after transaction is complete by sending NAK.

Bit 4: ACKed Transaction. The ACKed Transaction bit is set whenever the SIE engages in a transaction to the register's endpoint that completes with an ACK packet. This bit is cleared by any writes to the register. ‘0’ is no ACKed transactions since bit was last cleared. ‘1’ indicates a transaction ended with an ACK.

Bits 3 to 0: Mode[3:0]. The mode controls how the USB SIE responds to traffic and how the USB SIE changes the mode of that endpoint as a result of host packets to the endpoint. Refer to “Mode Encoding for Control and Non-Control Endpoints” on page 142.

For additional information, refer to the EPx_CR0 register on page 218.

20.3.11 PMAx_WA Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,34h	PMA0_WA	Write Address[7:0]								RW : 00
1,35h	PMA1_WA	Write Address[7:0]								RW : 00
1,36h	PMA2_WA	Write Address[7:0]								RW : 00
1,37h	PMA3_WA	Write Address[7:0]								RW : 00
1,38h	PMA4_WA	Write Address[7:0]								RW : 00
1,39h	PMA5_WA	Write Address[7:0]								RW : 00
1,3Ah	PMA6_WA	Write Address[7:0]								RW : 00
1,3Bh	PMA7_WA	Write Address[7:0]								RW : 00
1,44h	PMA8_WA	Write Address[7:0]								RW : 00
1,45h	PMA9_WA	Write Address[7:0]								RW : 00
1,46h	PMA10_WA	Write Address[7:0]								RW : 00
1,47h	PMA11_WA	Write Address[7:0]								RW : 00
1,48h	PMA12_WA	Write Address[7:0]								RW : 00
1,49h	PMA13_WA	Write Address[7:0]								RW : 00
1,4Ah	PMA14_WA	Write Address[7:0]								RW : 00
1,4Bh	PMA15_WA	Write Address[7:0]								RW : 00

The PSoC Memory Arbiter Write Address Register (PMAx_WA) is used to set the beginning SRAM address for the PMA channel. A PMAx_WA register address uses the same physical register as the PMAx_RA register address. Therefore, when the read address is changed, the write address is also changed and the PMAx_RA and PMAx_WA registers always return the same value when read.

Bits 7 to 0: Address[7:0]. The value returned when this register is read depends on whether the PMA channel is being used by the USB SIE or by the M8C. In the USB case, this register always returns the beginning SRAM address for the PMA channel. In the M8C case, this register always returns the next SRAM address that is used by the PMA channel, if a byte is written to the channel's data register (PMAx_DR) by the M8C.

For additional information, refer to the PMAx_WA register on page 216.

20.3.12 PMAx_DR Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
0,58h	PMA0_DR	Data Byte[7:0]								RW : 00
0,59h	PMA1_DR	Data Byte[7:0]								RW : 00
0,5Ah	PMA2_DR	Data Byte[7:0]								RW : 00
0,5Bh	PMA3_DR	Data Byte[7:0]								RW : 00
0,5Ch	PMA4_DR	Data Byte[7:0]								RW : 00
0,5Dh	PMA5_DR	Data Byte[7:0]								RW : 00
0,5Eh	PMA6_DR	Data Byte[7:0]								RW : 00
0,5Fh	PMA7_DR	Data Byte[7:0]								RW : 00
0,64h	PMA8_DR	Data Byte[7:0]								RW : 00
0,65h	PMA9_DR	Data Byte[7:0]								RW : 00
0,66h	PMA10_DR	Data Byte[7:0]								RW : 00
0,67h	PMA11_DR	Data Byte[7:0]								RW : 00
0,68h	PMA12_DR	Data Byte[7:0]								RW : 00
0,69h	PMA13_DR	Data Byte[7:0]								RW : 00
0,6Ah	PMA14_DR	Data Byte[7:0]								RW : 00
0,6Bh	PMA15_DR	Data Byte[7:0]								RW : 00

The PSoC Memory Arbiter Data Register (PMAx_DR) is used to read and write to a particular PMA channel by either the USB SIE or the M8C. Note that a PMA channel may not be used simultaneously by both the USB SIE and the M8C.

Bits 7 to 0: Data Byte[7:0]. When the M8C writes to this register, the PMA registers the byte and then stores the value at the address in SRAM indicated by the PMAx_WA register.

After the value is written to SRAM, the PMAx_WA register is automatically incremented. When the USB SIE writes to this register, the PMA registers the byte and then stores the value in SRAM using the sum of the value of the PMAx_WA register and the USB SIEs received byte count. When the M8C reads this register, a pre-loaded value is returned and the PMAx_RA value is automatically incremented.

The new PMAx_RA value is used to fetch the next value from the SRAM, to be ready for the next read from the channel's PMAx_DR register. When the USB SIE reads the PMAx_DR register, it also receives a pre-loaded value, which triggers the PMA logic to fetch the next value in SRAM to be ready for the USB SIEs next read request. In all read cases, the initial pre-load of the first address of the channel is triggered by writing the first address of the channel to the channel's PMAx_RA register. Therefore, the PMAx_RA register must be written after data is stored for the channel.

For additional information, refer to the PMAx_DR register on page 179.

20.3.13 PMAx_RA Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,3Ch	PMA0_RA	Read Address[7:0]								RW : 00
1,3Dh	PMA1_RA	Read Address[7:0]								RW : 00
1,3Eh	PMA2_RA	Read Address[7:0]								RW : 00
1,3Fh	PMA3_RA	Read Address[7:0]								RW : 00
1,40h	PMA4_RA	Read Address[7:0]								RW : 00
1,41h	PMA5_RA	Read Address[7:0]								RW : 00
1,42h	PMA6_RA	Read Address[7:0]								RW : 00
1,43h	PMA7_RA	Read Address[7:0]								RW : 00
1,4Ch	PMA8_RA	Read Address[7:0]								RW : 00
1,4Dh	PMA9_RA	Read Address[7:0]								RW : 00
1,4Eh	PMA10_RA	Read Address[7:0]								RW : 00
1,4Fh	PMA11_RA	Read Address[7:0]								RW : 00
1,50h	PMA12_RA	Read Address[7:0]								RW : 00
1,51h	PMA13_RA	Read Address[7:0]								RW : 00
1,52h	PMA14_RA	Read Address[7:0]								RW : 00
1,53h	PMA15_RA	Read Address[7:0]								RW : 00

The PSoC Memory Arbiter Read Address Register (PMAx_RA) is used to set the beginning address for the PMA channel. A PMAx_RA register address uses the same physical register as the PMAx_WA register address. Therefore, when the read address is changed, the write address is also changed and the PMAx_WA and PMAx_RA registers always return the same value when read. When a PMAx_RA register is written, the address is stored and the value of the corresponding SRAM address is loaded into the channel's PMAx_DR. Therefore, this register must only be written after valid data is stored in SRAM for the channel.

Bits 7 to 0: Address[7:0]. The value returned when this register is read depends on whether the PMA channel is being used by the USB SIE or by the M8C. In the USB SIE case, this register always returns the beginning SRAM address for the PMA channel. In the M8C case, this register always returns the next SRAM address that is used by the PMA channel, if a byte is read from the channel's data register (PMAx_DR) by the M8C.

For additional information, refer to the PMAx_RA register on page 217.

20.3.14 USB_CR1 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,30h	USB_CR1						BusActivity	EnableLock	RegEnable	# : 0

The USB Control Register 1 (USB_CR1) is used to configure the internal regulator and the oscillator tuning capability.

Bit 2: BusActivity. The BusActivity bit is a “sticky” bit that detects any non-idle USB event that has occurred on the USB bus. After set to high by the SIE to indicate the bus activity, this bit retains its logical high value until firmware clears it. Writing a '0' to this bit clears it; writing a '1' preserves its value. ‘0’ is no activity. '1' is non-idle activity (D+ = low) was detected since the last time the bit was cleared.

Bit 1: EnableLock. Set this bit to turn on the automatic frequency locking of the internal oscillator to USB traffic. Unless an external clock is being provided, this bit must remain set for proper USB operation. ‘0’ is locking disabled. '1' is locking enabled.

Bit 0: RegEnable. This bit controls the operation of the internal USB regulator. For applications with device supply voltages in the 5-V range, set this bit high to enable the internal regulator. For device supply voltages in the 3.3-V range, clear this bit to connect the transceiver directly to the supply. ‘0’ is passthrough mode. Use for Vdd = 3.3-V range. '1' is regulating mode. Use for Vdd = 5-V range.

For additional information, refer to the USB_CR1 register on page 215.

20.3.15 USB_MISC_CR Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,BDh	USB_MISC_CR						USB_SE_EN	USB_ON	USB_CLK_ON	RW : 0

The USB Miscellaneous Control Register controls the clocks to the USB block, to make the IMO work with better accuracy for the USB part and to disable the single-ended input of the USBIO in the case of a non-USB part.

Bit 2: USB_SE_EN. The single-ended outputs of USBIO is enabled or disabled based upon this bit setting. Set this bit to '1' when using this part as a USB part for USB transactions to occur. Set this bit to '0' to disable single-ended outputs of USBIO. The DPO and DMO are held at logic high state and RSE0 is held at a low state.

Note Bit [1:0] of the USBIO_CR1 register is also affected depending on this register setting. When this bit is '0' (default), regardless of the DP and DM state, the DPO and DMO bits of USBIO_CR1 are '11b'.

Bit 1: USB_ON. This bit is used by the IMO DAC block to either work with better DNL consuming higher power, or with sacrificed DNL consuming lower power. Set this bit to '1' when the part is used as a USB part. A '0' runs the IMO with sacrificed DNL by consuming less power. A '1' runs the IMO with better DNL by consuming more power.

Bit 0: USB_CLK_ON. This bit either enables or disables the clocks to the USB block. It is used to save power in cases when the device need not respond to USB traffic. Set this bit to '1' when the device is used as a USB part.

When this bit is a ‘0’, all clocks to the USB block are driven. The device does not respond to USB traffic and none of the USB registers, except IMO_TR, IMO_TR1, and USBIO_CR1, listed in the Register Definitions on page 147 are writable.

When this bit is a ‘1’, clocks are not blocked to the USB block. The device responds to USB traffic depending on the other register settings mentioned under Register Definitions in the Full-Speed USB chapter on page 141.

For additional information, refer to the USB_MISC_CR register on page 220.

20.3.16 IMO_TR1 Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access
1,FAh	IMO_TR1						Fine Trim[2:0]			RW : 00

INTERNAL Register – The Internal Main Oscillator Trim Register 1 (IMO_TR1) fine tunes the IMO frequency.

For information on the other IMO trim register (IMO_TR), see the Internal Main Oscillator (IMO) chapter on page 67 or refer to the IMO_TR1 register on page 238 in the Register Details chapter.

Bits 2 to 0: Fine Trim[2:0]. These bits provide a fine tuning capability to the IMO trim. These three bits are the three LSb of the IMO trim with the IMO_TR register supplying the eight MSb.

For additional information, refer to the IMO_TR1 register on page 238.

Section D: Registers

The Registers section discusses the registers of the enCoRe V device. It lists all the registers in mapping tables, in address order. For easy reference, each register is linked to the page of a detailed description located in the next chapter. This section includes the following chapter:

- Register Reference chapter on page 163.

Register General Conventions

The register conventions specific to this section and the Register Reference chapter are listed in the following table:

Register Conventions

Convention	Description
Empty, grayed-out table cell	Illustrates a reserved bit or group of bits.
‘x’ before the comma in an address	Indicates the register exists in register bank 1 and register bank 2.
‘x’ in a register name	Indicates that there are multiple instances/address ranges of the same register.
R	Read register or bit(s).
W	Write register or bit(s).
O	Only a read/write register or bit(s).
L	Logical register or bit(s).
C	Clearable register or bit(s).
#	Access is bit specific.

Register Mapping Tables

The enCoRe V device has a total register address space of 512 bytes. The register space is also referred to as I/O space and is broken into two parts: Bank 0 (user space) and Bank 1 (configuration space). The XIO bit in the Flag register (CPU_F) determines which bank the user is currently in. When the XIO bit is set, the user is said to be in the “extended” address space or the “configuration” registers.

Refer to the individual enCoRe V device datasheets for device-specific register mapping information.

Register Map Bank 0 Table: User Space

Name	Addr (0,Hex)	Access	Page	Name	Addr (0,Hex)	Access	Page	Name	Addr (0,Hex)	Access	Page	Name	Addr (0,Hex)	Access	Page
PRT0DR	00	RW	164	EP1_CNT0	40	#	177		80				C0		
PRT0IE	01	RW	165	EP1_CNT1	41	RW	178		81				C1		
	02			EP2_CNT0	42	#	177		82				C2		
	03			EP2_CNT1	43	RW	178		83				C3		
PRT1DR	04	RW	164	EP3_CNT0	44	#	177		84				C4		
PRT1IE	05	RW	165	EP3_CNT1	45	RW	178		85				C5		
	06			EP4_CNT0	46	#	177		86				C6		
	07			EP4_CNT1	47	RW	178		87				C7		
PRT2DR	08	RW	164	EP5_CNT0	48	#	177		88			I2C_XCFG	C8	RW	186
PRT2IE	09	RW	165	EP5_CNT1	49	RW	178		89				C9		
	0A			EP6_CNT0	4A	#	177		8A			I2C_ADDR	CA	RW	187
	0B			EP6_CNT1	4B	RW	178		8B				CB		
PRT3DR	0C	RW	164	EP7_CNT0	4C	#	177		8C				CC		
PRT3IE	0D	RW	165	EP7_CNT1	4D	RW	178		8D				CD		
	0E			EP8_CNT0	4E	#	177		8E				CE		
	0F			EP8_CNT1	4F	RW	178		8F				CF		
PRT4DR	10	RW	164		50				90			CUR_PP	D0	RW	188
PRT4IE	11	RW	165		51				91			STK_PP	D1	RW	189
	12				52				92				D2		
	13				53				93			IDX_PP	D3	RW	190
	14				54				94			MVR_PP	D4	RW	191
	15				55				95			MVW_PP	D5	RW	192
	16				56				96			I2C_CFG	D6	RW	193
	17				57				97			I2C_SCR	D7	#	194
	18			PMA0_DR	58	RW	179		98			I2C_DR	D8	RW	195
	19			PMA1_DR	59	RW	179		99				D9		
	1A			PMA2_DR	5A	RW	179		9A			INT_CLR0	DA	RW	196
	1B			PMA_DR	5B	RW	179		9B			INT_CLR1	DB	RW	198
	1C			PMA4_DR	5C	RW	179		9C			INT_CLR2	DC	RW	200
	1D			PMA5_DR	5D	RW	179		9D				DD		
	1E			PMA6_DR	5E	RW	179		9E			INT_MSK2	DE	RW	202
	1F			PMA7_DR	5F	RW	179		9F			INT_MSK1	DF	RW	203
	20				60				A0			INT_MSK0	E0	RW	204
	21				61				A1			INT_SW_EN	E1	RW	205
	22				62				A2			INT_VC	E2	RC	206
	23				63				A3			RES_WDT	E3	W	207
	24			PMA8_DR	64	RW	179		A4				E4		
	25			PMA9_DR	65	RW	179		A5				E5		
	26			PMA10_DR	66	RW	179		A6				E6		
	27			PMA11_DR	67	RW	179		A7				E7		
	28			PMA12_DR	68	RW	179		A8				E8		
SPI_TXR	29	W	166	PMA13_DR	69	RW	179		A9				E9		
SPI_RXR	2A	R	167	PMA14_DR	6A	RW	179		AA				EA		
SPI_CR	2B	#	168	PMA15_DR	6B	RW	179		AB				EB		
	2C			TMP_DR0	6C	RW	219		AC				EC		
	2D			TMP_DR1	6D	RW	219		AD				ED		
	2E			TMP_DR2	6E	RW	219		AE				EE		
	2F			TMP_DR3	6F	RW	219		AF				EF		
	30				70			PT0_CFG	B0	RW	181		F0		
USB_SOF0	31	R	169		71			PT0_DATA1	B1	RW	183		F1		
USB_SOF1	32	R	170		72			PT0_DATA0	B2	RW	183		F2		
USB_CR0	33	RW	171		73			PT1_CFG	B3	RW	184		F3		
USBIO_CR0	34	#	172		74			PT1_DATA1	B4	RW	183		F4		
USBIO_CR1	35	#	173		75			PT1_DATA0	B5	RW	183		F5		
EP0_CR	36	#	174		76			PT2_CFG	B6	RW	185		F6		
EP0_CNT0	37	#	175		77			PT2_DATA1	B7	RW	183	CPU_F	F7	RL	208
EP0_DR0	38	RW	176	CMP_RDC	78	#		PT2_DATA0	B8	RW	183		F8		
EP0_DR1	39	RW	176	CMP_MUX	79	RW			B9				F9		
EP0_DR2	3A	RW	176	CMP_CR0	7A	RW			BA				FA		
EP0_DR3	3B	RW	176	CMP_CR1	7B	RW			BB				FB		
EP0_DR4	3C	RW	176	CMP_LUT	7C	RW			BC				FC		
EP0_DR5	3D	RW	176		7D				BD				FD		

Gray fields are reserved. # Access is bit specific.

Name	Addr (0,Hex)	Access	Page	Name	Addr (0,Hex)	Access	Page	Name	Addr (0,Hex)	Access	Page	Name	Addr (0,Hex)	Access	Page
EP0_DR6	3E	RW	176		7E				BE			CPU_SCR1	FE	#	210
EP0_DR7	3F	RW	176		7F				BF			CPU_SCR0	FF	#	211

Gray fields are reserved. # Access is bit specific.

Register Map Bank 1 Table: Configuration Space

Name	Addr (1,Hex)	Access	Page	Name	Addr (1,Hex)	Access	Page	Name	Addr (1,Hex)	Access	Page	Name	Addr (1,Hex)	Access	Page
PRT0DM0	00	RW	212	PMA4_RA	40	RW	217		80				C0		
PRT0DM1	01	RW	213	PMA5_RA	41	RW	217		81				C1		
	02			PMA6_RA	42	RW	217		82				C2		
	03			PMA7_RA	43	RW	217		83				C3		
PRT1DM0	04	RW	212	PMA8_WA	44	RW	216		84				C4		
PRT1DM1	05	RW	213	PMA9_WA	45	RW	216		85				C5		
	06			PMA10_WA	46	RW	216		86				C6		
	07			PMA11_WA	47	RW	216		87				C7		
PRT2DM0	08	RW	212	PMA12_WA	48	RW	216		88				C8		
PRT2DM1	09	RW	213	PMA13_WA	49	RW	216		89				C9		
	0A			PMA14_WA	4A	RW	216		8A				CA		
	0B			PMA15_WA	4B	RW	216		8B				CB		
PRT3DM0	0C	RW	212	PMA8_RA	4C	RW	217		8C				CC		
PRT3DM1	0D	RW	213	PMA9_RA	4D	RW	217		8D				CD		
	0E			PMA10_RA	4E	RW	217		8E				CE		
	0F			PMA11_RA	4F	RW	217		8F				CF		
PRT4DMO	10	RW	213	PMA12_RA	50	RW	217		90				D0		
PRT4DM1	11	RW	213	PMA13_RA	51	RW	217		91				D1		
	12			PMA14_RA	52	RW	217		92			ECO_ENBUS	D2	RW	221
	13			PMA15_RA	53	RW	217		93			ECO_TRIM	D3	RW	222
	14			EP1_CR0	54	#	218		94				D4		
	15			EP2_CR0	55	#	218		95				D5		
	16			EP3_CR0	56	#	218		96				D6		
	17			EP4_CR0	57	#	218		97				D7		
	18			EP5_CR0	58	#	218		98			MUX_CR0	D8	RW	223
	19			EP6_CRO	59	#	218		99			MUX_CR1	D9	RW	223
	1A			EP7_CR0	5A	#	218		9A			MUX_CR2	DA	RW	223
	1B			EP8_CR0	5B	#	218		9B			MUX_CR3	DB	RW	223
	1C				5C				9C			IO_CFG1	DC	RW	224
	1D				5D				9D			OUT_P1	DD	RW	225
	1E				5E				9E			IO_CFG2	DE	RW	227
	1F				5F				9F			MUX_CR4	DF	RW	223
	20				60				A0			OSC_CR0	E0	RW	228
	21				61				A1			ECO_CFG	E1	#	229
	22				62				A2			OSC_CR2	E2	RW	230
	23				63				A3			VLT_CR	E3	RW	231
	24				64				A4			VLT_CMP	E4	R	232
	25				65				A5				E5		
	26				66				A6				E6		
	27				67				A7				E7		
	28				68				A8			IMO_TR	E8	W	233
SPI_CFG	29	RW	214		69				A9			ILO_TR	E9	W	234
	2A				6A				AA				EA		
	2B				6B				AB			SLP_CFG	EB	RW	235
	2C			TMP_DR0	6C	RW	219		AC			SLP_CFG2	EC	RW	236
	2D			TMP_DR1	6D	RW	219		AD			SLP_CFG3	ED	RW	237
	2E			TMP_DR2	6E	RW	219		AE				EE		
	2F			TMP_DR3	6F	RW	219		AF				EF		
USB_CR1	30	#	215		70				B0				F0		
	31				71				B1				F1		
	32				72				B2				F2		
	33				73				B3				F3		
PMA0_WA	34	RW	216		74				B4				F4		
PMA1_WA	35	RW	216		75				B5				F5		

Gray fields are reserved # Access is bit specific.

Name	Addr (1,Hex)	Access	Page	Name	Addr (1,Hex)	Access	Page	Name	Addr (1,Hex)	Access	Page	Name	Addr (1,Hex)	Access	Page
PMA2_WA	36	RW	216		76				B6				F6		
PMA3_WA	37	RW	216		77				B7			CPU_F	F7	RL	208
PMA4_WA	38	RW	216		78				B8				F8		
PMA5_WA	39	RW	216		79				B9				F9		
PMA6_WA	3A	RW	216		7A				BA			IMO_TR1	FA	RW	238
PMA7_WA	3B	RW	216		7B				BB				FB		
PMA0_RA	3C	RW	217		7C				BC				FC		
PMA1_RA	3D	RW	217		7D			USB_MISC_CR	BD	RW	220		FD		
PMA2_RA	3E	RW	217		7E				BE				FE		
PMA3_RA	3F	RW	217		7F				BF				FF		

Gray fields are reserved # Access is bit specific.

21. Register Reference

This chapter is a reference for all the enCoRe V device registers in address order, for Bank 0 and Bank 1. The most detailed descriptions of the enCoRe V registers are in the Register Definitions section of each chapter. The registers that are in both banks are incorporated with the Bank 0 registers, designated with an ‘x’, rather than a ‘0’ preceding the comma in the address. Bank 0 registers are listed first and begin on page 164. Bank 1 registers are listed second and begin on page 212. A condensed view of all the registers is shown in the register mapping tables starting on page 159.

21.1 Maneuvering Around the Registers

For ease-of-use, this chapter is formatted so that there is one register per page, although some registers use two pages. On each page, from top to bottom, there are four sections:

1. Register name and address (from lowest to highest).
2. Register table showing the bit organization, with reserved bits grayed out.
3. Written description of register specifics or links to additional register information.
4. Detailed register bit descriptions.

Use the register tables, in addition to the detailed register bit descriptions, to determine which bits are reserved. Reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of ‘0’. For all registers, an ‘x’ before the comma in the address field indicates that the register can be accessed or written to no matter what bank is used. For example, the M8C flag register’s (CPU_F) address is ‘x,F7h’ meaning it is located in bank 0 and bank 1 at F7h.

21.2 Register Conventions

The following table lists the register conventions that are specific to this chapter.

Register Conventions

Convention	Example	Description
‘x’ in a register name	PRTxIE	Multiple instances/address ranges of the same register.
R	R : 00	Read register or bit(s).
W	W : 00	Write register or bit(s).
O	RO : 00	Only a read/write register or bit(s).
L	RL : 00	Logical register or bit(s).
C	RC : 00	Clearable register or bit(s).
00	RW : 00	Reset value is 0x00 or 00h.
XX	RW : XX	Register is not reset.
0,	0,04h	Register is in bank 0.
1,	1,23h	Register is in bank 1.
x,	x,F7h	Register exists in register bank 0 and register bank 1.
Empty, grayed-out table cell		Reserved bit or group of bits, unless otherwise stated.

21.3 Bank 0 Registers

The following registers are all in bank 0 and are listed in address order. An ‘x’ before the comma in the register’s address indicates that the register can be accessed in Bank 0 and Bank 1, independent of the XIO bit in the CPU_F register. Registers that are in both Bank 0 and Bank 1 are listed in address order in Bank 0. For example, the RDIxLT1 register has an address of x,B4h and is listed only in Bank 0 but is accessed in both Bank 0 and Bank 1.

21.3.1 PRTxDR

Port Data Registers

Individual Register Names and Addresses:

PRT0DR : 0,00h PRT1DR : 0,04h PRT2DR : 0,08h PRT3DR : 0,0Ch
PRT4DR : 0,10h

	7	6	5	4	3	2	1	0
Access : POR	RW : 00							
Bit Name	Data[7:0]							

These registers allow write or read access, or the current logical equivalent, of pin voltage.

The upper nibble of the PRT4DR register returns the last data bus value when read. You need to mask it off before using this information. For additional information, refer to the Register Definitions on page 56 in the GPIO chapter.

Bit	Name	Description
7:0	**Data[7:0]**	Write value to port or read value from port. Reads return the state of the pin, not the value in the PRTxDR register.

21.3.2 PRTxIE

Port Interrupt Enable Registers

Individual Register Names and Addresses:

PRT0IE : 0,01h PRT1IE : 0,05h PRT2IE : 0,09h PRT3IE : 0,0Dh
PRT4IE : 0,11h

	7	6	5	4	3	2	1	0
Access : POR	RW : 00							
Bit Name	Interrupt Enables[7:0]							

These registers enable or disable interrupts from individual GPIO pins.

The upper nibble of the PRT4IE register returns the last data bus value when read and must be masked off before using this information. For additional information, refer to the Register Definitions on page 56 in the GPIO chapter.

Bit	Name	Description
7:0	**Interrupt Enables[7:0]**	These bits enable the corresponding port pin interrupt. Only four LSB are used because this port has four pins. 0 Port pin interrupt disabled for the corresponding pin. 1 Port pin interrupt enabled for the corresponding pin. Interrupt mode is determined by the IOINT bit in the IO_CFG1 register.

21.3.3 SPI_TXR

SPI Transmit Data Register

Individual Register Names and Addresses:

SPI_TXR : 0,29h

	7	6	5	4	3	2	1	0
Access : POR	W : 00							
Bit Name	Data[7:0]							

This register is the SPI's transmit data register.

For additional information, refer to the Register Definitions on page 125 in the SPI chapter.

Bit	Name	Description
7:0	**Data[7:0]**	Data for selected function.

21.3.4 SPI_RXR

SPI Receive Data Register

Individual Register Names and Addresses:

SPI_RXR : 0,2Ah

	7	6	5	4	3	2	1	0
Access : POR	R : 00							
Bit Name	Data[7:0]							

This register is the SPI's receive data register.

For additional information, refer to the Register Definitions on page 125 in the SPI chapter.

Bit	Name	Description
7:0	**Data[7:0]**	Data for selected function.

21.3.5 SPI_CR

SPI Control Register

Individual Register Names and Addresses:

SPI_CR : 0,2Bh

	7	6	5	4	3	2	1	0
Access : POR	RW : 0	R : 0	R : 0	R : 1	R : 0	RW : 0	RW : 0	RW : 0
Bit Name	LSb First	Overrun	SPI Complete	TX Reg Empty	RX Reg Full	Clock Phase	Clock Polarity	Enable

This register is the SPI control register.

The LSb First, Clock Phase, and Clock Polarity bits are configuration bits. Do not change them when the block is enabled. These bits can be set at the same time that the block is enabled. For additional information, refer to the Register Definitions on page 125 in the SPI chapter.

Bit	Name	Description
7	**LSb First**	Do not change this bit during an SPI transfer. 0 Data is shifted out MSb first. 1 Data is shifted out LSb first.
6	**Overrun**	0 No overrun has occurred. 1 Overrun has occurred. Indicates that a new byte is received and loaded into the RX Buffer before the previous one is read. It is cleared on a read of this (CR0) register.
5	**SPI Complete**	0 Indicates that a byte may still be in the process of shifting out, or no transmission is active. 1 Indicates that a byte is shifted out and all associated clocks are generated. It is cleared on a read of this (CR0) register. Optional interrupt.
4	**TX Reg Empty**	Reset state and the state when the block is disabled is ‘1’. 0 Indicates that a byte is currently buffered in the TX register. 1 Indicates that a byte is written to the TX register and cleared on write of the TX Buffer (DR1) register. This is the default interrupt. This status is initially asserted on block enable; however, the TX Reg Empty interrupt occurs only after the first data byte is written and transferred into the shifter.
3	**RX Reg Full**	0 RX register is empty. 1 A byte is received and loaded into the RX register. It is cleared on a read of the RX Buffer (DR2) register.
2	**Clock Phase**	0 Data is latched on the leading clock edge. Data changes on the trailing edge (modes 0, 1). 1 Data changes on the leading clock edge. Data is latched on the trailing edge (modes 2, 3).
1	**Clock Polarity**	0 Non-inverted, clock idles low (modes 0, 2). 1 Inverted, clock idles high (modes 1, 3).
0	**Enable**	0 SPI function is not enabled. 1 SPI function is enabled.

21.3.6 USB_SOF0

USB Start-of-Frame Register 0

Individual Register Names and Addresses:

USB_SOF0 : 0,31h

	7	6	5	4	3	2	1	0
Access : POR	R : 00							
Bit Name	Frame Number[7:0]							

This register is a USB Start-of-Frame register 0.

For additional information, refer to the Register Definitions on page 147 in the Full-Speed USB chapter.

Bit	Name	Description
7:0	**Frame Number[7:0]**	Contains the lower eight bits of the frame number.

21.3.7 USB_SOF1

USB Start-of-Frame Register 1

Individual Register Names and Addresses:

USB_SOF1 : 0,32h

	7	6	5	4	3	2	1	0
Access : POR						R : 0		
Bit Name						Frame Number[10:8]		

This register is a USB Start-of-Frame register 1.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 147 in the Full-Speed USB chapter.

Bit	Name	Description
2:0	**Frame Number[10:8]**	Contains the upper three bits of the frame number.

21.3.8 USB_CR0

USB Control Register 0

Individual Register Names and Addresses:

USB_CR0 : 0,33h

	7	6	5	4	3	2	1	0
Access : POR	RW : 0	RW : 00						
Bit Name	USB Enable	Device Address[6:0]						

This register is a USB control register 0. The USB_MISC_CR register on page 220 must be set correctly for the bits in this register to function as described here.

For additional information, refer to the Register Definitions on page 147 in the Full-Speed USB chapter.

Bit	Name	Description
7	**USB Enable**	This bit enables the enCoRe V device to respond to USB traffic. 0 USB disabled. 1 USB enabled.
6:0	**Device Address**	These bits specify the USB address to which the SIE responds.

21.3.9 USBIO_CR0

USB I/O Control Register 0

Individual Register Names and Addresses:

USBIO_CR0 : 0,34h

	7	6	5	4	3	2	1	0
Access : POR	RW : 0	RW : 0	RW : 0					R : 0
Bit Name	TEN	TSE0	TD					RD

This register is a USBIO manual control register 0. This register is used to manually control or read the USB differential state of the D+ and D– pins. The USB_MISC_CR register on page 220 must be set correctly for the bits in this register to function as described here.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 147 in the Full-Speed USB chapter.

Bit	Name	Description
7	**TEN**	Transmit Enable. This bit is used to manually transmit on D+, D– pins. Normally, this bit must be cleared to allow the internal SIE to drive the pins. The most common reason for manually transmitting is to force a resume state on the bus. 0 Manual transmission off. 1 Manual transmission enabled.
6	**TSE0**	Both D+ and D– are low. There is no effect if TEN=0.
5	**TD**	This bit transmits a USB J or K state on the USB bus. There is no effect if TEN=0 or TSE0=1. 0 Force USB K state. 1 Force USB J state.
0	**RD**	This read-only bit gives the state of the USB differential receiver. 0 D+ < D– or D+ = D– = 0. 1 D+ > D–.

Note This note applies when you use the block in compatibility mode with the hardware address feature enabled and before going to I^2C sleep mode. You need to disable and then re-enable the block to avoid any data corruption from I^2C slave when the master is reading the data from the device's I^2C slave.

21.3.10 USBIO_CR1

USB I/O Control Register 1

Individual Register Names and Addresses:

USBIO_CR1 : 0,35h

	7	6	5	4	3	2	1	0
Access : POR	RW : 0	RW : 0	RW : 0	RW : 0	RW : 0	RW : 0	R : 1	R : 1
Bit Name	IOMode	Drive Mode	DPI	DMI	PS2PUEN	USBPUEN	DPO	DMO

This register is a USBIO manual control register 1. This register is used to manually control or read the drive mode and state of the D+ and D– USB pins. The USB_MISC_CR register on page 220 must be set correctly for the bits in this register to function as described here.

For additional information, refer to the Register Definitions on page 147 in the Full-Speed USB chapter.

Bit	Name	Description
7	**IOMode**	This bit enables the D+ and D– pins to be manually set or read with this register. 0 Manual access mode disabled. 1 Manual access mode enabled.
6	**Drive Mode**	This bit configures the D+ and D– pins for either CMOS drive mode or open drain drive mode. This bit has no effect if IOMode = 0. 0 Open drain drive mode. 1 CMOS drive mode.
5	**DPI**	This bit drives the D+ pin. This bit has no effect if IOMode = 0. 0 D+ pin drives a logic level LOW. 1 If Drive Mode bit is 0, D+ pin is tri-stated. If Drive Mode bit is 1, D+ pin drives a logic level HIGH.
4	**DMI**	This bit drives the D– pin. This bit has no effect if IOMode = 0. 0 D– pin drives a logic level LOW. 1 If Drive Mode bit is 0, D– pin is tri-stated. If Drive Mode bit is 1, D– pin drives a logic level HIGH.
3	**PS2PUEN**	This bit controls the connection of the two internal 5-kΩ pull-up resistors to the D+ and D– pins. 0 Pull-up resistors not connected. 1 Pull-up resistors connected.
2	**USBPUEN**	This bit controls the connection of the internal 1.5-kΩ pull-up resistor to the D+ pin. 0 Pull-up resistor not connected. 1 Pull-up resistor connected. **Note** Set bit2 of “USBIO_MISC_CR” register to enable USBIO.
1	**DPO**	This read-only bit provides the D+ pin logic level state.
0	**DMO**	This read-only bit provides the D– pin logic level state.

21.3.11 EP0_CR

Endpoint 0 Control Register

Individual Register Names and Addresses:

EP0_CR : 0,36h

	7	6	5	4	3	2	1	0
Access : POR	RC : 0	RC : 0	RC : 0	RC : 0	RW : 0			
Bit Name	Setup Received	IN Received	OUT Received	ACKed Transaction	Mode[3:0]			

This register is an endpoint 0 control register.

For additional information, refer to the Register Definitions on page 147 in the Full-Speed USB chapter.

Bit	Name	Description
7	**Setup Received**	When set, this bit indicates a valid setup packet was received and ACKed.
6	**IN Received**	When set, this bit indicates a valid IN packet was received.
5	**OUT Received**	When set, this bit indicates an OUT packet was received.
4	**ACKed Transaction**	When set, this bit indicates a valid OUT packet has been received and ACKed.
3:0	**Mode[3:0]**	The mode bits control how the USB SIE responds to traffic and how the USB SIE changes the mode of that endpoint as a result of host packets to the endpoint.

21.3.12 EP0_CNT

Endpoint 0 Count Register

Individual Register Names and Addresses:

EP0_CNT : 0,37h

	7	6	5	4	3	2	1	0
Access : POR	RW : 0	RC : 0			RW : 0			
Bit Name	Data Toggle	Data Valid			Byte Count[3:0]			

The Endpoint 0 Count register (EP0_CNT) configures endpoint 0.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 147 in the Full-Speed USB chapter.

Bit	Name	Description
7	**Data Toggle**	This bit selects the data packet's toggle state.
6	**Data Valid**	This bit indicates whether there were errors in OUT or setup transactions.
3:0	**Byte Count[3:0]**	These bits indicate the number of data bytes in a transaction.

21.3.13 EP0_DRx

Endpoint 0 Data Registers

Individual Register Names and Addresses:

EP0_DR0 : 0,38h
EP0_DR1 : 0,39h
EP0_DR2 : 0,3Ah
EP0_DR3 : 0,3Bh
EP0_DR4 : 0,3Ch
EP0_DR5 : 0,3Dh
EP0_DR6 : 0,3Eh
EP0_DR7 : 0,3Fh

	7	6	5	4	3	2	1	0
Access : POR	RW : 00							
Bit Name	Data Byte[7:0]							

These registers are endpoint 0 data registers.

For additional information, refer to the Register Definitions on page 147 in the Full-Speed USB chapter.

Bit	Name	Description
7:0	**Data Byte[7:0]**	These registers are shared for both transmit and receive.

21.3.14 EPx_CNT0

Endpoint Count 0 Registers

Individual Register Names and Addresses:

EP1_CNT0 : 0,40h
EP2_CNT0 : 0,42h
EP3_CNT0 : 0,44h
EP4_CNT0 : 0,46h
EP5_CNT0 : 0,48h
EP6_CNT0 : 0,4Ah
EP7_CNT0 : 0,4Ch
EP8_CNT0 : 0,4Eh

	7	6	5	4	3	2	1	0
Access : POR	RW : 0	RC : 0						# : 0
Bit Name	Data Toggle	Data Valid						Count MSB

These registers are endpoint count 0 registers.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 147 in the Full-Speed USB chapter.

Bit	Name	Description
7	**Data Toggle**	This bit selects the data packet's toggle state.
6	**Data Valid**	This bit is used for OUT transactions only and is read only.
0	**Count MSB**	This bit is the 1 MSb of a 9-bit counter.

21.3.15 EPx_CNT1

Endpoint Count 1 Registers

Individual Register Names and Addresses:

EP1_CNT1 : 0,41h
EP2_CNT1 : 0,43h
EP3_CNT1 : 0,45h
EP4_CNT1 : 0,47h
EP5_CNT1 : 0,49h
EP6_CNT1 : 0,4Bh
EP7_CNT1 : 0,4Dh
EP8_CNT1 : 0,4Fh

	7	6	5	4	3	2	1	0
Access : POR	RW : 00							
Bit Name	Data Count[7:0]							

These registers are endpoint count 1 registers.

For additional information, refer to the Register Definitions on page 147 in the Full-Speed USB chapter.

Bit	Name	Description
7:0	**Data Count[7:0]**	These bits are the eight LSb of a 9-bit counter. The MSb is the Count MSb of the EPx_CNT0 register.

21.3.16 PMAx_DR

PSoC Memory Arbiter Data Registers

Individual Register Names and Addresses:

PMA0_DR : 0,58h	PMA1_DR : 0,59h	PMA2_DR : 0,5Ah	PMA3_DR : 0,5Bh
PMA4_DR : 0,5Ch	PMA5_DR : 0,5Dh	PMA6_DR : 0,5Eh	PMA7_DR : 0,5Fh
PMA8_DR : 0,64h	PMA9_DR : 0,65h	PMA10_DR : 0,66h	PMA11_DR : 0,67h
PMA12_DR : 0,68h	PMA13_DR : 0,69h	PMA14_DR : 0,6Ah	PMA15_DR : 0,6Bh

	7	6	5	4	3	2	1	0
Access : POR	RW : 00							
Bit Name	Data Byte[7:0]							

These registers are PSoC Memory Arbiter write address registers.

For additional information, refer to the Register Definitions on page 147 in the Full-Speed USB chapter.

Bit	Name	Description
7:0	**Data Byte[7:0]**	When the M8C writes to this register, the PMA registers the byte and then stores the value at the address in SRAM indicated by the PMAx_WA register.

21.3.17 CMP_MUX

Comparator Multiplexer Register

Individual Register Names and Addresses:

CMP_MUX : 0,79h

	7	6	5	4	3	2	1	0
Access : POR	RW : 0		RW : 0		RW : 0		RW : 0	
Bit Name	INP1[1:0]		INN1[1:0]		INP0[1:0]		INN0[1:0]	

This register contains control bits for input selection of comparators 0 and 1.

For additional information, refer to the Register Definitions on page 111 in the Comparators chapter.

Bit	Name	Description
7:6	**INP1[1:0]**	Comparator 1 Positive Input Select 00b Analog Global Mux Bus 01b Reserved 10b P0[1] pin 11b P0[3] pin
5:4	**INN1[1:0]**	Comparator 1 Negative Input Select 00b VREF (1.0 V) 01b Ref Lo (approximately 0.6 V) 10b Ref Hi (approximately 1.2 V) 11b Reserved
3:2	**INP0[1:0]**	Comparator 0 Positive Input Select 00b Analog Global Mux Bus 01b Reserved 10b P0[1] pin 11b P0[3] pin
1:0	**INN0[1:0]**	Comparator 0 Negative Input Select 00b VREF (1.0 V) 01b Ref Lo (approximately 0.6 V) 10b Ref Hi (approximately 1.2 V) 11b Reserved

21.3.18 PT0_CFG

Programmable Timer 0 Configuration Register

Individual Register Names and Addresses:

PT0_CFG : 0,B0h

	7	6	5	4	3	2	1	0
Access : POR						RW : 0	RW : 0	RW : 0
Bit Name						CLKSEL	One Shot	START

This register configures the programmable timer 0.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 139 in the Programmable Timer chapter.

Bit	Name		Description
2	**CLKSEL**		This bit determines if the timer runs on the 32-kHz clock or CPU clock. If the bit is set to 1, the timer runs on the CPU clock, otherwise, the timer runs on the 32-kHz clock.
1	**One Shot**	0	Continuous count mode. Timer reloads the count value from the data registers upon each terminal count, and continues counting.
		1	One-shot mode. Timer goes through one complete count period and then stops. Upon completion, the START bit in this register is cleared.
0	**START**	0	Timer held in reset.
		1	Timer counts down from a full count determined from its data registers (PT_DATA1, PT_DATA0). When complete, it either stops or reloads and continues, based on the One Shot bit in this register.

21.3.19 PTx_DATA1

Programmable Timers Data Register 1

Individual Register Names and Addresses:

PT0_DATA1 : 0,B1h PT1_DATA1 : 0,B4h PT2_DATA1 : 0,B7h

	7	6	5	4	3	2	1	0
Access : POR	RW : 00							
Bit Name	DATA[7:0]							

These registers hold the eight bits of the programmable timer's count value for the device.

For additional information, refer to the Register Definitions on page 139 in the Programmable Timer chapter.

Bit	Name	Description
7:0	**DATA[7:0]**	This is the upper byte of a 16-bit timer. The lower byte is in the corresponding PTx_DATA0 register.

21.3.20 PTx_DATA0

Programmable Timers Data Register 0

Individual Register Names and Addresses:

PT0_DATA0 : 0,B2h PT1_DATA0 : 0,B5h PT2_DATA0 : 0,B8h

	7	6	5	4	3	2	1	0
Access : POR	RW : 00							
Bit Name	DATA[7:0]							

These registers provide the programmable timer with its lower eight bits of the count value.

For additional information, refer to the Register Definitions on page 139 in the Programmable Timer chapter.

Bit	Name	Description
7:0	**DATA[7:0]**	This is the lower byte of a 16-bit timer. The upper byte is in the corresponding PTxDATA1 register.

21.3.21 PT1_CFG

Programmable Timer 1 Configuration Register

Individual Register Names and Addresses:

PT1_CFG : 0,B3h

	7	6	5	4	3	2	1	0
Access : POR						RW : 0	RW : 0	RW : 0
Bit Name						CLKSEL	One Shot	START

This register configures the programmable timer 1.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 139 in the Programmable Timer chapter.

Bit	Name		Description
2	**CLKSEL**		This bit determines if the timer runs on the 32-kHz clock or CPU clock. If the bit is set to 1, the timer runs on the CPU clock, otherwise, the timer runs on the 32-kHz clock.
1	**One Shot**	0	Continuos count mode. Timer reloads the count value from the data registers upon each terminal count, and continues counting.
		1	One-shot mode. Timer goes through one complete count period and then stops. Upon completion, the START bit in this register is cleared.
0	**START**	0	Timer held in reset.
		1	Timer counts down from a full count determined from its data registers (PT_DATA1, PT_DATA0). When complete, it either stops or reloads and continues, based on the One Shot bit in this register.

21.3.22 PT2_CFG

Programmable Timer 2 Configuration Register

Individual Register Names and Addresses:

PT2_CFG : 0,B6h

	7	6	5	4	3	2	1	0
Access : POR						RW : 0	RW : 0	RW : 0
Bit Name						CLKSEL	One Shot	START

This register configures the programmable timer 2.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 139 in the Programmable Timer chapter.

Bit	Name		Description
2	**CLKSEL**		This bit determines if the timer runs on the 32-kHz clock or CPU clock. If the bit is set to 1, the timer runs on the CPU clock; otherwise, the timer runs on the 32-kHz clock.
1	**One Shot**	0	Continuous count mode. Timer reloads the count value from the data registers upon each terminal count, and continues counting.
		1	One-shot mode. Timer goes through one complete count period and then stops. Upon completion, the START bit in this register is cleared.
0	**START**	0	Timer held in reset.
		1	Timer counts down from a full count determined from its data registers (PT_DATA1, PT_DATA0). When complete, it either stops or reloads and continues, based on the One Shot bit in this register.

21.3.23 I2C_XCFG

I²C Extended Configuration Register

Individual Register Names and Addresses:

I2C_XCFG: 0,C8h

	7	6	5	4	3	2	1	0
Access : POR								RW : 0
Bit Name								HW Addr En

This register configures enhanced features. The Enable bit (bit 0) of the I2C_CFG (0,D6h) register should be set to '1' for the I2C enhanced features to work.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Always write reserved bits with a value of '0'. For additional information, refer to the Register Definitions on page 106 in the I2C Slave chapter.

Bit	Name	Description
0	**HW Addr En**	When this bit is set to a '1', hardware address compare is enabled. When enabled, bit 3 in the I2C_SCR register is not set. Upon a compare, the address is automatically ACKed, and upon a mismatch, the address is automatically NAKed and the hardware reverts to an idle state waiting for the next Start detection. You must configure the compare address in the I2C_ADDR register. When this bit is a '0', bit 3 of the I2C_SCR register is set and the bus stalls, and the received address is available in the I2C_DR register to enable the CPU to do a firmware address compare. The functionality of this bit is independent of the data buffering mode.

21.3.24 I2C_ADDR

I²C Slave Address Register

Individual Register Names and Addresses:

I2C_ADDR : 0,CAh

	7	6	5	4	3	2	1	0
Access : POR		RW : 00						
Bit Name		Slave Address[6:0]						

This register holds the slave's 7-bit address.

When hardware address compare mode is not enabled in the I2C_XCFG register, this register is not in use. In the table, note that the reserved bit is a grayed table cell and not described in the bit description section. Always write reserved bits with a value of '0'. For additional information, refer to the Register Definitions on page 106 in the I2C Slave chapter.

Bit	Name	Description
6:0	**Slave Address[6:0]**	These seven bits hold the slave's own device address.

21.3.25 CUR_PP

Current Page Pointer Register

Individual Register Names and Addresses:

CUR_PP : 0,D0h

	7	6	5	4	3	2	1	0
Access : POR						RW : 0		
Bit Name						Page Bits[2:0]		

This register is used to set the effective SRAM page for normal memory accesses in a multi-SRAM page enCoRe V device. It is only used when a device has more than one SRAM page.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 41 in the RAM Paging chapter.

Bit	Name	Description
2:0	**Page Bits[2:0]**	Bits determine which SRAM page is used for generic SRAM access. See the RAM Paging chapter on page 38 for more information. 000b SRAM Page 0 001b SRAM Page 1 010b SRAM Page 2 011b SRAM Page 3 100b SRAM Page 4 101b SRAM Page 5 110b SRAM Page 6 111b SRAM Page 7

21.3.26 STK_PP

Stack Page Pointer Register

Individual Register Names and Addresses:

STK_PP : 0,D1h

	7	6	5	4	3	2	1	0
Access : POR						RW : 0		
Bit Name						Page Bits[2:0]		

This register is used to set the effective SRAM page for stack memory accesses in a multi-SRAM page enCoRe V device. It is only used when a device has more than one SRAM page.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 41 in the RAM Paging chapter.

Bit	Name	Description
2:0	**Page Bits[2:0]**	Bits determine which SRAM page is used to hold the stack. See the RAM Paging chapter on page 38 for more information. 000b SRAM Page 0 001b SRAM Page 1 010b SRAM Page 2 011b SRAM Page 3 100b SRAM Page 4 101b SRAM Page 5 110b SRAM Page 6 111b SRAM Page 7

21.3.27 IDX_PP

Indexed Memory Access Page Pointer Register

Individual Register Names and Addresses:

IDX_PP : 0,D3h

	7	6	5	4	3	2	1	0
Access : POR						RW : 0		
Bit Name						Page Bits[2:0]		

This register is used to set the effective SRAM page for indexed memory accesses in a multi-SRAM page enCoRe V device.

This register is only used when a device has more than one page of SRAM. In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 41 in the RAM Paging chapter.

Bit	Name	Description
2:0	**Page Bits[2:0]**	Bits determine which SRAM page an indexed memory access operates on. See the Register Definitions on page 41 for more information on when this register is active. 000b SRAM Page 0 001b SRAM Page 1 010b SRAM Page 2 011b SRAM Page 3 100b SRAM Page 4 101b SRAM Page 5 110b SRAM Page 6 111b SRAM Page 7

21.3.28 MVR_PP

MVI Read Page Pointer Register

Individual Register Names and Addresses:

MVR_PP : 0,D4h

	7	6	5	4	3	2	1	0
Access : POR						RW : 0		
Bit Name						Page Bits[2:0]		

This register is used to set the effective SRAM page for MVI read memory accesses in a multi-SRAM page enCoRe V device.

This register is only used when a device has more than one page of SRAM. In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 41 in the RAM Paging chapter.

Bit	Name	Description
2:0	**Page Bits[2:0]**	Bits determine which SRAM page an MVI Read instruction operates on. 000b SRAM Page 0 001b SRAM Page 1 010b SRAM Page 2 011b SRAM Page 3 100b SRAM Page 4 101b SRAM Page 5 110b SRAM Page 6 111b SRAM Page 7

21.3.29 MVW_PP

MVI Write Page Pointer Register

Individual Register Names and Addresses:

MVW_PP : 0,D5h

	7	6	5	4	3	2	1	0
Access : POR						RW : 0		
Bit Name						Page Bits[2:0]		

This register is used to set the effective SRAM page for MVI write memory accesses in a multi-SRAM page enCoRe V device.

This register is only used when a device has more than one page of SRAM. In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 41 in the RAM Paging chapter.

Bit	Name	Description
2:0	**Page Bits[2:0]**	Bits determine which SRAM page an MVI Write instruction operates on. 000b SRAM Page 0 001b SRAM Page 1 010b SRAM Page 2 011b SRAM Page 3 100b SRAM Page 4 101b SRAM Page 5 110b SRAM Page 6 111b SRAM Page 7

21.3.30 I2C_CFG

I²C Configuration Register

Individual Register Names and Addresses:

I2C_CFG : 0,D6h

	7	6	5	4	3	2	1	0
Access : POR		RW : 0		RW : 0	RW : 0			RW : 0
Bit Name		PSelect		Stop IE	Clock Rate[1:0]			Enable

This register is used to set the basic operating modes, baud rate, and interrupt selection.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Always write reserved bits with a value of '0'. For additional information, refer to the Register Definitions on page 106 in the I2C Slave chapter.

Bit	Name	Description
6	**P Select**	I2C Pin Select. 0 P1[5] and P1[7]. 1 P1[0] and P1[1]. **Note** Read the I²C Slave chapter on page 103 for a discussion of the side effects of choosing the P1[0] and P1[1] pair of pins.
4	**Stop IE**	Stop Interrupt Enable. 0 Disabled. 1 Enabled. An interrupt is generated on the detection of a Stop condition.
3:2	**Clock Rate[1:0]**	00b 100K Standard Mode. 01b 400K Fast Mode. 10b 50K Standard Mode. 11b Reserved.
0	**Enable**	0 Disabled. 1 Enabled.

21.3.31 I2C_SCR

I²C Status and Control Register

Individual Register Names and Addresses:

I2C_SCR : 0,D7h

	7	6	5	4	3	2	1	0
Access : POR	RC : 0		RC : 0	RW : 0	RC : 0	RW : 0	RC : 0	RC : 0
Bit Name	Bus Error		Stop Status	ACK	Address	Transmit	LRB	Byte Complete

This register is used by the slave to control the flow of data bytes and to keep track of the bus state during a transfer.

Bits in this register are held in reset until one of the enable bits in I2C_CFG is set. In the table, note that the reserved bit is a grayed table cell and not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 106 in the I2C Slave chapter.

Bit	Name	Description
7	**Bus Error**	0 Status bit. It must be cleared by firmware by writing a '0' to the bit position. It is never cleared by the hardware. 1 A misplaced Start or Stop condition was detected.
5	**Stop Status**	0 Status bit. It must be cleared by firmware with a write of '0' to the bit position. It is never cleared by the hardware. 1 A Stop condition was detected.
4	**ACK**	Acknowledge Out. Bit is automatically cleared by hardware upon a Byte Complete event. 0 NACK the last received byte. 1 ACK the last received byte
3	**Address**	0 Status bit. It must be cleared by firmware with a write of '0' to the bit position. 1 The received byte is a slave address.
2	**Transmit**	Bit is set by firmware to define the direction of the byte transfer. Any Start detect or a write to the Start or Restart generate bits when operating in master mode also clears the bit. 0 Receive mode. 1 Transmit mode.
1	**LRB**	Last Received Bit. The value of the 9th bit in a Transmit sequence, which is the acknowledge bit from the receiver. Any Start detect or a write to the Start or Restart generate bits when operating in master mode also clears the bit. 0 Last transmitted byte was ACKed by the receiver. 1 Last transmitted byte was NAKed by the receiver.
0	**Byte Complete**	Transmit/Receive Mode: 0 No completed transmit/receive since last cleared by firmware. Any Start detect or a write to the Start or Restart generate bits when operating in master mode also clears the bit. Transmit Mode: 1 Eight bits of data have been transmitted and an ACK or NACK has been received. Receive Mode: 1 Eight bits of data have been received.

21.3.32 I2C_DR

I²C Data Register

Individual Register Names and Addresses:

I2C_DR : 0,D8h

	7	6	5	4	3	2	1	0
Access : POR	RW : 00							
Bit Name	Data[7:0]							

This register provides read/write access to the Shift register.

This register is read only for received data and write only for transmitted data. For additional information, refer to the Register Definitions on page 106 in the I2C Slave chapter.

Bit	Name	Description
7:0	**Data[7:0]**	Read received data or write data to transmit.

21.3.33 INT_CLR0

Interrupt Clear Register 0

Individual Register Names and Addresses:

INT_CLR0 : 0,DAh

	7	6	5	4	3	2	1	0
Access : POR	RW : 0	RW : 0	RW : 0	RW : 0	RW : 0	RW : 0	RW : 0	RW : 0
Bit Name	I2C	Sleep	SPI	GPIO	Timer0	Reserved	Reserved	V Monitor

This register enables the individual interrupt sources' ability to clear posted interrupts.

When bits in this register are read, a '1' is returned for every bit position that has a corresponding posted interrupt. When bits in this register are written with a '0' and ENSWINT is not set, posted interrupts are cleared at the corresponding bit positions. If there is no posted interrupt, there is no effect. When bits in this register are written with a '1' and ENSWINT is set, an interrupt is posted in the interrupt controller.

For additional information, refer to the Register Definitions on page 46 in the Interrupt Controller chapter.

Bit	Name	Description
7	**I2C**	Read 0 No posted interrupt for I2C. Read 1 Posted interrupt present for I2C. Write 0 AND ENSWINT = 0 Clear posted interrupt if it exists. Write 1 AND ENSWINT = 0 No effect. Write 0 AND ENSWINT = 1 No effect. Write 1 AND ENSWINT = 1 Post an interrupt for I2C.
6	**Sleep**	Read 0 No posted interrupt for sleep timer. Read 1 Posted interrupt present for sleep timer. Write 0 AND ENSWINT = 0 Clear posted interrupt if it exists. Write 1 AND ENSWINT = 0 No effect. Write 0 AND ENSWINT = 1 No effect. Write 1 AND ENSWINT = 1 Post an interrupt for sleep timer.
5	**SPI**	Read 0 No posted interrupt for SPI. Read 1 Posted interrupt present for SPI. Write 0 AND ENSWINT = 0 Clear posted interrupt if it exists. Write 1 AND ENSWINT = 0 No effect. Write 0 AND ENSWINT = 1 No effect. Write 1 AND ENSWINT = 1 Post an interrupt for SPI.
4	**GPIO**	Read 0 No posted interrupt for general-purpose inputs and outputs (GPIO) (pins). Read 1 Posted interrupt present for GPIO (pins). Write 0 AND ENSWINT = 0 Clear posted interrupt if it exists. Write 1 AND ENSWINT = 0 No effect. Write 0 AND ENSWINT = 1 No effect. Write 1 AND ENSWINT = 1 Post an interrupt for general-purpose inputs and outputs (pins).

(continued on next page)

21.3.33 INT_CLR0 *(continued)*

3	**Timer0**	Read 0 No posted interrupt for Timer. Read 1 Posted interrupt present for Timer. Write 0 AND ENSWINT = 0 Clear posted interrupt if it exists. Write 1 AND ENSWINT = 0 No effect. Write 0 AND ENSWINT = 1 No effect. Write 1 AND ENSWINT = 1 Post an interrupt for Timer.
0	**V Monitor**	Read 0 No posted interrupt for Supply Voltage Monitor. Read 1 Posted interrupt present for Supply Voltage Monitor. Write 0 AND ENSWINT = 0 Clear posted interrupt if it exists. Write 1 AND ENSWINT = 0 No effect. Write 0 AND ENSWINT = 1 No effect. Write 1 AND ENSWINT = 1 Post an interrupt for Supply Voltage Monitor.

21.3.34 INT_CLR1

Interrupt Clear Register 1

Individual Register Names and Addresses:

INT_CLR1 : 0,DBh

	7	6	5	4	3	2	1	0
Access : POR	RW : 0	RW : 0	RW : 0	RW : 0	RW : 0	RW : 0	RW : 0	RW : 0
Bit Name	Endpoint3	Endpoint2	Endpoint1	Endpoint0	USB_SOF	USB_BUS_RST	Timer2	Timer1

This register is used to enable the individual interrupt sources' ability to clear posted interrupts.

When bits in this register are read, a '1' is returned for every bit position that has a corresponding posted interrupt. When bits in this register are written with a '0' and ENSWINT is not set, posted interrupts are cleared at the corresponding bit positions. If there is no posted interrupt, there is no effect. When bits in this register are written with a '1' and ENSWINT is set, an interrupt is posted in the interrupt controller.

For additional information, refer to the Register Definitions on page 46 in the Interrupt Controller chapter.

Bit	Name	Description
7	**Endpoint3**	Read 0 No posted interrupt for USB Endpoint3. Read 1 Posted interrupt present for USB Endpoint3. Write 0 AND ENSWINT = 0 Clear posted interrupt if it exists. Write 1 AND ENSWINT = 0 No effect. Write 0 AND ENSWINT = 1 No effect. Write 1 AND ENSWINT = 1 Post an interrupt for USB Endpoint3.
6	**Endpoint2**	Read 0 No posted interrupt for USB Endpoint2. Read 1 Posted interrupt present for USB Endpoint2. Write 0 AND ENSWINT = 0 Clear posted interrupt if it exists. Write 1 AND ENSWINT = 0 No effect. Write 0 AND ENSWINT = 1 No effect. Write 1 AND ENSWINT = 1 Post an interrupt for USB Endpoint2.
5	**Endpoint1**	Read 0 No posted interrupt for USB Endpoint1. Read 1 Posted interrupt present for USB Endpoint1. Write 0 AND ENSWINT = 0 Clear posted interrupt if it exists. Write 1 AND ENSWINT = 0 No effect. Write 0 AND ENSWINT = 1 No effect. Write 1 AND ENSWINT = 1 Post an interrupt for USB Endpoint1.

(continued on next page)

21.3.34 INT_CLR1 (*continued*)

4	**Endpoint0**	Read 0 No posted interrupt for USB Endpoint0. Read 1 Posted interrupt present for USB Endpoint0. Write 0 AND ENSWINT = 0 Clear posted interrupt if it exists. Write 1 AND ENSWINT = 0 No effect. Write 0 AND ENSWINT = 1 No effect. Write 1 AND ENSWINT = 1 Post an interrupt for USB Endpoint0.
3	**USB_SOF**	Read 0 No posted interrupt for USB Start of Frame (SOF). Read 1 Posted interrupt present for USB Start of Frame (SOF). Write 0 AND ENSWINT = 0 Clear posted interrupt if it exists. Write 1 AND ENSWINT = 0 No effect. Write 0 AND ENSWINT = 1 No effect. Write 1 AND ENSWINT = 1 Post an interrupt for USB Start of Frame (SOF).
2	**USB_BUS_RST**	Read 0 No posted interrupt for USB Bus Reset. Read 1 Posted interrupt present for USB Bus Reset. Write 0 AND ENSWINT = 0 Clear posted interrupt if it exists. Write 1 AND ENSWINT = 0 No effect. Write 0 AND ENSWINT = 1 No effect. Write 1 AND ENSWINT = 1 Post an interrupt for USB Bus Reset.
1	**Timer2**	Read 0 No posted interrupt for Timer2. Read 1 Posted interrupt present for Timer2. Write 0 AND ENSWINT = 0 Clear posted interrupt if it exists. Write 1 AND ENSWINT = 0 No effect. Write 0 AND ENSWINT = 1 No effect. Write 1 AND ENSWINT = 1 Post an interrupt for Timer2.
0	**Timer1**	Read 0 No posted interrupt for Timer1 Read 1 Posted interrupt present for Timer1. Write 0 AND ENSWINT = 0 Clear posted interrupt if it exists. Write 1 AND ENSWINT = 0 No effect. Write 0 AND ENSWINT = 1 No effect. Write 1 AND ENSWINT = 1 Post an interrupt for Timer1.

21.3.35 INT_CLR2

Interrupt Clear Register 2

Individual Register Names and Addresses:

INT_CLR2 : 0,DCh

	7	6	5	4	3	2	1	0
Access : POR			RW : 0	RW : 0	RW : 0	RW : 0	RW : 0	RW : 0
Bit Name			USB_WAKE	Endpoint8	Endpoint7	Endpoint6	Endpoint5	Endpoint4

This register is used to enable the individual interrupt sources' ability to clear posted interrupts.

When bits in this register are read, a '1' is returned for every bit position that has a corresponding posted interrupt. When bits in this register are written with a '0' and ENSWINT is not set, posted interrupts are cleared at the corresponding bit positions. If there is no posted interrupt, there is no effect. In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. When bits in this register are written with a '1' and ENSWINT is set, an interrupt is posted in the interrupt controller. For additional information, refer to the Register Definitions on page 46 in the Interrupt Controller chapter.

Bit	Name	Description
5	**USB_WAKE**	Read 0 No posted interrupt for USB Wake. Read 1 Posted interrupt present for USB Wake. Write 0 AND ENSWINT = 0 Clear posted interrupt if it exists. Write 1 AND ENSWINT = 0 No effect. Write 0 AND ENSWINT = 1 No effect. Write 1 AND ENSWINT = 1 Post an interrupt for USB Wake.
4	**Endpoint8**	Read 0 No posted interrupt for USB Endpoint8. Read 1 Posted interrupt present for USB Endpoint8. Write 0 AND ENSWINT = 0 Clear posted interrupt if it exists. Write 1 AND ENSWINT = 0 No effect. Write 0 AND ENSWINT = 1 No effect. Write 1 AND ENSWINT = 1 Post an interrupt for USB Endpoint8.

(continued on next page)

21.3.35 INT_CLR2 (*continued*)

3	**Endpoint7**	Read 0 No posted interrupt for USB Endpoint7. Read 1 Posted interrupt present for USB Endpoint7. Write 0 AND ENSWINT = 0 Clear posted interrupt if it exists. Write 1 AND ENSWINT = 0 No effect. Write 0 AND ENSWINT = 1 No effect. Write 1 AND ENSWINT = 1 Post an interrupt for USB Endpoint7.
2	**Endpoint6**	Read 0 No posted interrupt for USB Endpoint6. Read 1 Posted interrupt present for USB Endpoint6. Write 0 AND ENSWINT = 0 Clear posted interrupt if it exists. Write 1 AND ENSWINT = 0 No effect. Write 0 AND ENSWINT = 1 No effect. Write 1 AND ENSWINT = 1 Post an interrupt for USB Endpoint6.
1	**Endpoint5**	Read 0 No posted interrupt for USB Endpoint5. Read 1 Posted interrupt present for USB Endpoint5. Write 0 AND ENSWINT = 0 Clear posted interrupt if it exists. Write 1 AND ENSWINT = 0 No effect. Write 0 AND ENSWINT = 1 No effect. Write 1 AND ENSWINT = 1 Post an interrupt for USB Endpoint5
0	**Endpoint4**	Read 0 No posted interrupt for USB Endpoint4. Read 1 Posted interrupt present for USB Endpoint4. Write 0 AND ENSWINT = 0 Clear posted interrupt if it exists. Write 1 AND ENSWINT = 0 No effect. Write 0 AND ENSWINT = 1 No effect. Write 1 AND ENSWINT = 1 Post an interrupt for USB Endpoint4.

21.3.36 INT_MSK2

Interrupt Mask Register 2

Individual Register Names and Addresses:

INT_MSK2 : 0,DEh

	7	6	5	4	3	2	1	0
Access : POR			RW : 0	RW : 0	RW : 0	RW : 0	RW : 0	RW : 0
Bit Name			USB Wakeup	Endpoint8	Endpoint7	Endpoint6	Endpoint5	Endpoint4

This register enables the individual sources' ability to create pending interrupts.

When an interrupt is masked off, the mask bit is '0'. The interrupt continues to post in the interrupt controller. Clearing the mask bit only prevents a posted interrupt from becoming a pending interrupt. In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 46 in the Interrupt Controller chapter.

Bit	Name		Description
5	**USB Wakeup**	0 1	Mask USB Wakeup interrupt. Unmask USB Wakeup interrupt.
4	**Endpoint8**	0 1	Mask USB Endpoint8 interrupt. Unmask USB Endpoint8 interrupt.
3	**Endpoint7**	0 1	Mask USB Endpoint7 interrupt. Unmask USB Endpoint7 interrupt.
2	**Endpoint6**	0 1	Mask USB Endpoint6 interrupt. Unmask USB Endpoint6 interrupt.
1	**Endpoint5**	0 1	Mask USB Endpoint5 interrupt. Unmask USB Endpoint5 interrupt.
0	**Endpoint4**	0 1	Mask USB Endpoint4 interrupt. Unmask USB Endpoint4 interrupt.

21.3.37 INT_MSK1

Interrupt Mask Register 1

Individual Register Names and Addresses:

INT_MSK1 : 0,DFh

	7	6	5	4	3	2	1	0
Access : POR	RW : 0	RW : 0	RW : 0	RW : 0	RW : 0	RW : 0	RW : 0	RW : 0
Bit Name	Endpoint3	Endpoint2	Endpoint1	Endpoint0	USB SOF	USB Bus Reset	Timer2	Timer1

This register enables the individual sources' ability to create pending interrupts.

When an interrupt is masked off, the mask bit is '0'. The interrupt continues to post in the interrupt controller. Clearing the mask bit only prevents a posted interrupt from becoming a pending interrupt.

For additional information, refer to the Register Definitions on page 46 in the Interrupt Controller chapter.

Bit	Name	Description	
7	**Endpoint3**	0 1	Mask USB Endpoint3 interrupt. Unmask USB Endpoint3 interrupt.
6	**Endpoint2**	0 1	Mask USB Endpoint2 interrupt. Unmask USB Endpoint2 interrupt.
5	**Endpoint1**	0 1	Mask USB Endpoint1 interrupt. Unmask USB Endpoint1 interrupt.
4	**Endpoint0**	0 1	Mask USB Endpoint0 interrupt. Unmask USB Endpoint0 interrupt.
3	**USB SOF**	0 1	Mask USB SOF interrupt. Unmask USB SOF interrupt.
2	**USB Bus Reset**	0 1	Mask USB Bus Reset interrupt. Unmask USB Bus Reset interrupt.
1	**Timer2**	0 1	Mask Timer2 interrupt. Unmask Timer2 interrupt.
0	**Timer1**	0 1	Mask Timer1 interrupt. Unmask Timer1 interrupt.

21.3.38 INT_MSK0

Interrupt Mask Register 0

Individual Register Names and Addresses:

INT_MSK0 : 0,E0h

	7	6	5	4	3	2	1	0
Access : POR	RW : 0	RW : 0	RW : 0	RW : 0	RW : 0	RW : 0	RW : 0	RW : 0
Bit Name	I2C	Sleep	SPI	GPIO	Timer0	Reserved	Reserved	V Monitor

This register enables the individual sources' ability to create pending interrupts.

When an interrupt is masked off, the mask bit is '0'. The interrupt continues to post in the interrupt controller. Clearing the mask bit only prevents a posted interrupt from becoming a pending interrupt. For additional information, refer to the Register Definitions on page 46 in the Interrupt Controller chapter.

Bit	Name	Description	
7	**I2C**	0 1	Mask I2C interrupt. Unmask I2C interrupt.
6	**Sleep**	0 1	Mask Sleep interrupt. Unmask Sleep interrupt.
5	**SPI**	0 1	Mask SPI interrupt. Unmask SPI interrupt.
4	**GPIO**	0 1	Mask GPIO interrupt. Unmask GPIO interrupt.
3	**Timer0**	0 1	Mask Timer0 interrupt. Unmask Timer0 interrupt.
0	**V Monitor**	0 1	Mask Voltage Monitor interrupt. Unmask Voltage Monitor interrupt.

21.3.39 INT_SW_EN

Interrupt Software Enable Register

Individual Register Names and Addresses:

INT_SW_EN : 0,E1h

	7	6	5	4	3	2	1	0
Access : POR								RW : 0
Bit Name								ENSWINT

This register is used to enable software interrupts.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 46 in the Interrupt Controller chapter.

Bit	Name	Description	
0	**ENSWINT**	0	Disable software interrupts.
		1	Enable software interrupts.

21.3.40 INT_VC

Interrupt Vector Clear Register

Individual Register Names and Addresses:

INT_VC : 0,E2h

	7	6	5	4	3	2	1	0
Access : POR	RC : 00							
Bit Name	Pending Interrupt[7:0]							

This register returns the next pending interrupt and clears all pending interrupts when written.

For additional information, refer to the Register Definitions on page 46 in the Interrupt Controller chapter.

Bit	Name	Description	
7:0	**Pending Interrupt[7:0]**	Read	Returns vector for highest priority pending interrupt.
		Write	Clears all pending and posted interrupts.

21.3.41 RES_WDT

Reset Watchdog Timer Register

Individual Register Names and Addresses:

RES_WDT : 0,E3h

	7	6	5	4	3	2	1	0
Access : POR	W : 00							
Bit Name	WDSL_Clear[7:0]							

This register is used to clear the watchdog timer alone, or clear both the watchdog timer and the sleep timer together.

For additional information, refer to the Register Definitions on page 82 in the Sleep and Watchdog chapter.

Bit	Name	Description
7:0	**WDSL_Clear[7:0]**	Any write clears the watchdog timer. A write of 38h clears both the watchdog and sleep timers.

21.3.42 CPU_F

M8C Flag Register

Individual Register Names and Addresses:

CPU_F : x,F7h

	7	6	5	4	3	2	1	0
Access : POR	RL : 0		RL : 0	RL : 0		RL : 0	RL : 0	RL : 0
Bit Name	PgMode[1:0]		BINC	XIO		Carry	Zero	GIE

This register provides read access to the M8C flags.

The AND f, expr; OR f, expr; and XOR f, expr flag instructions are used to modify this register. In the table, note that the reserved bit is a grayed table cell and is not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 31 in the M8C chapter and the Register Definitions on page 46 in the Interrupt Controller chapter.

Bit	Name	Description	
7:6	**PgMode[1:0]**	00b	Direct Address mode and Indexed Address mode operands are referred to RAM Page 0, regardless of the values of CUR_PP and IDX_PP. Note that this condition prevails upon entry to an Interrupt Service Routine when the CPU_F register is cleared.
		01b	Direct Address mode instructions are referred to Page 0. Indexed Address mode instructions are referred to the RAM page specified by the stack page pointer, STK_PP.
		10b	Direct Address mode instructions are referred to the RAM page specified by the current page pointer, CUR_PP. Indexed Address mode instructions are referred to the RAM page specified by the index page pointer, IDX_PP.
		11b	Direct Address mode instructions are referred to the RAM page specified by the current page pointer, CUR_PP. Indexed Address mode instructions are referred to the RAM page specified by the stack page pointer, STK_PP.
5	**BINC**	Bit Implemented Not Connected.	
4	**XIO**	0	Normal register address space.
		1	Extended register address space. Primarily used for configuration.
2	**Carry**	Set by the M8C CPU Core to indicate whether there was a carry in the previous logical/arithmetic operation.	
		0	No carry.
		1	Carry.

(continued on next page)

21.3.42 CPU_F *(continued)*

1	**Zero**	Set by the M8C CPU Core to indicate whether there was a zero result in the previous logical/arithmetic operation.	
		0	Not equal to zero.
		1	Equal to zero.
0	**GIE**	0	M8C does not process any interrupts.
		1	Interrupt processing enabled.

21.3.43 CPU_SCR1

System Status and Control Register 1

Individual Register Names and Addresses:

CPU_SCR1: x,FEh

	7	6	5	4	3	2	1	0
Access : POR	R : 0			RW : 0				RW : 0
Bit Name	IRESS			SLIMO[1:0]				IRAMDIS

This register is used to convey the status and control of events related to internal resets and watchdog reset.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 116 in the System Resets chapter.

Bit	Name	Description
7	**IRESS**	This bit is read only. 0 Boot phase only executed once. 1 Boot phase occurred multiple times.
4:3	**SLIMO[1:0]**	These bits set the frequency range for the IMO. **Note** When changing from the default setting, the corresponding trim value must be loaded into the IMO_TR register for highest frequency accuracy. **SLIMO** **CY7C6xxxx** 00 12 01 6 10 24 11 Reserved
0	**IRAMDIS**	0 SRAM is initialized to 00h after POR, XRES, and WDR. 1 Addresses 03h - D7h of SRAM Page 0 are not modified by WDR.

21.3.44 CPU_SCR0

System Status and Control Register 0

Individual Register Names and Addresses:

CPU_SCR0 : x,FFh

	7	6	5	4	3	2	1	0
Access : POR	R : 0		RC : 0	RC : 1	RW : 0			RW : 0
Bit Name	GIES		WDRS	PORS	Sleep			STOP

This register is used to convey the status and control of events for various functions of a enCoRe V device.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 116 in the System Resets chapter.

Bit	Name	Description
7	**GIES**	Global Interrupt Enable Status. It is recommended that the user read the Global Interrupt Enable Flag bit from the CPU_F register on page 208. This bit is read only for GIES. Its use is discouraged, as the Flag register is now readable at address x,F7h (read only).
5	**WDRS**	Watchdog Reset Status. This bit may not be set by user code; however, it may be cleared by writing a '0'. 0 No watchdog reset has occurred. 1 Watchdog reset has occurred.
4	**PORS**	Power-On-Reset Status. This bit may not be set by user code; however, it may be cleared by writing a '0'. 0 Power-on-reset has not occurred and watchdog timer is enabled. 1 Is set after external reset or power-on-reset.
3	**Sleep**	Set by the user to enable the CPU sleep state. CPU remains in Sleep mode until any interrupt is pending. 0 Normal operation. 1 Sleep.
0	**STOP**	0 M8C is free to execute code. 1 M8C is halted and is only cleared by POR, XRES, or WDR.

21.4 Bank 1 Registers

The following registers are all in bank 1 and are listed in address order. Registers that are in both Bank 0 and Bank 1 are listed in address order in the section titled Bank 0 Registers on page 164.

21.4.1 PRTxDM0

Port Drive Mode Bit Registers 0

Individual Register Names and Addresses:

PRT0DM0 : 1,00h PRT1DM0 : 1,04h PRT2DM0 : 1,08h PRT3DM0 : 1,0Ch
PRT4DM0 : 1,10h

	7	6	5	4	3	2	1	0
Access : POR	RW : 00							
Bit Name	Drive Mode 0[7:0]							

This register is one of two registers where the combined value determines the unique drive mode of each bit in a GPIO port.

In register PRTxDM0 there are four possible drive modes for each port pin. Two mode bits are required to select one of these modes, and these two bits are spread into two different registers (PRTxDM0 and PRTxDM1 on page 213). The bit position of the effected port pin (for example, Pin[2] in Port 0) is the same as the bit position of each of the two Drive Mode register bits that control the drive mode for that pin (for example, bit[2] in PRT0DM0 and bit[2] in PRT0DM1). The two bits from the two registers are treated as a group. These are referred to as DM1 and DM0, or together as DM[1:0].

All drive mode bits are shown in the sub-table ([1**0**] refers to the combination (in order) of bits in a given bit position); however, this register only controls the ***least significant bit (LSb)*** of the drive mode.

The upper nibble of the PRT4DM0 register returns the last data bus value when read. You need to mask it off prior to using this information. For additional information, refer to the Register Definitions on page 56 in the GPIO chapter.

Bit	Name	Description
7:0	**Drive Mode 0[7:0]**	Bit 0 of the drive mode, for each of 8-port pins, for a GPIO port.

[1**0**]	Pin Output High	Pin Output Low	Notes
0**0**b	Resistive	Strong	
0**1**b	Strong	Strong	
1**0**b	High-Z	High-Z	Reset state. Digital input disabled for zero power.
1**1**b	High-Z	Strong	I2C compatible mode. For digital inputs, use this mode with data bit (PRTxDR register) set high.

Note A bold digit in the table signifies that the digit is used in this register.

21.4.2 PRTxDM1

Port Drive Mode Bit Registers 1

Individual Register Names and Addresses:

PRT0DM1 : 1,01h PRT1DM1 : 1,05h PRT2DM1 : 1,09h PRT3DM1 : 1,0Dh
PRT4DM1 : 1,11h

	7	6	5	4	3	2	1	0
Access : POR	RW : FF							
Bit Name	Drive Mode 1[7:0]							

This register is one of three registers where the combined value determines the unique drive mode of each bit in a GPIO port.

In register PRTxDM1 there are four possible drive modes for each port pin. Two mode bits are required to select one of these modes, and these two bits are spread into two different registers (PRTxDM1 and PRTxDM0 on page 212). The bit position of the effected port pin (for example, Pin[2] in Port 0) is the same as the bit position of each of the two Drive Mode register bits that control the drive mode for that pin (for example, bit[2] in PRT0DM0 and bit[2] in PRT0DM1). The two bits from the two registers are treated as a group. These are referred to as DM1 and DM0, or together as DM[1:0].

All drive mode bits are shown in the sub-table ([**1**0] refers to the combination (in order) of bits in a given bit position); however, this register only controls the most significant bit (MSb) of the drive mode.

The upper nibble of the PRT4DM1 register returns the last data bus value when read. You need to mask it off before using this information. For additional information, refer to the Register Definitions on page 56 in the GPIO chapter.

Bit	Name	Description
7:0	**Drive Mode 1[7:0]**	Bit 1 of the drive mode, for each of 8-port pins, for a GPIO port.

[10**]**	**Pin Output High**	**Pin Output Low**	**Notes**
00b	Resistive	Strong	
01b	Strong	Strong	
10b	High-Z	High-Z	Reset state. Digital input disabled for zero power.
11b	High-Z	Strong	I2C compatible mode. For digital inputs, use this mode with data bit (PRTxDR register) set high.

Note A bold digit in the table signifies that the digit is used in this register.

21.4.3 SPI_CFG

SPI Configuration Register

Individual Register Names and Addresses:

SPI_CFG : 1,29h

	7	6	5	4	3	2	1	0
Access : POR	RW : 0			RW : 0	RW : 0	RW : 0	RW : 0	RW : 0
Bit Name	Clock Sel [2:0]			Bypass	SS_	SS_EN_	Int Sel	Slave

This register is used to configure the SPI.

Do not change the values in this register while the block is enabled. For additional information, refer to the Register Definitions on page 125 in the SPI chapter.

Bit	Name	Description
7:5	**Clock Sel [2:0]**	SYSCLK in Master mode. 000b / 2 001b / 4 010b / 8 011b / 16 100b / 32 101b / 64 110b / 128 111b / 256
4	**Bypass**	Bypass Synchronization. 0 All pin inputs are doubled and synchronized. 1 Input synchronization is bypassed.
3	**SS_**	Slave Select in Slave mode. 0 Slave selected. 1 Slave not selected.
2	**SS_EN_**	Internal Slave Select Enable. 0 Slave selection determined from SS_ bit. 1 Slave selection determined from external SS_ pin.
1	**Int Sel**	Interrupt Select. 0 Interrupt on TX Reg Empty. 1 Interrupt on SPI Complete.
0	**Slave**	0 Operates as a master. 1 Operates as a slave.

21.4.4 USB_CR1

USB Control Register 1

Individual Register Names and Addresses:

USB_CR1 : 1,30h

	7	6	5	4	3	2	1	0
Access : POR						RC : 0	RW : 0	RW : 0
Bit Name						BusActivity	EnableLock	RegEnable

This register is used to configure the internal regulator and the oscillator tuning capability.

This register is only used by the CY8C20XX6A/AS/LCY7C64215 enCoRe V devices. In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 147 in the Full-Speed USB chapter.

Bit	Name	Description
2	**BusActivity**	Monitors activity on USB bus. This bit is only set by the hardware. Writing a '0' clears this bit. Writing a '1' preserves its present state. 0 No activity. 1 Non-idle activity (D+ = Low) was detected since the last time the bit was cleared.
1	**EnableLock**	Controls the automatic tuning of the internal oscillator. Hardware locks the internal oscillator based on the frequency of incoming USB data when this bit is set. Normally, this is set when the device is used in a USB application. 0 Locking disabled. 1 Locking enabled.
0	**RegEnable**	Configures USB regulator for appropriate power supply range. 0 Pass-through mode. Use for Vdd = 3.3-V range. 1 Regulating mode. Use for Vdd = 5-V range.

21.4.5 PMAx_WA

PSoC Memory Arbiter Write Address Registers

Individual Register Names and Addresses:

PMA0_WA : 1,34h	PMA1_WA : 1,35h	PMA2_WA : 1,36h	PMA3_WA : 1,37h
PMA4_WA : 1,38h	PMA5_WA : 1,39h	PMA6_WA : 1,3Ah	PMA7_WA : 1,3Bh
PMA8_WA : 1,44H	PMA9_WA : 1,45H	PMA10_WA : 1,46H	PMA11_WA : 1,47H
PMA12_WA : 1, 48H	PMA13_WA : 1,49h	PMA14_WA : 1,4Ah	PMA15_WA : 1,4Bh

	7	6	5	4	3	2	1	0
Access : POR	RW : 00							
Bit Name	Write Address[7:0]							

These registers are PSoC Memory Arbiter write address registers.

For additional information, refer to the Register Definitions on page 147 in the Full-Speed USB chapter.

Bit	Name	Description
7:0	**Write Address[7:0]**	The value returned when this register is read depends on whether the PMA channel is being used by the USB SIE or by the M8C.

21.4.6 PMAx_RA

PSoC Memory Arbiter Read Address Registers

Individual Register Names and Addresses:

PMA0_RA : 1,3Ch	PMA1_RA : 1,3Dh	PMA2_RA : 1,3Eh	PMA3_RA : 1,3Fh
PMA4_RA : 1,40h	PMA5_RA : 1,41h	PMA6_RA : 1,42h	PMA7_RA : 1,43h
PMA8_RA : 1,4Ch	PMA9_RA : 1,4Dh	PMA10_RA : 1,4Eh	PMA11_RA : 1,4Fh
PMA12_RA : 1, 50h	PMA13_RA : 1,51h	PMA14_RA : 1,52h	PMA15_RA : 1,53h

	7	6	5	4	3	2	1	0
Access : POR	RW : 00							
Bit Name	Read Address[7:0]							

These registers are PSoC Memory Arbiter read address registers.

For additional information, refer to the Register Definitions on page 147 in the Full-Speed USB chapter.

Bit	Name	Description
7:0	**ReadAddress[7:0]**	The value returned when this register is read depends on whether the PMA channel is being used by the USB SIE or by the M8C. In the USB case, this register always returns the beginning SRAM address for the PMA channel.

21.4.7 EPx_CR0

Endpoint Control Registers 0

Individual Register Names and Addresses:

EP1_CR0 : 1,54h
EP2_CR0 : 1,55h
EP3_CR0 : 1,56h
EP4_CR0 : 1,57h
EP5_CR0 : 1,58h
EP6_CR0 : 1,59h
EP7_CR0 : 1,5Ah
EP7_CR0 : 1,5Bh

	7	6	5	4	3	2	1	0
Access : POR	RW : 0		RW : 0	RC : 0	RW : 0			
Bit Name	Stall		NAK_INT_EN	ACKed Tx	Mode[3:0]			

These registers endpoint control registers.

In the table, note that the reserved bit is a grayed table cell and is not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 147 in the Full-Speed USB chapter.

Bit	Name	Description
7	**Stall**	When this bit is set, the SIE stalls an OUT packet if the Mode bits are set to ACK-OUT. The SIE stalls an IN packet if the Mode bits are set to ACK-IN. This bit must be clear for all other modes.
5	**NAK_INT_EN**	When set, this bit causes an endpoint interrupt to be generated even when a transfer completes with a NAK.
4	**ACKed Tx**	The ACKed transaction bit is set whenever the SIE engages in a transaction to the register's endpoint that completes with an ACK packet.
3:0	**Mode[3:0]**	The mode controls how the USB SIE responds to traffic and how the USB SIE changes the mode of that endpoint as a result of host packets to the endpoint.

21.4.8 TMP_DRx

Temporary Data Registers

Individual Register Names and Addresses:

TMP_DR0 : x,6Ch TMP_DR1 : x,6Dh TMP_DR2 : x,6Eh TMP_DR3 : x,6Fh

	7	6	5	4	3	2	1	0
Access : POR	RW : 00							
Bit Name	Data[7:0]							

These registers enhance the performance in multiple SRAM page enCoRe V devices.

All bits in this register are reserved for enCoRe V devices with 256 bytes of SRAM. For additional information, refer to the Register Definitions on page 41 in the RAM Paging chapter.

Bit	Name	Description
7:0	**Data[7:0]**	General-purpose register space

21.4.9 USB_MISC_CR

USB Miscellaneous Control Register

Individual Register Names and Addresses:

USB_MISC_CR: 1,BDh

	7	6	5	4	3	2	1	0
Access : POR						RW : 0	RW : 0	RW : 0
Bit Name						USB_SE_EN	USB_ON	USB_CLK_ON

The USB Miscellaneous Control Register controls the clocks to the USB block to make IMO work with better accuracy for the USB part and to disable the single-ended input of USBIO in the case of a non-USB part.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of ‘0’. For additional information, refer to the Register Definitions on page 99 in the Digital Clocks chapter.

Bit	Name	Description
2	**USE_SE_EN**	The single-ended outputs of USBIO is enabled or disabled based upon this bit setting. Set this bit to '1' when using this part as a USB part. 0 The single-ended outputs of USBIO are disabled. The DPO and DMO is held at logic high state and RSEO is held at a low state. 1 The single-ended output of USBIO is enabled and USB transactions can occur. **Note** Bit [1:0] of the USBIO_CR1 register is also affected by this register setting. When this bit is '0' (default) regardless of the DP and DM state, the DPO and DMO bits of USBIO_CR1 are '11b'.
1	**USB_ON**	This bit is used by the IMO DAC block to either work with better DNL consuming higher power, or with sacrificed DNL consuming lower power. Set this bit to '1' when the part is used as a USB part. 0 The IMO runs with sacrificed DNL by consuming less power. 1 The IMO runs with better DNL by consuming more power.
0	**USB_CLK_ON**	This bit either enables or disables the clocks to the USB block. It is used to save power in cases when the device need not respond to USB traffic. Set this bit to '1' when the device is used as a USB part. 0 All clocks to the USB block are driven as '0'. The device does not respond to USB traffic and none of the USB registers, except IMO_TR, IMO_TR1 and USBIO_CR1, listed in the Register Definitions on page 147 are writable. 1 Clocks are not blocked to the USB block. The device responds to USB traffic depending upon the other register settings mentioned in Register Definitions on page 147 in the Full-Speed USB chapter.

21.4.10 ECO_ENBUS

External Oscillator ENBUS Register

Individual Register Names and Addresses:

ECO_ENBUS : 1,D2h

	7	6	5	4	3	2	1	0
Access : POR						RW : 7		
Bit Name						ECO_ENBUS[2:0]		

The ECO_ENBUS register is used to disable and enable the external crystal oscillator (ECO). In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits should always be written with a value of '0'. See the Application Overview on page 75 for the proper sequence for enabling the ECO.

Bits	Name	Description
2:0	**ECO_ENBUS[2:0]**	These bits should be written with a value of 011b to allow the ECO to be enabled by bits in the ECO_CFG register, or a value of 111b (default) to disable the ECO. Other values in this register are reserved.

21.4.11 ECO_TRIM

External Oscillator Trim Register

Individual Register Names and Addresses:

ECO_TRIM : 1,D3h

	7	6	5	4	3	2	1	0
Access : POR				RW : 4			RW : 1	
Bit Name				ECO_XGM[2:0]			ECO_LP[1:0]	

This register trims the external oscillator gain and power settings.

These settings in this register should not be changed from their default state.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 91 in the I/O Analog Multiplexer chapter.

Bits	Name	Description
4:2	**ECO_XGM[2:0]**	These bits set the amplifier gain. The high-power mode (ECO_LPM=0) step size of the current source is approximately 400 nA. In low-power mode (ECO_LPM=1), the overall power is approximately 5% lower with the '000' setting than with the '111' setting. 000 Highest gain setting. Lowest power setting in low-power mode. 111 Lowest gain setting. 101 Typical Value (factory trimmed).
1:0	**ECO_LP[1:0]**	These regulate low-power mode settings. 00 Highest Power Setting. 11 Lowest power setting (30% power reduction). 00 Default value.

21.4.12 MUX_CRx

Analog Mux Port Bit Enable Registers

Individual Register Names and Addresses:

MUX_CR0 : 1,D8h MUX_CR1 : 1,D9h MUX_CR2 : 1,DAh MUX_CR3 : 1,DBh
MUX_CR4 : 1,DFh

	7	6	5	4	3	2	1	0
Access : POR	RW : 00							
Bit Name	ENABLE[7:0]							

This register is used to control the connection between the analog mux bus and the corresponding pin.

Port 4 is a 4-bit port, so the upper 4 bits of the MUX_CR4 register are reserved and return zeros when read.

For additional information, refer to the Register Definitions on page 91 in the I/O Analog Multiplexer chapter.

Bits	Name	Description
7:0	**ENABLE[7:0]**	Each bit controls the connection between the analog mux bus and the corresponding port pin. For example, MUX_CR2[3] controls the connection to bit 3 in Port 2. Any number of pins may be connected at the same time. 0 No connection between port pin and analog mux bus. 1 Connect port pin to analog mux bus.

21.4.13 IO_CFG1

Input/Output Configuration Register 1

Individual Register Names and Addresses:

IO_CFG1 : 1,DCh

	7	6	5	4	3	2	1	0
Access : POR	RW : 0		RW : 0		RW : 0	RW : 0	RW : 0	RW : 0
Bit Name	StrongP		Range[1:0]		P1_LOW_THRS	SPICLK_ON_P10	REG_EN	IO INT

This register is used to configure the Port 1 output regulator and set the interrupt mode for all GPIO.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 56 in the GPIO chapter.

Bits	Name	Description
7	**StrongP**	Setting this bit increases the drive strength and edge ratio for high outputs.
5:4	**Range[1:0]**	Selects the high output level for Port 1 outputs. 00 3.0 volts 01 3.0 volts 10 1.8 volts 11 2.5 volts
3	**P1-LOW_THRS**	This bit reduces the threshold voltage of the P1 port input buffers so that there are no compatibility issues when Port 1 is communicating at regulated voltage levels. 0 Standard threshold of VIH, VIL 1 Reduce threshold of VIH, VIL
2	**SPICLK_ON_P10**	When set to '1', the SPI clock is mapped to Port 1 pin 0. Otherwise, it is mapped to Port 1 pin 3.
1	**REG_EN**	Controls the regulator on Port 1 outputs. 0 Regulator disabled, so Port 1 strong outputs drive to Vdd. 1 Regulator enabled, so Port 1 strong outputs drive to approximately 3 V (for Vdd > 3 V).
0	**IO INT**	Sets the GPIO interrupt mode for all pins in the enCoRe V device. GPIO interrupts are also controlled at each pin by the PRTxIE registers, and by the global GPIO bit in the INT_MSK0 register. 0 GPIO interrupt configured for interrupt when pin is low. 1 GPIO interrupt configured for interrupt when pin state changes from last time port was read.

21.4.14 OUT_P1

Output Override to Port 1 Register

Individual Register Names and Addresses:

OUT_P1: 1,DDh

	7	6	5	4	3	2	1	0
Access : POR	RW : 0	RW : 0				RW : 0		RW : 0
Bit Name	P16D	P16EN	RSVD	RSVD	RSVD	P12EN	RSVD	P10EN

This register enables specific internal signals to be output to Port 1 pins.

The GPIO drive modes must be specified to support the desired output mode (registers PRT1DM1 and PRT1DM0). If a pin is enabled for output by a bit in this register, the corresponding signal has priority over any other internal function that may be configured to output to that pin. Reserved bits must always be written with a value of '0'.

For additional information, refer to the Register Definitions on page 99 in the Digital Clocks chapter.

Bit	Name	Description
7	**P16D**	Bit selects the data output to P1[6] when P16EN is high. 0 Select Timer output (TIMEROUT) 1 Select CLK32
6	**P16EN**	Bit enables pin P1[6] for output of the signal selected by the P16D bit. 0 No internal signal output to P1[6] 1 Output the signal selected by P16D to P1[6]
2	**P12EN**	Bit enables pin P1[2] to output the main system clock (SYSCLK). 0 No internal signal output to P1[2] 1 Output SYSCLK to P1[2]

(continued on next page)

21.4.14 OUT_P1 (continued)

0	**P10EN**	Bit enables pin P1[0] to output the sleep interrupt (SLPINT). 0 No internal signal output to P1[0] 1 Output SLPINT to P1[0]

21.4.15 IO_CFG2

Input/Output Configuration Register 2

Individual Register Names and Addresses:

IO_CFG2 : 1,DEh

	7	6	5	4	3	2	1	0
Access : POR			RW : 0				RW : 0	
Bit Name			REG_LEVEL[2:0]				REG_CLOCK[1:0]	

The Input/Output Configuration 2 Register (IO_CFG2) selects output regulated supply and clock rates.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits should always be written with a value of '0'. For additional information, refer to the IO_CFG2 Register on page 58 in the GPIO chapter.

Bits	Name	Description
5:3	REG_LEVEL[2:0]	These bits select output regulated supply

REG_LEVEL[2:0]	Approx. Regulated Supply (V)		
000	3	2.5	1.8
001	3.1	2.6	1.9
010	3.2	2.7	2.0
011	3.3	2.8	2.1
100	3.4	2.9	2.2
101	3.5	3.0	2.3
110	3.6	3.1	2.4
111	3.7	3.2	2.5

Bits	Name	Description
1:0	REG_CLOCK[1:0]	The Regulated I/O charge pump can operate with a maximum clock speed of 12 MHZ. The REG_CLOCK[1:0] bits select clocking options for the regulator. Setting REG_CLOCK[1:0] to '10' should be used with 24-MHz SYSCLK and '01' should be used with 6/12-MHz SYSCLK.

REG_CLOCK[1:0]	SYSCLK Clock Rate
10	24 MHz
01	6/12 MHz

21.4.16 OSC_CR0

Oscillator Control Register 0

Individual Register Names and Addresses:

OSC_CR0: 1,E0h

	7	6	5	4	3	2	1	0
Access : POR	RW : 0	RW : 0	RW : 0	RW : 0		RW : 010b		
Bit Name	X32ON	Disable Buzz	No Buzz	Sleep[1:0]		CPU Speed[2:0]		

This register is used to configure various features of internal clock sources and clock nets.

In the table, note that the reserved bit is a grayed table cell and is not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 99 in the Digital Clocks chapter.

Bit	Name	Description
7	**X32ON**	Select bit for the external 32-kHz external crystal oscillator (ECO). See the Application Overview on page 75 for the proper sequence for enabling the ECO. 0 The internal 32 kHz oscillator is the source of the 32K clock. 1 The external crystal oscillator is the source of the 32K clock.
6	**Disable Buzz**	Option to disable buzz during sleep. This bit has lower priority than the No Buzz bit. Therefore, if No Buzz = 1, the Disable Buzz bit has no effect. 0 No effect on buzz modes. 1 Buzz is disabled during sleep, with bandgap powered down. No periodic wakeup of the bandgap during sleep.
5	**No Buzz**	This bit allows the bandgap to stay powered during sleep. 0 Buzz bandgap during power down. 1 Bandgap is always powered even during sleep.
4:3	**Sleep[1:0]**	Sleep interval. 00b: **For 32-kHz ILO** 1.95 ms (512 Hz); **For 1-kHz ILO** 64 ms (15.6 Hz) 01b: 15.6 ms (64 Hz); 512 ms (1.95 Hz) 10b: 125 ms (8 Hz); 4 s (0.244 Hz) 11b: 1s (1 Hz); 32 s (0.031 Hz)
2:0	**CPU Speed[2:0]**	These bits set the CPU clock speed, based on the system clock (SYSCLK). SYSCLK is 12 MHz by default, but it can also be set to other frequencies (6 and 24 MHz), or driven from an external clock. **Note** During USB operation, the CPU speed can be set to any setting. Be aware that USB throughput decreases with a decrease in CPU speed. For maximum throughput, the CPU clock should be made equal to the system clock. The system clock must be 24 MHz for USB operation. 000b: **6-MHz IMO** 750 kHz; **12-MHz IMO** 1.5 MHz; **24-MHz IMO** 3 MHz; **External Clock** EXTCLK/8 001b: 1.5 MHz; 3 MHz; 6 MHz; EXTCLK/4 010b: 3 MHz; 6 MHz; 12 MHz; EXTCLK/2 (Reset State) 011b: 6 MHz; 12 MHz; 24 MHz; EXTCLK/1 100b: 375 kHz; 750 kHz; 1.5 MHz; EXTCLK/16 101b: 187.5 kHz; 375 kHz; 750 kHz; EXTCLK/32 110b: 46.9 kHz; 93.7 kHz; 187.5 kHz; EXTCLK/128 111b: 23.4 kHz; 46.8 kHz; 93.7 kHz; EXTCLK/256

21.4.17 ECO_CFG

External Oscillator Trim Configuration Register

Individual Register Names and Addresses:

ECO_CFG : 1,E1h

	7	6	5	4	3	2	1	0
Access : POR						RW : 0	RW : 0	RW : 0
Bit Name						ECO_LPM	ECO_EXW	ECO_EX

This register provides ECO status and control information.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of ‘0’. For additional information, refer to the ECO_CFG Register on page 76 in the External Crystal Oscillator chapter.

Bits	Name	Description
2	**ECO_LPM**	This bit enables the ECO lower power mode.
1	**ECO_EXW**	This is a status bit that indicates that the ECO_EX bit was previously written to. When this bit is a '1', this indicates that the ECO_CONFIG register was written to and is now locked. When this bit is a '0', the register was not written to since the last reset event.
0	**ECO_EX**	The ECO Exists bit is a flag to the hardware indicating that an external crystal oscillator exists in the system.If the bit is '0', a switch-over to the ECO is locked out by hardware. If the bit is '1', hardware allows the firmware to freely switch between the ECO and ILO. It must be written as early as possible after a power-on-reset (POR) or external reset (XRES) event.

21.4.18 OSC_CR2

Oscillator Control Register 2

Individual Register Names and Addresses:

OSC_CR2 : 1,E2h

	7	6	5	4	3	2	1	0
Access : POR				RW : 0		RW : 0	RW : 0	
Bit Name				CLK48MEN		EXTCLKEN	RSVD	

This register is used to configure various features of internal clock sources and clock nets.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 99 in the Digital Clocks chapter.

Bit	Name	Description
4	**CLK48MEN**	This is the 48-MHz clock enable bit. 0 Disables the 48-MHz clock. 1 Enables the 48-MHz clock.
2	**EXTCLKEN**	External Clock Mode Enable. 0 Disabled. Operate from internal main oscillator. 1 Enabled. Operate from the clock supplied at P1[4] based upon the TSYNC bit in CPU_SCR1.
1	**RSVD**	This is a reserved bit. It should always be 0.

21.4.19 VLT_CR

Voltage Monitor Control Register

Individual Register Names and Addresses:

VLT_CR: 1,E3h

	7	6	5	4	3	2	1	0
Access : POR	RW : 0		RW : 0		RW : 0	RW : 0		
Bit Name	HPOR		PORLEV[1:0]		LVDTBEN	VM[2:0]		

This register is used to set the trip points for POR and LVD.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 122 in the POR chapter.

Bit	Name	Description
7	**HPOR**	This bit along with PORLEV sets one of the four values for the PPOR trip voltage. See Table 21-1. See the "DC POR and LVD Specifications" table in the Electrical Specifications section of the enCoRe V device datasheet for voltage tolerances for each setting.
5:4	**PORLEV[1:0]**	These bits along with HPOR sets one of the four values for the PPOR trip voltage. See Table 21-1.
3	**LVDTBEN**	Enables reset of the CPU speed register by LVD comparator output.
2:0	**VM[2:0]**	Sets the LVD levels per the DC Electrical Specifications in the enCoRe V device datasheet, for those devices with this feature. 000b Lowest voltage setting. 001b 010b 011b 100b 101b 110b 111b Highest voltage setting.

Table 21-1. PPOR Range Setting

HPOR, PORLEV[1:0] (VLT_CR[7], VLT_CR[5:4])	Typical PPOR Trip Voltage
000b	1.66 V
001b, 010b, 011b	Do not use
100b	2.36 V
101b	2.60 V
110b	2.82 V
111b	Do not use

21.4.20 VLT_CMP

Voltage Monitor Comparators Register

Individual Register Names and Addresses:

VLT_CMP : 1,E4h

	7	6	5	4	3	2	1	0
Access : POR							R : 0	
Bit Name							LVD	

This register reads the state of the internal supply voltage monitors.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 122 in the POR chapter.

Bit	Name	Description
1	LVD	This bit reads the state of the LVD comparator. 0 Vdd is above LVD trip point. 1 Vdd is below LVD trip point.

21.4.21 IMO_TR

Internal Main Oscillator Trim Register

Individual Register Names and Addresses:

IMO_TR : 1,E8h

	7	6	5	4	3	2	1	0
Access : POR	RW : 00							
Bit Name	Trim[7:0]							

This register is used to manually center the Internal Main Oscillator's (IMO) output to a target frequency.

It is strongly recommended that you do not alter this register's values except to load factory trim settings when changing IMO range.

When changing ranges, the new trim value for this range must be read from flash using a Table Read operation. The new value must be written at the lower frequency range. That is, when moving to a higher frequency range, change the IMO_TR value and then change the range (SLIMO[1:0] in CPU_SCR1). When moving to a lower frequency, change the range first and then update IMO_TR.

For additional information, refer to the Register Definitions on page 68 in the Internal Main Oscillator chapter.

Bit	Name	Description
7:0	**Trim[7:0]**	The value of this register is used to trim the Internal Main Oscillator. Its value is set to the best value for the device during boot.

21.4.22 ILO_TR

Internal Low-speed Oscillator Trim Register

Individual Register Names and Addresses:

ILO_TR : 1,E9h

	7	6	5	4	3	2	1	0
Access : POR		RW : 0	RW : 0		RW : 08			
Bit Name		PD_MODE	ILOFREQ		Freq Trim[3:0]			

This register sets the adjustment for the Internal Low-speed Oscillator (ILO).

It is strongly recommended that you do not alter this register's Freq Trim[3:0] values. The trim bits are set to factory specifications and must not be changed.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Always write reserved bits with a value of '0'. For additional information, refer to the Register Definitions on page 73 in the Internal Low-speed Oscillator chapter.

Bit	Name	Description
6	**PD_MODE**	This bit selects power down mode. Setting this bit high disables oscillator and current bias, which results in slower startup time. Power down mode: 0 Partial oscillator power down for faster startup (100 nA nominal). 1 Full oscillator power down for lower power (0 nA nominal).
5	**ILOFREQ**	Selects oscillator nominal frequency. 0 32 kHz 1 1 kHz
3:0	**Freq Trim[3:0]**	These bits trim the oscillator frequency.

21.4.23 SLP_CFG

Sleep Configuration Register

Individual Register Names and Addresses:

SLP_CFG : 1,EBh

	7	6	5	4	3	2	1	0
Access : POR	RW : 0							
Bit Name	PSSDC[1:0]							

This register sets up the sleep duty cycle.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 82 in the Sleep and Watchdog chapter.

Bit	Name	Description
7:6	**PSSDC[1:0]**	Sleep Duty Cycle. Controls the ratios (in numbers of 32.768-kHz clock periods) of "on" time versus "off" time for PORLVD, bandgap reference, and pspump. 00b 1 / 256 (8 ms). 01b 1 / 1024 (31.2 ms). 10b 1 / 64 (2 ms). 11b 1 / 16 (500 µs).

Note The buzz rate for Isb1 spec in the datasheet is with 01 setting in the ALT_Buzz bits of the SLP_CFG2 register.

21.4.24 SLP_CFG2

Sleep Configuration Register 2

Individual Register Names and Addresses:

SLP_CFG2 : 1,ECh

	7	6	5	4	3	2	1	0
Access : POR					RW : 0		RW : 0	RW : 0
Bit Name					ALT_Buzz[1:0]		I2C_ON	LSO_OFF

This register holds the configuration for I2C sleep, deep sleep, and buzz.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 82 in the Sleep and Watchdog chapter.

Bit	Name	Description
3:2	**ALT_Buzz[1:0]**	These bits control additional selections for POR/LVD buzz rates. 00 Compatibility mode, buzz rate is determined by PSSDC bits. 01 Duty cycle is 1/32768. 10 Duty cycle is 1/8192. 11 Reserved.
1	**I2C_ON**	This bit enables the standby regulator in I2C sleep mode at a level sufficient to supply the I2C circuitry to detect I2C address in the bus and also to ensure data retention in 32-byte I2C buffer during sleep state.
0	**LSO_OFF**	This bit disables the LSO oscillator when in sleep state.

Note The buzz rate for Isb1 spec in the datasheet is with 01 setting in the ALT_Buzz bits.

21.4.25 SLP_CFG3

Sleep Configuration Register 3

Individual Register Names and Addresses:

SLP_CFG3 : 1,EDh

	7	6	5	4	3	2	1	0
Access : POR		RW : 1	RW : 11		RW : 11		RW : 11	
Bit Name		DBL_TAPS	T2TAP[1:0]		T1TAP[1:0]		T0TAP[1:0]	

This register holds the configuration of the wakeup sequence taps.

It is strongly recommended to not alter this register setting.

In the table, note that the reserved bit is a grayed table cell and is not described in the bit description section. Reserved bits must always be written with a value of '0'. For additional information, refer to the Register Definitions on page 82 in the Sleep and Watchdog chapter.

Bit	Name	Description
6	**DBL_TAPS**	When this bit is set all the tap values (T0, T1, and T2) are doubled for the wakeup sequence.
5:4	**T2TAP[1:0]**	These bits control the duration of the T2-T4 sequence (see Figure 11-2 on page 80) by selecting a tap from the WakeupTimer. **Note** The T2 delay is only valid for the wakeup sequence. It is not used for the buzz sequence. 00 1 µs 01 2 µs 10 5 µs 11 10 µs
3:2	**T1TAP[1:0]**	These bits control the duration of the T1-T2 sequence (see Figure 11-2 on page 80) by selecting a tap from the Wakeup Timer. 00 3 µs 01 4 µs 10 5 µs 11 10 µs
1:0	**T0TAP[1:0]**	These bits control the duration of the T0-T1 sequence (see Figure 11-2 on page 80) by selecting a tap from the Wakeup Timer. 00 10 µs 01 14 µs 10 20 µs 11 30 µs

21.4.26 IMO_TR1

Internal Main Oscillator Trim Register 1

Individual Register Names and Addresses:

IMO_TR1 : 1,FAh

	7	6	5	4	3	2	1	0
Access : POR						W : 0		
Bit Name						FineTrim[2:0]		

This register is used to fine tune the IMO frequency.

It is strongly recommended that the user not alter this register's values.

In the table, note that reserved bits are grayed table cells and are not described in the bit description section. Reserved bits should always be written with a value of '0'. For additional information, refer to the Register Definitions on page 68 in the Internal Main Oscillator chapter.

Bit	Name	Description
7:0	**FineTrim[2:0]**	These bits provide ability to fine tune the IMO frequency. These values are normally only changed by the oscillator-locking function. These are the lower 3 bits of the 11-bit oscillator trim. IMO_TR holds the MSb.

Section E: Glossary

The Glossary section explains the terminology used in this technical reference manual. Glossary terms are characterized in ***bold, italic font*** throughout the text of this manual.

A

accumulator	In a CPU, a register in which intermediate results are stored. Without an accumulator, it is necessary to write the result of each calculation (addition, subtraction, shift, and so on) to main memory and read them back. Access to main memory is slower than access to the accumulator, which usually has direct paths to and from the arithmetic and logic unit (ALU).
active high	1. A logic signal having its asserted state as the logic 1 state. 2. A logic signal having the logic 1 state as the higher voltage of the two states.
active low	1. A logic signal having its asserted state as the logic 0 state. 2. A logic signal having its logic 1 state as the lower voltage of the two states: inverted logic.
address	The label or number identifying the memory location (RAM, ROM, or register) where a unit of information is stored.
algorithm	A procedure for solving a mathematical problem in a finite number of steps that frequently involve repetition of an operation.
ambient temperature	The temperature of the air in a designated area, particularly the area surrounding the enCoRe Vdevice.
analog	As opposed to digital, signals that are on or off or ‘1’ or ‘0’. Analog signals vary in a continuous manner. See also ***analog signals***.
analog output	An output that is capable of driving any voltage between the supply rails, instead of just a logic 1 or logic 0.
analog signals	A signal represented in a continuous form with respect to continuous times, as contrasted with a digital signal represented in a discrete (discontinuous) form in a sequence of time.
analog-to-digital (ADC)	A device that changes an analog signal to a digital signal of corresponding magnitude. Typically, an ADC converts a voltage to a digital number. The ***digital-to-analog (DAC)*** converter performs the reverse operation.
AND	See ***Boolean Algebra***.
API (Application Programming Interface)	A series of software routines that comprise an interface between a computer application and lower-level services and functions (for example, user modules and libraries). APIs serve as building blocks for programmers that create software applications.

array An array, also known as a vector or list, is one of the simplest data structures in computer programming. Arrays hold a fixed number of equally-sized data elements, generally of the same data type. Individual elements are accessed by index using a consecutive range of integers, as opposed to an associative array. Most high level programming languages have arrays as a built-in data type. Some arrays are multi-dimensional, meaning they are indexed by a fixed number of integers; for example, by a group of two integers. One- and two-dimensional arrays are the most common. Also, an array can be a group of capacitors or resistors connected in some common form.

assembly A symbolic representation of the machine language of a specific processor. Assembly language is converted to machine code by an assembler. Usually, each line of assembly code produces one machine instruction, though the use of macros is common. Assembly languages are considered low level languages; where as C is considered a high level language.

asynchronous A signal whose data is acknowledged or acted upon immediately, irrespective of any clock signal.

attenuation The decrease in intensity of a signal as a result of absorption of energy and of scattering out of the path to the detector, but not including the reduction due to geometric spreading. Attenuation is usually expressed in dB.

B

bandgap reference A stable voltage reference design that matches the positive temperature coefficient of V_T with the negative temperature coefficient of V_{BE}, to produce a zero temperature coefficient (ideally) reference.

bandwidth
1. The frequency range of a message or information processing system measured in hertz.
2. The width of the spectral region over which an amplifier (or absorber) has substantial gain (or loss). It is sometimes represented more specifically as, for example, full width at half maximum.

bias
1. A systematic deviation of a value from a reference value.
2. The amount by which the average of a set of values departs from a reference value.
3. The electrical, mechanical, magnetic, or other force (field) applied to a device to establish a reference level to operate the device.

bias current The constant low level DC current that is used to produce a stable operation in amplifiers. This current can sometimes be changed to alter the bandwidth of an amplifier.

binary The name for the base 2 numbering system. The most common numbering system is the base 10 numbering system. The base of a numbering system indicates the number of values that may exist for a particular positioning within a number for that system. For example, in base 2, binary, each position may have one of two values (0 or 1). In the base 10, decimal, each position may have one of ten values (0, 1, 2, 3, 4, 5, 6, 7, 8, and 9).

bit A single digit of a binary number. Therefore, a bit may only have a value of ‘0’ or ‘1’. A group of 8 bits is called a byte. Because the enCoRe V M8C is an 8-bit microcontroller, the enCoRe V native data chunk size is a byte.

bit rate (BR) The number of bits occurring per unit of time in a bit stream, usually expressed in bits per second (bps).

block

1. A functional unit that performs a single function, such as an oscillator.
2. A functional unit that may be configured to perform one of several functions, such as a digital block or an analog block.

Boolean Algebra

In mathematics and computer science, Boolean algebras or Boolean lattices, are algebraic structures which “capture the essence” of the logical operations AND, OR and NOT as well as set the theoretic operations union, intersection, and complement. Boolean algebra also defines a set of theorems that describe how Boolean equations can be manipulated. For example, these theorems are used to simplify Boolean equations which reduces the number of logic elements needed to implement the equation.

The operators of Boolean algebra may be represented in various ways. Often they are simply written as AND, OR, and NOT. In describing circuits, NAND (NOT AND), NOR (NOT OR), XNOR (exclusive NOT OR), and XOR (exclusive OR) may also be used. Mathematicians often use + (for example, A+B) for OR and • for AND (for example, A*B) (because in some ways those operations are analogous to addition and multiplication in other algebraic structures) and represent NOT by a line drawn above the expression being negated (for example, ~A, A_, !A).

break-before-make

The elements involved go through a disconnected state entering (‘break”) before the new connected state (“make”).

buffer

1. A storage area for data that is used to compensate for a speed difference, when transferring data from one device to another. Usually refers to an area reserved for I/O operations into which data is read or from which data is written.
2. A portion of memory set aside to store data, often before it is sent to an external device or as it is received from an external device.
3. An amplifier used to lower the output impedance of a system.

bus

1. A named connection of nets. Bundling nets together in a bus makes it easier to route nets with similar routing patterns.
2. A set of signals performing a common function and carrying similar data. Typically represented using vector notation; for example, address[7:0].
3. One or more conductors that serve as a common connection for a group of related devices.

byte

A digital storage unit consisting of 8 bits.

C

C

A high level programming language.

capacitance

A measure of the ability of two adjacent conductors, separated by an insulator, to hold a charge when a voltage differential is applied between them. Capacitance is measured in units of Farads.

capture

To extract information automatically through the use of software or hardware, as opposed to hand-entering of data into a computer file.

checksum

The checksum of a set of data is generated by adding the value of each data word to a sum. The actual checksum can simply be the result sum or a value that must be added to the sum to generate a pre-determined value.

chip

A single monolithic Integrated Circuit (IC). See also ***integrated circuit (IC)***.

clear

To force a bit/register to a value of logic 0.

clock	The device that generates a periodic signal with a fixed frequency and duty cycle. A clock is sometimes used to synchronize different logic blocks.
clock generator	A circuit that is used to generate a clock signal.
CMOS	The logic gates constructed using ***MOS*** transistors connected in a complementary manner. CMOS is an acronym for complementary metal-oxide semiconductor.
comparator	An electronic circuit that produces an output voltage or current whenever two input levels simultaneously satisfy pre-determined amplitude requirements.
compiler	A program that translates a high level language, such as C, into machine language.
configuration	In a computer system, an arrangement of functional units according to their nature, number, and chief characteristics. Configuration pertains to hardware, software, firmware, and documentation. The configuration affects system performance.
configuration space	In enCoRe V devices, the register space accessed when the XIO bit in the CPU_F register is set to '1'.
crowbar	A type of over-voltage protection that rapidly places a low resistance shunt (typically an SCR) from the signal to one of the power supply rails, when the output voltage exceeds a pre-determined value.
crystal oscillator	An oscillator in which the frequency is controlled by a piezoelectric crystal. Typically a piezoelectric crystal is less sensitive to ambient temperature than other circuit components.
CSP	Chip scale package.
cyclic redundancy check (CRC)	A calculation used to detect errors in data communications, typically performed using a linear feedback shift register. Similar calculations may be used for a variety of other purposes such as data compression.

D

data bus	A bidirectional set of signals used by a computer to convey information from a memory location to the central processing unit and vice versa. More generally, a set of signals used to convey data between digital functions.
data stream	A sequence of digitally encoded signals used to represent information in transmission.
data transmission	The sending of data from one place to another by means of signals over a channel.
debugger	A hardware and software system that allows the user to analyze the operation of the system under development. A debugger usually allows the developer to step through the firmware one step at a time, set break points, and analyze memory.
dead band	A period of time when neither of two or more signals are in their active state or in transition.
decimal	A base 10 numbering system, which uses the symbols 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9 (called digits) together with the decimal point and the sign symbols + (plus) and - (minus) to represent numbers.

default value Pertaining to the pre-defined initial, original, or specific setting, condition, value, or action a system assumes, uses, or takes in the absence of instructions from the user.

device The device referred to in this manual is the enCoRe V chip, unless otherwise specified.

die An unpackaged Integrated Circuit (IC), normally cut from a wafer.

digital A signal or function, the amplitude of which is characterized by one of two discrete values: '0' or '1'.

digital blocks The 8-bit logic blocks that can act as a counter, timer, serial receiver, serial transmitter, CRC generator, pseudo-random number generator, or SPI.

digital logic A methodology for dealing with expressions containing two-state variables that describe the behavior of a circuit or system.

digital-to-analog (DAC) A device that changes a digital signal to an analog signal of corresponding magnitude. The ***analog-to-digital (ADC)*** converter performs the reverse operation.

direct access The capability to obtain data from a storage device, or to enter data into a storage device, in a sequence independent of their relative positions by means of addresses that indicate the physical location of the data.

duty cycle The relationship of a clock period ***high time*** to its ***low time***, expressed as a percent.

E

emulator Duplicates (provides an emulation of) the functions of one system with a different system, so that the second system appears to behave similar to the first system.

External Reset (XRES) An active high signal that is driven into the enCoRe V device. It causes all operation of the CPU and blocks to stop and return to a pre-defined state.

F

falling edge A transition from a logic 1 to a logic 0. Also known as a negative edge.

feedback The return of a portion of the output, or processed portion of the output, of a (usually active) device to the input.

filter A device or process by which certain frequency components of a signal are attenuated.

firmware The software that is embedded in a hardware device and executed by the CPU. The software may be executed by the end user but it may not be modified.

flag Any of various types of indicators used for identification of a condition or event (for example, a character that signals the termination of a transmission).

Flash An electrically programmable and erasable, non***volatile*** technology that provides users with the programmability and data storage of EPROMs, plus in-system erasability. Nonvolatile means that the data is retained when power is off.

Flash bank A group of flash ROM blocks where flash block numbers always begin with ‘0’ in an individual flash bank. A flash bank also has its own block level protection information.

Flash block The smallest amount of flash ROM space that may be programmed at one time and the smallest amount of flash space that may be protected. A flash block holds 64 bytes.

flip-flop A device having two stable states and two input terminals (or types of input signals) each of which corresponds with one of the two states. The circuit remains in either state until it is made to change to the other state by application of the corresponding signal.

frequency The number of cycles or events per unit of time, for a periodic function.

G

gain The ratio of output current, voltage, or power to input current, voltage, or power, respectively. Gain is usually expressed in dB.

ground

1. The electrical neutral line having the same potential as the surrounding earth.
2. The negative side of DC power supply.
3. The reference point for an electrical system.
4. The conducting paths between an electric circuit or equipment and the earth, or some conducting body serving in place of the earth.

H

hardware A comprehensive term for all of the physical parts of a computer or embedded system, as distinguished from the data it contains or operates on, and the software that provides instructions for the hardware to accomplish tasks.

hardware reset A reset that is caused by a circuit, such as a POR, watchdog reset, or external reset. A hardware reset restores the state of the device as it was when it was first powered up. Therefore, all registers are set to the POR value as indicated in register tables throughout this manual.

hexadecimal A base 16 numeral system (often abbreviated and called hex), usually written using the symbols 0-9 and A-F. It is a useful system in computers because there is an easy mapping from four bits to a single hex digit. Thus, one can represent every byte as two consecutive hexadecimal digits. Compare the binary, hex, and decimal representations:

bin	**=**	**hex**	**=**	**dec**
0000b	=	0x0	=	0
0001b	=	0x1	=	1
0010b	=	0x2	=	2
...				
1001b	=	0x9	=	9
1010b	=	0xA	=	10
1011b	=	0xB	=	11
...				
1111b	=	0xF	=	15

So the decimal numeral 79 whose binary representation is 0100 1111b can be written as 4Fh in hexadecimal (0x4F).

high time The amount of time the signal has a value of ‘1’ in one period, for a periodic digital signal.

I

I2C A two-wire serial computer bus by Philips Semiconductors (now NXP Semiconductors). I2C is an inter-integrated circuit. It is used to connect low-speed peripherals in an embedded system. The original system was created in the early 1980s as a battery control interface, but it was later used as a simple internal bus system for building control electronics. I2C uses only two bidirectional pins, clock and data, both running at +5 V and pulled high with resistors. The bus operates at 100 kbps in standard mode and 400 kbps in fast mode.

ICE The in-circuit emulator that allows users to test the project in a hardware environment, while viewing the debugging device activity in a software environment (PSoC Designer™).

idle state A condition that exists whenever user messages are not being transmitted, but the service is immediately available for use.

impedance
1. The resistance to the flow of current caused by resistive, capacitive, or inductive devices in a circuit.
2. The total passive opposition offered to the flow of electric current. Note the impedance is determined by the particular combination of resistance, inductive reactance, and capacitive reactance in a given circuit.

input A point that accepts data in a device, process, or channel.

input/output (I/O) A device that introduces data into or extracts data from a system.

instruction An expression that specifies one operation and identifies its operands, if any, in a programming language such as C or assembly.

integrated circuit (IC) A device in which components such as resistors, capacitors, diodes, and ***transistors*** are formed on the surface of a single piece of semiconductor.

interface The means by which two systems or devices are connected and interact with each other.

interrupt A suspension of a process, such as the execution of a computer program, caused by an event external to that process and performed in such a way that the process can be resumed.

interrupt service routine (ISR) A block of code that normal code execution is diverted to when the M8C receives a hardware interrupt. Many interrupt sources may each exist with its own priority and individual ISR code block. Each ISR code block ends with the RETI instruction, returning the device to the point in the program where it left normal program execution.

J

jitter
1. A misplacement of the timing of a transition from its ideal position. A typical form of corruption that occurs on serial data streams.
2. The abrupt and unwanted variations of one or more signal characteristics, such as the interval between successive pulses, the amplitude of successive cycles, or the frequency or phase of successive cycles.

K

keeper A circuit that holds a signal to the last driven value, even when the signal becomes un-driven.

L

latency The time or delay that it takes for a signal to pass-through a given circuit or network.

least significant bit (LSb) The binary digit, or bit, in a binary number that represents the least significant value (typically the right-hand bit). The bit versus byte distinction is made by using a lower case “b” for bit in LSb.

least significant byte (LSB) The byte in a multi-byte word that represents the least significant value (typically the right-hand byte). The byte versus bit distinction is made by using an upper case “B” for byte in LSB.

load The electrical demand of a process expressed as power (watts), current (amps), or resistance (ohms).

logic function A mathematical function that performs a digital operation on digital data and returns a digital value.

low time The amount of time the signal has a value of ‘0’ in one period, for a periodic digital signal.

low-voltage detect (LVD) A circuit that senses Vdd and provides an interrupt to the system when Vdd falls below a selected threshold.

M

M8C An 8-bit Harvard Architecture microprocessor. The microprocessor coordinates all activity inside the enCoRe V device by interfacing to the flash, SRAM, and register space.

macro A programming language macro is an abstraction whereby a certain textual pattern is replaced according to a defined set of rules. The interpreter or compiler automatically replaces the macro instance with the macro contents when an instance of the macro is encountered. Therefore, if a macro is used 5 times and the macro definition required 10 bytes of code space, 50 bytes of code space are needed in total.

mask

1. To obscure, hide, or otherwise prevent information from being derived from a signal. It is usually the result of interaction with another signal, such as noise, static, jamming, or other forms of interference.
2. A pattern of bits that can be used to retain or suppress segments of another pattern of bits in computing and data processing systems.

master device A device that controls the timing for data exchanges between two devices. Or when devices are cascaded in width, the master device is the one that controls the timing for data exchanges between the cascaded devices and an external interface. The controlled device is called the ***slave device***.

microcontroller An integrated circuit ***chip*** that is designed primarily for control systems and products. In addition to a CPU, a microcontroller typically includes memory, timing circuits, and I/O circuitry. The reason for this is to permit the realization of a controller with a minimal quantity of chips, thus achieving maximal possible miniaturization. This, in turn, reduces the volume and the cost of the controller. The microcontroller is normally not used for general-purpose computation as is a microprocessor.

mixed signal The reference to a circuit containing both analog and digital techniques and components.

mnemonic
1. A tool intended to assist the memory. Mnemonics rely on not only repetition to remember facts, but also on creating associations between easy-to-remember constructs and lists of data.
2. A two to four character string representing a microprocessor instruction.

mode A distinct method of operation for software or hardware. For example, the digital block may be in either counter mode or timer mode.

modulation A range of techniques for encoding information on a carrier signal, typically a sine-wave signal. A device that performs modulation is known as a modulator.

Modulator A device that imposes a signal on a carrier.

MOS An acronym for metal-oxide semiconductor.

most significant bit (MSb) The binary digit, or bit, in a binary number that represents the most significant value (typically the left-hand bit). The bit versus byte distinction is made by using a lower case “b” for bit in MSb.

most significant byte (MSB) The byte in a multi-byte word that represents the most significant value (typically the left-hand byte). The byte versus bit distinction is made by using an upper case “B” for byte in MSB.

multiplexer (mux)
1. A logic function that uses a binary value, or address, to select between a number of inputs and conveys the data from the selected input to the output.
2. A technique which allows different input (or output) signals to use the same lines at different times, controlled by an external signal. Multiplexing is used to save on wiring and I/O ports.

N

NAND See ***Boolean Algebra***.

negative edge A transition from a logic 1 to a logic 0. Also known as a falling edge.

net The routing between devices.

nibble A group of four bits, which is one-half of a byte.

noise
1. A disturbance that affects a signal and that may distort the information carried by the signal.
2. The random variations of one or more characteristics of any entity such as voltage, current, or data.

NOR	See ***Boolean Algebra***.
NOT	See ***Boolean Algebra***.

O

OR	See ***Boolean Algebra***.
oscillator	A circuit that may be crystal controlled and is used to generate a clock frequency.
output	The electrical signal or signals which are produced by an analog or digital block.

P

parallel	The means of communication in which digital data is sent multiple bits at a time, with each simultaneous bit being sent over a separate line.
parameter	Characteristics for a given block that have either been characterized or may be defined by the designer.
parameter block	A location in memory where parameters for the SSC instruction are placed prior to execution.
parity	A technique for testing transmitting data. Typically, a binary digit is added to the data to make the sum of all the digits of the binary data either always even (even parity) or always odd (odd parity).
path	1. The logical sequence of instructions executed by a computer. 2. The flow of an electrical signal through a circuit.
pending interrupts	An interrupt that is triggered but is not serviced, either because the processor is busy servicing another interrupt or global interrupts are disabled.
phase	The relationship between two signals, usually the same frequency, that determines the delay between them. This delay between signals is either measured by time or angle (degrees).
Phase-Locked Loop (PLL)	An electronic circuit that controls an ***oscillator*** so that it maintains a constant phase angle relative to a reference signal.
pin	A terminal on a hardware component. Also called lead.
pinouts	The pin number assignment: the relation between the logical inputs and outputs of the enCoRe V device and their physical counterparts in the printed circuit board (PCB) package. Pinouts involve pin numbers as a link between schematic and PCB design (both being computer generated files) and may also involve pin names.
port	A group of pins, usually eight.
positive edge	A transition from a logic 0 to a logic 1. Also known as a rising edge.
posted interrupts	An interrupt that has been detected by the hardware but may or may not be enabled by its mask bit. Posted interrupts that are not masked become pending interrupts.

Power-on-Reset (POR) A circuit that forces the enCoRe V device to reset when the voltage is below a pre-set level. This is one type of ***hardware reset***.

program counter The instruction pointer (also called the program counter) is a register in a computer processor that indicates where in memory the CPU is executing instructions. Depending on the details of the particular machine, it holds either the address of the instruction being executed or the address of the next instruction to be executed.

protocol A set of rules. Particularly the rules that govern networked communications.

PSoC Designer™ The software for Cypress' Programmable System-on-Chip technology.

pulse A rapid change in some characteristic of a signal (for example, phase or frequency) from a base-line value to a higher or lower value, followed by a rapid return to the baseline value.

R

RAM An acronym for random access memory. A data-storage device from which data can be read out and new data can be written in.

register A storage device with a specific capacity, such as a bit or byte.

reset A means of bringing a system back to a known state. See ***hardware reset*** and ***software reset***.

resistance The resistance to the flow of electric current measured in ohms for a conductor.

revision ID A unique identifier of the enCoRe V device.

ripple divider An asynchronous ripple counter constructed of flip-flops. The clock is fed to the first stage of the counter. An n-bit binary counter consisting of n flip-flops that can count in binary from 0 to 2^n - 1.

rising edge *See* ***positive edge***.

ROM An acronym for read-only memory. A data-storage device from which data can be read out, but new data cannot be written in.

routine A block of code, called by another block of code, that may have some general or frequent use.

routing Physically connecting objects in a design according to design rules set in the reference library.

S

sampling The process of converting an analog signal into a series of digital values or reversed.

schematic A diagram, drawing, or sketch that details the elements of a system, such as the elements of an electrical circuit or the elements of a logic diagram for a computer.

seed value An initial value loaded into a linear feedback shift register or random number generator.

serial	1. Pertaining to a process in which all events occur one after the other. 2. Pertaining to the sequential or consecutive occurrence of two or more related activities in a single device or channel.
set	To force a bit/register to a value of logic 1.
settling time	The time it takes for an output signal or value to stabilize after the input has changed from one value to another.
shift	The movement of each bit in a word, one position to either the left or right. For example, if the hex value 0x24 is shifted one place to the left, it becomes 0x48. If the hex value 0x24 is shifted one place to the right, it becomes 0x12.
shift register	A memory storage device that sequentially shifts a word either left or right to output a stream of serial data.
sign bit	The most significant binary digit, or bit, of a signed binary number. If set to a logic 1, this bit represents a negative quantity.
signal	A detectable transmitted energy that can be used to carry information. As applied to electronics, any transmitted electrical impulse.
silicon ID	A unique identifier of the enCoRe V silicon.
skew	The difference in arrival time of bits transmitted at the same time, in parallel transmission.
slave device	A device that allows another device to control the timing for data exchanges between two devices. Or when devices are cascaded in width, the slave device is the one that allows another device to control the timing of data exchanges between the cascaded devices and an external interface. The controlling device is called the master device.
software	A set of computer programs, procedures, and associated documentation concerned with the operation of a data processing system (for example, compilers, library routines, manuals, and circuit diagrams). Software is often written first as source code and then converted to a binary format that is specific to the device on which the code is executed.
software reset	A partial reset executed by software to bring part of the system back to a known state. A software reset restores the M8C to a known state but not blocks, systems, peripherals, or registers. For a software reset, the CPU registers (CPU_A, CPU_F, CPU_PC, CPU_SP, and CPU_X) are set to 0x00. Therefore, code execution begins at flash address 0x0000.
SRAM	An acronym for static random access memory. A memory device allowing users to store and retrieve data at a high rate of speed. The term static is used because, after a value is loaded into an SRAM cell, it remains unchanged until it is explicitly altered or until power is removed from the device.
SROM	An acronym for supervisory read-only memory. The SROM holds code that is used to boot the device, calibrate circuitry, and perform flash operations. The functions of the SROM may be accessed in normal user code, operating from flash.
stack	A stack is a data structure that works on the principle of Last In First Out (LIFO). This means that the last item put on the stack is the first item that can be taken off.
stack pointer	A stack may be represented in a computer's inside blocks of memory cells, with the bottom at a fixed location and a variable stack pointer to the current top cell.

state machine The actual implementation (in hardware or software) of a function that can be considered to consist of a set of states through which it sequences.

sticky A bit in a register that maintains its value past the time of the event that caused its transition has passed.

stop bit A signal following a character or block that prepares the receiving device to receive the next character or block.

switching The controlling or routing of signals in circuits to execute logical or arithmetic operations, or to transmit data between specific points in a network.

synchronous 1. A signal whose data is not acknowledged or acted upon until the next active edge of a clock signal.
2. A system whose operation is synchronized by a clock signal.

T

tap The connection between two blocks of a device created by connecting several blocks/components in a series, such as a shift register or resistive voltage divider.

terminal count The state at which a counter is counted down to zero.

threshold The minimum value of a signal that can be detected by the system or sensor under consideration.

transistors The transistor is a solid-state semiconductor device used for amplification and switching, and has three terminals. A small current or voltage applied to one terminal controls the current through the other two. It is the key component in all modern electronics. In digital circuits, transistors are used as very fast electrical switches, and arrangements of transistors can function as logic gates, RAM-type memory, and other devices. In analog circuits, transistors are essentially used as amplifiers.

tri-state A function whose output can adopt three states: 0, 1, and Z (high-impedance). The function does not drive any value in the Z state and, in many respects, may be considered to be disconnected from the rest of the circuit, allowing another output to drive the same ***net***.

U

user The person using the enCoRe V device and reading this manual.

user modules Pre-build, pre-tested hardware/firmware peripheral functions that take care of managing and configuring the lower level analog and digital blocks. User modules also provide high level ***API (Application Programming Interface)*** for the peripheral function.

user space The bank 0 space of the register map. The registers in this bank are more likely to be modified during normal program execution and not just during initialization. Registers in bank 1 are most likely to be modified only during the initialization phase of the program.

V

Vdd	A name for a power net meaning “voltage drain.” The most positive power supply signal. Usually 5 or 3.3 volts.
volatile	Not guaranteed to stay the same value or level when not in scope.
Vss	A name for a power net meaning “voltage source.” The most negative power supply signal.

W

watchdog timer	A timer that must be serviced periodically. If it is not serviced, the CPU resets after a specified period of time.
waveform	The representation of a signal as a plot of amplitude versus time.
WLCSP	Wafer level chip scale package.

X

XOR	See ***Boolean Algebra***.

Index

#

32 kHz clock selection 72
32-Pin Part Pinout 16
48-pin OCD part pinout 17, 19
48-Pin Part Pinout 18, 20

A

ACK bit 194
acronyms 14
ADC, *See* Analog-to-Digital Converter
Address bits
  in I2C_SCR register 194
address spaces, CPU core 26
analog input, GPIO 53
Analog-to-Digital Converter (ADC) 59
architecture
  PSoC core 23
  system resources 92
  top level 10

B

bank 0 registers 164
  register mapping table 160
bank 1 registers 212
  register mapping table 161
basic paging in RAM paging 38
bias generator in regulated IO 88
Bias Trim bits in ILO_TR register 234
Bus Error bit 194
Bypass bit 214
Byte Complete bit 194

C

Calibrate0 function in SROM 36
Calibrate1 function in SROM 36
Carry bit 208
charge pump in regulated IO 88
Checksum function in SROM 36
Clock Phase bit in SPI_CR 168
Clock Polarity bit 168
Clock Rate bits 193
Clock Sel bit 214
clock, external digital 97
  switch operation 97
clocks digital, *See* digital clocks
CMP_MUX register 180
comparator in regulated IO 88
configuration register in SPI
  SPI_CFG register 128
control register in SPI
  SPI_CR register 127
conventions, documentation
  acronyms 14
  numeric naming 13
  register conventions 159
core, *See* enCoRe V core
CPU core 26
  instruction formats 29
  instruction set summary 27–28
  overview 26
  register definitions 31
CPU Speed bits 228
CPU_F register 31, 208
CPU_SCR0 register 117, 211
CPU_SCR1 register 69, 210
CUR_PP register 41, 188
current page pointer in RAM paging 39

D

DATA 182, 183
DATA bits
  in PT_DATA0 register 182, 183
Data bits
  in I2C_DR register 195
  in PRTxDR register 56, 164
  in SPI_RXR register 126, 167
  in SPI_TXR register 125, 166
  in TMP_DRx register 41, 219
data bypass in GPIO 55
data registers in SPI 125
development kits 12
digital clocks 96
  architecture 96
  internal low speed oscillator 96
  internal main oscillator 96
  register definitions 99
  system clocking signals 96
digital IO, GPIO 53
documentation

history 12
overview 9
Drive Mode 0 bits 212
Drive Mode 1 bits 213

E

Enable bit 193
ENABLE bits
in MUX_CRx registers 223
in SPI_CR register 168
ENSWINT bit 50, 205
EP0_CNT register 150, 175
EP0_CR register 149, 153
EP0_DRx register 150, 176
EPx_CNT1 register 151, 152
EraseAll function in SROM 35
EraseBlock function in SROM 35
EXTCLKEN bit 230
external digital clock 97
external reset 118

F

Flash
memory organization 34
Freq Trim bits for ILO_TR 234
full-speed USB 141
architecture 141
memory arbiter 142
register definitions 147
suspend mode 145
USB SIE 141
USB SRAM 142

G

general purpose IO 52
analog and digital input 53
architecture 52
block interrupts 54
data bypass 55
digital IO 53
drive modes 57
interrupt modes 54
port 1 distinctions 53
register definitions 56
GIE bit 209
GIES bit 211
GPIO bit
in INT_CLR0 register 196
in INT_MSK0 register 204
GPIO block interrupts 54
GPIO, *See* general purpose IO

H

help, getting
development kits 12
support 12
upgrades 12

I

I2C bit
in INT_CLR0 register 196
in INT_MSK0 register 204
I2C slave 103
application overview 105
architecture 103
basic data transfer 104
basic IO timing 111
clock generation timing 111
operation 105
register definitions 106
stall timing 113
status timing 112
I2C_CFG register 107, 193
I2C_DR register 110, 195
I2C_SCR register 109, 194
IDX_PP register 42, 190
ILO, *See* internal low speed oscillator
IMO_TR register 233
IMO_TR1 register 158
IMO, *See* internal main oscillator
IMODIS bit 230
index memory page pointer in RAM paging 40
INNx bits 180
INPx bits 180
instruction formats
1-byte instructions 29
2-byte instructions 29
3-byte instructions 30
instruction set summary 27–28
Int Sel bit 214
INT_CLR0 register 46, 47, 48, 76, 196, 198, 199
INT_MSK0 register 49, 50, 202, 203, 204
INT_SW_EN register 50, 205
INT_VC register 51, 206
internal low speed oscillator 72
32 kHz clock selection 72
architecture 72
in digital clocks 96
register definitions 73
internal M8C registers 26
internal main oscillator 67
architecture 67
in digital clocks 96
register definitions 68
interrupt controller 44
application overview 45
architecture 44
interrupt table 45

latency and priority 45
posted vs pending interrupts 45
register definitions 46
Interrupt Enables bits 165
interrupt modes in GPIO 54
interrupt table 45
interrupts in RAM paging 39
IO analog multiplexer 90
architecture 90
register definitions 91
IO_CFG register 58, 224
IRAMDIS bit 210
IRESS bit 210

L

low voltage detect (LVD)
See POR and LVD
LRB bit 194
LSb First bit 168
LVD bit 232
LVDTBEN bits 231

M

M8C, *See* CPU core
mapping tables, registers 159
master function for SPI 124
measurement units 13
memory arbiter in USB 142
MUX_CRx register 91, 222, 223, 229
MVI instructions in RAM paging 39
MVR_PP register 42, 191
MVW_PP register 43, 192

N

No Buzz bit 228
numeric naming conventions 13

O

One Shot bit 181, 184, 185
OSC_CR0 register 228
OSC_CR2 register 69, 230
OUT_P1 register 99, 225, 226
Overrun bit 168
overviews
enCoRe V core 23
register tables 159
system resources 92

P

P10EN bit 226
P12EN bit 225
P16D bit 225
P16EN bit 225
Page bits
in CUR_PP register 188
in IDX_PP register 190
in MVR_PP register 191
in MVW_PP register 192
in STK_PP register 189
pass transistors in regulated IO 88
Pending Interrupt bits 206
PgMode bits 208
pin behavior during reset 114
pin information, *See* pinouts
pinouts
16-pin part 15
32-pin part 16
48-pin part 17
48-pin OCD part 17, 19
PMAx_CUR register 156
POR and LVD 121
architecture 121
register definitions 122
PORLEV bits 231
PORS bit 211
power modes
system resets 120
power on reset (POR)
See POR and LVD
power on reset in system resets 118
product upgrades 12
programmable timer 137
architecture 137
register definitions 139
ProtectBlock function in SROM 35
protocol function for SPI 123
PRTxDM0 register 57, 212
PRTxDM1 register 57, 213
PRTxDR register 56, 164
PRTxIE register 56, 165
PSelect bit 193
PSoC core
architecture 23
overview 10
register summary 24
See also CPU core
PSSDC bits 235, 236, 237
PT_CFG register 139, 140, 181
PT_DATA1 register 140
PTx_DATA0 register 140, 183
PTx_DATA1 register 140, 182

R

RAM paging 38
architecture 38
basic paging 38
current page pointer 39

index memory page pointer 40
interrupts 39
MVI instructions 39
register definitions 41
stack operations 38
ReadBlock function in SROM 34
reference of all registers 163
register conventions 13, 163
register definitions
CPU core 31
digital clocks 99
general purpose IO 56
I2C slave 106
internal low speed oscillator 73
internal main oscillator 68
interrupt controller 46
IO analog multiplexer 91
POR and LVD 122
programmable timer 139
RAM paging 41
sleep and watchdog 82
SPI 125
supervisory ROM 37
system performance controller 64
system resets 116
USB, full-speed 147
register mapping tables
bank 0 registers 160
bank 1 registers 161
registers
bank 0 registers 164
bank 1 registers 212
core register summary 24
internal M8C registers 26
maneuvering around 163
mapping tables 159
reference of all registers 163
system resources register summary 93
regulated IO 87
application overview 75, 88
architecture 74, 87
bias generator 88
charge pump 88
comparator 88
pass transistors 88
replica structure 88
replica structure in regulated IO 88
RES_WDT register 82, 207
RX Reg Full bit 168

S

serial peripheral interconnect, *See* SPI
Slave bit 214
slave function for SPI 124
slave operation, I2C 105
sleep and watchdog 78
application overview 81
architecture 78
bandgap refresh 85
register definitions 82
sleep sequence 84
sleep timer 81
timing diagrams 84
wake up sequence 85
watchdog timer 86
Sleep bit
in CPU_SCR0 register 211
in INT_CLR0 register 196
in INT_MSK0 register 204
Sleep bits 228
sleep timer 81
SLIMO bit 210
SLP_CFG register 82, 235, 236, 237
SPI 123
architecture 123
configuration register 128
control register 127
data registers 125
master data register definitions 126
master function 124
protocol function 123
register definitions 125
slave data register definitions 126
slave function 124
timing diagrams 129
SPI bit
in INT_CLR0 register 196
in INT_MSK0 register 204
SPI Complete bit 168
SPI timing diagrams
SPI mode timing 129
SPIM timing 130
SPIS timing 134
SPI_CFG register 128, 214
SPI_CR register 127, 168
SPI_RXR register 126, 167
SPI_TXR register 125, 166
SRAM with USB 142
SROM, *See* supervisory ROM
SS_ bit 214
SS_EN_ bit 214
stack operations in RAM paging 38
START bit 181, 184, 185
start, how to 12
STK_PP register 42, 189
STOP bit 211
Stop IE bit 193
Stop Status bit 194
summary of registers
mapping tables 159
PSoC core 24
system resources 93
supervisory ROM
architecture 32

Calibrate0 function 36
Calibrate1 function 36
EraseAll function 35
EraseBlock function 35
function descriptions 33
ProtectBlock function 35
ReadBlock function 34
SWBootReset function 33
TableRead function 35
WriteBlock function 34
SWBootReset function in SROM 33
switch operation in digital clocks 97
system performance controller
application overview 65
architecture 59
system resets 114
architecture 114
external reset 118
functional details 120
pin behavior 114
power modes 120
power on reset 118
register definitions 116
timing diagrams 118
watchdog timer reset 118
system resources
architecture 92
overview 10, 92
register summary 93

T

TableRead function in SROM 35
technical support 12
Timer1/0 bits
in INT_CLR0 register 197
in INT_MSK0 register 204
timing diagrams
I2C slave 111
sleep and watchdog 84
SPI 129
system resets 118
TMP_DRx register 41
Transmit bit 194
Trim bits in IMO_TR register 233, 238
TX Reg Empty bit 168

U

units of measure 13
upgrades 12
USB_CR0 171
USB_CR0 register 147, 171
USB_CR1 register 157, 215
USB_SOFx register 147, 169
USB, *See* full-speed USB
USBIO_CR0 register 148, 172
USBIO_CR1 register 148, 173

V

V Monitor bit
in INT_CLR0 register 197, 199
in INT_MSK0 register 204
VLT_CMP register 122, 232
VLT_CR register 122, 231
VM bits 231

W

watchdog timer reset 118
WDRS bit 211
WDSL_Clear bits 207
WriteAndVerify function in SROM 36
WriteBlock function in SROM 34

X

XIO bit 208
XRES reset 118

Z

Zero bit 209