Application Note, V1.1, Aug. 2008

AP16147

XC22xx/XC22xxM

Interfacing XC22xx-USIC_SPI to TLE6209R

Microcontrollers

Never stop thinking

Edition 2008-08-21

Published by
Infineon Technologies AG
81726 München, Germany

LEGAL DISCLAIMER

THE INFORMATION GIVEN IN THIS APPLICATION NOTE IS GIVEN AS A HINT FOR THE IMPLEMENTATION OF THE INFINEON TECHNOLOGIES COMPONENT ONLY AND SHALL NOT BE REGARDED AS ANY DESCRIPTION OR WARRANTY OF A CERTAIN FUNCTIONALITY, CONDITION OR QUALITY OF THE INFINEON TECHNOLOGIES COMPONENT. THE RECIPIENT OF THIS APPLICATION NOTE MUST VERIFY ANY FUNCTION DESCRIBED HEREIN IN THE REAL APPLICATION. INFINEON TECHNOLOGIES HEREBY DISCLAIMS ANY AND ALL WARRANTIES AND LIABILITIES OF ANY KIND (INCLUDING WITHOUT LIMITATION WARRANTIES OF NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OF ANY THIRD PARTY) WITH RESPECT TO ANY AND ALL INFORMATION GIVEN IN THIS APPLICATION NOTE.

Information

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

Warnings

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

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

AP16147		
Revision History:	2008-08	V1.1
Previous Version:	none	
Page	Subjects (major changes since last revision)	
V1.0	Initial Version	
V1.1	Formal changes	

We Listen to Your Comments

Any information within this document that you feel is wrong, unclear or missing at all?
Your feedback will help us to continuously improve the quality of this document.
Please send your proposal (including a reference to this document) to:

mcdocu.comments@infineon.com

Table of Contents **Page**

1 Introduction

Embedded Microcontroller applications may require diagnostic information from various devices by using different kind of protocols. SPI protocol is one among them. In this context, one device (Slave) will provide Diagnostic information for the other device (Master) to process.

Infineon XC22xx or Xc22xxM Microcontrollers can be used to process such diagnostic information's.

In embedded systems the SPI protocol is used as synchronous full duplex communication. The Devices communicate in master/slave mode where the master device initiates the clock to communicate data in either or both directions. To choose a particular in slave device, master has to send active slave select signal.

1.1 Purpose of Application Note

This Application Note is intended to facilitate user to generate SPI protocol using XC22xx Microcontroller. An illustration provided in this application note is to interface XC2287 Microcontroller with TLE 6209R DC-Motor controller. This should facilitate user to write there own application to generate SPI protocol using XC22xx or XC22xxM Microcontrollers.

In this application note, the SPI protocol is used to set the control word and to extract the diagnostic information from the TLE7209R controller.

1.2 Scope

The Application note provides information on

- Utilized XC2287 micro controller(as master) to interface TLE6209R(as slave) for illustration
- USIC peripheral of XC2287 microcontroller is used for SPI protocol generation
- Schematic or circuit diagram provided to interface XC2287 to TLE6209R
- The illustartion software package provided to interface TLE6209R

The following information is part of Illustration software to facilitate interface between XC2287 and TLE6209R. But the details on these information's are out off scope of this application note.

- System Clock Generation
- Port Pin Configurations
- PWM Signal Generation
- Running DC Motor using TLE6209R IC
- USIC_ASC/UART- Software for Hyper Terminal communication

1.3 Prerequisites to understand App Note

- Knowledge on SPI protocol
- Knowledge on XC22xx and XC22xxM microcontrollers
- Knowledge on peripheral module USIC
- Knowledge on XC2287 Easy Kit hardware
- Knowledge on TLE6209 H-Bridge Motor Controller

1.4 Abbreviations

Table 1 Abbreviations used in this App Note

Abbreviation	Explanation
API	Application Programming Interface
ASC	Asynchronous Serial Channel
CCU6	Capture Compare Unit6 (Microcontroller peripheral unit6)
CPU	Central Processing Unit
HW	Hardware
IFX	Infineon Technologies
PPP	Pre Protocol Processor
PWM	Pulse Width Modulation
MRST	Master Receive Slave Transmit
MTSR	Mater Transmit Slave Receive
SFR	Special Function Register
SCLK	Shift Clock on SPI bus
SCU	System Control Unit (Hardware Module)
SCS	System Configuration Software
SPI	Serial Peripheral Interface
SSC	Synchronous Serial Channel referred for SPI
SW	Software
UART	Universal Asynchronous Receiver and Transmitter
USIC	Universal Serial Interface Channel (Peripheral Hardware Unit)

2 SPI Overview

The SPI (Serial Peripheral Interface) has a 4-wire synchronous serial interface.

- SCLK — Serial Clock (output from master)
- MOSI/SIMO/MTSR/SDI — Master Output or Transmit, Slave Input or Receive (output from master)
- MISO/SOMI/MRST/SDO — Master Input or Receive, Slave Output or Transmit (output from slave)
- SS / SLSx — Slave Select Signal (active low; output from master)

To communicate with any Slave Device, first master has to select Slave Device by using Slave Select Signal. Parameters called clock polarity and clock phase determine the edges of the master clock signal on which the data are driven and sampled.

The following **Figure 1** represents the Single Master and 3 Slave SPI communications.

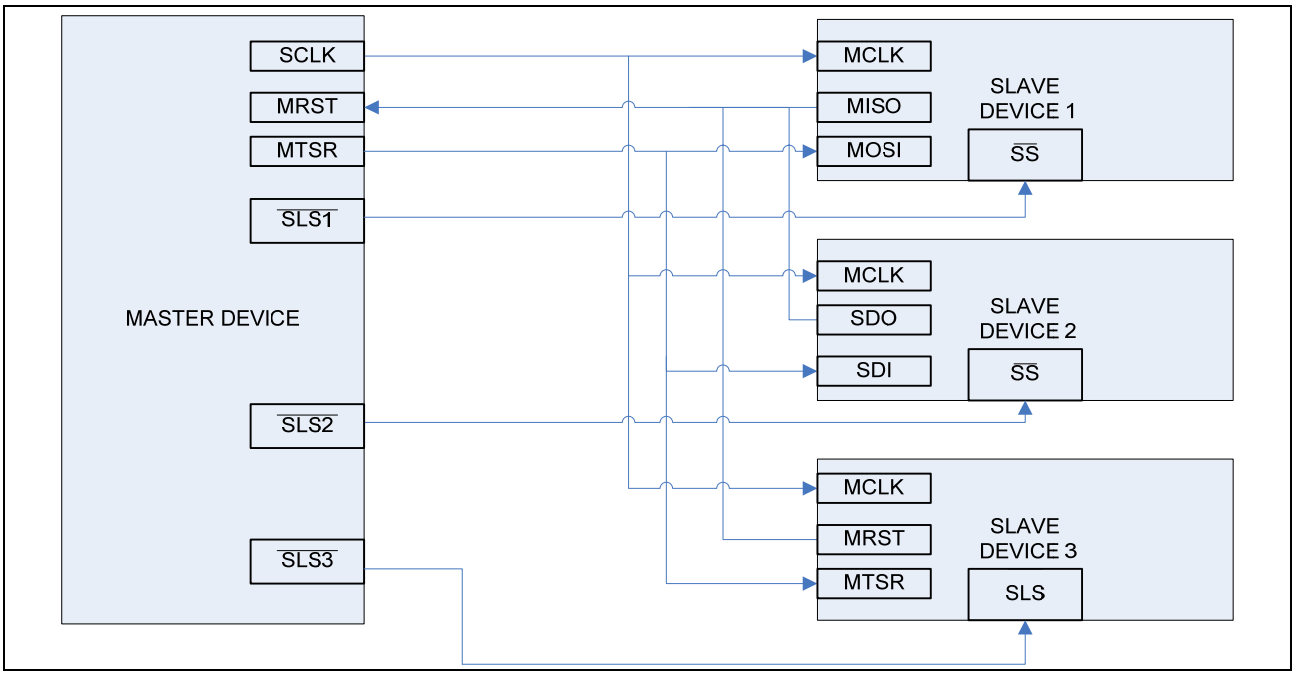


Figure 1 SPI Signals

3 XC22xx/XC22xxM Hardware Overview

Note: XC22xxM is enhanced version of XC22xx Microcontroller. Unless otherwise mentioned, the information regarding XC22xx holds good for XC22xxM also. In this App Note XC2200 is referred for XC22xx micro.

Note: XC2000/XE166 family microcontrollers containes USIC and CCU6 peripheral modules. In case user would like to interface TLE6209R using XC2000/XE166 family microcontrollers, user can use this APP Note TLE6209R-interface concept.

The XC22xx devices are members of Infineon XC2000 Family with full featured 16-bit single-chip CMOS microcontrollers. The XC22xx combines the extended functionality and performance of the C166SV2 Core with powerful on-chip peripheral subsystems and on-chip memory units and provides a means for power reduction on various levels.

The XC2200 derivatives are designed for Body Applications such as:

• Body Control Module

• Gateway

• Door

• HVAC

Infineon's XC2200 devices were specifically designed to fulfill the requirements of today's and future body applications by providing high performance (control and DSP) at low power consumption. All XC2200 members are fully software and pin compatible. Designers of automotive body systems can choose the optimal combination of memory, peripherals, temperature, and packaging to match the application's requirements. Infineon's high quality standards make the XC2200 Controllers an ideal choice for flexible and future-oriented body systems. USIC hardware module is one among on-chip peripherals supported by XC22xx microcontroller. USIC is a flexible interface module that supports the serial communication protocols like ASC, SSC, IIC and IIS.

The architecture of the XC2200 core combines the advantages of both RISC and CISC processors in a very well-balanced way. This computing and controlling power is completed by the DSP-functionality of the MAC-unit. The XC2200 integrates this powerful CPU core with a set of powerful peripheral units into one chip and connects them very efficiently. On-chip memory blocks with dedicated buses and control units store code and data. This combination of features results in a high performance microcontroller, which is the right choice not only for today's applications, but also for future engineering challenges.

The architecture of both XC22xx and XC22xxM microcontrollers can be seen in the below figures. Also user can see that, the marked red colored peripheral module (USIC Channels) is used for TLE6209R interface.

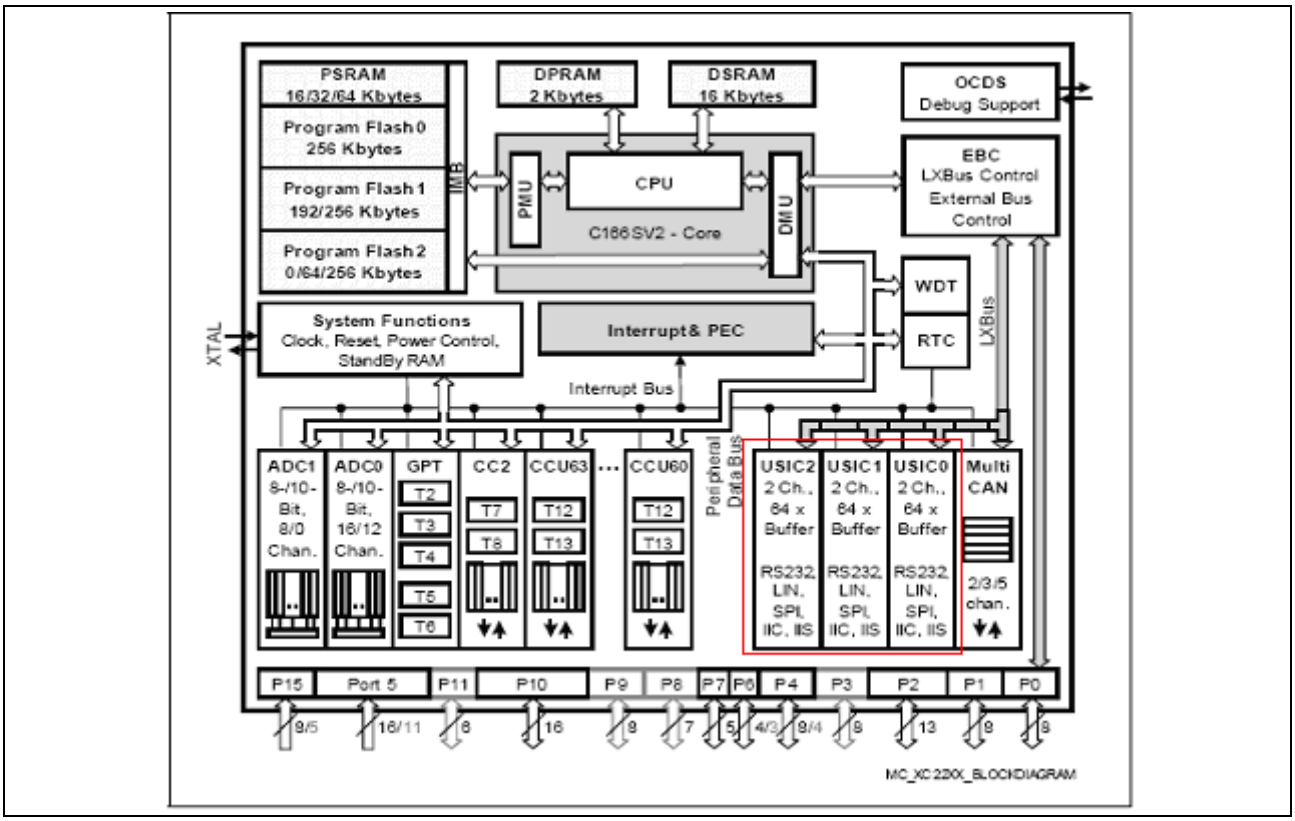


Figure 2 XC22xx Architecture

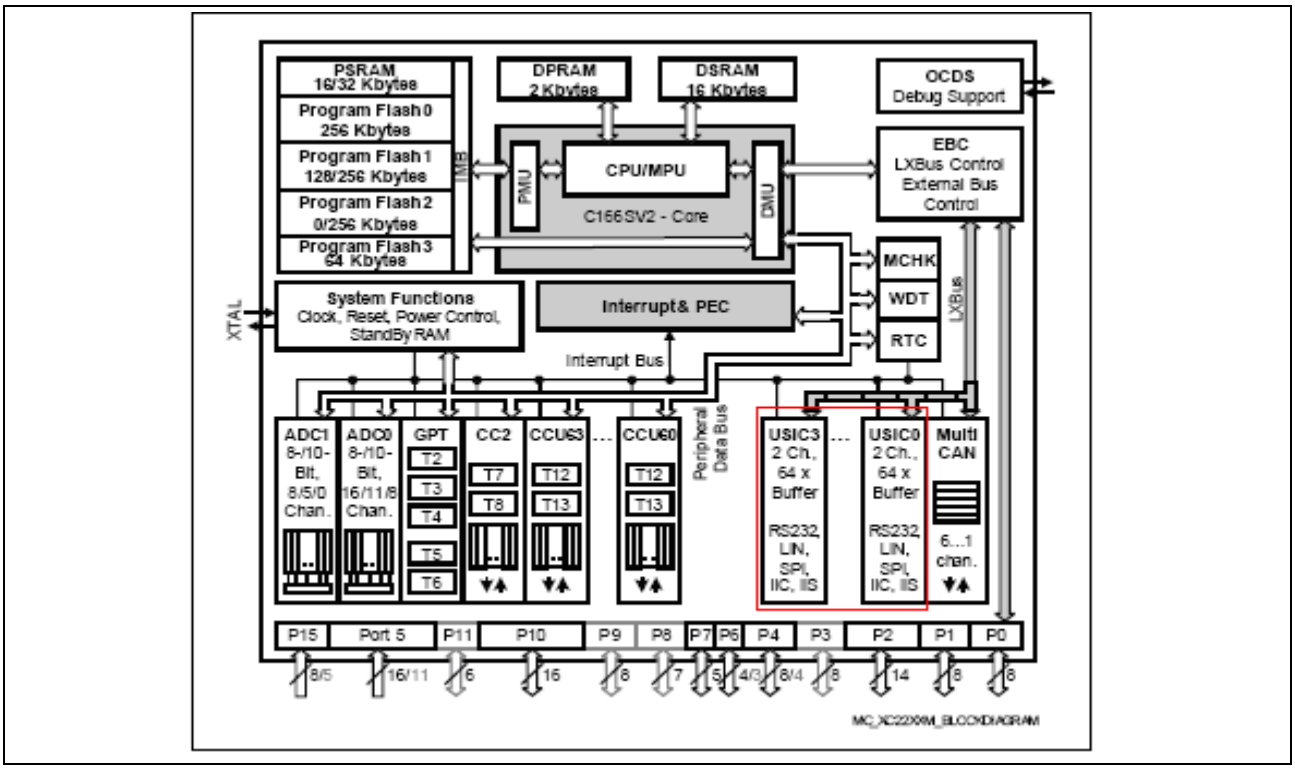


Figure 3 XC22xxM Architecture

4 Software Architecture

The overall objective of this Application Note is to illustrate the process of SPI communication with TLE6209R. To achieve this, the below architecture is used.

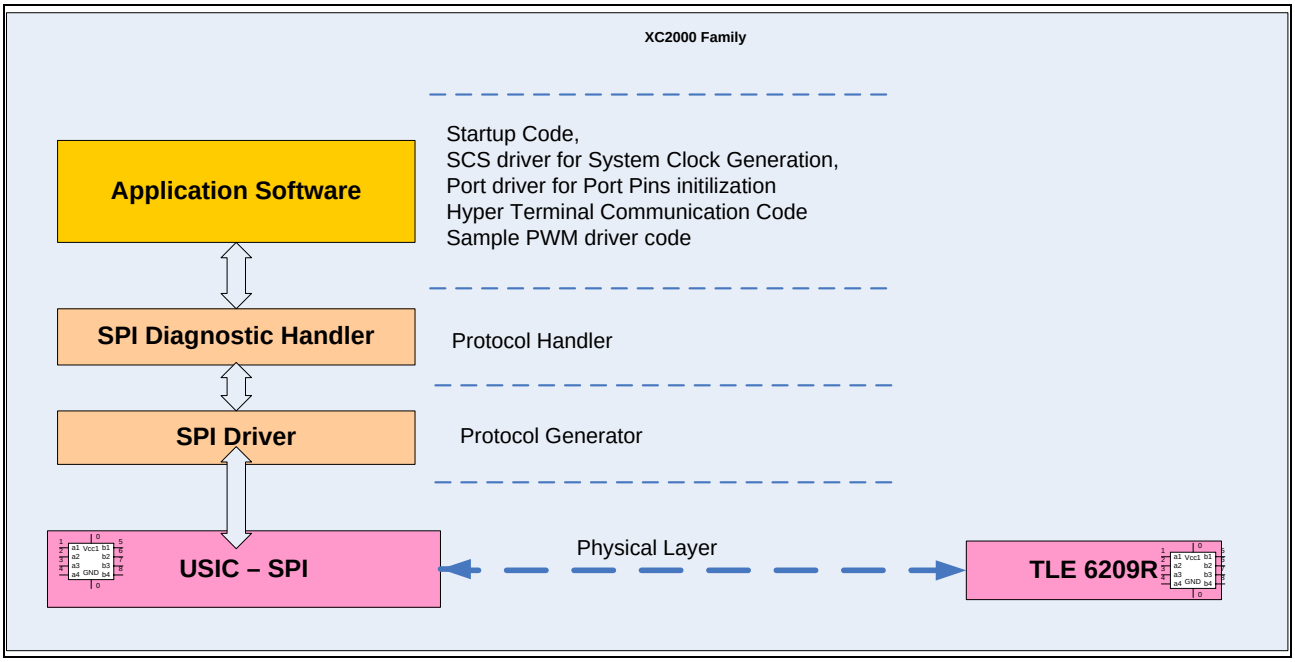


Figure 4 Software Architecture

The details on each layer of Software Architecture can be found in the following chapters.

5 Application Layer

This layer performs

- The initialization of System clock,
- Port configuration,
- Setting Interrupt priorities and processing Interrupt Service Routines
- Sample PWM signal generation
- Hyper Terminal Communication

5.1 System Clock Configuration

The USIC peripheral module clock and the required baud rate are basically derived from the System Clock configured. System Configuration Software is implemented to generate 80Mhz System Frequency.

5.2 Port Configuration

The port pins configuration is necessary to route the signals from various peripheral units through the appropriate Microcontroller pins.

Once the choice of the pins needed for the Spi Bus (USIC Channels) is made, the ports must be initialized with corresponding configuations. The programming descriptions of port-pins initialization is out of scope for this application note, but software has been provided.

The following table represents the possible port pins that can be configured for each USIC Channel,

Table 2 USIC Channel - MRST(MISO), MTSR(MOSI), SCLK Pin Connections

USIC Channel	MRST (MISO)	MTSR (MOSI)	SCLK OUT (Shift Clock)	Chip Select
U0C0	DX0A: P10.0 DX0B: P10.1 DX0C: P10.6 DX0D: P7.4 DX0E: P2.3 DX0F: P2.4	P 2.3 P 7.3 P 10.1 P 10.6	P2.5 P10.2	SELO0: P 2.6, 10.10 SELO1: P 2.7 SELO2: P 2.11 SELO3: P 2.10,10.4 SELO4: P 3.4,10.9,2.12 SELO5: P 3.5 SELO6: P 3.6 SELO7: P 10.15
U0C1	DX0A: P10.0 DX0B: P10.7 DX0C: P10.14 DX0E: P 2.10 DX0F: P 7.3	P 2.9 P 2.10 P 7.3 P 7.4 P 10.0 P 10.7 P 10.14 P 10.15	P 2.8 P 7.4 P 10.5	SELO0: P 2.7, 10.8 SELO1: P 2.6 SELO2: P 2.11 SELO3: P 2.12
U1C0	DX0A: P 0.0 DX0B: P 0.1 DX0C: P 10.12 DX0D: P 10.13	P 0.0 P 0.1 P 10.12 P 10.13 P 10.15	P 0.2 P 10.11	SELO0: P 0.3, 10.6 SELO1: P 0.4,10.14 SELO2: P 0.5, 10.15 SELO3: P 0.7, 10.13 SELO4: P 1.0 SELO5: P 1.1 SELO6: P 1.2 SELO7: P 1.3
U1C1	DX0A: P 0.6 DX0B: P 0.7 DX0E: P 6.0	P 0.6 P 0.7 P 6.0 P 6.1	P 0.5 P 6.2	SELO0: P 0.4, 6.3 SELO1: P 0.3 SELO2: P 1.6 SELO3: P 1.5 SELO4: P 1.4
U2C0	DX0A: P 3.0 DX0B: P 3.1 DX0C: P 1.5 DX0D: P 1.6 DX0E: P 9.5	P 3.0 P 3.1 P 1.6 P 9.4 P 9.5	P 3.2 P 1.7	SELO0: P 3.3 SELO1: P 3.4 SELO2: P 3.5 SELO3: P 3.7 SELO4: P 1.3 SELO5: P 1.4
U2C1	DX0A: P 3.6 DX0B: P 3.7 DX0C: P 1.1 DX0D: P 1.2	P 3.6 P 3.7 P 1.1	P 3.5 P 1.2	SELO0: P 3.4 SELO1: P 3.3
U3C0	DX0A: P 10.3 DX0B: P 4.5	P10.4 P4.5	P10.14 P4.2	SELO0: P10.11 SELO1: P10.2

		P4.6		SELO2: P4.4 SELO3: P4.1
U3C1	DX0A: P 2.10 DX0B: P 11.4	P2.11 P11.4 P11.2	P11.0	SELO0: P11.1 SELO1: P11.5
DX0<A-G>: Pins for Input Stage. Lines A to G. SELO<0-7>: Slave Select Output lines 0 to 7				

5.3 Interrupt Priorities and Service Routines

The Interrupt priorities for the various Interrupt Requests (In this case USIC Service Requests) will be set in Hwal_Irq.c file

5.4 PWM Signal Generation (Sample)

To create open load and close load diagnostic error message it is necessary to provide PWM signal to TLE6209R IC. To generate PWM signal CCU6 hardware resource is utilized

5.5 Hyper Terminal Communication

USB interface is used between PC and XC2287 easy kit to establish the Hyper Terminal communication.

It is recommended for the user to go through XC22xx Easy Kit User Manual to know connection details and DIP switch positions to establish communication between PC and Easy Kit via USB-UART Bridge.

USIC0 Channel0 (U0C0) is used for UART (ASC) communication in this application software. The port pins configuration used for hyper terminal communication is given in the below table.

Table 3 Port Pins used for Hyper Terminal communication

Port Pins	SCLK (Shift Clock) Chip Select
P7.3	ALT3 Output : TXD (DOUT of U0C0)
P7.4	Input : DX0D of U0C0

The user has to do the following settings in the Hyper Terminal for establishing the communication

Baud Rate: 19200, **Parity**: None, **Data Bits**: 8, **Stop Bits**: 1

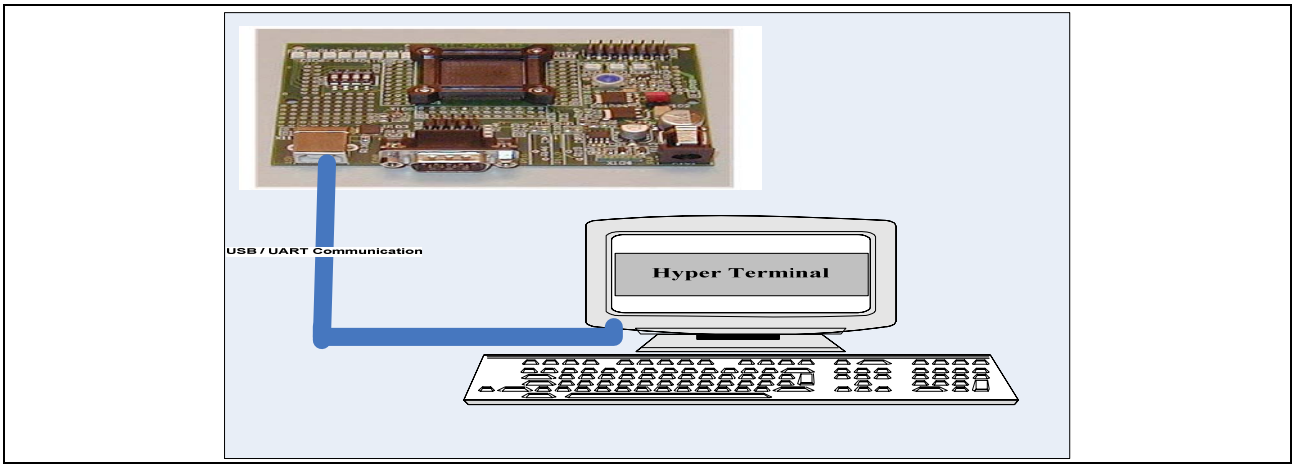


Figure 5 Hyper Terminal Connection

6 SPI Diagnostic Handler

This layer consists of software which facilitates the user to send the control word to TLE6209R and to display the received diagnostic data from TLE6209R.

The menu for user interaction is shown in the below figure.

```
/*****************************************************************************/

                          SPI HANDLER
                    ENTER YOUR OPTION
              < 1 >  Write Control word to TLE6209R
              < 2 >  Read Diagnostic data from TLE6209R
              < x >  Back To Main Menu

/*****************************************************************************/
```

Figure 6 SPI Diagnostic Handler Menu

6.1 Handler Option 1:

- Asks the user to enter the Control word to TLE6209R.
- Writes the entered control word on SPI bus.
- Reads the Diagnosis data from TLE6209R.
- Displays the diagnostic data

6.2 Handler Option 2:

- Writes the previously used control word on SPI bus
- Prints the diagnostic data previously read

6.3 Handler Option 3:

This will de-initialize the SPI driver and takes the control back to main menu

7 Spi Driver Layer

The SPI layer provides the following Application Programming Interfaces.

7.1 Spi_Init

Table 4 Spi_Init() API

Function Name	Spi_Init
Syntax	void Spi_Init(Spi_ConfigType *SpiConfigPtr)
Parameters (In)	Spi_ConfigType *SpiConfigPtr Where Spi_ConfigType **[7.5.1]** is of struct type.
Parameters (Out)	None
Description	This API initializes Initializes all SPI relevant registers, variables for all configured SPI channels (i.e. uses all sub structures of main structure pointed by parameter SpiConfigPtr) This function has to be called before any other function call.

7.2 Spi_DeInit

Table 5 Spi_DeInit

Function Name	Spi_DeInit
Syntax	void Spi_DeInit(void)
Parameters (In)	None
Parameters (Out)	None
Description	Service for SPI Deinitialization. Does De-initialization of all channels which have initialized via Spi_Init() API

7.3 Spi_WriteData

Table 6 Spi_WriteData

Function Name	Spi_WriteData
Syntax	void Spi_WriteData(uint8 SpiChannel, const uint8 *DataBufferPtr uint16 NoOfData)
Parameters (In)	SpiChannel - Spi Channel or configuration index DataBufferPtr - Pointer to source data buffer NoOfData - Number of Data's to be transmitted
Parameters (Out)	None
Description	Service to transmit data on the SPI bus. Function initiates transmission, and continues transmission in the background.

7.4 Spi_ReadData

Table 7 Spi_ReadData

Function Name	Spi_ReadData
Syntax	void Spi_ReadData (uint8 SpiChannel, uint8 *DataBufferPtr, uint16 NoOfData)
Parameters (In)	SpiChannel - Spi Channel or configuration index DataBufferPtr - Pointer to copy data received from TLE6209R or any configured slave device NoOfData - Number of Data's to be received
Parameters (Out)	None
Description	This function copies data received from slave device to the pointer DataBufferPtr.

7.5 Spi Driver Configuration

This section summaries all configuration parameters and their description.

To configure the Spi driver, Spi_Lcfg.c and Spi_Cfg.h files have to be used

7.5.1 Spi_Lcfg.c

The parameters which needs to be configured in Spi_Lcfg.c file are mentioned below

Table 8 Spi_Lcfg.c configuration parameters

Name of the parameter	Description
ShiftDirection	This parameter defines the first starting bit or Shift direction of data words for transmission/reception. Type : uint16 Allowed Range : SPI_DATA_LSB_FIRST SPI_DATA_MSB_FIRST ***Attention: For TLE6209R Interface, this parameter is configured as SPI_DATA_LSB_FIRST***
DataWidth	Data width: This parameter is the width of a transmitted data bits. Note that, the number of data bits transmitted/received is equal to configured value plus one. Type : uint8 Allowed Range : 0 to 15 ***Attention: For TLE6209R Interface, this parameter is configured as : 7*** ***[i.e number of data bits transmitted/received = 8]***
AssignedUsicChannel	Specifies the identification (ID) for a hardware USIC Channel (Microcontroller Dependent) Type : uint8 Allowed Range : USIC0_CHANNEL0 USIC0_CHANNEL1 USIC1_CHANNEL0 USIC1_CHANNEL1 USIC2_CHANNEL0 USIC2_CHANNEL1 USIC3_CHANNEL0 USIC3_CHANNEL1 ***Attention: For TLE6209R Interface, this parameter is configured as : USIC2_CHANNEL0*** *Note: The variants of XC22xx and XC22xxM microcontrollers are having various numbers of USIC channels. So it is recommended for the user to configure this parameter depending on the availability of USIC channels on a particular micro.* Example: In XC2287 microcontroller USIC3_CHANNEL0 and USIC3_CHANNEL1 are not provided.

SlaveSelectOut	Slave Select Output line (Microcontroller Dependent) Type : uint8 Allowed Range : SLS_SELO0 SLS_SELO1 SLS_SELO2 SLS_SELO3 SLS_SELO4 SLS_SELO5 SLS_SELO6 SLS_SELO7 ***Attention: For TLE6209R Interface, this parameter is configured as : SLS_SELO0*** *Note: It is recomnded for the user to configure this parameter depending on the availability of Slave Select Signal pins in a particular microcontroller.*
ChipSelectPolarity	This parameter defines the active polarity of Chip Select signal. Type : uint8 Allowed Range : SPI_CS_POLARITY_LOW SPI_CS_POLARITY_HIGH ***Attention: For TLE6209R Interface, this parameter is configured as : SPI_CS_POLARITY_LOW***
SCLKConfig	Shift Clock Output Configuration: This parameter defines the Shift Clock polarity on which the data is shifted out or in. Type : uint8 Allowed Range : SCLKOUT_SCLK SCLKOUT_NOT_SCLK SCLKOUT_SCLK_DELAY_FPDIV SCLKOUT_NOT_SCLK_DELAY_FPDIV ***Attention: For TLE6209R Interface, this parameter is configured as : SCLKOUT_SCLK*** Refer **10.2.5]** for details on “Configuring Shift Clock for Data Transmission / Reception”
DividerMode	Divider Mode: Normal Divider or Fractional Divider Type : uint16 Allowed Range : SPI_NORMAL_DIVIDER SPI_FRACTIONAL_DIVIDER ***Attention: For TLE6209R Interface, this parameter is configured as : SPI_NORMAL_DIVIDER*** Refer **[10.2.4]** to know more on the impact of this parameter selection on baudrate generation.
STEP	Step value to facilitate required baud generation Type : uint16 Allowed Range : 0 to 1023

	Attention: For TLE6209R Interface, this parameter is configured as : 191 Refer **[10.2.4]** to know more on the impact of this parameter selection on baudrate generation.
PDIV	Baud Reload Value based on the SPI System Clock and the configured baud rate Type : uint16 Allowed Range : 0 to 1023 ***Attention: For TLE6209R Interface, this parameter is configured as : 4*** Refer **[10.2.4]** to know more on the impact of this parameter selection on baudrate generation.
CTQSEL	Input Clock selection for time quanta counters (Transmitter and Receiver) and also for generating leading and trailing delays. Type : uint16 Allowed Range : CTQ_SEL_FPDIV CTQ_SEL_FPPP CTQ_SEL_SCLK CTQ_SEL_MCLK ***Attention: For TLE6209R Interface, this parameter is configured as : CTQ_SEL_FPPP*** Refer **[10.2.7]** to know more on the impact of this parameter selection on timing delay generation.
PCTQ	Pre-Divider for Time Quanta Counter : Divider factors to generate the Leading delay and trailing delay Type : uint16 Allowed Range : 0 to 3 ***Attention: For TLE6209R Interface, this parameter is configured as : 0*** Refer **[10.2.7]** to know more on the impact of this parameter selection on timing delay generation.
DCTQ	Denominator for Time Quanta Counter : Divider factors to generate the Leading delay and trailing delay Type : uint16 Allowed Range : 0 to 31 ***Attention: For TLE6209R Interface, this parameter is configured as : 4*** Refer **[10.2.7]** to know more on the impact of this parameter selection on timing delay generation.
CTQSEL1	Input Clock selection for time quanta counters and also for inter word delay and next frame delays. Type : uint16 Allowed Range : CTQ_SEL_FPDIV CTQ_SEL_FPPP CTQ_SEL_SCLK CTQ_SEL_MCLK ***Attention: For TLE6209R Interface, this parameter is configured as : CTQ_SEL_FPDIV*** Refer **[10.2.7]** to know more on the impact of this parameter selection on timing

	delay generation.
PCTQ1	Pre-Divider for Time Quanta Counter : Divider factors to generate the inter word delay and next frame delay Type : uint16 Allowed Range : 0 to 3 ***Attention: For TLE6209R Interface, this parameter is configured as : 0*** Refer **[10.2.7]** to know more on the impact of this parameter selection on timing delay generation.
DCTQ1	Denominator for Time Quanta Counter : Divider factors to generate the inter word delay and next frame delay Type : uint16 Allowed Range : 0 to 31 ***Attention: For TLE6209R Interface, this parameter is configured as : 0*** Refer **[10.2.7]** to know more on the impact of this parameter selection on timing delay generation.
PassiveDataLevel	This parameter defines the output level at the data output pin when no data is available for transmission. The configured value or level is the output with the first relevant transmit shift clock edge of a data word. Type : uint16 Allowed Range : SPI_PASSIVE_DATA_LOW SPI_PASSIVE_DATA_HIGH ***Attention: For TLE6209R Interface, this parameter is configured as : SPI_PASSIVE_DATA_HIGH***
InputVector	This parameter defines the input data source for the corresponding input line for the used Usic Channel Type : uint16 Allowed Range : SPI_SELECT_DATALINE_A SPI_SELECT_DATALINE_B SPI_SELECT_DATALINE_C SPI_SELECT_DATALINE_D SPI_SELECT_DATALINE_E SPI_SELECT_DATALINE_F SPI_SELECT_DATALINE_G SPI_SELECT_DATALINE_H ***Attention: For TLE6209R Interface, this parameter is configured as : SPI_SELECT_DATALINE_A*** *Note: Use SPI_SELECT_DATALINE_H for the input to make always High* To under stand how this parameter is utilized refer section :**[10.2.3]**

7.5.2 Spi_Cfg.h

The parameters which needs to be configured in Spi_Cfg.h file are mentioned below

Table 9

Name of the parameter	Description
SPI_HW_FIFO_USAGE	This parameter defines whether Spi driver has to use USIC HW FIFO or TBUF and RBUF for data transmission and reception. Type : #define Range : OFF ON ***Attention: For TLE6209R Interface, SPI_HW_FIFO_USAGE is defined with : OFF***
SPI_CHANNEL_NAME	This parameter is the symbolic name to be used in place of SPI channel while calling different API's. This symbolic name allows accessing Channel data. This informs the configuration index Type : #define and type casted value with (uint8) Example : #define SPI_CHANNEL_TO_TLE6209R ((uint8)0) #define SPI_CHANNEL_SLAVE_X ((uint8)1) #define SPI_CHANNEL_SLAVE_Y ((uint8)2) ***Attention: For TLE6209R Interface, SPI_CHANNEL_WITH_TLE6209R is used as symbolic name for Spi Chanel 0***
MAX_NUM_OF_SPI_CHANNELS	This parameter defines number of configured SPI channels Type : #define and type casted value with (uint8) Example : #define MAX_NUM_OF_SPI_CHANNELS ((uint8)3) ***Attention: For TLE6209R Interface, only single SPI channel is used. So it has been defined with ((uint8)1)***
USICx_CHy_SPI Where x [Usic Module] = 0 to 3 y [Channel number] = 0 to 1	This parameter defines whether user configured particular USIC Channel for SPI driver. If user uses, then make it to ON otherwise OFF Type : #define ***Attention: For TLE6209R Interface, the following definitions have been used.*** #define USIC0_CH0_SPI (OFF) #define USIC0_CH1_SPI (OFF) #define USIC1_CH0_SPI (OFF) #define USIC1_CH1_SPI (OFF) #define USIC2_CH0_SPI (ON) #define USIC2_CH1_SPI (OFF) #define USIC3_CH0_SPI (OFF) #define USIC3_CH1_SPI (OFF)

7.5.3 Code Generator Tool (Spi_ConfigTool.xls)

Along with this App note, Excel VB based tool is been provided. This tool will facilitate the user to generate Spi_Lcfg.c and Spi_Cfg.h configuration files. To understand usage of this tool, refer section **[10.3].**

8 Physical Layer

The physical layer consists of

- XC2287 Microcontroller as a master.
- TLE 6209R as a Slave device.
- SPI bus interface

8.1 XC2287 USIC-SPI as master

XC2287 is one of microcontroller in the family of XC22xx, which has been used to communicate with TLE6209R device.

In this application USIC channel is programmed to work as SSC or SPI protocol.

Refer Appendix section **[10.1]** for details on USIC hardware peripheral module that is used for SPI communication.

8.2 TLE6209R as SPI Slave

The TLE 6209R is an integrated power H-Bridge for driving bidirectional loads such as DC-Motors. Operation modes forward, reverse and brake are invoked by two control pins PWM and DIR. Protection and a reliable diagnosis of over-current, open-load, short-circuit to ground, to the supply voltage or across the load are integrated. The SPI protocol is used for bidirectional communication with the control unit. Application note covers only the SPI communication part for setting the control information and extracting diagnostic data from TLE 6209R chip.

For more details about the TLE6209R please refer to the TLE6209R Data Sheet

8.3 SPI Bus Interface

To set control word for TLE6209R and to retrieve diagnostic data from TLE-6209R, the USIC module always acts as master device on the SPI bus which will provide shift clock for SPI communication.

Control words are shifted out of the USIC module into the TLE6209R using line “Master Transmit Slave Receive” (MTSR). The diagnostic data from TLE6209R will be read on line “Master Receive Slave Transmit “(MRST) in sync with Shift Clock.

The 8-bit programming word or control word is read in via the SDI serial data input of TLE6209R and this is synchronized with the serial clock input SCLK. The status word appears synchronously at the SDO serial data output which is read by USIC-SPI Master. At each SPI transmission, the diagnosis bits as currently valid in the error logic are transmitted.

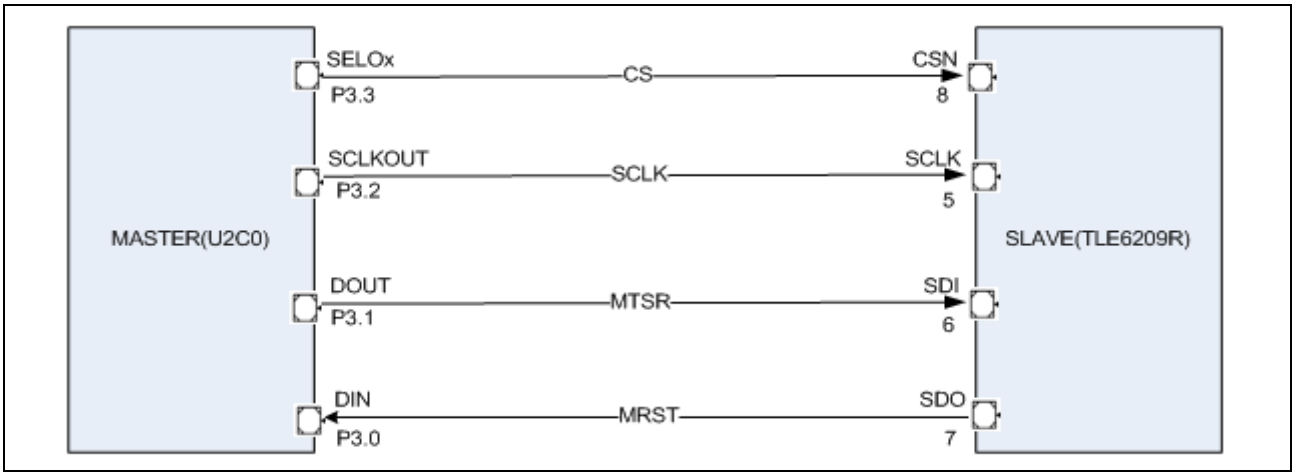


Figure 7 SPI Bus Interface

8.4 Spi Timing Diagrams

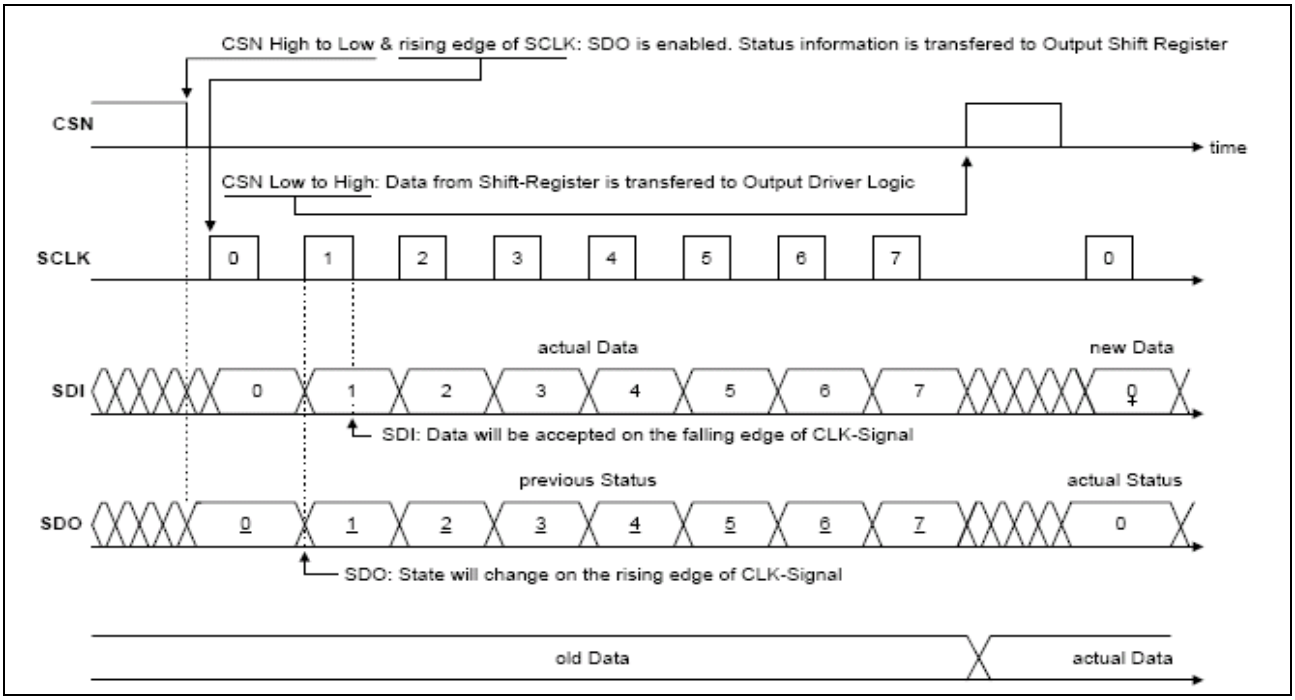


Figure 8 Standard Data Transfer Timing

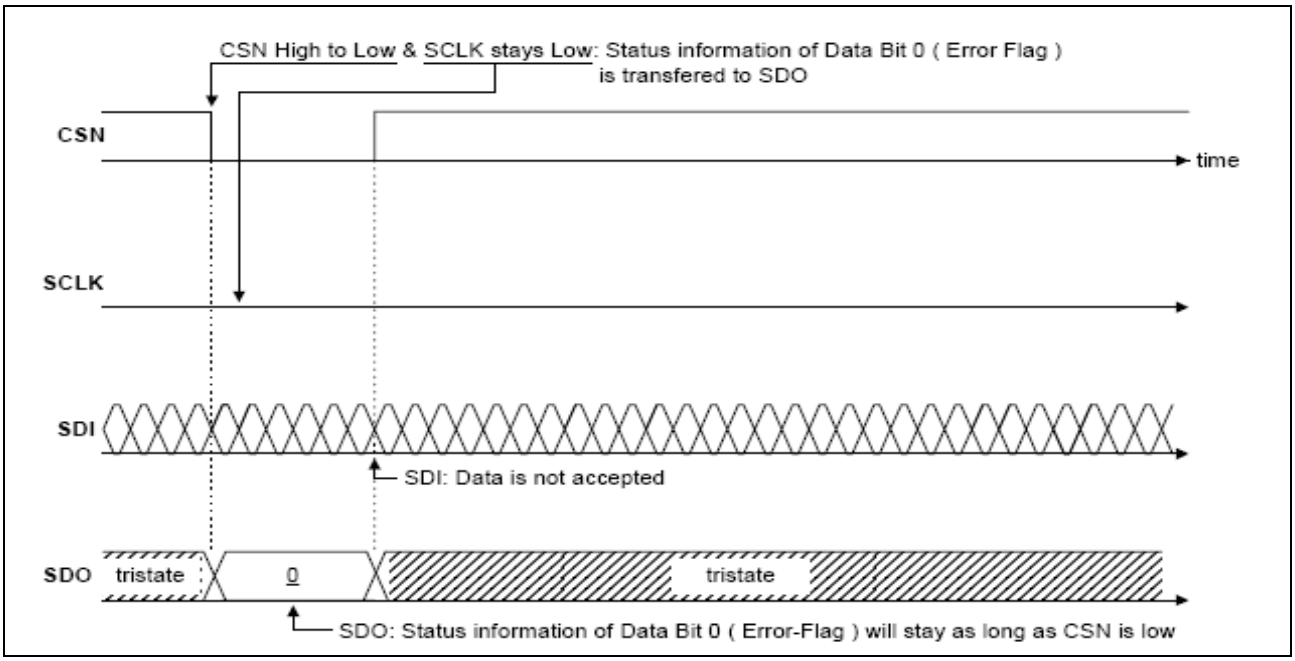


Figure 9 Timing for diagnostic error detection

8.5 Circuit Diagram

The circuit diagram used to interface XC2287 with TLE6209R is given below

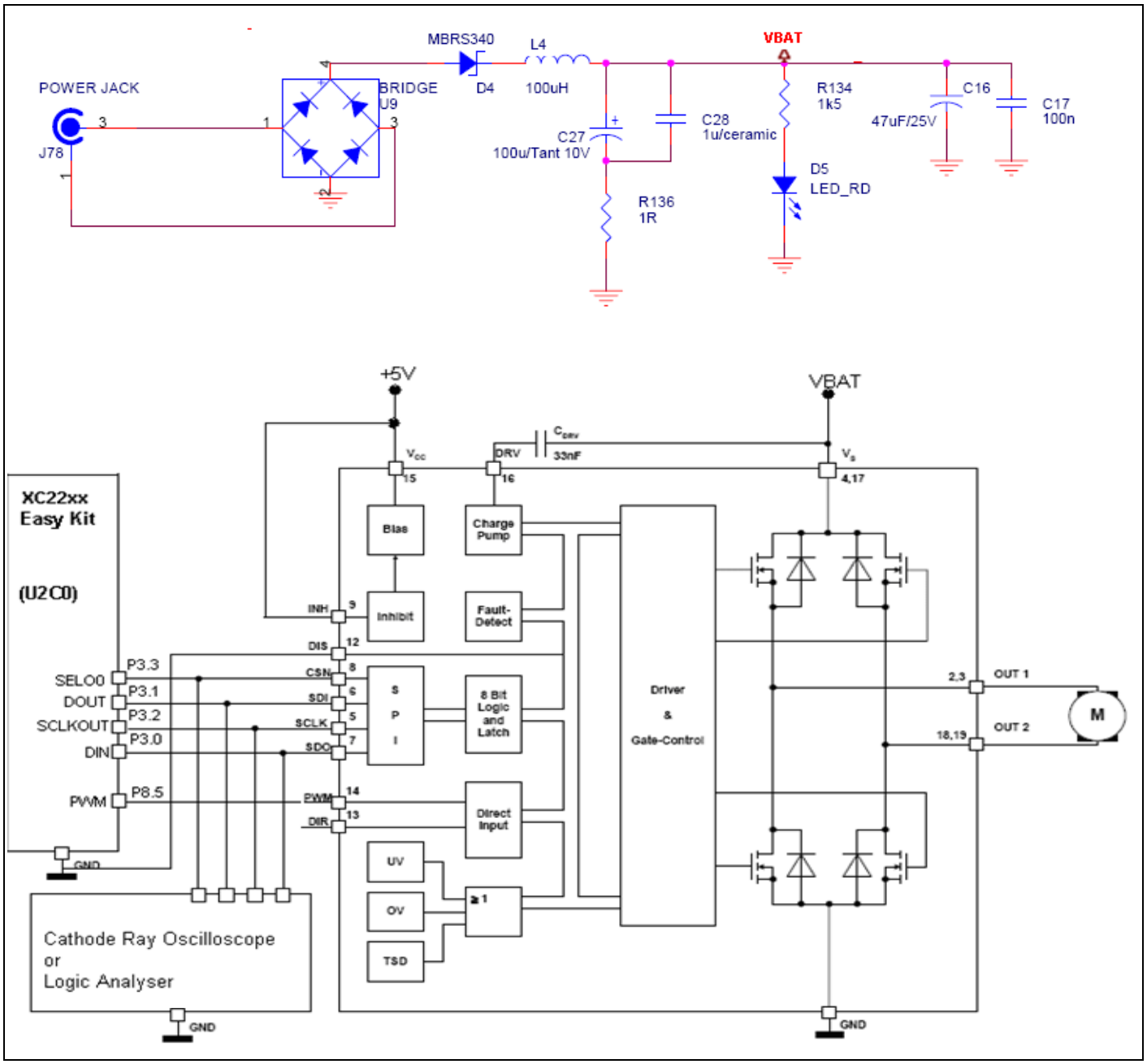


Figure 10 Circuit Diagram for XC2287 to TLE6209R interface

8.6 Sample Test Result Waveforms

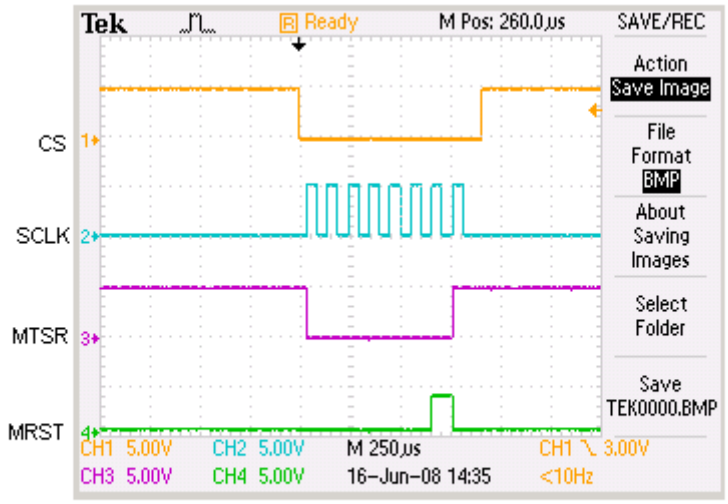


Figure 11 Diagnostic data without error

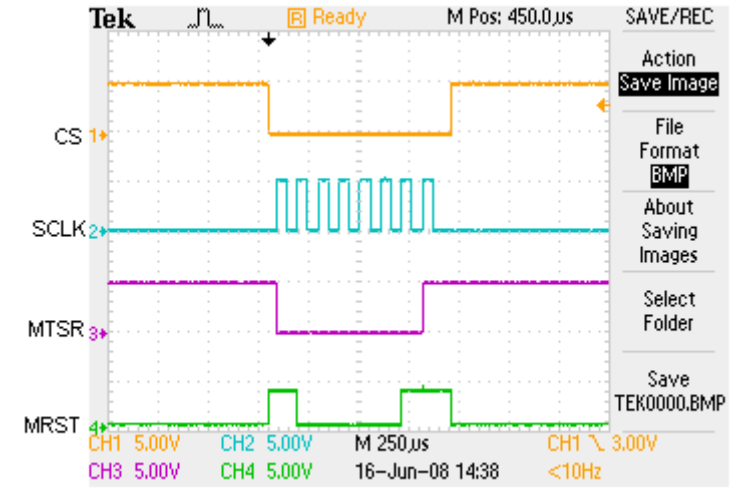


Figure 12 Diagnostic data during Power Supply Fail error

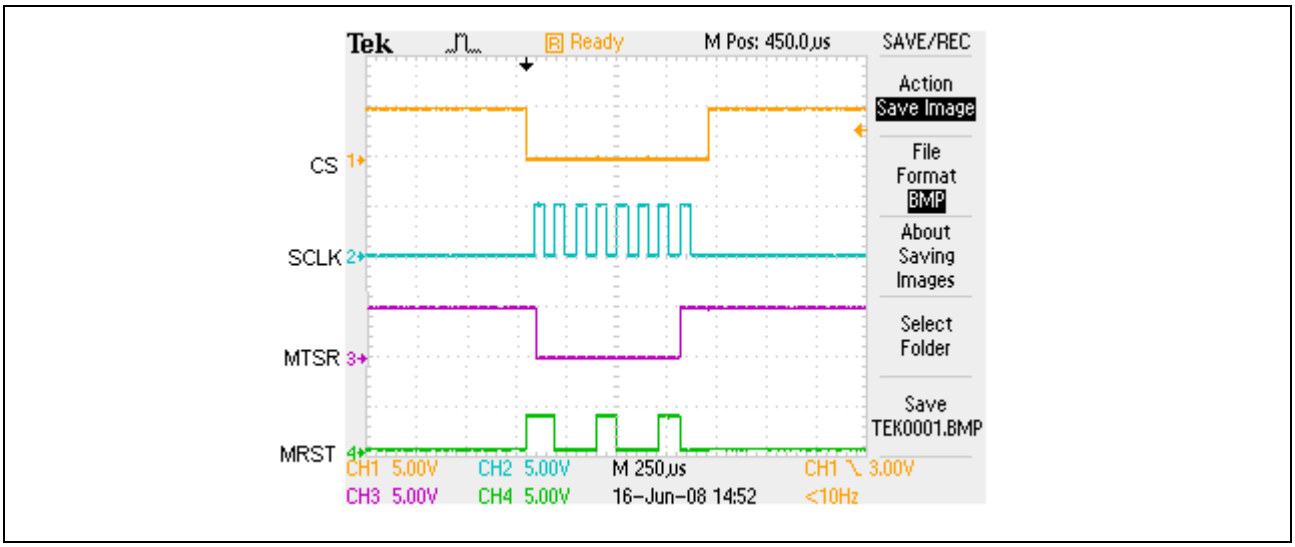


Figure 13 Diagnostic data during Open Load Error

9 Software Package Descriptions

There are two software packages which are provided along with this Application note.

XC22xxSpi_TLE6209R_TaskingClassicSoftware :

- This package is been developed to use Tasking Classic Tool Suit
- The package has been verified with compiler version 8.6r2.
- cstart.asm file used for startup code
- Made COMPILER_TYPE as TASKING_CLASSIC in Hwal.h file to use package with Classic Tool chain
- classic_compiler_option.opt is the option file used for Tasking Classic EDE compiler settings

XC22xxSpi_TLE6209R_TaskingViperSoftware :

- This package is been developed to use Tasking Viper Tool Suit
- The package has been verified with compiler version 2.2r5
- cstart.c and cstart.h file used for startup code
- Made COMPILER_TYPE as TASKING_VIPER in Hwal.h file to use package with Viper Tool chain

The content and descriptions of source files are mentioned in the below table.

Table 10 Usage of Source Files

File Name	File Contents
DemoApp_Main.c	This file contains the main demo application [main() function definition] This file does the following tasks 1. Calling API's for (a) System Clock Initialization (b) Port Initialization (c) Interrupt Priority Initialization (d) Initialization for Hyper Terminal Print 2. Displays Main Menu and asks for Spi communication demo application entry
DemoApp_Main.h	Header file for DemoApp_Main.c.
DemoApp_Spi.c	This file contains demo code for Spi Diagnostic handler. This file does the following tasks 1. Calls API for Spi driver initialization 2. Displays menu for the user to set control word for TLE6209R and also provides information on Diagnostic data received from TLE6209R
DemoAppSpi.h	Header file for DemoApp_Spi.c.
Hwal.h	This file exports Hardware Abstraction or Physical Layer common (a) Global compiler dependent macros (b) Type Definitions (c) Definitions for various Interrupt or Trap Vectors (d) Function definition of Sys_Protection() : Useful while accessing protected registers of micro controller. (e) This file contains COMPILER_TYPE definition. Whenever user would like to change the compiler, then it is recommended to change the definition here. Example : Whenever user would like to use Tasking Classic Tool chain, then define COMPILER_TYPE as #define COMPILER_TYPE (TASKING_CLASSIC) In case user would like use the software with Tasking Viper tool set, then define the COMPILER_TYPE as #define COMPILER_TYPE (TASKING_VIPER)

	Note: The software suit is having definitions for Tasking Classic as well as Tasking Viper. In case user would like to use the suit with other compiler, it is recommended user to define the compiler dependent macros separately.
Hwal_Irq.c	This file contains the code to initialize the used Interrupt priorities
Hwal_Irq.h	Header file for Hwal_Irq.h file. This file contains definitions of various used Interrupt Priorities. Whenever user would like to change the priority of USIC interrupt requests, this file can be used
Mcureg,h	This file can be used whenever user would like to include own developed sfr file.
Port.c	This file contains the function definition for Port Pins initialization
Port.h	Header file for Port.c. Exports definitions to configure Port pins using file Port_Cfg.h file
Port_Cfg.h	User configuration file for Port driver: This file provides user to configure various port pins. User can change this file according his/her need to initialize the port pins
Pwm_Ccu6.c	This file contains the PWM Signal Generation code using CCU6 HW
Pwm_Ccu6.h	Header file for Pwm_Ccu6.c
RegXC22xx_Usic.h	This file exports SFR file for USIC register definitions. Struct based register definitions can be found in the file
SCS_Poweron.c	This file contains System Clock Generation code
SCS_Poweron.h	Header file for Scs_Poweron.c.
SCS_Config_Poweron.h	User Configuration file for SCS driver
Spi.c	SPI driver Function implementation
Spi.h	Header file for Spi.c. This file contains definitions for the user to configure the required value for the Spi driver.
Spi_Cfg.h and Spi_Lcfg.c	Configuration files for SPI driver.
Std_Types.h	Contains various data type definitions
Test_Print.c	Contains various API's to communicate with Hyper Terminal.
Test_Prin.h	Header file for Test_Print.c.
Usic_Irq.c	This file contains the USIC Interrupt Service Routines
start.asm or cstart.c and csart.h	These files contain compiler dependent start up code. For Tasking Classic Tools Chain : start.asm is used For Tasking Viper Tool Chain : cstart.c and cstart.h files are used

10 Appendix

10.1 USIC Peripheral Module Overview

XC22xx Microcontroller family provides various number of USIC peripheral modules viz USIC0, USIC1, USIC2, USIC3 etc. Each USIC module provides 2 independent communication channels and each channel can be configured as one of the supported protocols like ASC, SSC, IIS, IIC. The user can program each communication channel with the desired protocols during run-time. The protocol can also be changed during run-time without a reset.

Example:

1. XC2287 supports USIC0, USIC1, USIC2 modules

2. XC2287M supports USIC0, USIC1, USIC2, USIC3 modules

The main advantage of using USIC module includes

- Higher flexibility through configuration with same look-and-feel for data management
- Reduced complexity for low-level drivers serving different protocols
- Wide range of protocols, but improved performances (baud rate, buffer handling)

The Universal Serial Interface Channel (USIC) module is based on a generic data shift (DSU) and data storage structure (DBU) which is identical for all supported serial communication protocols. Each channel supports complete full-duplex operation with a basic data buffer structure (one transmit buffer and two receive buffer stages). In addition, the data handling software can use FIFOs.

The protocol part (generation of shift clock/data/control signals) is independent from the general part and is handled by protocol-specific preprocessors (PPPs).

The USIC's input/output lines are connected to pins by a pin routing unit, so the inputs and outputs of each USIC channel can be assigned to different interface pins providing great flexibility to the application software. All assignments can be done during runtime.

The general structure of USIC module is as shown in the below figure

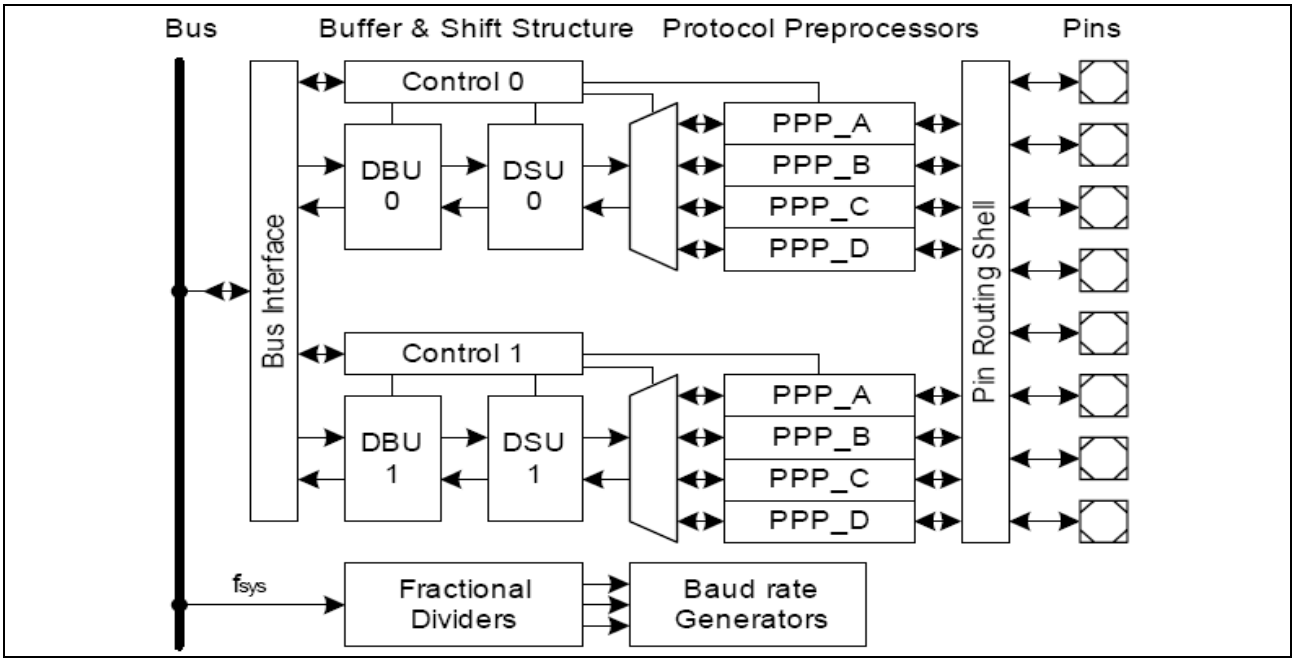


Figure 14 General structure of USIC Hardware Module

10.2 Programming USIC Module for SPI Communication

The below steps are followed to configure USIC Channel as SPI master.

- Activate USIC Channel.
- SSC / SPI protocol selection for the activated USIC Channel.
- Connecting Input and Output lines.
- Generating the required Baud Rate.
- Configuring Shift Clock for Data Transmission/Reception.
- Chip Select Configuration.
- Programming the control registers.

10.2.1 Enabling USIC Channel

To enable the USIC channel, MODEN has to be set. While writing 1 to MODEN, BPMODEN should also be set. So code line looks like. UxCy_KSCFG = 3U. Where x is USIC module number and y is channel number

KSCFG
Kernel State Configuration Register (0C$_H$) **Reset Value: 0000$_H$**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BP COM	0	COMCFG		BP SUM	0	SUMCFG		BP NOM	0	NOMCFG		ERR	ACK	BP MOD EN	MOD EN
w	r	rw		w	r	rw		w	r	rw		rh	rh	w	rw

Field	Bits	Type	Description
MODEN	0	rw	**Module Enable** This bit enables the module kernel clock and the module functionality. 0 The module is switched off immediately (without respecting a stop condition). It does not react on mode control actions and the module clock is switched off. The module does not react on read accesses and ignores write accesses (except to KSCFG). 1 The module is switched on.
BPMODEN	1	w	**Bit Protection for MODEN** This bit enables the write access to the bit MODEN. It always reads 0. It is only active during the write access cycle. 0 The bit MODEN is not changed. 1 The bit MODEN is updated with the written value.

Figure 15 Enabling USIC Channel – Register Usage

10.2.2 Protocol Selection

To select the SSC/SPI protocol CCR.MODE register field has to be programmed with the value one.

CCR
Channel Control Register **(10_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AI EN	RI EN	TBI EN	TSI EN	DLI EN	RSI EN	PM		0				MODE			
rw	rw	rw	rw	rw	rw	rw		r				rw			

Field	Bits	Type	Description
MODE	[3:0]	rw	**Operating Mode** This bit field defines which protocol is selected for this USIC channel. When switching between two protocols, the value 0000_B has to be programmed before selecting a new protocol. In this case, registers PCR and PSR have to be cleared or updated by SW. 0000 The USIC channel is disabled. All protocol-related state machines are set to an idle state. 0001 The SSC (SPI) protocol is selected. 0010 The ASC (SCI) protocol is selected. 0011 The IIS protocol is selected. 0100 The IIC protocol is selected. else reserved

Figure 16 Protocol Selection – Register Usage

10.2.3 Connecting input and output lines

The I/O associated with the USIC-SSC peripheral must be configured to route the signals through the appropriate Microcontroller pins. The master receive slave transmit (MRST) pin must be configured as an input. The chip select (CS), SPI clock (SCLK), and master transmit slave receive (MTSR) pins must all be configured as outputs.

Input Structure: In order to provide a wide range of flexibility in signal routing, each input signal can be selected from a vector of 7 input signals (the signal input H is a permanent internal 1). The different elements of an input vector are numbered A, B, C, D, E, F, and G. In turn the data input vector is connected to port pin for reception of data.

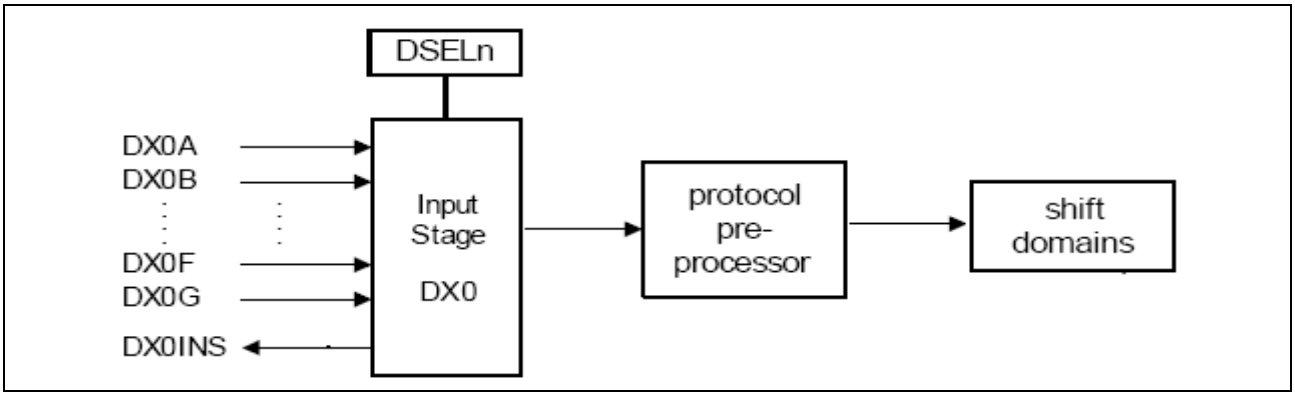


Figure 17 Input Structure

Input Control Register: The input control registers contain the bit fields to define the characteristics of the input stages.

DX0CR
Input Control Register 0 **(20_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DXS	0			CM		SF SEL	D POL	SF EN	DS EN	DF EN	IN SW	0	DSEL		
rh	r			rw		rw	rw	rw	rw	rw	rw	r	rw		

Field	Bits	Type	Description
DSEL	[2:0]	rw	**Data Selection for Input Signal** This bit field defines the input data source for the corresponding input line for protocol pre-processor. The selection can be made from one line of the input vector [G:A]. 000 The data input A is selected. 001 The data input B is selected. ... ... 110 The data input G is selected. 111 The data input is always 1.
INSW	4	rw	**Input Switch** This bit switches between the data input path DXn, and the enables the synchronous outputs PPOn of the protocol pre-processors. 0 The input of the shift registers are connected to the corresponding PPOn output. This setting is used for protocol pre-processors treating data, such as the ASC protocol. 1 The input is connected to the selected data input line (DXn). This setting is used if the signals are directly coming from an input without treatment of the protocol pre-processor, such as for SSC.

Figure 18 Input line selection – Register usage

10.2.4 Baud Rate Generation

The Baud rate generator provides the frequencies needed for the different protocols. Each channel has independent baud rate generator. To generate the required baud rate following formulae can be used

$$\text{if PPPEN} = 0 \begin{cases} f_{SCLK} = f_{SYS} \times \frac{1}{1024 - STEP} \times \frac{1}{PDIV + 1} \times \frac{1}{2} & \text{if Normal Divider} \\ f_{SCLK} = f_{SYS} \times \frac{STEP}{1024} \times \frac{1}{PDIV + 1} \times \frac{1}{2} & \text{if Fractional Divider} \end{cases}$$

$$\text{if PPPEN} = 1 \begin{cases} f_{SCLK} = f_{SYS} \times \frac{1}{1024 - STEP} \times \frac{1}{PDIV + 1} \times \frac{1}{4} & \text{if Normal Divider} \\ f_{SCLK} = f_{SYS} \times \frac{STEP}{1024} \times \frac{1}{PDIV + 1} \times \frac{1}{4} & \text{if Fractional Divider} \end{cases}$$

Where
Normal Divider or Fractional Divider mode can be selected using FDRL.DM register field
STEP is equal to programmed FDRL.STEP register field value
PDIV is equal to programmed BRGH.PDIV register field value
PPPEN is equal to programmed BRGL.PPPEN register field value
fSSC or fSCLK is the generated Shift Clock or Baud rate
fSYS is System Clock Frequency

Figure 19 Formulae for baud rate generation (Shift clock frequency selection)

10.2.5 Configuring the Shift Clock for data transmission or reception

The Shift Clock configuration allows the SPI clock signal to be adjusted to meet the requirements of the device connected to the microcontroller. Due to the multitude of different SSC applications, in master mode, there are different ways to configure the shift clock output signal SCLKOUT with respect to SCLK. This is done in the block SCLKCFG (shift clock configuration) by bit field BRGH.SCLKCFG, allowing 4 possible settings, as shown in figure below.

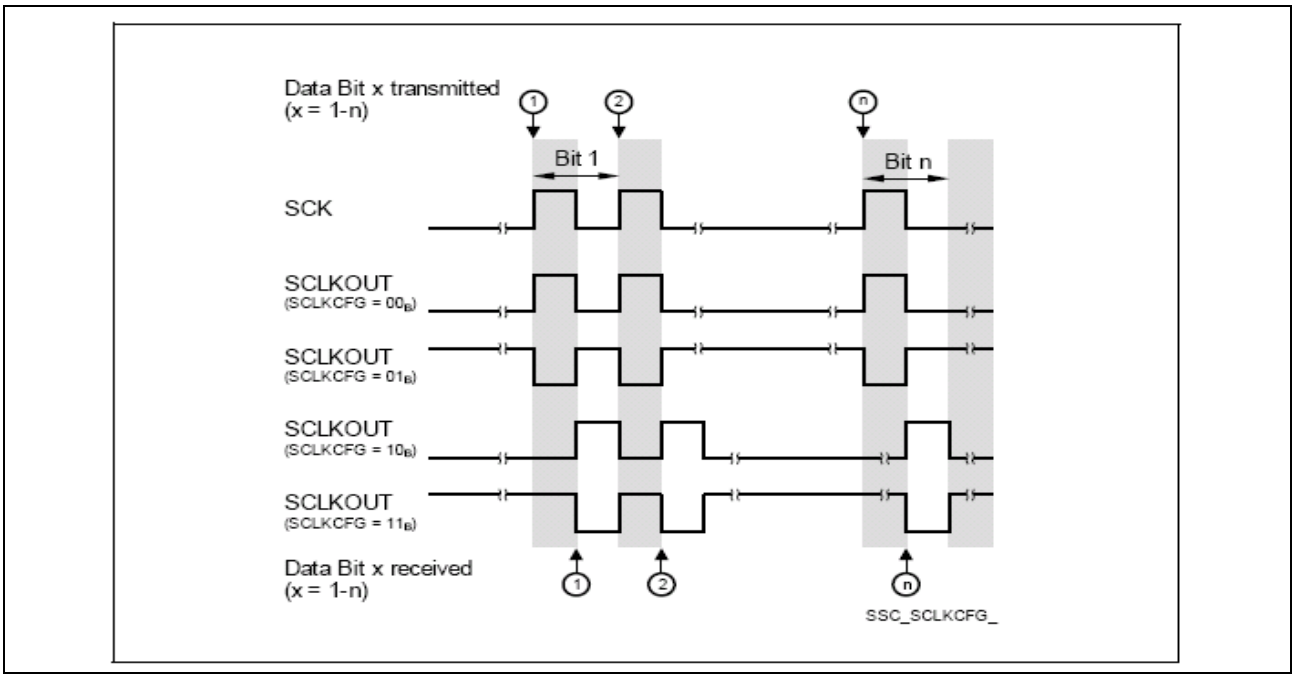


Figure 20 Shift Clock Configuration

10.2.6 Chip Select Configuration

The slave select (chip select) signals indicate the start and the end of a data frame and are also used by the communication master to individually select the desired slave device. The polarity of chip select signal is configured depending upon the slave device to which it has to communicate.

In master mode, a master slave select signal MSLS is generated by USIC using the internal slave select generator. In order to address different external slave devices independently, the internal MSLS signal is made available externally via up to 8 SELOx output signals that can be configured by the block SELCFG (select configuration).

SLSx or SELOx signal become active a certain time before the communication starts (leading delay Tld) and become inactive again a certain time after the transfer of the last bit (trailing delay Ttd). If data frames are transferred back-to back one after the other, the minimum time between the deactivation of the slave select and the next activation of a slave select is programmable (next-frame delay Tnf). If a data frame consists of more than one data word, an optional delay between the data words can also be programmed (inter-word delay Tiw).

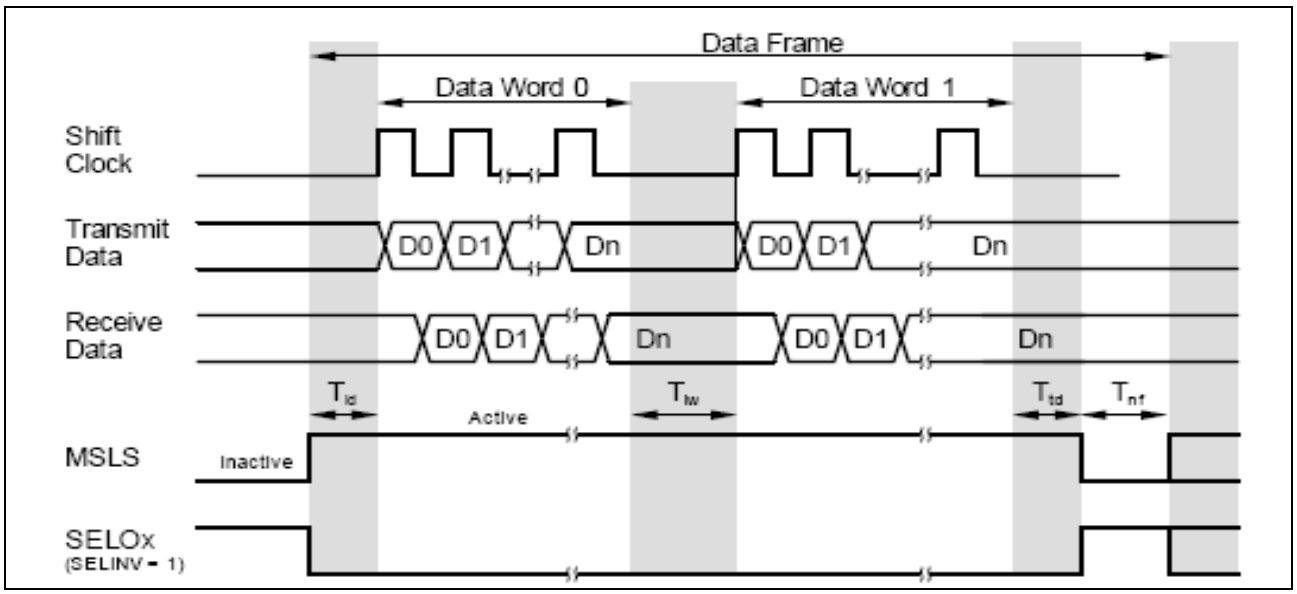
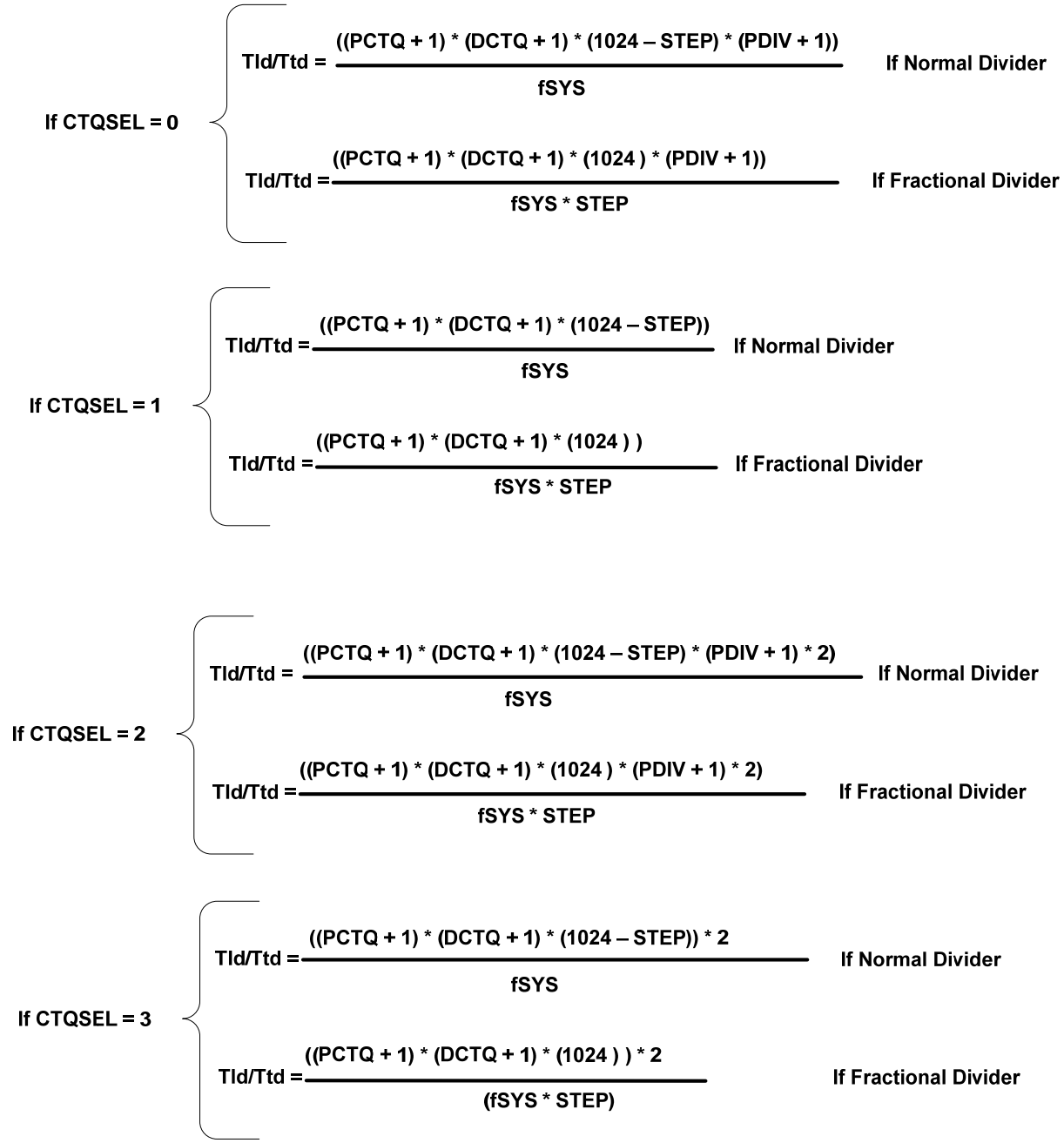


Figure 21 USIC - SPI Bus information

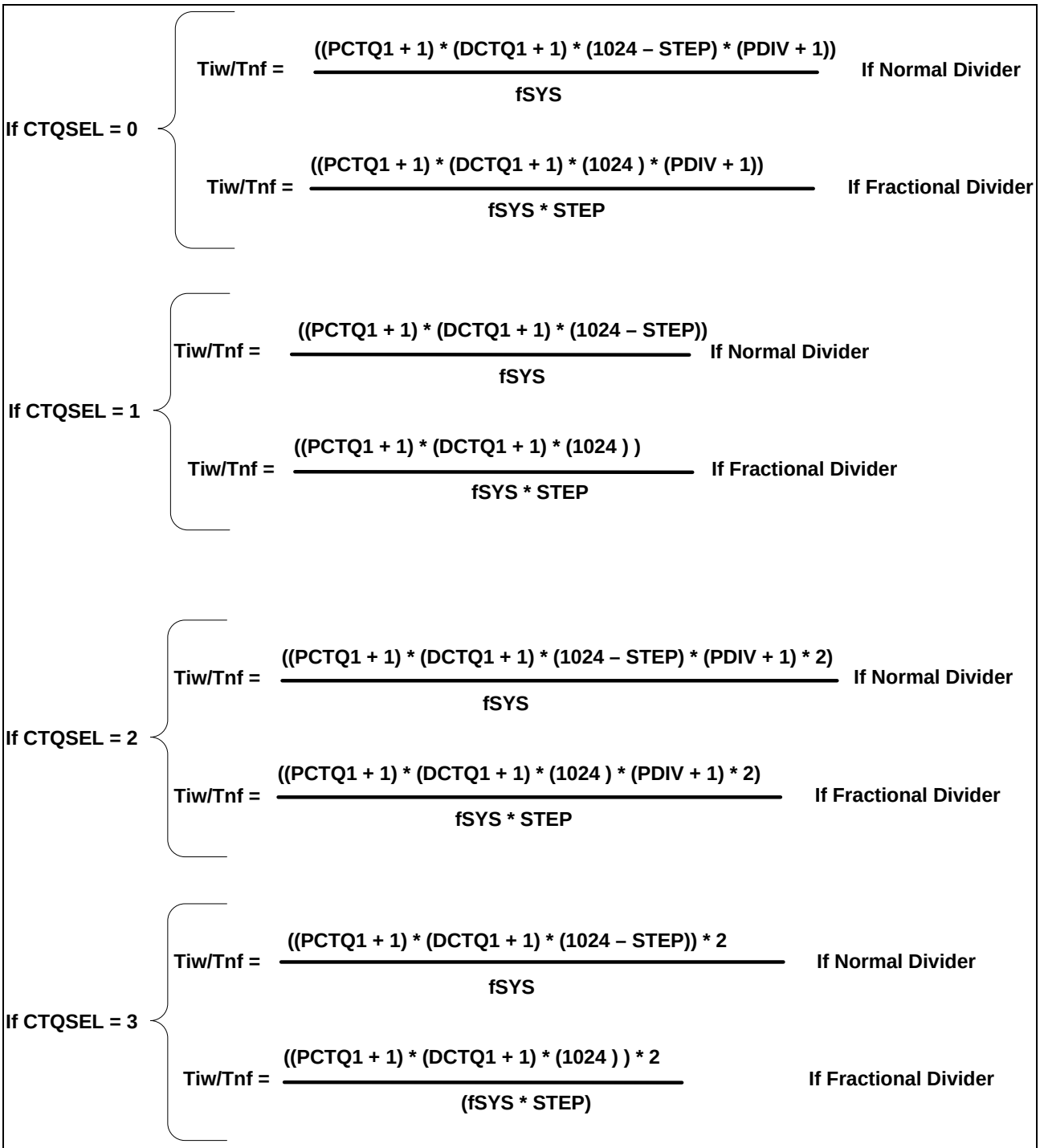
10.2.7 Timing Delays

The various formulae which can be used to generate timing delays are as given below



Where
Normal Divider or Fractional Divider mode can be selected using FDRL.DM register field
STEP is equal to programmed FDRL.STEP register field value
CTQSEL is equal to programmed BRGL.CTQSEL value
PCTQ is equal to programmed BRGL.PCTQ register field value
DCTQ is equal to programmed BRGL.DCTQ register field value
fSYS is System Clock Frequency
Tld is Leading Delay
Ttd is Trailing Delay

Figure 22 Leading and Trailing delays (tdelay)



Where
Normal Divider or Fractional Divider mode can be selected using FDRL.DM register field
STEP is equal to programmed FDRL.STEP register field value
CTQSEL1 is equal to programmed PCRL.CTQSEL1 value
PCTQ1 is equal to programmed PCRL.PCTQ1 register field value
DCTQ1 is equal to programmed PCRL.DCTQ1 register field value
fSYS is System Clock Frequency
Tiw is Leading Delay
Tnf is Next Frame Delay

Figure 23 InterWord and NextFrameDelay (tdelay)

10.2.8 Interrupt Structure

The notification to the user about events occurring during data traffic and data handling is based on:

- Data transfer events related to the transmission or reception of a data word, independent of the selected protocol.
- Protocol-specific events depending on the selected protocol.
- Data buffer events related to data handling by the optional FIFO data buffers.

An interrupt can be generated if the corresponding enable bit is set. The interrupt node pointer bit field allows defining which output line becomes active at which event (grouping of event to a single signal possible).

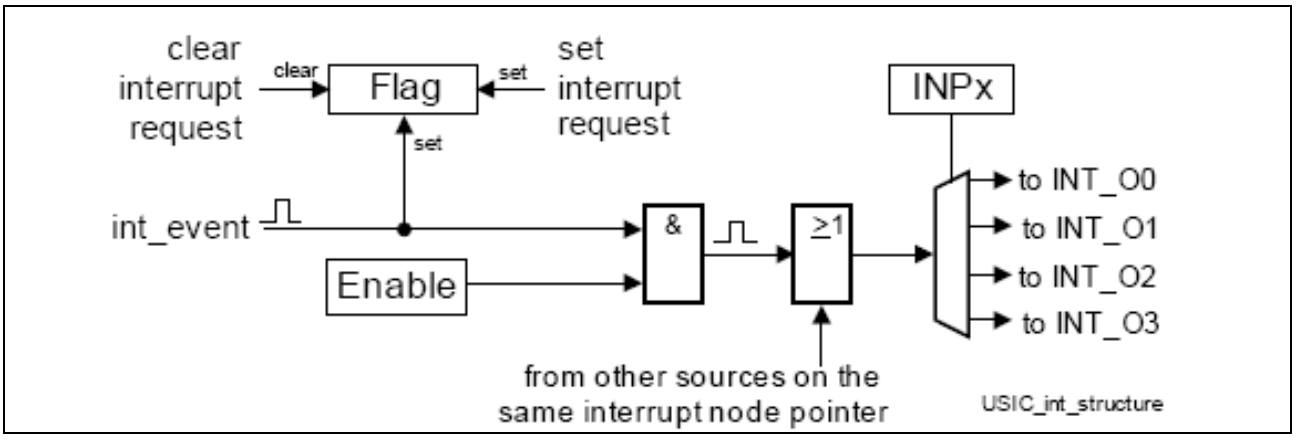


Figure 24 Interrupt Structure

Only receive interrupt which belongs to the category of Data Transfer interrupt is used in this application.

The Receive interrupt can be activated each time a received word becomes available in the receive buffer.

10.2.9 Programming Control Registers

The details of control register PCR (PCRL and PCRH) used to program for SPI communication is mentioned below

PCRL
Protocol Control Register L **(40_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR 15	CTR 14	CTR 13	CTR 12	CTR 11	CTR 10	CTR 9	CTR 8	CTR 7	CTR 6	CTR 5	CTR 4	CTR 3	CTR 2	CTR 1	CTR 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

PCRH
Protocol Control Register H **(42_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR 31	CTR 30	CTR 29	CTR 28	CTR 27	CTR 26	CTR 25	CTR 24	CTR 23	CTR 22	CTR 21	CTR 20	CTR 19	CTR 18	CTR 17	CTR 16
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Bit	Alias	Description
CTR0	MSLSEN	MSLS Enable
CTR1	SELCTR	Select Control
CTR2	SELINV	Select Inversion
CTR3	FEM	Frame End Mode
CTR [5:4]	CTQ SEL1	Input Selection for CTQ
CTR [7:6]	PCTQ1	Divider Factor to Generate f_{PDIV}
CTR [12:8]	DCTQ1	Divider Factor to Generate $f_{TCTQ\,/\,RCTQ}$
CTR14	MSLS IEN	MSLS Interrupt Enable
CTR15	DX2C IEN	DX2C Interrupt Enable
CTR [23:16]	SELO [7:0]	Select Output
CTR24	TIWEN	Enable Inter-Word Delay T_{IW}
CTR31	MCLK	Start Stop of MCLK
other CTRx		Reserved

Figure 25 Protocol Control Register

10.3 Code Generator Tool (Spi_ConfigTool.xls)

The below given steps needs to be followed to generate the Spi_Lcfg.c and Spi_Cfg.h files using Spi_ConfigTool

1. Open Spi_ConfigTool.xls file with Macros Enabled

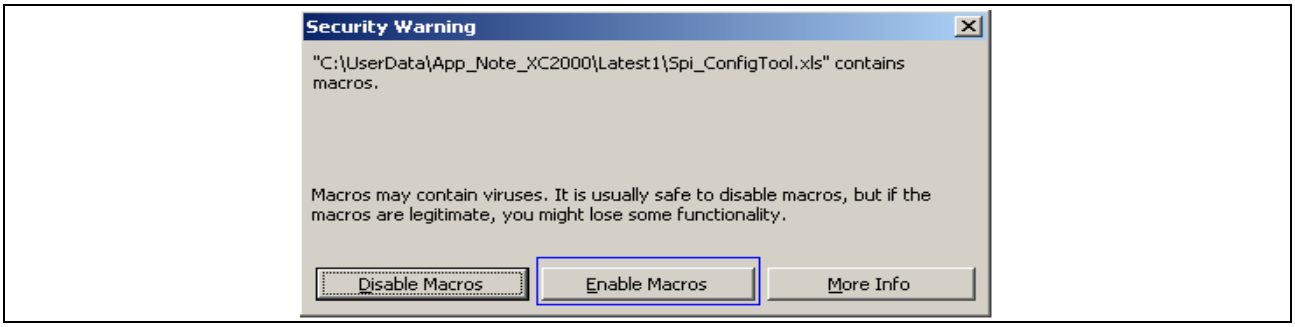


Figure 26 Config Tool Menu : Enable Macros before opening Excel

2. After clicking Enable Macros: the following menu will be displayed.

   Here user has to select the required number of Spi channels to be configured.

   Example: Assume that user would like to use two SPI channels. So select 2.

select: 1st option to configure the 1st channel or

select: 2nd option to close the window [if user feels, don't want to do anything]

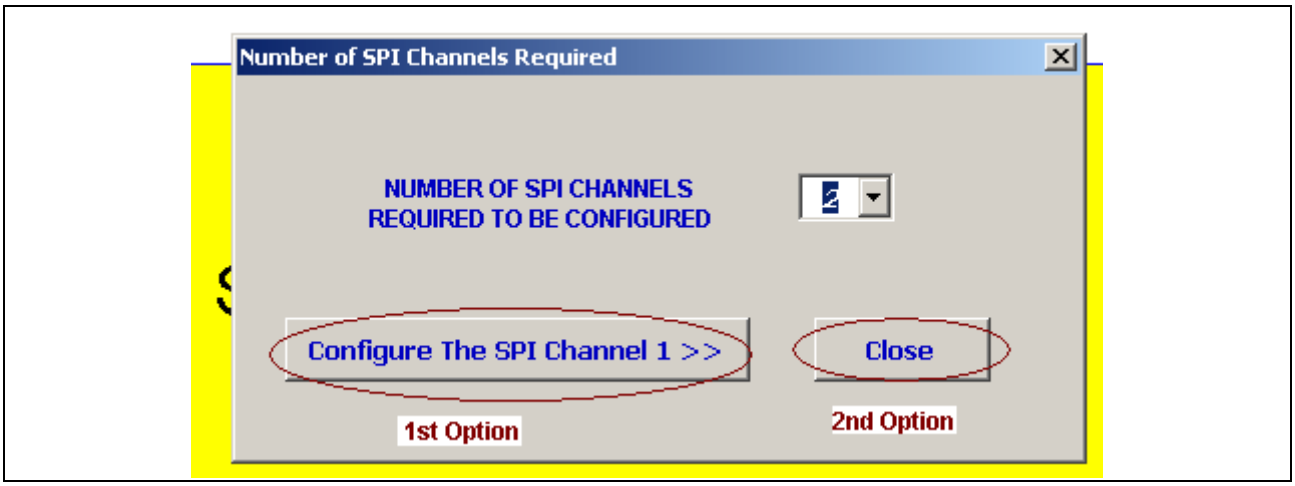


Figure 27 Config Tool Menu : Number of SPI Channels selection

3. After clicking 1st option, the following menu (with default configuration values) will be displayed. In this menu, user can change the required configuration.

   To configure next SPI channels (in this case 2nd channel), user can use red colored button.

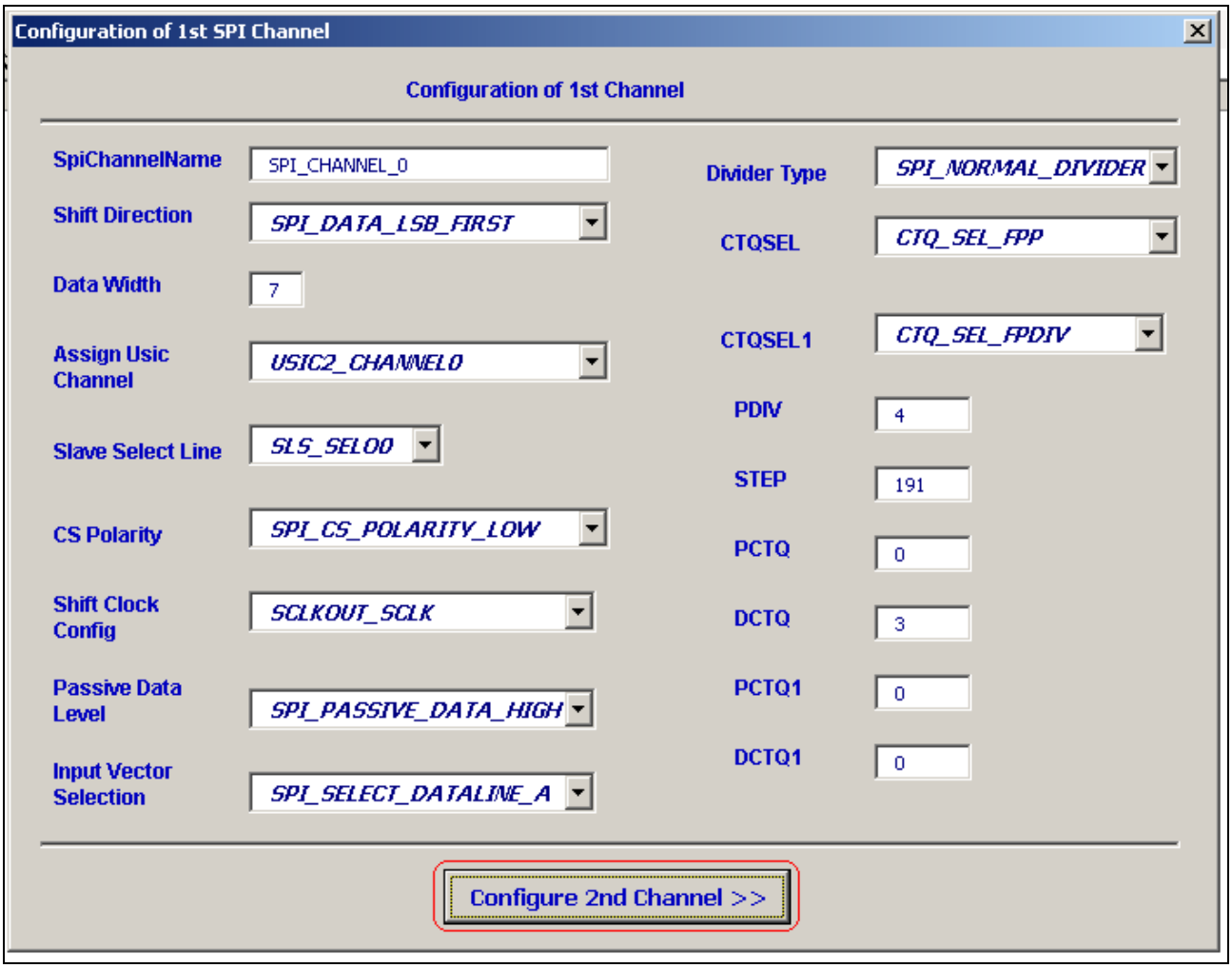


Figure 28 Config Tool Menu : Configure for next channel

4. After clicking configure next (2nd) channel button the next channel configuration menu will be displayed as shown below

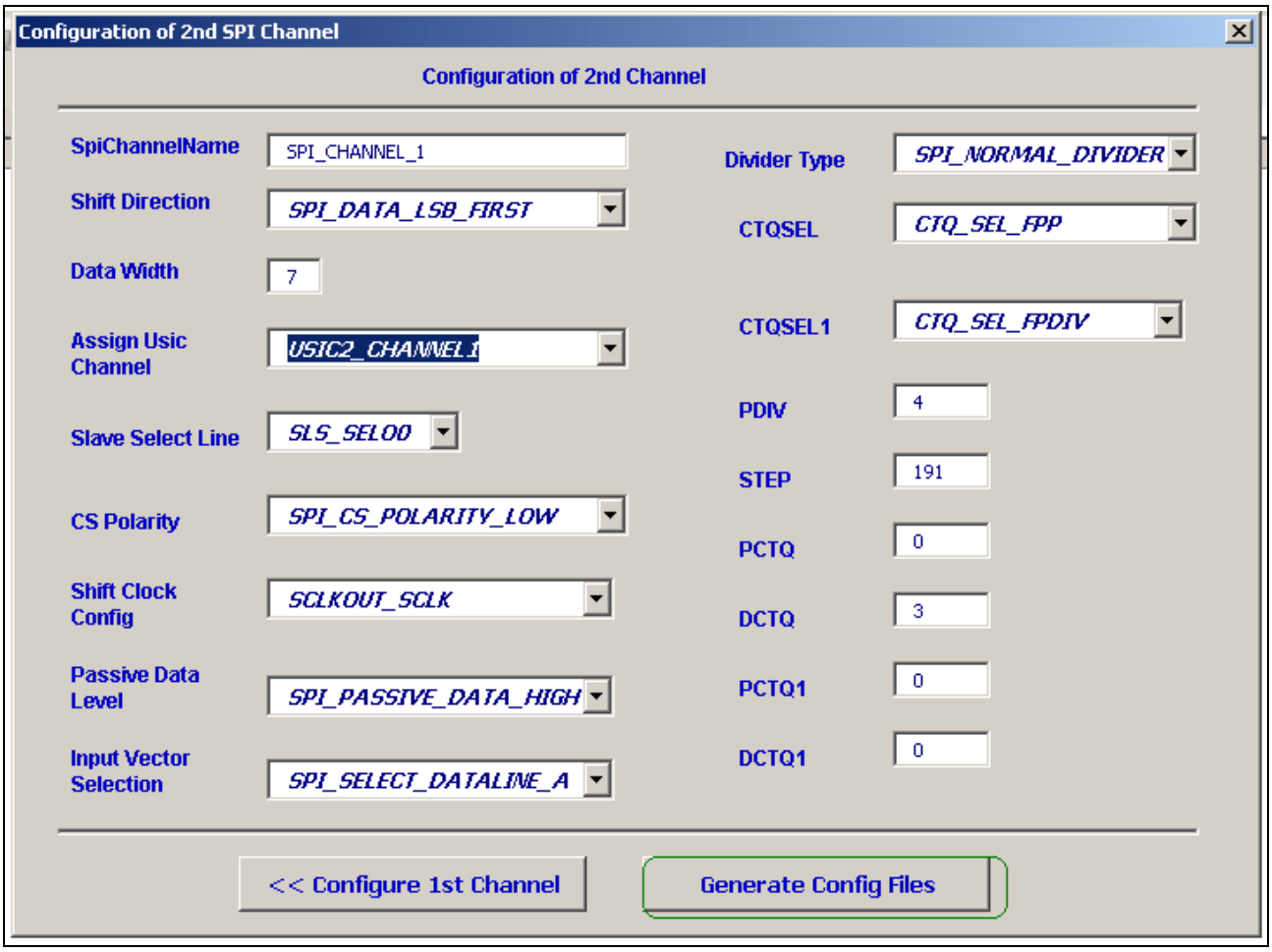


Figure 29 Config Tool Menu : Generate Config Files

In this way, user can configure all the channels (note: the number of channels was been decided during step-2).

After configuring all the channels, in the nth or Maximum channel configuration menu [In this case configuration of 2nd channel], user can see “Generate Config Files” button.

5. After clicking “Generate Config Files” button the following menu will be displayed with the default path for the generation of configuration files being the location of Spi_ConfigTool.xls file.

In this menu, user can change the location in which the configured Spi_Lcfg.c and Spi_Cfg.h files needs to be generated.

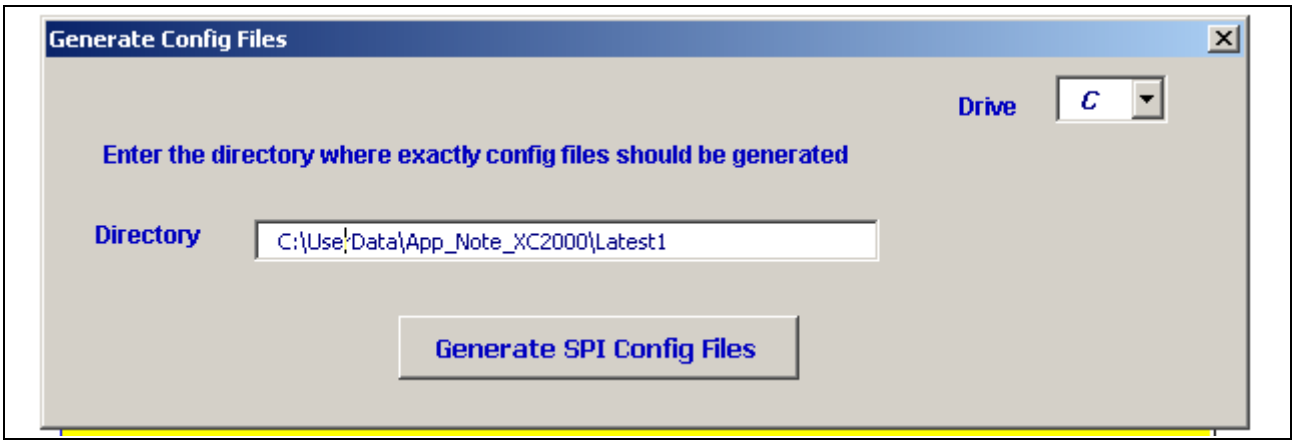


Figure 30 Config Tool Menu : Selecting directory for config files generation

6. After clicking “Generate SPI Config Files” button, the below menu will be displayed.

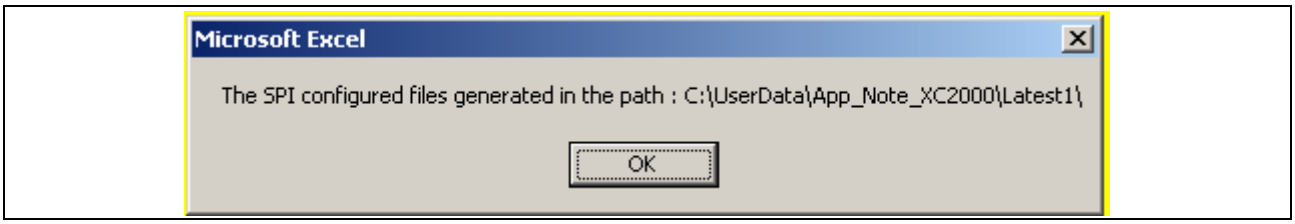


Figure 31 Config Tool Menu : After generating config files

After this user can see the generated Spi_Lcfg.c and Spi_Cfg.h files in the path mentioned.

7. If the user wants to re-generate Spi_Lcfg.c and Spi_Cfg.h files, user can use the below main menu option in Excel sheet.

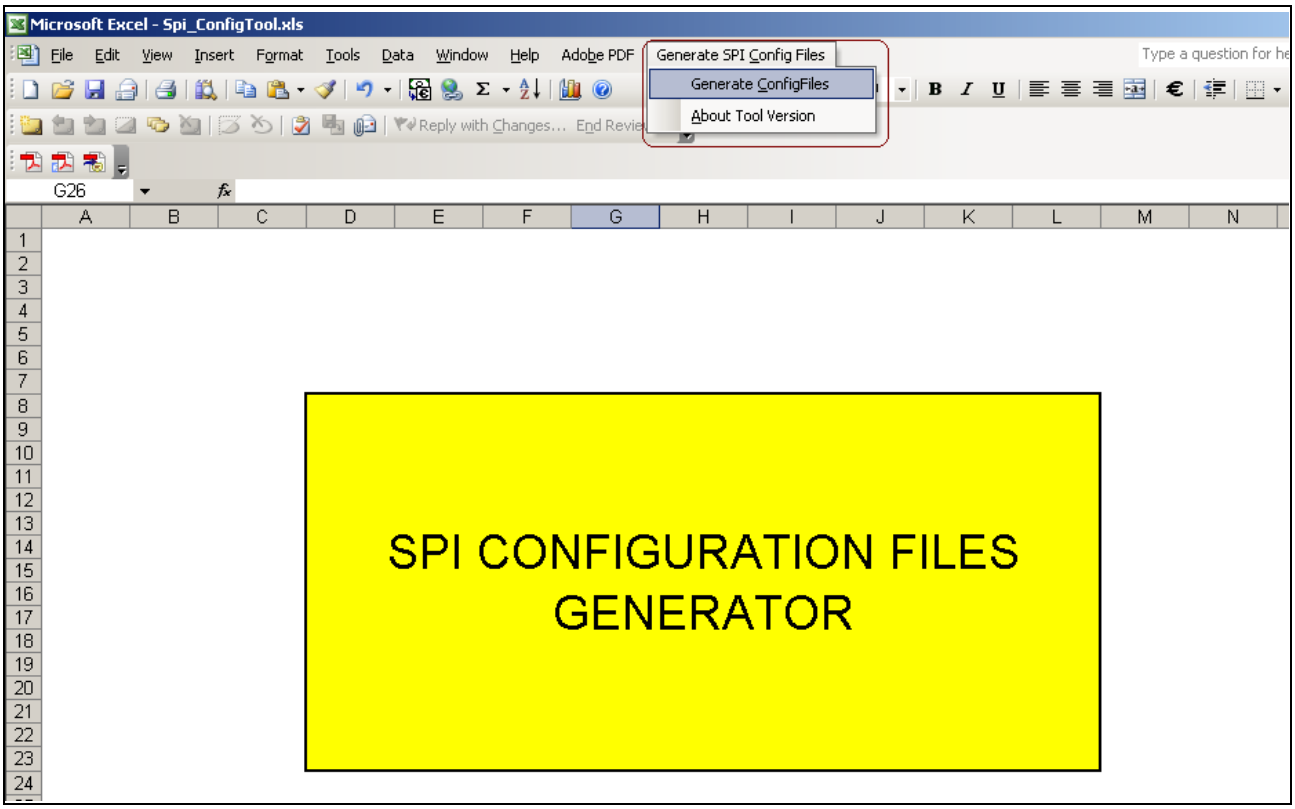


Figure 32 Config Tool Menu : To regenerate confg files

Attention: This tool is tested on Windows-2000 Platform.

http://www.infineon.com

Published by Infineon Technologies AG