Application Note, V 2.0, November 2007

AP16129

XE166/XC2000

USIC Getting Started

Microcontrollers

Never stop thinking

Edition 2008-12

Published by
Infineon Technologies AG
81726 München, Germany

LEGAL DISCLAIMER

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.

XC2000/XE166

Revision History:	**2008-12**	V 2.0
Previous Version: V1.0, 2007-11	-	

Page	Subjects (major changes since last revision)
30	the DX2 stage has to deliver a (permanent) 1-level (by programming bit field DSEL=111b) to the data shift unit
34	PCTQ=9, DCTQ=9 -> PCTQ=1, DCTQ=15
35	PCRL.PCTQ1/DCTQ1=0/0 -> PCRL.PCTQ1/DCTQ1=1/15
37	new section 3.4 and section 3.5

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 Overview

The USIC is a newly designed module in the Infineon Microcontroller XC2000/XE16x family. It is a flexible interface module and covers several serial communication protocols. This application note gives a brief overview on the module structure and some detailed operating information. It will allow the user to quickly start their configuration for different communication protocols using USIC module.

Corresponding to the configuration of desired the communication protocol, SW settings in USIC parts are explained in detail. Example C code is included in the Appendix and can run with an Infineon XC2000/XE16x Starter Kit. For more information about the Starter Kit, see **www.infineon.com**.

2 USIC

2.1 Introduction

XE166 contains 3 USIC modules. Each module has 2 channels and each USIC channel has the same structure and consists of:

- input stages
- output signals
- baud rate generator
- data shift unit
- channel events and interrupt generation unit
- FIFO structure for data transmission and reception.

2.1.1 Input Stages

For each protocol up to 3 input signals are available. Each input stage is handled by an input stage (DX0, DX1 and DX2). It can:

- select the input signal (DXnA...DXnG) via bits field DSEL in register DXxCR
- switch to directly use the input data (DXxCR.INSW=1b) or the output of the protocol pre-processor (DXxCR.INSW=0b)

  if the input signal is used (INSW=1b), then

  – the edge can be defined as trigger signal (via DXxCR.ICM)
  – digital filter can be used (via DXxCR.DFEN)
  – data synchronization can be enabled (via DXxCR.DSEN)
- loop back mode is possible (DXxCR.DSEL=DXnG)
- the input signal can be inverted (via bit DPOL).

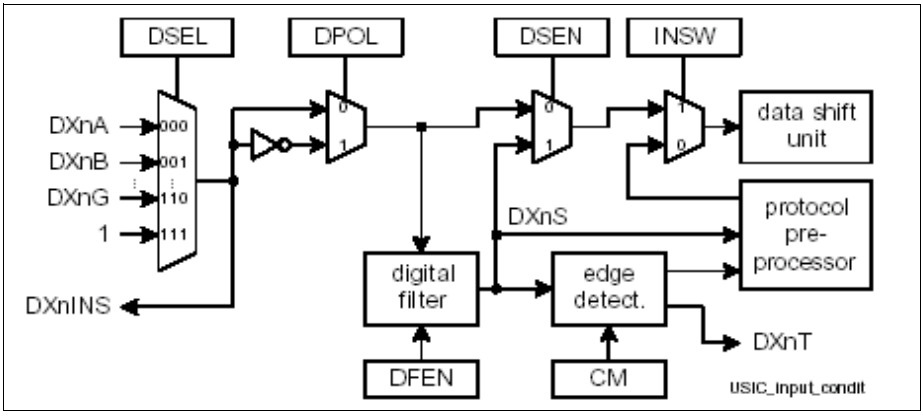


Figure 1 Structure of the input Structure

2.1.2 Output Signals

For each protocol up to 11 output signals are available (DOUT, SCLKOUT, SELO[7..0], MCLKOUT).

- The polarity of the master clock output signal MCLKOUT can be configured via BRGH.MCLKCFG.
- The polarity of the shift clock output signal SCLKOUT can be configured and a delay of one period of f_{PDIV} (half SCLK period) can be created (BRGH.SCLKCFG).

2.1.3 Baud Rate Generator

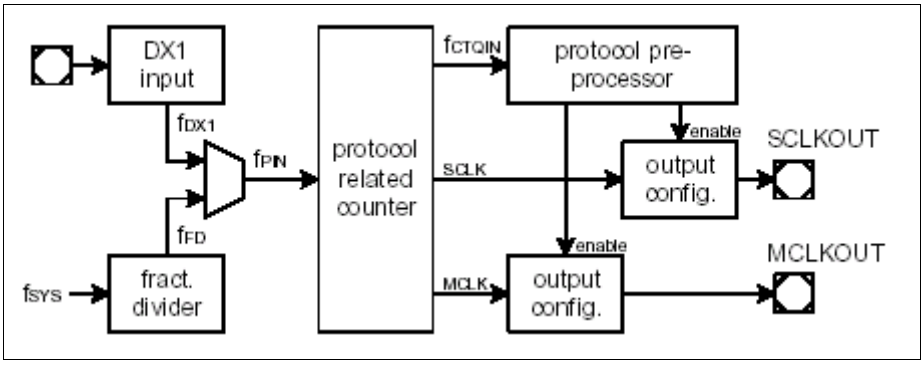


Figure 2 Structure of the Baud Rate Generator

2.1.3.1 Clock Input DX1

If it is used, then it holds $f_{PIN} = f_{DX1}$ for baud rate generation based on an external signal. It is normally used for salve mode (e.g. for the baud rate detection). In this case, an external input signal at the DX1 input stage can be optionally filtered and optionally synchronized with f_{SYS}.

- If BRGL.CLKSEL=10b, the trigger signal DX1T determines f_{DX1}. The rising/falling/ both edges of the input signal can be used for baud rate generation. The active edge is selected by bit field DX1CTR.CM.
- If BRGL.CLKSEL=11b, the rising edges of the input signal can be used for baud rate generation. The external signal is synchronized. The rising edges of DX1S is used for the synchronization.

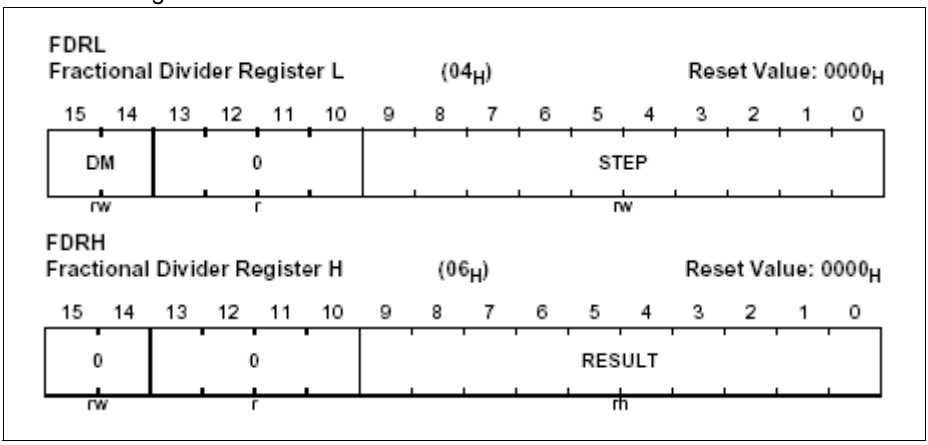
2.1.3.2 Fractional Divider

If it is used, then it holds $f_{PIN}=f_{FD}$ for baud rate generation based on f_{SYS}.

- normal divider mode (FDRL.DM=01b): $f_{FD}=f_{SYS}$ x 1/n, n=1024-STEP.
- fractional divider mode (FDRL.DM=10b): $f_{FD}=f_{SYS}$ x n/1024, n=STEP.

Note: the fractional divider mode allows to program the average output clock with a finer granularity. But f_{FD} can have a max. period jitter of one f_{SYS} period.

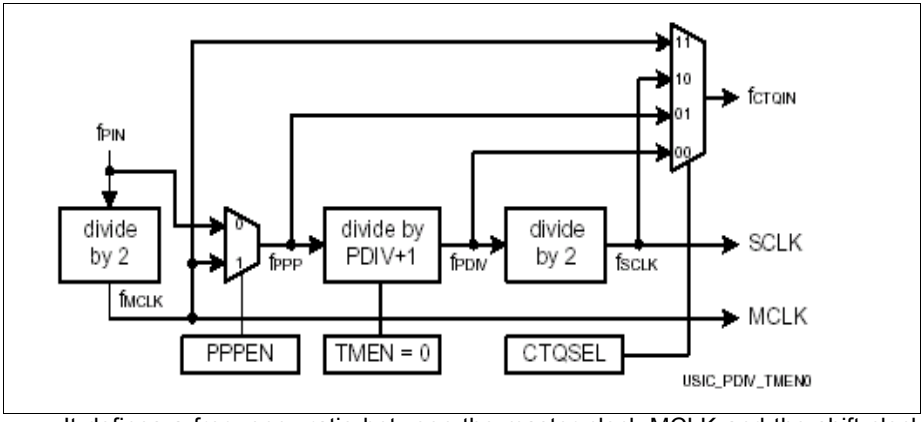
The related registers:



2.1.3.3 Protocol-Related Counter

It provides MCLK, SCLK and other protocol-related signals for baud rate detection or the shift clock.

- **protocol-related counter in divider mode**



– It defines a frequency ratio between the master clock MCLK and the shift clock SLCK.
– It is used for an integer division delivering the p_{PIDV}.
– 2 output signal MCLK and SCLK have 50% duty cycle.

the related registers:

BRGL
Baud Rate Generator Register L (1C$_H$) Reset Value: 0000$_H$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	DCTQ					PCTQ		CTQSEL		0	PPP EN	TM EN	0	CLKSEL	
r	rw					rw		rw		r	rw	rw	r	rw	

BRGH
Baud Rate Generator Register H (1E$_H$) Reset Value: 0000$_H$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCLKCFG		MCLK CFG	0			PDIV									
rw		rw	r			rwh									

SW configuration for baud rate generation based on f_{SYS}:

- f_{FD}=f(f_{SYS}) via bits filed DM and STEP in register FDRL
- f_{PIN}=f_{FD} via bits CLKSEL=00b in register BRGL
- f_{PPP}=f_{PIN} or f_{MCLK} via bit PPPEN in register BRGL
- f_{PDIV}=f(f_{PPP}) via bits field PDIV in register BRGH
- select f_{CTQIN} via bits field CTQSEL in register BRGL

  CTQSEL=00b -> f_{CTRQ}=f_{PDIV}

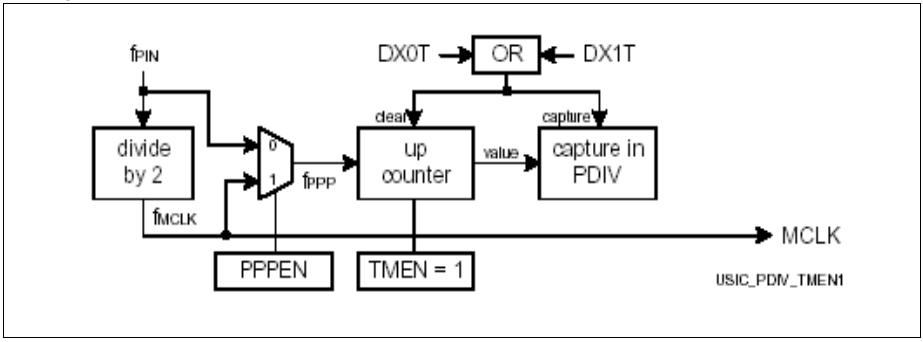
  CTQSEL=01b -> f_{CTRQ}=f_{PPP}

  CTQSEL=10b -> f_{CTRQ}=f_{SCLK}

  CTQSEL=11b -> f_{CTRQ}=f_{MCLK}

- **protocol-related counter in capture mode**

  it is used for time internal measurement (BRFL.TMEN=1). For example, measure the baud rate in the slave mode before starting data transfers (the time between two edges of DX0T and DX1T).



2.1.3.4 Protocol Pre-Processor for the Data Shift Unit

It is used to generate time intervals for protocol-specific purposes. It has a time quanta counter and is used for bit timing control.

– PCTQ: pre-divider for time quanta counter (division of f_{CTQIN} by 1,2,3 or 4)
– DCTQ: denominator for time quanta counter.

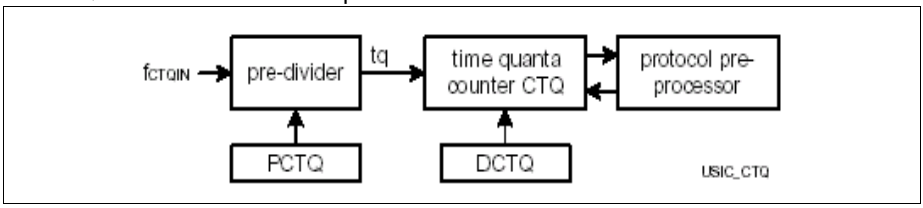


Figure 3 Time Quanta Counter

The PPP supports communication protocols:

UART:

– max. frequency is $f_{SYS}/4$ (20MHz @ f_{SYS}=80MHz)
– baud rate: 1,2kBaud to 3,5 MBaud
– number of data bits: 1 to 63
– PCTQ: the length of a time quantum
– DCTQ: the number of time quanta per **bit time**, a standard setting is DCTQ+1=16, SP=8 or 9 (SP<DCTQ, recommended: DCTQ>=4)

SPI:

– max. frequency is $f_{SYS}/2$ (40MHz @ f_{SYS}=80MHz)
– baud rate: 2kBaud to 10MBaud (theoretic up to 40MBaud)
– number of data bits: 1 to 63
– PCTQ: define the length of a time quantum for delay T_{ld} and T_{td}
– DCTQ: the number of time quanta for the delay generation for T_{ld} and T_{td}

$T_{ld}=T_{td}=(PCTQ+1)x(DCTQ+1)/f_{CTQIN}$

IIC:

– 7bit and 10bit addressing
– PCTQ: the length of a time quantum
– DCTQ: the number of time quanta per **symbol**

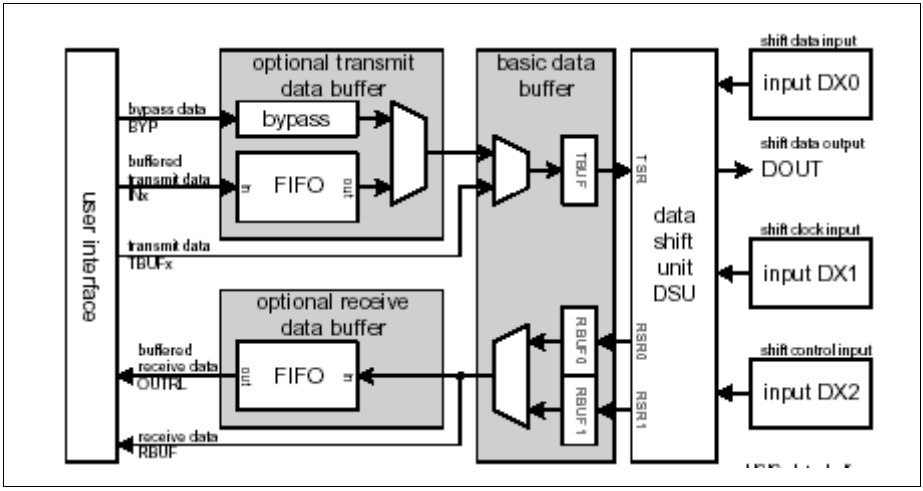
100kBaud (PCRH.STIM=0b): f_{SYS}>=2MHz, 1 symbol timing=10 tq (DCTQ=9)

400kBaud (PCRH.STIM=1b): f_{SYS}>=10MHz, 1 symbol timing =25 tq (DCTQ=24)

IIS:

– module frequency: receiver: max. f_{SYS}, transmitter: max. $f_{SYS}/2$
– baud rate: up to 26 MBaud

2.1.4 Data Shift Unit



The data handling is based on the data shift unit. For data transmit/receive it contains

- transmit buffering (registers TBUFx)/receive buffering (registers RBUF0/1)
- transmit control information (register TCSRL)
  - frame length and word length:

  ASC: FLE=0..62 (63 is not allowed), parity bit can be enabled via bits field CCR.PM

  SSC: FLE=0..63, PM=0

  IIC: for 7-bit addressing: WLE=7, unlimited data flow (SCTRH.FLE=3FFh), PM=0

  - the shift control signal (TRM)

  ASC: TRM= 01b, the shift control signal is active if it is at 1-level

  SSC: TRM=01b, the shift control signal is active if it is at 1-level

  IIC: TRM=11b, active without referring to the actual signal level

  - Data output configuration (DOCFG)

  ASC: DOCFG=00b (DOCFG=01b for IrDA signal, the DOUT value is then inverted)

  SSC: DOCFG=00b, the DOUT value not inverted

  IIC: DOCFG=00b, he DOUT value not inverted

  - passive data level (PDL)

  ASC: PDL=1b, the passive data level=1

  SSC: PDL=1b, the passive data level=1

  IIC: PDL=0b, the passive data level=0

  - shift direction control (SDIR)

ASC: SDIR=0b, the LSB first

SSC: SDIR=1b/0b, the MSB/LSB first

IIC: SDIR=1b, the MSB first

the related registers:

SCTRL
Shift Control Register L (30_H) Reset Value: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						TRM		DOCFG		0				PDL	SDIR
r						rw		rw		r				rw	rw

SCTRH
Shift Control Register H (32_H) Reset Value: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				WLE				0		FLE					
r				rwh				r		rwh					

- transmit data validation information (not in receive operation)

  If TBUF data has the single short mode (TDSSM=1b), the data in TBUF is considered as invalid after it has been loaded into the shift register. TDEN must be sent to 01b to allow send out data from TBUF if TDV=1. Bit TDV is a HW controlled bit. It is automatically set when data is moved to TBUF (by writing to one of the transmit buffers).

  – **ASC** and **IIC**: TCSRL.TVD is cleared in single short mode with the transmit buffer interrupt event (bit TBIF in register PSR)
  – **SSC** and **IIS**: TCSRL.TVD is cleared in single short mode with the receive start interrupt event (bit RSIF in register PSR)

the related register:

TCSRL
Transmit Control/Status Register L ($3C_H$) Reset Value: 0000_H

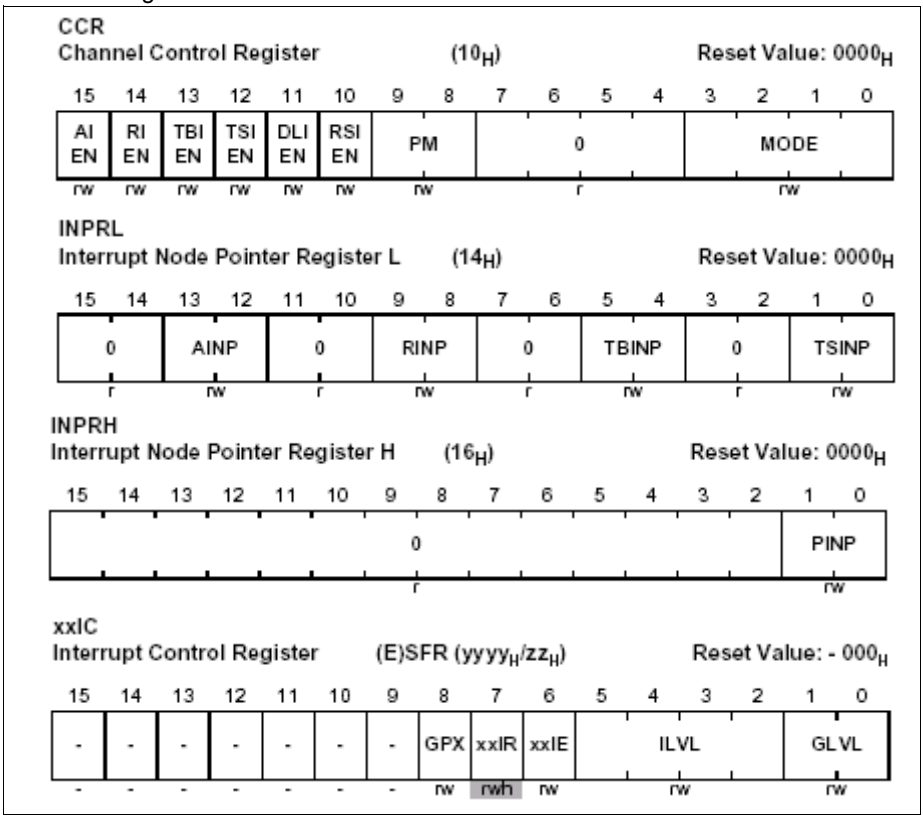
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		WA	TD VTR	TDEN		0	TD SSM	TDV	EOF	SOF	0	WA MD	FLE MD	SEL MD	WLE MD
r		rwh	rw	rw		r	rw	rh	rwh	rwh	r	rw	rw	rw	rw

2.1.5 Channel Events and Interrupt Generation Unit

Each USIC channel provides 4 service request output SRx (x=0,1,2,3). For each channel there are 3 types of interrupt events available

- data transfer events related to the transmission or reception (independent of the selected protocol)
- protocol-specific events (dependent of the selected protocol)
- FIFO data buffer events

the related registers:



– register CCR defines the general interrupt generation
– register INPRL and INPRH defines which SRx will be activated if the corresponding event occurs
– register PCRL/H defines protocol-specific interrupts
– interrupt control register UxCy_zIC (x=0,1,2; y=0,1; z=0,1,2) defines interrupt level and enable/disable the interrupt generation

3 ASC (UART) Mode

3.1 ASC for Full-Duplex Communication

For full-duplex communication, an independent communication line is needed for each transfer direction.

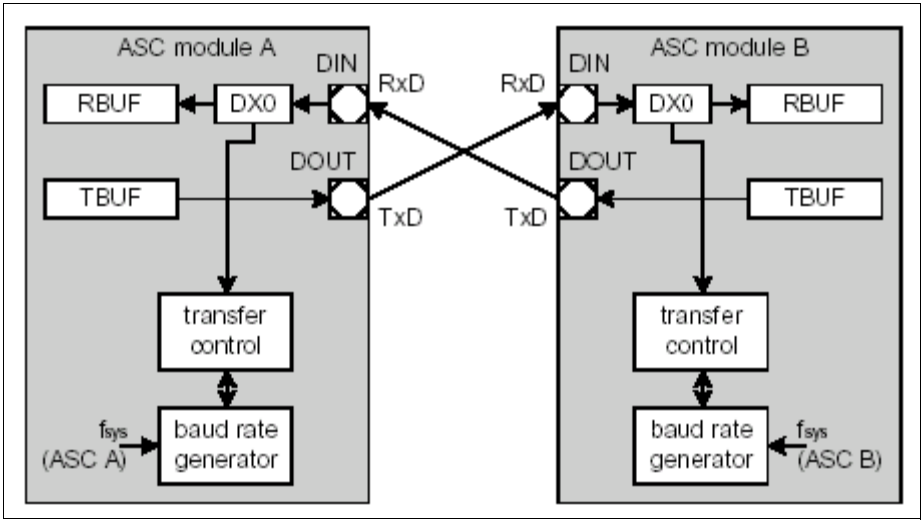


Figure 4 ASC Signal Connection for Full-Duplex Communication

3.1.1 Input Stages

- shift data input (DX0) is used as RxD. the PPP is used for the data shift unit
- shift clock input (DX1) is **optional** for collision detection in half-duplex communication.

3.1.2 Output Signal

- shift data output (DOUT) is used as TxD

3.1.3 Baud Rate Generation

- protocol pre-processor (PPP)

In ASC mode the PPP is responsible for reception and transmission of asynchronous data frames. The protocol specific bits (SOF, parity bit, stop bit) are automatically handled by the ASC protocol state machine and do not appear in the data flow via the receive and transmit buffer.

- baud rate generation:

$$fasc = \frac{fsys}{1024 - STEP} \times \frac{1}{PDIV + 1} \times \frac{1}{PCTQ + 1} \times \frac{1}{DCTQ + 1}$$

under condition: CLKSEL=0, PPPEN=0, CTQSEL=00b, and normal divider is used

3.1.4 ASC Frame

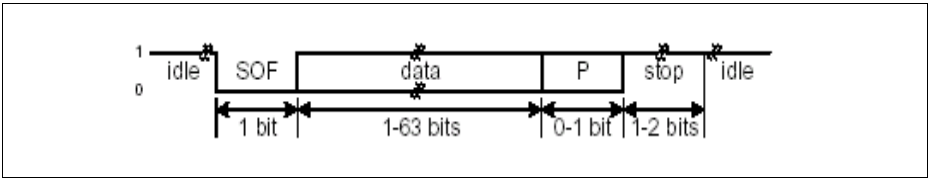


Figure 5 Standard ASC Frame Format

For a detailed description of data shift unit please see **Chapter 2.1.4**

– word and data length: FLE=0..62 (63 is not allowed),
– parity bit can be enabled via bit PM in register CCR
– shift control signal: TRM=01b
– data output configuration (DOCFG=00b)
– passive data level: PDL=1b, the passive data level ’1’
– shift direction control: SDIR=0b (the LSB first)

3.1.5 ASC Protocol-Related Control and Status Information

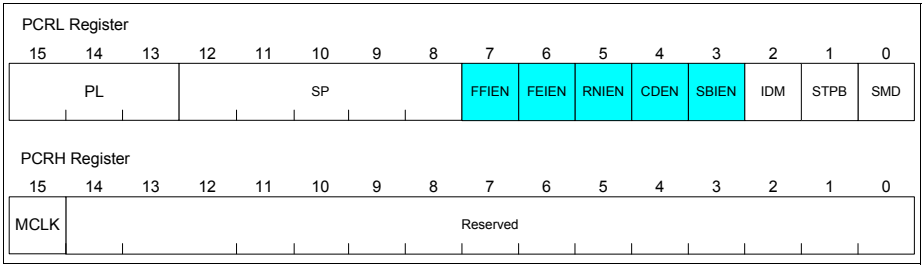


Table 1 Signification of PCR bits for ASC

bit 0	SMD	sample mode
bit 1	STPB	stop mode
bit 2	IDM	idle detection mode
bit 3	SBIEN	sync.-break interrupt enable
bit 4	CDEN	collision detection enable
bit 5	RNIEN	receive noise detection interrupt enable
bit 6	FEIEN	format error interrupt enable
bit 7	FFIEN	frame finished interrupt enable
bit 8-12	SP	sample position
bit 13-15	PL	pulse length
bit 31	MCLK	master clock enable

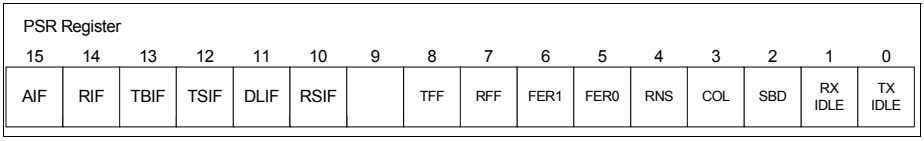


Table 2 Signification of PSR bits for ASC

bit 0	TXIDLE	transmission idle
bit 1	RXIDLE	reception idle
bit 2	SBD	synchronization break detected
bit 3	COL	collision detected
bit 4	RNS	receiver noise detected
bit 5	FER0	format error in stop bit 0
bit 6	FER1	format error in stop bit1
bit 7	RFF	receiver frame finished

bit 8	TFF	transmitter frame finished
bit 10	RSIF	receiver start indication flag
bit 11	DLIF	data lost flag
bit 12	TSIF	transmit shift flag
bit 13	TBIF	tinsmith buffer flag
bit 14	RIF	receive flag
bit 15	AIF	alternative receive flag

- **ASC general interrupt event**
  - RSIF: after the sample point of the first data bit of a data word
  - DLIF: RBUF becomes overwritten
  - TSIF: after the start of the last data bit of a data word
  - TBIF: after the start of the first data bit of a data word
  - RIF and AIF:

  a) if the parity bit is disabled, RIF is set after the SP of the last data bit of a data word
  b) if the parity bit is enabled,

  - RIF is set after the SP of the parity bit and no parity error has been detected
  - AIF is set after the SP of the parity bit and a parity error has been detected
- **ASC protocol-specific interrupt events**
  - SBD: used in LIN networks
  - COL: used in data transfer over a single data line
  - RNS: The 3 input samples are not identical. This is only active during the reception of a frame and using the three sample mode.
  - FER0: sampled stop bit is 0 instead of 1
  - FER1: if the sample process is active and the sampled stop bit is 0 instead of 1
  - RFF: The receiver has left the state of the last stop bit. RxD assignment can be changed.
  - TFF: The transmitter has left the state of the last stop bit. TxD assignment can be changed.

3.1.6 Frame Transmission Control

The data transmit path is based on the transmit buffer unit, the shift control unit (for data transmission and reception) and the transfer trigger logic unit.

- The internal TSR register (only readable) indicates the status information of the TBUF. If a currently transmitted data word is finished and new data is valid for transmission, the register TSR is updated automatically.
- The data transfer parameter (frame/word length, output polarity, shift direction...) is controlled by register TCSRL/H.
- The data word in the TBUF can only be triggered if it is indicated as varied. Bit TCSRL.TDEN defines the transfer gating logic.

When the transmitter is in idle state (PSR.TXIDLE is set), the data in the transmit buffer TBUF is valid (TCSRL.TDV is set) and the baud rat generator is not in timing measurement mode (BRGL.TMEN=0), the frame transmission can start.

There are two ways for TXIDLE/RXIDLE to have value ’1’:

– if the idle detection mode is used (bit PCR.IDM is set), the TXIDLE/RXIDLE will be set,
  – in the case with parity bit (PM=00b), when the bus has a consecutive passive level with length of FLE+2 bit times.
  – in the case without parity bit (PM=1xb), when the bus has a consecutive passive level with length of FLE+3 bit times.

  SW should reset bit TXIDLE/RXIDLE.
– if the idle detection mode is not used (bit PCR.IDM is cleared), bit TXIDLE/RXIDLE is set by HW automatically.

As the transmission starts, data output is switched to protocol pre-processor and the start bit ’0’ is sent on the TxD line.

At the end of the start bit, the data output is switched to the data shift unit. According to the configuration in register TCSRL/H the data in TBUF is shifted out on the TxD. When reaching the end point of the last bit, the PPP will generate the parity bit (optional) and send the stop bit with level ’1’. The level of the idle state is provided by PPP with the passive data level (defined in SCTRL.PDL).

3.1.7 Frame Receive Control

The data receive path is based on the receive buffer unit and the shift control unit

- the internal RSR0/1 registers (only readable) indicate the status information of the RBUF0/1. If a complete data word has been received or the frame is finished, the register RSR0/1 is updated automatically.
- the data receive parameter (frame/word length, output polarity, shift direction...) is controlled by TCSR registers (commonly control for send and receive).

When the USIC channel is in the receive idle state (PSR.RXIDLE is set) and a falling edge is detected on the DX0 input, the frame reception can be started.

3.1.8 SW Configuration for ASC Full-Duplex Communication

Example 1: f_{SYS}=80MHz, baud rate=19200Baud, U2C0

- input stage and output signal
  – input stage (DX0CR)

  DX0(DX0B)=DIN=P3.1(RxD): DPOL=0 (not inverted), INSW=0 (PPP used)

  – output signal

  TxD=DOUT=P3.0
- baud rate generation and the PPP configuration (BRGL/H, FDRL)

  - fractional divider (FDRL): DM=2, STEP=786
  - baud rate generator: PCTQ=1, DCTQ=15, PDIV=99, CTQSEL=0, PPPEN=0

f_{ASC}=80,000 x 786 x (1/1024) x (1/100) x (1/2) x (1/16) = 19,189KBaud (real value)

  - data format configuration (data shift control via SCTRL):

TRM=01b, DOCFG=0 (DOUT not inverted), PDL=1, SDIR=0 (LSB first)

  - word and frame length (SCTRH): WLE=7, FLE=7

- data transmission control (TCSRL): use TBUF single shot mode (TDEN=1, TDSSM=1)
- protocol-related information (PCR)
  - sample mode: one sample point (SDM=0)
  - stop bit: one stop bit (STPB=0)
  - sample position: SP=7
  - pulse length: PL=0 (the 0 level is during the complete bit time with bit '0')
- parity mode (CCR): PM=0, here parity mode is not used.
- interrupt configuration: not used
- input/output pins configuration
  - output P3.0: P3_IOCR00=0x0090 (ALT1, push-pull)

3.2 ASC for Half-Duplex Communication

In a half-duplex configuration, only one data line is shared between the communication partners and used for both reception and transmission of data. In this case, the user SW has to take care that only one transmitter is active at a time. There are two ways to avoid collisions on the data exchange line:

- only the transmitting channel may enable its transmit pin driver (enable/disable push/pull drivers)
- devices use open-drain outputs to allow the wired-AND connection in a multi-transmitter communication.

Module A in **Figure 6** has an internal connection with only the TxD pin, whereas module B uses an external connection between TxD and RxD pin.

Note: For transmission in case of a wired AND connection with open-drain drivers the internal pull-up resistor on pin RxD in the external connection (module B) can be used. If the internal connection (module A) is used, an external pull-up resister must be connected to V_{DDP}.

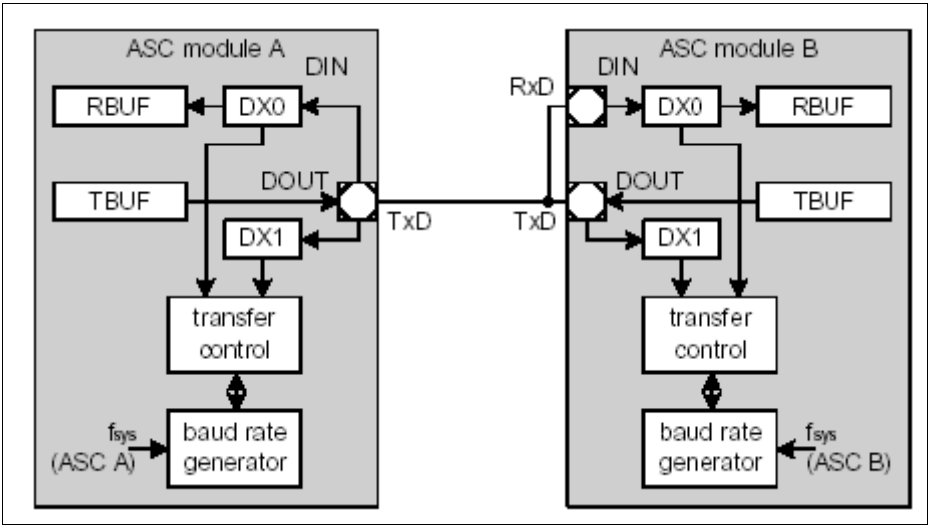


Figure 6 ASC Signal Connection for Half-Duplex Communication

3.2.1 Collision Detection

Collision detection can only be active during the transmission of a frame.

In order to support transmitter collision detection, the input stage DX1 can be used to monitor the level of the transmit line. In this case the actual transmit level is feed back and the USIC state machine can check if the line is in idle state or if a collision occurred.

When a collision is detected, the transmitter immediately aborts the current transmission (transmit shift register is invalid), enters into the idle state, releases the data output (sending passive data level) and sets the status flag (bit COL in register PSR).

Note: to send the data again the SW has to program Tx buffer again.

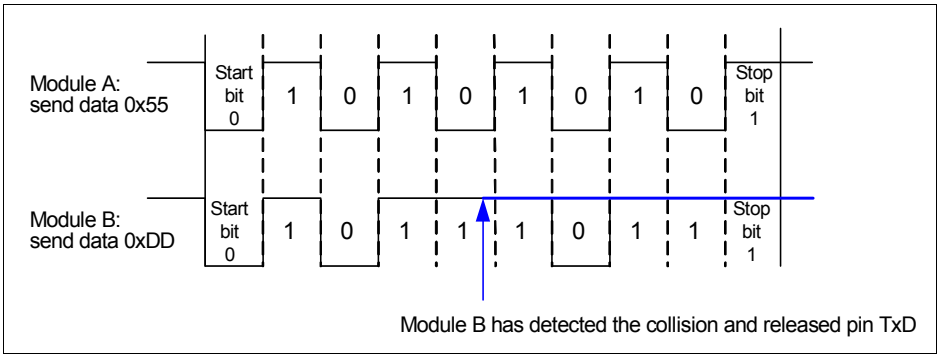


Figure 7 Collision Detection

3.2.2 SW Configuration for Half-Duplex Communication

Example 2: f_{SYS}=80MHz, baud rate=19200bps, U2C0 with external connection

- input stage and output signal
  - input stages (DXxCR):

  DX0(DX0B)=DIN=P3.1 (RxD): DPOL=0 (not inverted), INSW=0 (PPP used)

  DX1(DX1A)=collision detection=DOUT=P3.0 (TxD)

  - output signal:

  TxD=DOUT=P3.0
- baud rate generation and the PPP configuration (BRGL/H, FDRL): the same as in the **Example 1**
- data format configuration (SCTRL/H): the same as in the **Example 1**
- data transmission control (TCSRL/H): the same as in the **Example 1**
- protocol-related information (PCR):
  - sample mode: one sample point (SDM=0)
  - stop bit: one stop bit (STPB=0)
  - sample position: SP=7
  - pulse length: PL=0 (the 0 level is during the complete bit time with bit '0')
  - enable collision detection: CDEN=1 and assign it to the interrupt point SR2 (INPRH=0x0002)
- parity mode (CCR): PM=0, the parity bit is not used
- interrupt configuration:
  - configure the interrupt control register (U2C0_2IC) for collision detection
- input/output pin configuration
  - input P3.1: P3_IOCR01=0x0020 (direct input with pull-up)
  - output P3.0: P3_IOCR00=0x00D0 (ALT1, open-drain)

3.3 IrDA Mode

3.3.1 Introduction

IrDA (Infrared Data Association) is a communications protocol standard and has a widely use in personal area networks. IrDA data communications operate normally in a half-duplex mode. For the transmission rates the IrDA has 3 different encodings (SIR, MIR and FIR). SIR (serial infrared) covers transmission speeds normally supported by using a standard UART port (9600 bits/s...115,2 kbits/s).

the asynchronous data format in IrDA mode is shown in **Figure 8**.

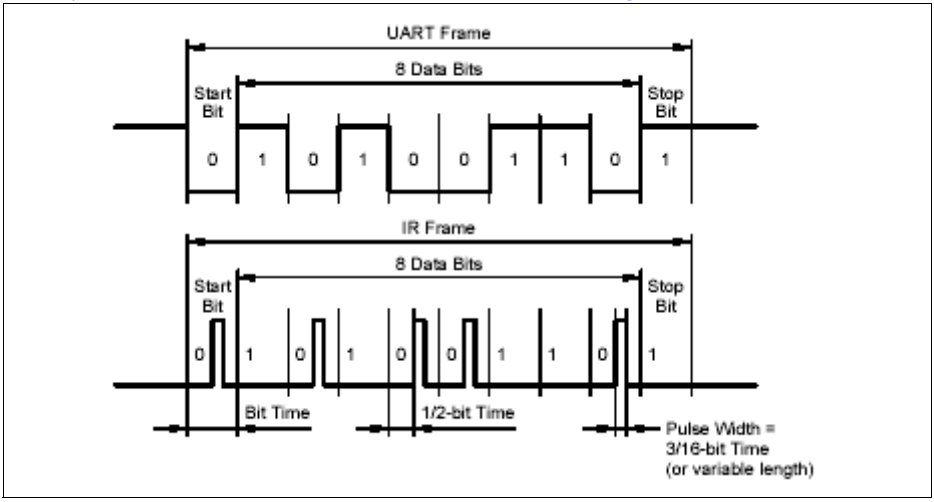


Figure 8 IrDA data frame encoding/decoding

According to the IrDA specification, the signal generated in SIR is as follows:

- on logic '1' the LED is off
- on logic '0' a pulse is created, starting from the center of the bit duration and lasting 3/16 of bit duration or 1,6us (3/16 bit times at 115,2 kbps), depending on the current settings

The USIC module in XE166/XC2000 family provides an IrDA SIR encoder for ASC transmitter. For the IrDA pulse generation the length of the '0' at the output signal can be configured by bit field PL in PCRL register.

Note: The difference to the IrDA specification is that in USIC the 0-pulse always starts at the beginning instead of in the middle of the bit time. USIC can select the IrDA transmit polarity via bit DOCFG in register SCTRL.

3.3.2 SW Configuration for IrDA Mode

Configuration the ASC with the pulse shape with the settings according to the IrDA specification (pulse length =3 tq, 1 bit time=16 tq)

– standard ASC signal (ensured by programming PCRH.PL=000b)

bit value 0: the '0' level during the complete bit time

bit value 1: the '1' level during the complete bit time

– IrDA signal (ensured by programming PCRH.PL!=000b)

bit value 0: 0 pulse (pulse length = PL), the remaining tq are driven with '1' level

bit value 1: the '1' level during the complete bit time

Example 3: f_{SYS}=80MHz, baud rate=19200Baud, U2C0, full-duplex communication

- input stage and output signal
  – input stage (DX0CR):

  DX0(DX0B)=DIN=P3.1(RxD): **DPOL=1** (input signal is inverted), INSW=0 (PPP used)

Note: For the configuration in IrDA mode to receive data frame, the signal polarity of the input signal should be inverted (set bit DPOL in register DX0CR).

  – output signal:

  TxD=DOUT=P3.0

- baud rate generation and the PPP configuration (BRGL/H, FDRL): the same as in **Example 1**
- data format configuration:
  – data shift control (SCTRL): TRM=01b, **DOCFG=1** (DOUT inverted), PDL=1, SDIR=0 (LSB first)
  – word and frame length (SCTRH): WLE=7, FLE=7
- data transmission control (TCSRL): the same as in **Example 1**
- protocol-related information (PCR):
  – sample mode: one sample point (SDM=0)
  – stop bit: one stop bit (STPB=0)
  – sample position: **SP=1**
  – pulse length: **PL=2**
- parity mode (CCR): PM=0, her the parity mode is not used
- interrupt configuration: not used
- input/output pin configuration
  – output P3.0: P3_IOCR00=0x0090 (ALT1, push-pull)

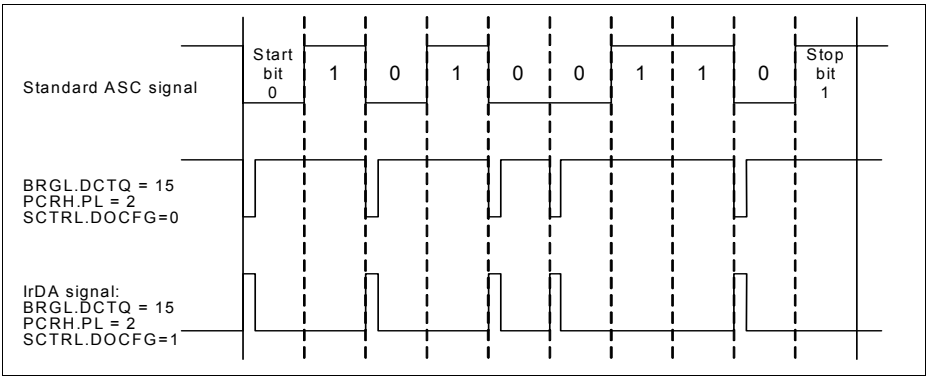


Figure 9 SW Generated IrDA Signal

3.4 LIN (Local Interconnect Network)

3.4.1 Introduction

LIN is a low cost serial asynchronous communication system. The communication is based on the single-wire UART interface. The LIN bus is master-slave bus system with a single master and multiple slaves. The maximum transmission speed is 20 kbps. The low cost bus has been widely used in automotive system.

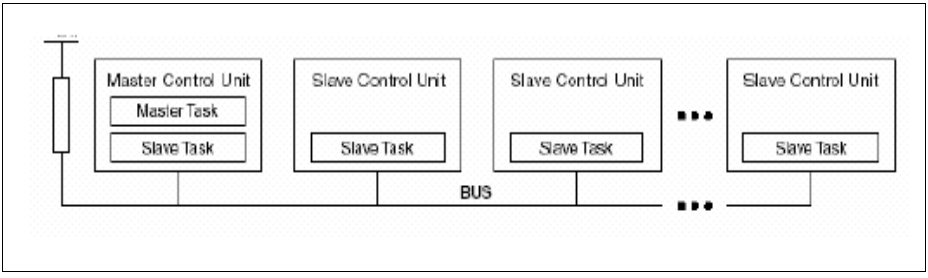


Figure 10 LIN Bus with one Master and Slave Nodes

A LIN cluster consists of one master task and several slave tasks. A master node contains the master task as well as a slave task. All other slave modes contain only a slave task.

The master task decides when and which frame shall be transferred on the bus. The slave tasks provide the data transport by each frame.

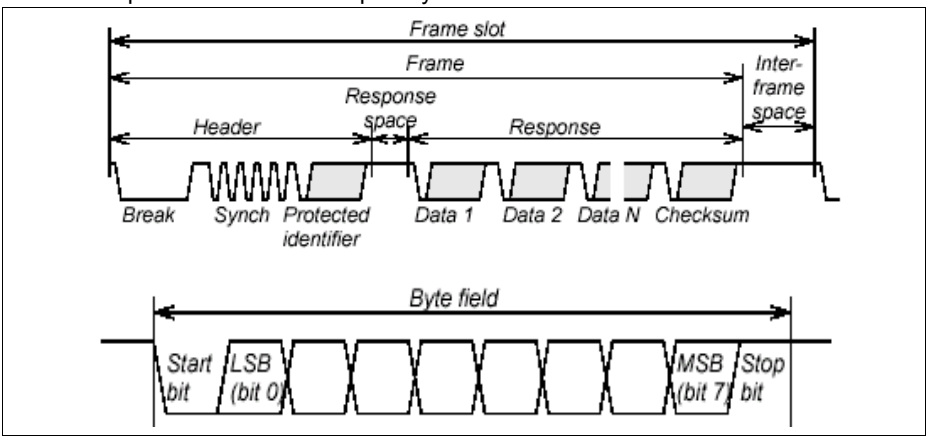


Figure 11 LiN Data Frame

The structure of a frame is shown in **Figure 11**. The frame is constructed of a break followed by four to eleven bytes fields. Each byte field is transmitted as a serial byte. The LSB of the data is sent first.

The master task is responsible for sending out a Break symbol, a one-byte Synch field, a one-byte identifier. The slave task will then respond to that identifier with 2, 4, or 8 data bytes, followed by a single-byte checksum. In LIN bus an identifier addresses a function, not a specific slave node.

- Break symbol is used to signal the beginning of a new frame. It is always generated by the master task and shall have at least 13 bits of dominant value (’0’), including the start bit, followed by a break delimiter with at least 1 nominal bit (’1’).
- Synch byte is a byte field with the data value 0x55. The slave node uses it for the baudrate detection.
- Protected identifier consists of the identifier (bit 0-5) and the identifier parity (bit 6-7).
- Checksum contains the inverted 8 bit sum with carry over all data bytes and the protected identifier.

3.4.2 USIC for LIN Communication

In order to support the LIN protocol, USIC implements the features like

- synchronization break detection (PSR.SBD) and the corresponding protocol interrupt
- baud rate measurement mode (BRGL.TMEN=1)
- automatic frame length control (TCSRL.FLEMD=1). With this setting the bit field SCTRL.FLE will be updated when new data is loaded to register TBUF. This function

can be used in all protocols to dynamically change the data frame length between 1 and 32 data bits per data frame.

3.4.3 SW Configuration for LIN Master Task

Example 4:

f_{SYS}=80MHz, baud rate=19200Baud, U2C0 with external connection

Attention: the standard connection between pin TxD/RxD of Mircocontroller and LIN bus uses LIN transceiver. In this example an alternative solution based on single wired structure is used.

- input stage and output signal
  - input stages (DXxCR):

  DX0(DX0B)=DIN=P3.1(RxD): DPOL=0 (not inverted), INSW=0 (PPP used)

  DX1(DX1B)=collision detection=DOUT=P3.0(TxD)

  - output signal:

  TxD=DOUT=P3.0
- baud rate generation and the PPP configuration (BRGL/H, FDRL)
  - normal divider (FDRL): DM=1, STEP=764
  - baud rate generator: PCTQ=0, DCTQ=15, PDIV=0, CTQSEL=0, PPPEN=0

  f_{ASC}=80,000 x (1/(1024-764)) x (1/1) x (1/1) x (1/16) = 19,231kbps (real value)
- data format configuration (SCTRL/H)
  - data shift control (SCTRL):

  TRM=01b, DOCFG=0(DOUT not inverted), PDL=1, SDIR=0 (LSB first)

  - word and frame length (SCTRH): WLE=12, FLE=12
- data transmission control (TCSRL)
  - use TBUF single shot mode (TDEN=1, TDSSM=1)
  - enable the automatic update of FLE (**FLEMD=1**)
- protocol-related information (PCR):
  - enable collision detection: CDEN=1 and assign it to the interrupt point SR1 (NPRH.PINP=01b)
- parity mode (CCR): PM=0
- interrupt configuration
  - configure the interrupt control register (U2C0_1IC) for collision detection
- input/output pin configuration
  - input P3.1: P3_IOCR01=0x0020 (direct input with pull-up)
  - output P3.0: P3_IOCR00=0x00D0 (ALT1, open-drain)

3.4.4 SW Configuration for LIN Slave Task

Example 5:

f_{SYS}=80MHz, baud rate=19200Baud, U2C1 with external connection

- input stage and output signal
  - input stages (DXxCR):

    DX0(DX0B)=DIN=P3.7(RxD): DPOL=0 (not inverted), INSW=0 (PPP used)

    DX1(DX1B)=collision detection=DOUT=P3.6(TxD)

  - output signal:

    TxD=DOUT=P3.6
- baud rate generation and the PPP configuration (BRGL/H, FDRL): the same as in **Example 4**
- data format configuration (SCTRL/H):
  - data shift control (SCTRL):

    TRM=01b, DOCFG=0 (DOUT not inverted), PDL=1, SDIR=0 (LSB first)

  - word and frame length (SCTRH): WLE=7, FLE=7
- data transmission control (TCSRL):
  - use TBUF single shot mode(TDEN=1, TDSSM=1)
  - disable the automatic update of FLE (FLEMD=0)
- protocol-related information (PCR): the same as **Example 4**
  - enable collision detection: CDEN=1 and assign it to the interrupt point SR1 (INPRH.PINP=01b)
  - enable synchronization break detection (**SBIEN=1**)
- parity mode (CCR): PM=0
- interrupt configuration:
  - configure the interrupt control register (U2C1_1IC) for collision and break detection
- input/output pin configuration
  - input P3.7: P3_IOCR07=0x0020 (direct input with pull-up)
  - output P3.6: P3_IOCR06=0x00D0 (ALT1, open-drain)

4 SSC (Synchronous Serial Channel) Mode

4.1 SSC for Full-Duplex Communication

A full-duplex system allows communication in both direction at the same time. A synchronous data transfer is characterized by a simultaneous transfer of a shift clock signal together with the transmit and receive data signal.

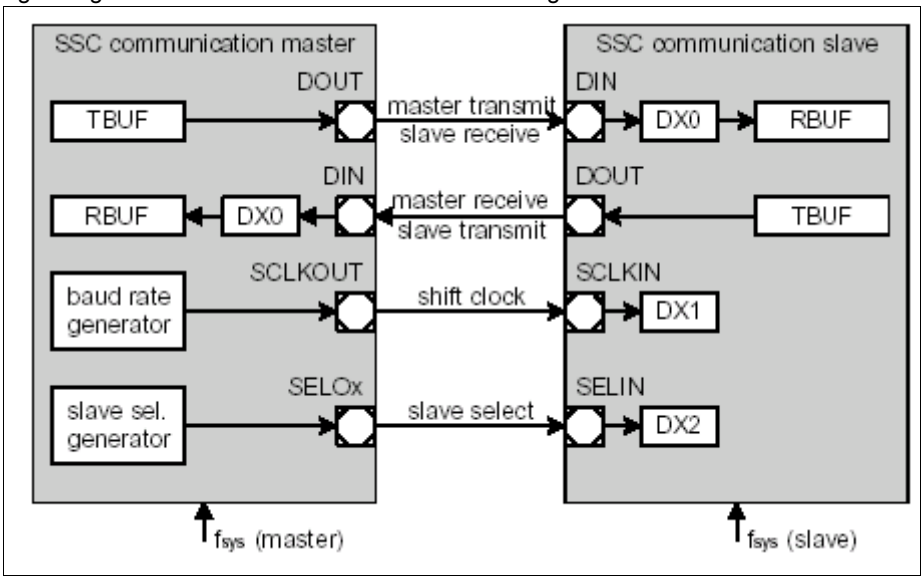


Figure 12 SSC Signal Connection for Full-Duplex Communication

4.1.1 Input/Output Stages and the Protocol Pre-Processor

Master mode

In SSC master mode the protocol pre-processor (PPP) covers DX1 and DX2 input stages. At DX1 the PPP uses the baud rate generator output SCLK directly as input for the data shift unit and gives the signal SCLKOUT on the shift clock output pin. At DX2 the PPP provides the output MSLS with the SSC specific delay (T_{ld}, T_{iw}, T_{td} and T_{nf} see **Figure 13**, and uses it as input for the data shift unit

- input stages
  - shift data input (DX0): is used as MRST(DIN)
  - shift clock input (DX1): optional
  - shift control input (DX2): optional
- output signals
  - shift data output (DOUT): is used as MTSR

– shift clock output (SCLKOUT): master shift clock. It is provided from the internal signal SCLK. The internal shift clock SCLK is generated by the internal baudrate generator. The configuration of SCLKOUT with respect to SCLK has 4 possible settings (via bits field SCLKCFG in register BRGH)
– slave select output (SELO[7:0]): slave select signal. It is provided from the internal MSLS signal. To generate the MSLS signal the bit PCRL.MSLSEN must be set. The duration of an active MSLS ('1') consists of the data frame length (FLE) and SSC specific delay (T_{ld}, T_{iw}, T_{td}).

Note: In application the output pin SLEO[7:0] is usually used as the chip select line (CS) for SSC device and it has normally an active 'low' level. In this case the polarity of the SELO signal has been inverted by set pin SELINV in the register PCR.

Slave mode

In SSC slave mode the protocol pre-processor (PPP) is not used in the input stages. DX0...DX2 signals are directly from the pins DIN, SCLKIN and SELIN.

- input stages:
  – shift data input (DX0): DIN, is used as MTSR
  – shift clock input (DX1): SCLKIN. In slave mode, the signal SCLKIN is received from an external master.
  – shift control input (DX2): SELIN

Note: Since the data shift unit used internal SCLK signal has an active 'high' level, if no SELIN is used, the DX2 stage has to deliver a (permanent) 1-level (by programming bit field DSEL=111b) to the data shift unit

Note: if a 'low' active chip select line (CS) is used as input signal for slave SSC device, its polarity must be inverted.

- output signals:
  – shift data output (DOUT): is used as MRST
  – shift clock output (SCLKOUT): optional
  – shift control outputs(SLEO[7:0]): not used

4.1.2 Baud Rate Generation

The baud rate of the SSC is defined by the frequency of the SCLK signal (one period of f_{SCLK} represents one data bit).

- in standard SSC application, the phase relation between MCLK and SCLK is not relevant, the PPP can be switched OFF (PPPEN=0)

  baud rate calculation (CLKSEL=0 and fractional divider is used):

$$fsclk = fsys \times \frac{STEP}{1024} \times \frac{1}{2} \times \frac{1}{PDIV + 1}$$

- If the phase relation is requested (e.g. using MCLK as clock reference for external devices), PPP must be switched ON (PPPEN=1)

  baud rate calculation (CLKSEL=0 and fractional divider is used):

$$fsclk = fsys \times \frac{1}{2} \times \frac{STEP}{1024} \times \frac{1}{2} \times \frac{1}{PDIV + 1}$$

4.1.3 SSC Frame

If the SSC module is in **master mode**, the slave select signal (the **internal** signal MSLS) is generated automatically by PPP.

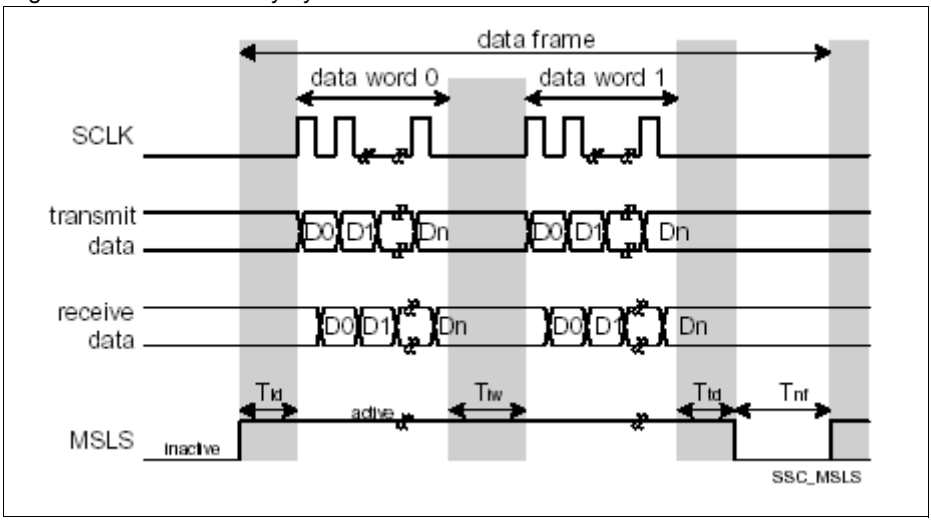


Figure 13 Standard SSC Frame Format with SCLKCFG=00b

In **slave mode**, while the **internal** signal MSLS is inactive, the shift unit does not react on shift clock edges. This ensures data consistency in the shift unit.

– T_{ld} (leading delay): starts if data is valid for transmission. The first shift clock edge (the data shift unit always uses the rising edge for data shifting and the latch edge for data receiving) of SLCK is generated after the leading delay.
– T_{td} (trailing delay): starts at the end of the last SCLK cycle of a data frame. At the end point of the training delay the MSLS becomes inactive. It corresponds to the salve hold time requirements.
– T_{nf} (next-frame delay): after the next-frame delay has elapsed, the frame is considered as finished.
– T_{iw} (inter-word delay): optionally it can be enabled/disabled by PCRH.TIWEN. it is used, if a data frame consists of more than one data word.

In the standard SSC application, like the setup and hold time the T_{ld} and T_{td} are mainly used to ensure stability on the input/output lines.

delay time calculation (CTQSEL=10b and CTQSEL1=10b):

$$Tld = Ttd = \frac{(PCTQ + 1) \times (DCTQ + 1)}{fsclk}$$

$$Tiw = Tnf = \frac{(PCTQ1 + 1) \times (DTCQ1 + 1)}{fsclk}$$

4.1.4 SSC Protocol-Related Control and Status Information

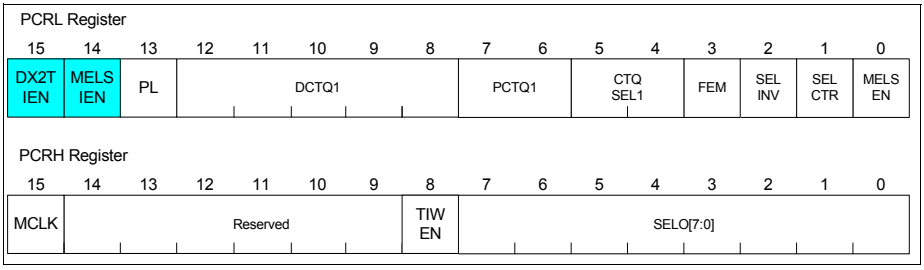


Table 3 Signification of PCR bits for SSC

bit 0	MSLSEN	MSLS enable
bit 1	SELCTR	select control, normally SELCTR is set to 1, in this case SELO is directly connected to an external slave device
bit 2	SELINV	select inversion
bit 3	FEM	frame end mode
bit 4-5	CTQSEL1	input selection for CTQ
bit 6-7	PCTQ1	divider factor to generate f_{PDIV}
bit 8-12	DCTQ1	divider factor to generate f_{TCTQ}/RCTQ
bit 14	MELSIEN	MSLSIEN interrupt enable
bit 15	DX2TIE	DX2T interrupt enable
bit 23-16	SELO	Select output
bit 24	TIWEN	enable inter-word delay T_{iw}
bit 31	MCLK	Master clock enable

PSR Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIF	RIF	TBIF	TSIF	DLIF	RSIF							DX2T EV	MSLS EV	DX2S	MSLS

Table 4 Signification of PSR bits for SSC

bit 0	MSLS	MSLS status
bit 1	DX2S	DX2S status
bit 2	MSLSEV	MSLS event detected
bit 3	DX2TEV	DX2T event detected
bit 10	RSIF	receiver start indication flag
bit 11	DLIF	data lost flag
bit 12	TSIF	transmit shift flag
bit 13	TBIF	transmit buffer flag
bit 14	RIF	receive flag
bit 15	AIF	alternative receive flag

- SSC general interrupt event:
  - RSIF: after the reception of the first data bit of a data word
  - DLIF: RBUF becomes overwritten
  - TSIF: after the start of the last data bit of a data word
  - TBIF: after the start of the first data bit of a data word
  - RIF: after the reception of the last data bit of a data word
  - AIF: is not used
- SSC protocol-specific interrupt events:
  - MSLS event: only in **master mode**, indicates that a data frame has started and/or has been finished. The status bit MSLSEV is set when the event has been occurred. Bit MSLS in PSR register indicates the current MSLS status.
  - DX2T event: in **slave mode** the slave select input signal SELIN is handled by theDX2 stage. This interrupt indicates that a data frame has started and/or has been finished. The status bit DX2TEV is set when the event has been occurred. Bit DX2S in PSR register indicates the current DX2S status.

4.1.5 Master Mode Control

In master mode, if the slave select signal (SELOx) is used (bit PCRL.MSLSEN is set), this signal indicates the start and the end of a data transfer. Two ways can be selected for end of frame controlling:

– with configuration of the data frame length FLE<63 the frame is considered as finished and remaining data bits in the last data word are not transferred, if the programmed number of bits per fame is reached within a data word.
– with configuration of the data frame length FLE=63 the frame is considered as finished and remaining data bits in the last data word are not transferred, if a deactivation of MSLS is detected within a data word.

In **master** mode frame transmission/reception can be started when the data in the transmit buffer TBUF is valid (TCSRL.TDV is set). The internal signal MSLS is set together with the corresponding event flag (PSR.MSLS) and enters the first leading delay state. After the delay (T_{ld}) generated by PPP has elapsed, the internal shift clock SCLK is issued, the data is shift out at the rising edge of SCLK. For every processed data bit when the falling edge of the shift clock SCLK is reached, the level of the input signal is lathed.

4.1.6 Slave Mode Control

In the case that the SELIN input signal is used in **slave** mode, the data frame start/end detection is based on the edge detection of input DX2 in both transmission and reception process. Data frame length (SCTRH.FLE) should be set to its maximum value (63).

in the case that the SELIN input signal is not used in **slave** mode, the data frame length must be programmed to the known value (FLE<63).

4.1.7 SW Configuration for Master Mode in Full-Duplex SSC

Example 6: f_{SYS}=80MHz, baud rate=9600Baud, U1C1=maste mode

- input stage and output signal
  – input stage (DX0CR):

  DX0(DX0B)=DIN=MRST=P0.7: DPOL=0 (not inverted), INSW=1 (PPP not used)

  – output signals:

  DOUT=MTSR=P0.6

  SCLKOUT=P0.5: the shift clock SCLKOUT=internal SCLK (no delay, no polarity inversion via SCLKCFG=00b)

  SELO0=P0.4 as chip select line
- baud rate generation and the PPP configuration (BRGL/H, FDRL)
  – fractional divider (FDRL): DM=2, STEP=1
  – baud rate generator: PCTQ=1, DCTQ=15, PDIV=3, CTQSEL=0, PPPEN=0

  f_{SSC}=80,000 x (1/1024) x (1/2) x (1/4) = 9,765kBaud (real value)
  – $T_{ld} = T_{td}$ = (15+1) x (1+1) / (80,000 x (1/1024) x (1/4)) = 1,64 ms
- data format configuration
  – data shift control (SCTRL):

TRM=01b, DOCFG=0 (DOUT not inverted), PDL=1, SDIR=1(MSB first)

– word and frame length (SCTRH): WLE=15, FLE=15

- data transmission control (TCSRL): use TBUF single shot mode
- protocol-related information (PCR)
  – master slave select signal active: MSLSEN=1
  – master slave select signal control: SELCTR=1 (direct select mode)
  – SELOx polarity: SELINV=1 (inverted for a low active CS line)
  – enable one SEL line (SELO0): PCRH=00000001b
  – delay of T_{NF} and T_{iw}: used (PCRL.PCTQ1/DCTQ1=15/1 and PCRH.TIWEN=0)

  T_{nf} = (15+1) x (1+1) / (80,000 x (1/1024) x (1/4)) = 1,64 ms

  T_{iw} = 0
- parity mode (CCR): PM=0
- interrupt configuration: not used
- input/output pins configuration
  – input P0.7=MRST:

  P0_IOCR07=0x0020 (direct input with pull-up)

  – outputs:

  P0.6=DOUT=MTSR: P0_IOCR06=0x0090 (ALT1, push-pull)

  P0.5=SCLKOUT: P0_IOCR05=0x0090 (ALT1, push-pull)

  P0.4=SELO0: P0_IOCR04=0x0090 (ALT1, push-pull)

4.1.8 SW Configuration for Slave Mode in Full-Duplex SSC

Example 7: f_{SYS}=80MHz, baud rate=9600bps, U1C0

- input stage and output signal
  – input stages (DXxCR):

  DX0(DX0A)=DIN=MTSR=P0.0: DPOL=0 (not inverted), INSW=1 (PPP not used)

  DX1(DX0B)=SCLKIN=P0.2: DPOL=0 (not inverted), INSW=1 (PPP not used)

  DX2(DX0A)=SELIN=P0.3: DPOL=1 (inverted), INSW=1 (PPP not used)

  – output signals:

  DOUT=MRST=DOUT=P0.1
- baud rate generation and the PPP configuration (BRGL/H, FDRL): not needed in slave mode
- data format configuration
  – data shift control (SCTRL): TRM=01b, DOCFG=0 (DOUT not inverted), PDL=1, SDIR=1 (MSB first)
  – word and frame length (SCTRH): WLE=15, FLE=15

- data transmission control (TCSRL): use TBUF single shot mode (TDEN=1, TDSSM=1)
- protocol-related information (PCR): not needed for slave mode
- parity mode (CCR): PM=0
- interrupt configuration: not used
- input/output pins configuration
  – inputs

  P0.0=DIN=MTSR: P0_IOCR00=0x0000 (direct input no pull-up)

  P0.2=SCLKIN: P0_IOCR02=0x0000 (direct input no pull-up)

  P0.3=SELIN: P0_IOCR03=0x0000 (direct input no pull-up)

  – output:

  P0.1=DOUT=MRST: P0_IOCR001=0x0090 (ALT1, push-pull)

4.2 SSC for Half-Duplex Communication

In half-duplex mode, only one data line is used. MRST and MTSR are connected together. One signal is used as input, the other as output, depending on the data direction.

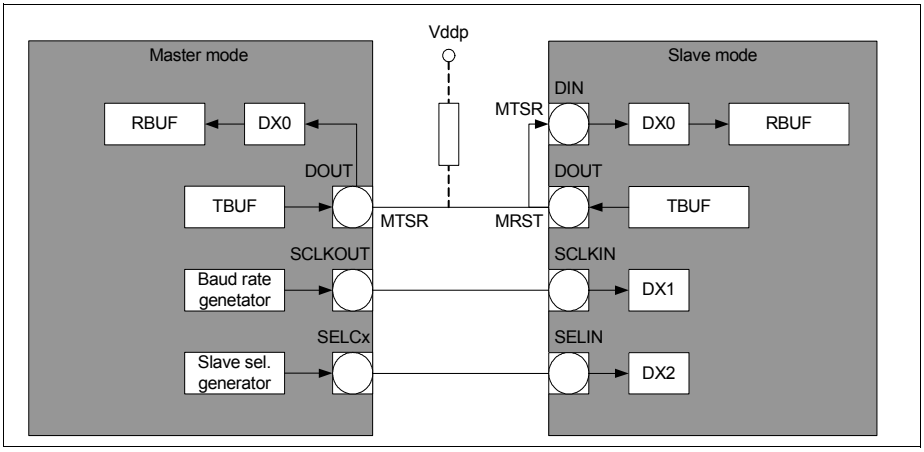


Figure 14 Signal Connection for SSC Half-Duplex Communication

In **Figure 14** the master mode uses an internal connection with the output pin DOUT, meanwhile the slave mode uses an external connection between DOUT and DIN pins. As mentioned in **Chapter 3.2** the user SW has to take care about the data direction to avoid data collision. There are two ways to avoid collisions on the data exchange line

- only the transmitting channel may enable its transmit pin driver (enable/disable push/pull drivers)
- devices use open-drain outputs to allow the wired-AND connection in a multi-transmitter communication

Since the SSC data transfer is synchronized by a simultaneous transfer of a shift clock signal together with the transmit and receive data signal, the dummy data of the register TBUF in slave mode should be prepared as all 1s.

4.3 End-of-Frame Control (CS Signal Inactive Control)

In USIC the number of bits for each data word (max. 16 bits) as well as the number of bits for each frame (max. 64) to be transferred can be individually regulated.

The flexible frame length control is very useful for the SW implementation for SPI communication. USIC provide generally two control modes.

- bits field SCTRH.FLE<63 means that frame has a known length. The MSLS signal is not necessarily required to indicate the start and the end data frame via SW setting bit TCSRL.SOF and TCSRL.EOF. See **Example 10**
- bits field SCTRH.FLE=63 means that the number of bits per data frame is not known. This method can be used for frame with more than 63 data bits.
  - method with SW setting bit SOF, see **Example 11**
  - method with WLEMOD=1, see **Example 13**

With every frame start the frame start signals (TCSRH.TSOF and RBUFSR.SOF) are issued, the protocol divider and transmit time quanta count is reset, the MSLS signal is set together with the flag PSR.MSLS, the frame length kept constant while a data frame is transferred. The word length is updated with the start of each data word.

After t_{ld} has elapsed delay or for every processed data bit the bit PSR.MSLS is checked. if the an abort (bit MSLE cleared by SW) is requested by SW, the delay parameter is activated. MSLS signal becomes inactive after finishing a currently running word transfer and the delay t_{td}.

The frame length FLE is checked also for every processed data bit. If the frame counter reaches the configured value (FLE<63), the delay parameter is activated. MSLS signal becomes inactive after finishing a currently running word transfer and the delay t_{td}.

The WLE is checked for every processed data bit.

- if it indicated as last word MSLS signal becomes inactive after the delay t_{td}.
- if it is not the last word the inter word delay, data valid flag and SW abort request will be checked. if none of the above conditions is met the HW enters 'wait for data valid' state.

Note: this wait state has to be entered normally if a short word length (WLE=2 bits) is used at high baud rates in the case that a word has been completely shifted but the TDV is not updated yet.

4.3.1 AT25128 Introduction

On the XC2000/XE166 easy board a 128Kbit AT25128 EEPROM is equipped. In this section some End-of-Frame control methods will be discussed.

Table 5 Instruction Set for the AT25128

Instruction Name	Instruction Format	Operation
WREN	0x06	set write enable latch
RDSR	0x05	read status register
READ	0x03	read data from memory array
WRITE	0x02	write data to memory array
WRDI	0x04	reset write enable latch
WRSR	0x01	write status register

The AT25128 utilizes an 8-bit instruction register. It is enabled through the chip select pin and accessed via a 3-wire interface consisting of data input SI, data output SD and clock SCK. Because the SCK is always an input, the AT25128 always operates as a slave. The list of their instructions and operation codes (OP_Code) are contained in **Table 5**. All instruction, addresses and data are transferred with MSB first and start with a high-to-low $\overline{CS}$ signal. For detailed information please refer the AT25128 data sheet.

Figure 15 shows EEPROM programming sequences used in this example.

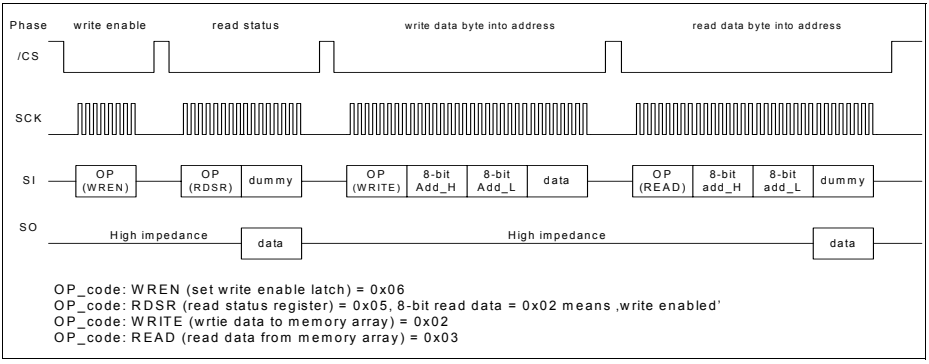


Figure 15 Write and Read one Data Byte into and from AT25128

EEPROM programming contains the following processes.

- read sequence: after the CS line is pulled low the read command is transmitted via the SI line followed by the 16-bit byte address to be read.
- write sequence:
  - first the EEPROM must be initialized to be 'write enabled' via WREN instruction.
  - then a write instruction may be executed.

– a write instruction requires the following sequence: CS is pulled low, write instruction is transmitted via SI line followed by 16-bit address and 8-bit data to be programmed.

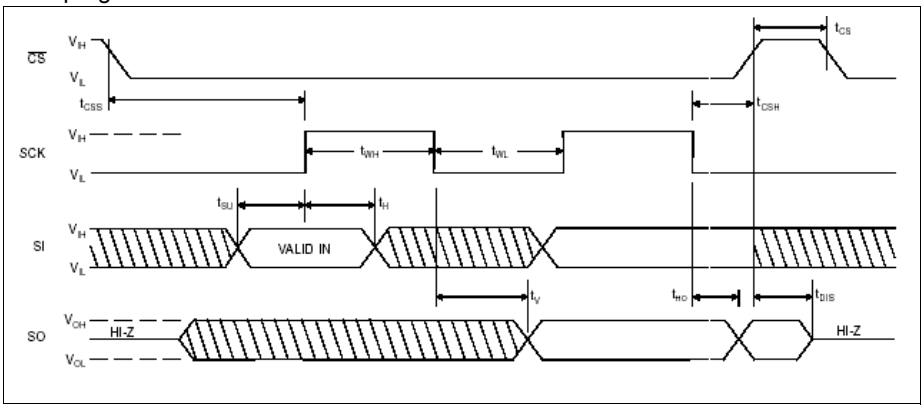


Figure 16 AT25128 Synchronous Data Timing Diagrams

To program the EEPROM successfully following issues have to be considered:

- regarding the AT25128 data timing showed in **Figure 16** the clock signal from XC2000 (SCKLOUT) should be configured with delay by 1/2 shift clock period with no polarity inversion (SCLKCFG=10b). whereas the first data bit is transmitted 1/2 shift clock period before the first rising edge of SCLKOUT.
- during execution of each AT25128 instruction command the CS signal must be kept low (active state). CS going high while each instruction will interrupt this command sequence and a new CS falling edge is required to start the instruction sequence again.
- by the configuration of USIC SSC the leading delay T_{ld}, trailing delay T_{td} and next-frame delay T_{nf} (see **Table 13**) must met the AT25128 parameter t_{CSS}, t_{CSH} and t_{CS}. In the example

  t_{ld} = 1,64 ms

  t_{td} = 1,64 ms

  t_{nf} = 1,64 ms

  t_{iw} = 0 (via PCRH.TIWEN=0b)
- another to be checked is SPI timing parameter.

In XC2000/XE166 family the SSC master mode can run with max. baud rate f_{sys}/2=40Mbaud (@f_{sys}=80MHz). For slave mode the SCLK signal don't have to synchronize and can be used directly. Therefore, the max. theoretical baud rate is 40Mbaud (@f_{sys}=80MHz). In master mode 5V operating range the XC2000 requires (please refer to the data sheet for details)

– t_4 input data (MRST) setup time before latch edge: min. 31 ns
– t_5 input data (MRST) hold time after latch edge: min. -7 ns

and provides

– t_3 transmit data output valid time: min. -6 ns -- max. 13 ns

USIC module needs 2 clock cycles pro bit. If the input data should be sampled in parallel the above setup time must be considered and it restricts the maximum baudrate (1/80ns = 12,5MBuad). In application the input/output driver and propagation delays have to be taken into consideration. If the filter structure is selected in the input stage of USIC, it will have an additional delay.

The AT25128 timing parameters as described in **Table 6**

Table 6 AC Parameters for the AT25128 (Operating Voltage = 5V)

Symbol	Parameter	
t_{CSS}	$\overline{CS}$ setup time	(min.) 100ns
t_{CSH}	$\overline{CS}$ hold time	(min.) 150ns
t_{CS}	CS high time	(min.) 150ns
t_{SU}	Data_In setup time	(min.) 30ns
t_H	Data_In hold time	(min.) 50ns
t_V	Output valid	(max.) 150ns
t_{HO}	output hold time	(min.) 0ns
t_{WH}/t_{WL}	SCK high/low time	(min.)150ns

In this example pro bit = 104us (9600pbs):

– MISO(DIN) signal:

the setup time t_4 and hold time t_5 are important times because they specify the requirements the XC2000 has on the interface of the slave. These times determine how long before the clock edge the slave has to have valid output data ready and how long after the clock edge this data has to be valid. If the setup and hold time are long enough the slave (transmit data) suits to the requirements of the XC2000 (receive data).

AT25128 provides data output valid time t_V = min. 0 ns - max 150ns. In this example, the real setup time (104/2us - 150ns(t_V) >> 31ns(t_4)) are long enough and it means AT25128 suits to the requirements of the XC2000.

– MOSI(DOUT) signal:

the time t_3 specifies the minimum time the XC2000 has valid output data ready before the clock edge occurs and the maximum time after which the XC2000 outputs the next data bit.

to check if the XC2000 suits to the requirements of the slave the XC2000 time t_3 must be compared to the AT25128 setup time t_{SU} (30ns). If the setup and hold time of

transmitter are long enough XC2000 (transmit data) suits to the requirements of AT25128 (receive data)

4.3.2 EEPROM Programming

In this section we use a very simple data write and read sequence as example and explain the important issues and timing parameters to be considered by EEPROM programming. In this example different alternate End-of-Frame control mechanisms implemented in USIC module will be discussed step by step.

Example 10: End-of-Frame handling is controlled by SCTRH.FLE.

If the number of bits per frame (<63 bits) is known in the application bits FLE can be set with a fixed value. The initialization routine is the same in **Example 6**.

- data format configuration
  - word and frame length in phase 'write enable': WLE=7, FLE=7
  - word and frame length in phase 'read status': WLE=15, FLE=15
  - word and frame length in phase 'write data byte into address': WLE=7, FLE=31
  - word and frame length in phase 'read data byte from address': WLE=7, FLE=31
- interrupt configuration: not used (polling system)

Note: flag RIF (bit_14 in PSR) is used to check whether the data word has been received.

Note: in XC2000 the flag AIF (bit_15 in PSR) is not defined, whereas in XC2000M this flag is indicated for the reception of the first word in a frame.

Note: in the example bit PCRL.FEM is set to 0 that means "an end of frame is assumed if the transmit buffer TBUF does not contain valid transmit data at the end of a data word transmission (TCSRL.TDV = 0 or in Stop Mode)". In this case, if single step is called in debug tool and TBUF runs empty, then CS is brought to high. Therefore, it is recommended to set bit PCRL.FEM while debug mode is activated in code verification phase.

Example 11: use SW-based Start-of-Frame indication SOF, FLE must be set to 63.

- data format configuration
  - word and frame length in phase 'write enable': WLE=7, FLE=63
  - word and frame length in phase 'read status': WLE=15, FLE=63
  - word and frame length in phase 'write data byte into address': WLE=7, FLE=63
  - word and frame length in phase 'read data byte from address': WLE=7, FLE=63
- interrupt configuration: not used (polling system)
- TBUF date transfer without FIFO
- bit TCSRL.SOF has to be set before writing OP_Code of EEPROM to TBUF in all four phase to generate a newly start MSLS signal after a current data word transfer has been finished completely and delay t_{td} has been elapsed.

Note: bit End-of-Frame flag TCSRL.EOF has a similar functionality like bit TCSRL.SOF.

Note: due to the bug USIC_AI.003 (refer to the errata sheet) bit TCSRL.EOF and TCSRL.SOF must be cleared by software if the example code runs in XC2000 and XE166 derivatives. This bug has been fixed in XC2000M/XE166M devices.

Example 12: use explicit End-of-Frame by software (reset bit_0 of register PSCR)

This write action immediately clears bit PSR.MSLS, whereas the SELOx signal becomes inactive after finishing a currently running word transfer (with the delays T_{td}).

- data format configuration is like in **Example 11**

Example 13: use automatic update by data handler via WLEMD=1

if TCSRL.WLEMD=1, the Word Length Control bit SCTRH.WLE and SCTRL.EOF are updated with TCI (Transmit Control Information). Using this feature the USIC can dynamically control the data word length between 1 and 16 data bits per data word. Additionally, SSC master mode can control CS signal to finish data frames.

initialization routine is the same as in the **Example 10**.

U1C1_TCSRL.WLEMD=1

U1C1_TCSRH.FLE=max.=63, U1C1_TCSRH.WLE=don't care

- data word length controlled by TCI[4-0]:
  - phase 'write enable': (8_bit with End-of-Frame)

    U1C1_TBUF23=EEPROM_CM_WREN
  - phase 'read status': (16_bit with End-of-Frame)

    U1C1_TBUF31=EEPROM_CM_READ_STATUS<<8| EEPROM_WRITE_DUMMY
  - phase 'write data byte into address':

    U1C1_TBUF7=EEPROM_CM_WRITE (8_bit without End-of-Frame)

    U1C1_TBUF7=EEPROM_ADDRESS_HIGH (8_bit without End-of-Frame)

    U1C1_TBUF7=EEPROM_ADDRESS_LOW (8_bit without End-of-Frame)

    U1C1_TBUF23=EEPROM_WriteData (8_bit with End-of-Frame)
  - phase 'read data byte from address':

    U1C1_TBUF7=EEPROM_CM_WRITE (8_bit without End-of-Frame)

    U1C1_TBUF7=EEPROM_ADDRESS_HIGH (8_bit without End-of-Frame)

    U1C1_TBUF7=EEPROM_ADDRESS_LOW (8_bit without End-of-Frame

    U1C1_TBUF23=EEPROM_WriteData (8_bit with End-of-Frame)
- interrupt configuration: not used (polling system)
- data FIFO is not used

Note: in this example we can also write transmit data to the FIFO registers U1C1_IN[31:0] instead of registers U1C1_TBUF[31:0]. In this case, the FIFO-based address related End-of-Frame handling is used and we will get the same behavior.

Note: bits TBCTRH.SIZE (buffer size) must be set to >0 in order to allow to write the FIFO input registers Iu1C1_IN[31:0].

Example 14: use PECx in EEPROM data transfer process.

In this example byte transfer is used in 'write' and 'read' phase. The receiver interrupt (RI) is enabled and assigned to the output line SR2. The receiver start interrupt (RSI) is enabled and assigned to the output line SR1

- PEC2 is defined as RSI (receiver start interrupt) for loading data to the transmit buffer, byte transfer with PEC transfer count = 4

- PEC3 is defined as RI (receiver interrupt) for data receive and store data, byte transfer with PEC transfer count = 4

- data format configuration
  - word and frame length in phase 'write data byte into address': WLE=7, FLE=31
  - word and frame length in phase 'read data byte from address': WLE=7, FLE=31

Note: using this code for XC2000M device the AI (alternative interrupt) must be enabled and assigned to SR2.

4.4 Multiple MSLS Output Signals Generation

The SSC module supports up to 8 different SELOx output signals for master mode operation in one USIC module. USIC provide generally two configuration modes to select the MSLS signal.

- direct control mode: write PCRH.SELO[7:0] as individual value for each SELOx line
- automatic update mode: it is enabled by TCSRL.SELMD=1. In this case, PCRH.SELO[4:0] is updated with TCI[4:0] and PCRH.SELO[7:5] is always '0'.

Each USIC module has the transmit buffer input locations TBUFx (x=00-31) which is addressed by using 32 consecutive addresses. If TCSRL.SELMD=1 data written to one of these locations will appear in a common register TBUF, additionally, the 5-bit TCI[4:0] coding will be updated accordingly. The relationship between TBUFx, TCI[x] and MSELx is listed in **Table 7**.

Table 7 MSLS Signals in Automatic Update Mode

Write to TBUFx	TCI[4:0]	PCRH.SELO[7:0]	SELOx signals
TBUF01	00001_b	$0000,0001_b$	SELO0 active
TBUF02	00010_b	$0000,0010_b$	SELO1 active
TBUF04	00100_b	$0000,0100_b$	SELO2 active
TBUF08	01000_b	$0000,1000_b$	SELO3 active
TBUF16	10000_b	$0001,0000_b$	SELO4 active
TBUF03	00011_b	$0000,0011_b$	SELO0/1 active
TBUF07	00111_b	$0000,0111_b$	SELO0/1/2 active

TBUF0	00000_b	$0000,0000_b$	no SELOx
.....	.....	.....	.....

Example 15: automatic MSLS signals update control

U1C1_SELO0: pin 0.4, ALT1 output, P0_IOCR04 = 0x0090

U1C1_SELO1: pin 0.3, ALT2 output, P0_IOCR03 = 0x00A0

Note: in this example we can also write transmit data to the FIFO registers U1C1_IN[31:0] instead of registers U1C1_TBUF[31:0]. In this case, the FIFO-based automatic MSLS update handling is used and we will get the same behavior.

Note: bits TBCTRH.SIZE (buffer size) must be set to >0 in order to allow to write the FIFO input register U1C1_IN[31:0].

5 Appendix: Source Code

5.1 Example 1

```
//*****************************************************************************
// @Function      void U2C0_ASC_Full_Duplex_vInit(void)
// @Date          15.11.2007
//*****************************************************************************
void U2C0_ASC_Full_Duplex_vInit(void)
{
  Sys_Protection(0);
  U2C0_KSCFG = 3;
  Sys_Protection(1);

  U2C0_CCR  = 0x0000;      // recommanded

  ///  ---------------------------------------------------------------------
  ///  1. baud rate generation and the PPP configuration
  ///  baud rate = 19200,
  ///  ---------------------------------------------------------------------
  U2C0_FDRL=((uword)2<<14)|(0x312<<0);    //DM=2 (fractional divider),STEP=0x312
  U2C0_BRGL=(15<<10)|(1<<8)|(0<<4)|(0<<0);//DCTQ=15,PCTQ=1,PPPEN=0,CLKSEL=0
  U2C0_BRGH=(0<<14)|(0<<13)|(99<<0);      //SCLKCFG=0,MCLKCFG=0,PDIV=99

  ///  ---------------------------------------------------------------------
  ///  2. Configuration of the U2C0 Input Control Register
  ///  ---------------------------------------------------------------------
            // CM  SFSEL   DPOL   DSEN    DFEN   INSW   DESL
  U2C0_DX0CR=((0<<10)|(0<<9)|(0<<8)|(0<<6)|(0<<5)|(0<<4)|(1<<0)); //P3.1(DX0B)

  ///  ---------------------------------------------------------------------
  ///  3. Data format: shift control signal, word/frame length control
  ///  ---------------------------------------------------------------------
        //TRM=01b(ASC)   DOCFG       PDL     SDIR=0(LSB first)
  U2C0_SCTRL = (1 << 8) | (0 << 6) | (1 << 1) | (0 << 0);
  U2C0_SCTRH = (7 << 8) | (7 << 0); //WLE=7, FLE=7;

  ///  ---------------------------------------------------------------------
  ///  4. data transmission control (TBUF single shot mode)
  ///  ---------------------------------------------------------------------
             // TDEN=1   TDSSM=1
  U2C0_TCSRL = (1 << 10) | (1 << 8);
  U2C0_TCSRH = 0x0000;

  ///  ---------------------------------------------------------------------
  //   5. protocol-related information
  // CTR[0]     SMD       sample mode
  // CTR[1]     STPB      stop bit
  // CTR[2]     IDM       idle detection mode
  // CTR[3]     SBIEN     Sync.-Break Interrupt Enable
  // CTR[4]     CDEN      Collision Detection Enable (for half-duplex mode)
  // CTR[5]     RNIEN     Receiver Noise Detection Interrupt Enable
  // CTR[6]     FEIEN     Format Error Interrupt Enable
  // CTR[7]     FFIEN     Frame Finished Interrupt Enable
  // CTR[8-12]  SP        Sample Position
  // CTR[13-15] PL        Pulse Length
  // CTR[31]    MCLK      master clock enable
  ///  ---------------------------------------------------------------------
           //  PL     SP       FFIEN    FFIEN    RNIEN     CDEN    SBIEN     IDM      STPB    SMD
U2C0_PCRL=(0<<13)|(7<<8)|(0<<7)|(0<<6)|(0<<5)|(0<<4)|(0<<3)|(0<<2)|(0<<1)|(0<<0);
  U2C0_PCRH=(0<<15); // MCLK not used

  ///  ---------------------------------------------------------------------
  ///  6. Configuration of the interrupts point and interrupt control registers
  ///  ---------------------------------------------------------------------
```

```
            //AINP=SR0, RINP=SR0,  TBINP=SR0, TSINP=SR0
  U2C0_INPRL = (0 << 12) | (0 << 8) | (0 << 4) | (0 << 0);
  U2C0_INPRH = (0 << 0); //PINP=SR0

  U2C0_0IC = (0x80 * 0 + 0x40 * 0 + ((10 << 2)|0)); //IR=0, IE=0, ILVL=10, GPX=0
  U2C0_1IC = (0x80 * 0 + 0x40 * 0 + ((10 << 2)|1)); //IR=0, IE=0, ILVL=10, GPX=1
  U2C0_2IC = (0x80 * 0 + 0x40 * 0 + ((10 << 2)|2)); //IR=0, IE=0, ILVL=10, GPX=2

  ///  ----------------------------------------------------------------------
  ///  7. Configuration of the U2C0 Channel Control Register
  //   enable/disable general interrupt
  //   enable/disable the parity mode
  //   select the protocol mode
  ///  ----------------------------------------------------------------------
           // AIEN=0,  RIEN=0,  TBIEN=0,  TSIEN=0,  DLIEN=0,  RSIEN=0
  U2C0_CCR  = (0 << 15)|(0 << 14)|(0 << 13)|(0 << 12)|(0 << 11)|(0 << 10);
  U2C0_CCR |= (0 << 8);  // PM=0(no parity generation)
  U2C0_CCR |= 0x0002;    // active ASC mode

  ///  ----------------------------------------------------------------------
  ///  8. input/output pins configuration
  ///  ----------------------------------------------------------------------
  P3_IOCR00 = 0x0090;  // P3.0=output(ALT1 Push/pull)
}


//****************************************************************************
// @Function      void Sys_Protection(int)
// @Date          15.11.2007
//****************************************************************************
void Sys_Protection(int on_off)
{
   if( on_off )
   {
      SLC = 0xaaaa;
      SLC = 0x5554;
      SLC = 0x96ff;
      SLC = 0x1800;
    }
   else
   {
      SLC = 0xaaaa;
      SLC = 0x5554;
      SLC = 0x96ff;
      SLC = 0x0000;
   }
}
```

5.2 Example 2

```
//****************************************************************************
// @Function      void U2C0_ASC_HalfDuplex_vInit(void)
// @Date          15.11.2007
//****************************************************************************
void U2C0_ASC_Half_Duplex_vInit(void)
{
  Sys_Protection(0);
  U2C0_KSCFG = 3;
  Sys_Protection(1);

  U2C0_CCR  = 0x0000;      // recommanded

  ///  ----------------------------------------------------------------------
  ///  1. baud rate generation and the PPP configuration
  ///  baud rate = 19200,
  ///  ----------------------------------------------------------------------
```

```
U2C0_FDRL=((uword)2<<14)|(0x312<<0);     //DM=2 (fractional divider), STEP=0x312
U2C0_BRGL=(15<<10)|(1<<8)|(0<<4)|(0<<0); //DCTQ=15,PCTQ=1,PPPEN=0,CLKSEL=0
U2C0_BRGH=(0<<14)|(0<<13)|(99<<0);       //SCLKCFG=0,MCLKCFG=0,PDIV=99

///  ----------------------------------------------------------------------
///  2. Configuration of the U2C0 Input Control Register
///  ----------------------------------------------------------------------
            // CM    SFSEL  DPOL   DSEN   DFEN   INSW   DESL
U2C0_DX0CR=((0<<10)|(0<<9)|(0<<8)|(0<<6)|(0<<5)|(0<<4)|(1<<0));//P3.1(DX0B)
// if the internal connection is used DESL=0b // P3.0(DX0A)
U2C0_DX1CR=((0<<10)|(0<<9)|(0<<8)|(0<<6)|(0<<5)|(0<<4)|(0<<0));//P3.0 colliion

///  ----------------------------------------------------------------------
///  3. Data format: shift control signal, word/frame length control
///  ----------------------------------------------------------------------
        //TRM=01b(ASC)   DOCFG       PDL     SDIR=0(LSB first)
U2C0_SCTRL = (1 << 8) | (0 << 6) | (1 << 1) | (0 << 0);
U2C0_SCTRH = (7 << 8) | (7 << 0); //WLE=7, FLE=7;

///  ----------------------------------------------------------------------
///  4. data transmission control (TBUF single shot mode)
///  ----------------------------------------------------------------------
             // TDEN=1   TDSSM=1
U2C0_TCSRL = (1 << 10) | (1 << 8);
U2C0_TCSRH = 0x0000;

///  ----------------------------------------------------------------------
//   5. protocol-related information
// CTR[0]     SMD       sample mode
// CTR[1]     STPB      stop bit
// CTR[2]     IDM       idle detection mode
// CTR[3]     SBIEN     Sync.-Break Interrupt Enable
// CTR[4]     CDEN      Collision Detection Enable (for half-duplex mode)
// CTR[5]     RNIEN     Receiver Noise Detection Interrupt Enable
// CTR[6]     FEIEN     Format Error Interrupt Enable
// CTR[7]     FFIEN     Frame Finished Interrupt Enable
// CTR[8-12]  SP        Sample Position
// CTR[13-15] PL        Pulse Length
// CTR[31]    MCLK      master clock enable
///  ----------------------------------------------------------------------
       // PL   SP    FFIEN  FFIEN  RNIEN   CDEN  SBIEN   IDM    STPB   SMD
U2C0_PCRL=(0<<13)|(7<<8)|(0<<7)|(0<<6)|(0<<5)|(1<<4)|(0<<3)|(0<<2)|(0<<1)|(0<<0);
U2C0_PCRH=(0<<15); // MCLK not used

///  ----------------------------------------------------------------------
///  6. Configuration of the interrupts point and interrupt control registers
///  ----------------------------------------------------------------------
             //AINP=SR0, RINP=SR0, TBINP=SR0, TSINP=SR0
U2C0_INPRL = (0 << 12) | (0 << 8) | (0 << 4) | (0 << 0);
U2C0_INPRH = (2 << 0); //PINP=SR2 for collision detection

U2C0_0IC = (0x80 * 0 + 0x40 * 0 + ((10 << 2)|0));  //IR=0, IE=0, ILVL=10, GPX=0
U2C0_1IC = (0x80 * 0 + 0x40 * 0 + ((10 << 2)|1));  //IR=0, IE=0, ILVL=10, GPX=1
U2C0_2IC = (0x80 * 0 + 0x40 * 1 + ((10 << 2)|2));  //IR=0, IE=1, ILVL=10, GPX=2

///  ----------------------------------------------------------------------
///  7. Configuration of the U2C0 Channel Control Register
//   enable/disable general interrupt
//   enable/disable the parity mode
//   select the protocol mode
///  ----------------------------------------------------------------------
            // AIEN=0, RIEN=0,  TBIEN=0,  TSIEN=0,  DLIEN=0,  RSIEN=0
U2C0_CCR  = (0 << 15)|(0 << 14)|(0 << 13)|(0 << 12)|(0 << 11)|(0 << 10);
U2C0_CCR |= (0 << 8);  // PM=0(no parity generation)
U2C0_CCR |= 0x0002;    // active ASC mode

///  ----------------------------------------------------------------------
```

```
  ///  8. input/output pins configuration
  ///  ------------------------------------------------------------------------
  P3_IOCR01 = 0x0020;  // P3.1=input(pull_up)
  P3_IOCR00 = 0x00D0;  // P3.0=output(ALT1 open-drain)
}


//****************************************************************************
// @Function      void U2C0_ASC_viProtocoll(void)
// @Date          15.11.2007
//****************************************************************************
_interrupt(U2C0_SRN2)  void U2C0_ASC_viProtocoll(void)
{
  U2C0_PSCR   |= 0x0008;      // clear PSR_COL Collesion
  while(1);
}

//****************************************************************************
// @Function      void U2C0_ASC_viProtocoll(void)
// @Date          15.11.2007
//****************************************************************************
_interrupt(U2C1_SRN2)  void U2C1_ASC_viProtocoll(void)
{
  U2C1_PSCR   |= 0x0008;      // clear PSR_COL Collesion
  while(1);
}
```

5.3 Example 3

```
//****************************************************************************
// @Function      void U2C0_ASC_HalfDuplex_IrDA_vInit(void)
// @Date          15.11.2007
//****************************************************************************
void U2C0_ASC_FullDuplex_IrDA_vInit(void)
{
  Sys_Protection(0);
  U2C0_KSCFG = 3;
  Sys_Protection(1);

  U2C0_CCR  = 0x0000;      // recommanded
  ///  ------------------------------------------------------------------------
  ///  1. baud rate generation and the PPP configuration
  ///  baud rate = 19200,
  ///  ------------------------------------------------------------------------
  U2C0_FDRL=((uword)2<<14)|(0x312<<0);     //DM=2 (fractional divider), STEP=0x312
  U2C0_BRGL=(15<<10)|(1<<8)|(0<<4)|(0<<0); //DCTQ=15,PCTQ=1,PPPEN=0,CLKSEL=0
  U2C0_BRGH=(0<<14)|(0<<13)|(99<<0);       //SCLKCFG=0,MCLKCFG=0,PDIV=99

  ///  ------------------------------------------------------------------------
  ///  2. Configuration of the U2C0 Input Control Register
  ///  ------------------------------------------------------------------------
            // CM    SFSEL   DPOL  DSEN    DFEN  INSW   DESL
  U2C0_DX0CR =((0<<10)|(0<<9)|(1<<8)|(0<<6)|(0<<5)|(0<<4)|(1<<0)); //P3.1(DX0B)

  ///  ------------------------------------------------------------------------
  ///  3. Data format: shift control signal, word/frame length control
  ///  ------------------------------------------------------------------------
          //TRM=01b(ASC)   DOCFG       PDL     SDIR=0(LSB first)
  U2C0_SCTRL = (1 << 8) | (1 << 6) | (1 << 1) | (0 << 0);
  U2C0_SCTRH = (7 << 8) | (7 << 0); //WLE=7, FLE=7;

  ///  ------------------------------------------------------------------------
  ///  4. data transmission control (TBUF single shot mode)
  ///  ------------------------------------------------------------------------
              // TDEN=1   TDSSM=1
  U2C0_TCSRL = (1 << 10) | (1 << 8);
```

```
  U2C0_TCSRH = 0x0000;

  ///  ----------------------------------------------------------------------
  //   5. protocol-related information
  // CTR[0]     SMD       sample mode
  // CTR[1]     STPB      stop bit
  // CTR[2]     IDM       idle detection mode
  // CTR[3]     SBIEN     Sync.-Break Interrupt Enable
  // CTR[4]     CDEN      Collision Detection Enable (for half-duplex mode)
  // CTR[5]     RNIEN     Receiver Noise Detection Interrupt Enable
  // CTR[6]     FEIEN     Format Error Interrupt Enable
  // CTR[7]     FFIEN     Frame Finished Interrupt Enable
  // CTR[8-12]  SP        Sample Position
  // CTR[13-15] PL       Pulse Length
  // CTR[31]    MCLK      master clock enable
  ///  ----------------------------------------------------------------------
           // PL   SP   FFIEN  FFIEN  RNIEN   CDEN   SBIEN  IDM   STPB   SMD
 U2C0_PCRL=(2<<13)|(1<<8)|(0<<7)|(0<<6)|(0<<5)|(0<<4)|(0<<3)|(0<<2)|(0<<1)|(0<<0);
  U2C0_PCRH=(0<<15); // MCLK not used

  ///  ----------------------------------------------------------------------
  ///  6. Configuration of the interrupts point and interrupt control registers
  ///  ----------------------------------------------------------------------
              //AINP=SR0, RINP=SR0, TBINP=SR0, TSINP=SR0
  U2C0_INPRL = (0 << 12) | (0 << 8) | (0 << 4) | (0 << 0);
  U2C0_INPRH = (2 << 0); //PINP=2 for collision detection

  U2C0_0IC = (0x80 * 0 + 0x40 * 0 + ((10 << 2)|0));  //IR=0, IE=0, ILVL=10, GPX=0
  U2C0_1IC = (0x80 * 0 + 0x40 * 0 + ((10 << 2)|1));  //IR=0, IE=0, ILVL=10, GPX=1
  U2C0_2IC = (0x80 * 0 + 0x40 * 0 + ((10 << 2)|2));  //IR=0, IE=1, ILVL=10, GPX=2

  ///  ----------------------------------------------------------------------
  ///  7. Configuration of the U2C0 Channel Control Register
  //   enable/disable general interrupt
  //   enable/disable the parity mode
  //   select the protocol mode
  ///  ----------------------------------------------------------------------
             // AIEN=0, RIEN=0,  TBIEN=0,  TSIEN=0,  DLIEN=0,  RSIEN=0
  U2C0_CCR  = (0 << 15)|(0 << 14)|(0 << 13)|(0 << 12)|(0 << 11)|(0 << 10);
  U2C0_CCR |= (0 << 8);  // PM=0(no parity generation)
  U2C0_CCR |= 0x0002;    // active ASC mode

  ///  ----------------------------------------------------------------------
  ///  8. input/output pins configuration
  ///  ----------------------------------------------------------------------
  P3_IOCR00 = 0x0090;  // P3.0=output(ALT1 Push/pull)

}
```

5.4 Example 4

```
//****************************************************************************
// @Function      void U2C0_LIN_Master_vInit(void)
// @Date          15.11.2007
//****************************************************************************
void U2C0_LIN_Master_vInit(void)
{
  Sys_Protection(0);
  U2C0_KSCFG = 3;
  Sys_Protection(1);

  U2C0_CCR       =  0x0000;      // recommanded
  ///  ----------------------------------------------------------------------
  ///  1. baud rate generation and the PPP configuration
  ///  baud rate = 19200,
  ///  fasc=80000 x 1000 /(1024-STEP) / (PDIV+1) / (PCTQ+1) / (DCTQ+1) = 19231
```

```
///  ---------------------------------------------------------------------
U2C0_FDRL=(((uword)1<<14)|(764<<0));       //DM=1(normal divider), STEP=764
U2C0_BRGL=((15<<10)|(0<<8)|(0<<4)|(0<<0)); //DCTQ=15,PCTQ=0,PPPEN=0,CLKSEL=0
U2C0_BRGH=((0<<14)|(0<<13)|(0<<0));        //SCLKCFG=0,MCLKCFG=0,PDIV=0

///  ---------------------------------------------------------------------
///  2. Configuration of the U2C1 Input Control Register and port pins
///  ---------------------------------------------------------------------
            // CM  SFSEL   DPOL   DSEN    DFEN   INSW   DESL
U2C0_DX0CR=((0<<10)|(0<<9)|(0<<8)|(0<<6)|(0<<5)|(0<<4)|(1<<0)); //P3.1(DX0B)
U2C0_DX1CR=((0<<10)|(0<<9)|(0<<8)|(0<<6)|(0<<5)|(0<<4)|(0<<0)); //P3.0 colliion

///  ---------------------------------------------------------------------
///  3. Data format: shift control signal, word/frame length control
///  ---------------------------------------------------------------------
       //TRM=01b(ASC)   DOCFG       PDL     SDIR=0(LSB first)
U2C0_SCTRL = (1 << 8) | (0 << 6) | (1 << 1) | (0 << 0);
U2C0_SCTRH = (12 << 8) | (12 << 0); //WLE=12, FLE=12;

///  ---------------------------------------------------------------------
///  4. data transmission control (TBUF single shot mode)
///  ---------------------------------------------------------------------
             // TDEN=1   TDSSM=1    FLEMD=1   WLEMD=0
U2C0_TCSRL = (1 << 10) | (1 << 8) | (1 << 2) | (0 << 0);
U2C0_TCSRH = 0x0000;

///  ---------------------------------------------------------------------
//   5. protocol-related information
// CTR[0]     SMD       sample mode
// CTR[1]     STPB      stop bit
// CTR[2]     IDM       idle detection mode
// CTR[3]     SBIEN     Sync.-Break Interrupt Enable
// CTR[4]     CDEN      Collision Detection Enable (for half-duplex mode)
// CTR[5]     RNIEN     Receiver Noise Detection Interrupt Enable
// CTR[6]     FEIEN     Format Error Interrupt Enable
// CTR[7]     FFIEN     Frame Finished Interrupt Enable
// CTR[8-12]  SP        Sample Position
// CTR[13-15] PL        Pulse Length
// CTR[31]    MCLK      master clock enable
///  ---------------------------------------------------------------------
         // PL   SP    FFIEN  FFIEN  RNIEN   CDEN  SBIEN   IDM    STPB   SMD
U2C0_PCRL=(0<<13)|(7<<8)|(0<<7)|(0<<6)|(0<<5)|(1<<4)|(0<<3)|(0<<2)|(0<<1)|(0<<0);
U2C0_PCRH=(0<<15); // MCLK not used

///  ---------------------------------------------------------------------
///  6. Configuration of the interrupts point and interrupt control registers
///  ---------------------------------------------------------------------
            //AINP=SR0, RINP=SR0,  TBINP=SR0, TSINP=SR0
U2C0_INPRL = (0 << 12) | (0 << 8) | (0 << 4) | (0 << 0);
U2C0_INPRH = (1 << 0); //PINP=SR1 for collision detection and SBIR

U2C0_0IC = (0x80 * 0 + 0x40 * 0 + ((10 << 2)|0)); //IR=0, IE=0, ILVL=10, GPX=0
U2C0_1IC = (0x80 * 0 + 0x40 * 1 + ((10 << 2)|1)); //IR=0, IE=1, ILVL=10, GPX=1
U2C0_2IC = (0x80 * 0 + 0x40 * 0 + ((10 << 2)|2)); //IR=0, IE=0, ILVL=10, GPX=2

///  ---------------------------------------------------------------------
///  7. Configuration of the U2C0 Channel Control Register
//   enable/disable general interrupt
//   enable/disable the parity mode
//   select the protocol mode
///  ---------------------------------------------------------------------
            // AIEN=0, RIEN=0,  TBIEN=0,  TSIEN=0,  DLIEN=0,  RSIEN=0
U2C0_CCR  = (0 << 15)|(0 << 14)|(0 << 13)|(0 << 12)|(0 << 11)|(0 << 10);
U2C0_CCR |= (0 << 8);  // PM=0(no parity generation)
U2C0_CCR |= 0x0002;    // active ASC mode

///  ---------------------------------------------------------------------
```

```
  ///  8. input/output pins configuration
  ///  ------------------------------------------------------------------------
  P3_IOCR01 = 0x0020;  // P3.1=input(pull_up)
  P3_IOCR00 = 0x00D0;  // P3.0=output(ALT1 open-drain)
}

//****************************************************************************
// @Function      void U2C0_ASC_Protocoll(void)
// @Date          15.11.2007
//****************************************************************************
_interrupt(U2C0_SRN1)  void U2C0_ASC_Protocoll(void)
{
  U2C0_PSCR   |= 0x0008;      // clear PSR_COL Collesion
  while(1);
}
```

5.5 Example 5

```
//****************************************************************************
// @Function      void U2C1_LIN_Slave_vInit(void)
// @Date          15.11.2007
//****************************************************************************
void U2C1_LIN_Slave_vInit(void)
{
  Sys_Protection(0);
  U2C1_KSCFG = 3;
  Sys_Protection(1);

  U2C1_CCR       =  0x0000;      // recommanded
  ///  ------------------------------------------------------------------------
  ///  1. baud rate generation and the PPP configuration
  ///  baud rate = 19200,
  ///  fasc=80000 x 1000 /(1024-STEP) / (PDIV+1) / (PCTQ+1) / (DCTQ+1) = 19231
  ///  ------------------------------------------------------------------------
  U2C1_FDRL=(((uword)1<<14)|(764<<0));       //DM=1(normal divider), STEP=764
  U2C1_BRGL=((15<<10)|(0<<8)|(0<<4)|(0<<0)); //DCTQ=15,PCTQ=0,PPPEN=0,CLKSEL=0
  U2C1_BRGH=((0<<14)|(0<<13)|(0<<0));        //SCLKCFG=0,MCLKCFG=0,PDIV=0

  ///  ------------------------------------------------------------------------
  ///  2. Configuration of the U2C1 Input Control Register and port pins
  ///  ------------------------------------------------------------------------
             // CM  SFSEL   DPOL   DSEN    DFEN   INSW   DESL
  U2C1_DX0CR=((0<<10)|(0<<9)|(0<<8)|(0<<6)|(0<<5)|(0<<4)|(1<<0)); //P3.7(DX0B)
  U2C1_DX1CR=((0<<10)|(0<<9)|(0<<8)|(0<<6)|(0<<5)|(0<<4)|(0<<0)); //P3.6 colliion

  ///  ------------------------------------------------------------------------
  ///  3. Data format: shift control signal, word/frame length control
  ///  ------------------------------------------------------------------------
        //TRM=01b(ASC)   DOCFG       PDL     SDIR=0(LSB first)
  U2C1_SCTRL = (1 << 8) | (0 << 6) | (1 << 1) | (0 << 0);
  U2C1_SCTRH = (7 << 8) | (7 << 0); //WLE=7, FLE=7;

  ///  ------------------------------------------------------------------------
  ///  4. data transmission control (TBUF single shot mode)
  ///  ------------------------------------------------------------------------
             // TDEN=1   TDSSM=1    FLEMD=0   WLEMD=0
  U2C1_TCSRL = (1 << 10) | (1 << 8) | (0 << 2) | (0 << 0);
  U2C1_TCSRH = 0x0000;

  ///  ------------------------------------------------------------------------
  //   5. protocol-related information
  // CTR[0]     SMD       sample mode
  // CTR[1]     STPB      stop bit
  // CTR[2]     IDM       idle detection mode
  // CTR[3]     SBIEN     Sync.-Break Interrupt Enable
  // CTR[4]     CDEN      Collision Detection Enable (for half-duplex mode)
```

```
  // CTR[5]     RNIEN     Receiver Noise Detection Interrupt Enable
  // CTR[6]     FEIEN     Format Error Interrupt Enable
  // CTR[7]     FFIEN     Frame Finished Interrupt Enable
  // CTR[8-12]  SP        Sample Position
  // CTR[13-15] PL       Pulse Length
  // CTR[31]    MCLK      master clock enable
  ///  ------------------------------------------------------------------------
          // PL   SP    FFIEN  FFIEN  RNIEN   CDEN  SBIEN   IDM    STPB   SMD
 U2C1_PCRL=(0<<13)|(7<<8)|(0<<7)|(0<<6)|(0<<5)|(1<<4)|(1<<3)|(0<<2)|(0<<1)|(0<<0);
  U2C1_PCRH=(0 << 15); // MCLK not used

  ///  ------------------------------------------------------------------------
  ///  6. Configuration of the interrupts point and interrupt control registers
  ///  ------------------------------------------------------------------------
             //AINP=SR0, RINP=SR0,  TBINP=SR0, TSINP=SR0
  U2C1_INPRL = (0 << 12) | (0 << 8) | (0 << 4) | (1 << 0);
  U2C1_INPRH = (1 << 0); //PINP=SR1 for collision detection and SBIR


  U2C1_0IC = (0x80 * 0 + 0x40 * 0 + ((10 << 2)|0)); //IR=0, IE=0, ILVL=10, GPX=0
  U2C1_1IC = (0x80 * 0 + 0x40 * 1 + ((10 << 2)|1)); //IR=0, IE=1, ILVL=10, GPX=1
  U2C1_2IC = (0x80 * 0 + 0x40 * 0 + ((10 << 2)|2)); //IR=0, IE=0, ILVL=10, GPX=2

  ///  ------------------------------------------------------------------------
  ///  7. Configuration of the U2C0 Channel Control Register
  //   enable/disable general interrupt
  //   enable/disable the parity mode
  //   select the protocol mode
  ///  ------------------------------------------------------------------------
             // AIEN=0, RIEN=0,  TBIEN=0,  TSIEN=0,  DLIEN=0,  RSIEN=0
  U2C1_CCR  = (0 << 15)|(0 << 14)|(0 << 13)|(0 << 12)|(0 << 11)|(0 << 10);
  U2C1_CCR |= (0 << 8);  // PM=0(no parity generation)
  U2C1_CCR |= 0x0002;    // active ASC mode

  ///  ------------------------------------------------------------------------
  ///  8. input/output pins configuration
  ///  ------------------------------------------------------------------------
  P3_IOCR07 = 0x0020;  // P3.7=input(pull_up)
  P3_IOCR06 = 0x00D0;  // P3.6=output(ALT1 open-drain)
}



/*****************************************************************************
// @Function      void U2C1_ASC_viProtocoll(void)
// @Date          15.11.2007
//*****************************************************************************
_interrupt(U2C1_SRN1)  void U2C1_ASC_Protocoll(void)
{
  uword uwTmp;
  if (U2C1_PSR & 0x0008) //  COL(bit3)=1 collision is detected
  {
    U2C1_PSCR |= 0x0008;
    uwState=5;  // LIN slave state: collision is detected
  }
  // LIN slave state machine actions upon synch break detection
  else if (U2C1_PSR & 0x0004)  // SBD=1(synch break detected)
  {
    U2C1_PSCR = 0xFFFF;  // clear all flags
     // set bit timing measuring mode
    U2C1_BRGL|= 0x0008 ;  // TMEN=1 (bit 3)
             // CM=2  SFSEL   DPOL  DSEN    DFEN  INSW    DESL
    U2C1_DX0CR=((2<<10)|(0<<9)|(0<<8)|(0<<6)|(0<<5)|(0<<4)|(1<<0));
    U2C1_CCR |= (1<<12);  // enable TSIEN
    uwState=1; // LIN slave state: synch. break is detected
  }
  // LIN slave sync field, baudrate measuring mode active
```

```
          // TMEN(bit3)==1 and TSIF(bit12)==1
  else if ((U2C1_BRGL & 0x0008) && (U2C1_PSR & 0x1000))
  {
    U2C1_PSCR = 0xffff;   // clear all flags
    if (uwTSIRCount == 0)
    {
      uwTSIRCount += 1;   // Start bit edge
    }
    else if (uwTSIRCount < 4)
    {
      uwPDIV[uwTSIRCount] = U2C1_BRGH & 0x03FF; // save PDIV
      uwTSIRCount += 1;
    }
    else
    {
      uwPDIV[uwTSIRCount] = U2C1_BRGH & 0x03FF; // save PDIV

      //check uwPDIV [uwTSIRCount]
      //the time between 2 dominant edges is 2 bit times
      uwTmp=(uwPDIV[uwTSIRCount]/2);
      uwTmp=uwTmp/(( U2C1_BRGL & 0xFC00)>>10);  //DTCQ
      U2C1_BRGH =  U2C1_BRGH & (~(uword)0x003F);
      U2C1_BRGH |= (uwTmp-1); //PFDIV

      // deactivate the baudrate measuring mode (reset TMEN)
      U2C1_BRGL &= ~((uword)0x0008);
      // disable the transmit shift interrupt (reset TSIEN)
      U2C1_CCR &= ~((uword)1<<12);

      uwState=2; // LIN slave state: baud rate is init.
    }
  }
  else
  {
    uwState=3;  // LIN slave state: undefined
  }
}
```

5.6 Example 6

```
//*****************************************************************************
// @Function:   void U1C1_SSC_MasterMode_init(void)
// DESCRIPTION: this function initializes a USIC module for SPI transfers
//              with a baudrate of 9600pbs from external osc with Fosc = 80MHz
//              module works as Master
// @Date:       15.11.2007
//*****************************************************************************
void U1C1_SSC_MasterMode_init(void)
{
  Sys_Protection(0);
  U1C1_KSCFG = 3;
  Sys_Protection(1);

  U1C1_CCR  = 0x0000;      // recommanded

  ///  ------------------------------------------------------------------------
  ///  1. baud rate generation and the PPP configuration
  ///  baud rate = 9766
  ///  ------------------------------------------------------------------------
  U1C1_FDRL = ((uword)2 << 14)|(1 << 0);  // DM=2 (fractional divider), STEP=1
              //DCTQ=15   PCTQ=1  CTQSEL=0  PPPEN=0  TMEN=0  CLKSEL=0
  U1C1_BRGL = (15 << 10)|(1 << 8)|(0 << 6)|(0 << 4)|(0 << 3)|(0 << 0);
              //SCLKCFG=0,MCLKCFG=0,PDIV=3
  U1C1_BRGH = (0 << 14)|(0 << 13)|(3 << 0);

  ///  ------------------------------------------------------------------------
```

```
  ///  2. Configuration of the U2C0 Input Control Register
  ///  ------------------------------------------------------------------------
              // CM   SFSEL   DPOL   DSEN   DFEN   INSW   DESL
  U1C1_DX0CR=((0<<10)|(0<<9)|(0<<8)|(0<<6)|(0<<5)|(1<<4)|(1<<0)); //P0.7(DX0B)

  ///  ------------------------------------------------------------------------
  ///  3. Data format: shift control signal, word/frame length control
  ///  ------------------------------------------------------------------------
         //TRM=01b(SSC)   DOCFG       PDL     SDIR=1(MSB first)
  U1C1_SCTRL = (1 << 8) | (0 << 6) | (1 << 1) | (1 << 0);
  U1C1_SCTRH = (15 << 8) | (15 << 0); //WLE=15, FLE=15;

  ///  ------------------------------------------------------------------------
  ///  4. data transmission control (TBUF single shot mode)
  ///  ------------------------------------------------------------------------
              // TDEN=1   TDSSM=1
  U1C1_TCSRL = (1 << 10) | (1 << 8);
  U1C1_TCSRH = 0x0000;

  ///  ------------------------------------------------------------------------
  //   5. protocol-related information
  // CTR[0]     MSLSEN     MSLS Enable
  // CTR[0]     SELCTR     Select Control
  // CTR[0]     SELINV     Select Inversion
  // CTR[0]     FEM        Frame End Mode
  // CTR[5:4]   CTQSEL1    Input Selection for CTQ
  // CTQ[7:6]   PCTQ1      Divide Factor to Generate fPDIV
  // CTQ[12:8]  DCTQ1      Divide Factor to Generate fTCTQ/RCTQ
  // CTR[14]    MSLSINEN   MSLS Interrupt Enable
  // CTR[15]    DX2CIEN    DX2C Interrupt Enable
  // CTR[23:16] SELO[7:0]  Select Output
  // CTR[24]    TIWEN      Enable Inter-World Delay Tiw
  // CTR[31]    MCLK       Start Stop of MCLK
  ///  ------------------------------------------------------------------------
             //DCTQ1   PCTQ1  CTQSEL1     FEM     SELINV  SELCTR   MSLSEN
  U1C1_PCRL=((15 << 8)|(1 << 6)|(0 << 4)|(0 << 3)|(1 << 2)|(1 << 1)|(1 << 0));
             //MCLK   TIMEN=0    SELO[7..0]
  U1C1_PCRH=(0 << 15)| (0 << 8) | (1 << 0); //no MCLK, no delay Tiw, SELO0 enable

  ///  ------------------------------------------------------------------------
  ///  6. Configuration of the interrupts point and interrupt control registers
  ///  ------------------------------------------------------------------------
             //AINP=SR2, RINP=SR2,  TBINP=SR1, TSINP=SR0
  U1C1_INPRL = (2 << 12) | (2 << 8) | (1 << 4) | (0 << 0);    // for MR+ AINP assinged to SR2
  U1C1_INPRH = (0 << 0); //PINP=SR0

  ///  ------------------------------------------------------------------------
  ///  7. Configuration of the U2C0 Channel Control Register
  //   enable/disable general interrupt
  //   enable/disable the parity mode
  //   select the protocol mode
  ///  ------------------------------------------------------------------------
           // AiEN=1, RIEN=1,  TBIEN=0,  TSIEN=0,  DLIEN=0,  RSIEN=1(uses TBINP)
 U1C1_CCR=(1 << 15)|(1 << 14)|(0 << 13)|(0 << 12)|(0 << 11)|(1 << 10); // for MR+ AI must be enabled
  U1C1_CCR |= (0 << 8);  // PM=0(no parity generation)
  U1C1_CCR |= 0x0001;    // active SSC mode

  ///  ------------------------------------------------------------------------
  ///  8. input/output pins configuration
  ///  master mode U1C1: P0.6=DOUT; P0.6=SLKOUT; P0.4=SELO; 0.7=MRST_IN
  ///  ------------------------------------------------------------------------
  P0_IOCR04  = 0x0090;  //ALT1 output(SELO0 U1C1)
  P0_IOCR05  = 0x0090;  //ALT1 output(SCLKOUT U1C1)
  P0_IOCR06  = 0x0090;  //ALT1 output for MTSR(DOUT U1C1)
  P0_IOCR07  = 0x0020;  //initiate pin as direct input, pull-up

}
```

5.7 Example 7

```
//*****************************************************************************
// @Function:   void U1C0_SSC_SlaveMode_init(void)
// DESCRIPTION: this function initializes a USIC module for SPI transfers
//              module works as slave
// @Date        15.11.2007
//*****************************************************************************
void U1C0_SSC_SlaveMode_init(void)
{

  Sys_Protection(0);
  U1C0_KSCFG = 3;
  Sys_Protection(1);

  ///  ------------------------------------------------------------------------
  ///  1. baud rate generation and the PPP configuration
  ///  --> not needed for slave mode
  ///  ------------------------------------------------------------------------


  ///  ------------------------------------------------------------------------
  ///  2. Configuration of the U2C0 Input Control Register
  ///  ------------------------------------------------------------------------
          // CM   SFSEL   DPOL   DSEN   DFEN   INSW   DESL
  U1C0_DX0CR=((0<<10)|(0<<9)|(0<<8)|(0<<6)|(0<<5)|(1<<4)|(0<<0)); //P0.0(DX0A)
  U1C0_DX1CR=((0<<10)|(0<<9)|(0<<8)|(0<<6)|(0<<5)|(1<<4)|(1<<0)); //P0.2(DX0B)
  U1C0_DX2CR=((0<<10)|(0<<9)|(1<<8)|(0<<6)|(0<<5)|(1<<4)|(0<<0)); //P0.3(DX0A)

 ///  ------------------------------------------------------------------------
  ///  3. Data format: shift control signal, word/frame length control
  ///  ------------------------------------------------------------------------
       //TRM=01b(SSC)   DOCFG       PDL     SDIR=1(MSB first)
  U1C0_SCTRL = (1 << 8) | (0 << 6) | (1 << 1) | (1 << 0);
  U1C0_SCTRH = (15 << 8) | (15 << 0); //WLE=15, FLE=15;

  ///  ------------------------------------------------------------------------
  ///  4. data transmission control (TBUF single shot mode)
  ///  ------------------------------------------------------------------------
           // TDEN=1   TDSSM=1
  U1C0_TCSRL = (1 << 10) | (1 << 8);
  U1C0_TCSRH = 0x0000;

  ///  ------------------------------------------------------------------------
  //   5. protocol-related information --> not needed for slave mode
  // CTR[0]     MSLSEN     MSLS Enable
  // CTR[0]     SELCTR     Select Control
  // CTR[0]     SELINV     Select Inversion
  // CTR[0]     FEM        Frame End Mode
  // CTR[5:4]   CTQSEL1    Input Selection for CTQ
  // CTQ[7:6]   PCTQ1      Divide Factor to Generate fPDIV
  // CTQ[12:8]  DCTQ1      Divide Factor to Generate fTCTQ/RCTQ
  // CTR[14]    MSLSINEN   MSLS Interrupt Enable
  // CTR[15]    DX2CIEN    DX2C Interrupt Enable
  // CTR[23:16] SELO[7:0]  Select Output
  // CTR[24]    TIWEN      Enable Inter-World Delay Tiw
  // CTR[31]    MCLK       Start Stop of MCLK
  ///  ------------------------------------------------------------------------


  ///  ------------------------------------------------------------------------
  ///  6. Configuration of the interrupts point and interrupt control registers
  ///  ------------------------------------------------------------------------
         //AINP=SR0, RINP=SR0,  TBINP=SR0, TSINP=SR0
  U1C0_INPRL = (0 << 12) | (0 << 8) | (0 << 4) | (0 << 0);
  U1C0_INPRH = (0 << 0); //PINP=SR0
```

```
  U1C0_0IC = (0x80 * 0 + 0x40 * 0 + ((8 << 2)|0));  //IR=0, IE=0, ILVL=8, GPX=0
  U1C0_1IC = (0x80 * 0 + 0x40 * 0 + ((8 << 2)|1));  //IR=0, IE=0, ILVL=8, GPX=1
  U1C0_2IC = (0x80 * 0 + 0x40 * 0 + ((8 << 2)|2));  //IR=0, IE=0, ILVL=8, GPX=2

  ///  ------------------------------------------------------------------------
  ///  7. Configuration of the U2C0 Channel Control Register
  //   enable/disable general interrupt
  //   enable/disable the parity mode
  //   select the protocol mode
  ///  ------------------------------------------------------------------------
              // AIEN=0, RIEN=0,  TBIEN=0,  TSIEN=0,  DLIEN=0,  RSIEN=0
  U1C0_CCR  = (0 << 15)|(0 << 14)|(0 << 13)|(0 << 12)|(0 << 11)|(0 << 10);
  U1C0_CCR |= (0 << 8);  // PM=0(no parity generation)
  U1C0_CCR |= 0x0001;    // active SSC mode

  ///  ------------------------------------------------------------------------
  ///  8. input/output pins configuration
  ///  slave mode U1C0: P0.0=MTSR_IN; P0.1=MRST(DOUT); P0.2=SCLK_IN; 0.3=SLS_IN
  ///  ------------------------------------------------------------------------
  P0_IOCR01  = 0x0090;  //ALT1 output(MTSR U1C0)
}
```

5.8 Example 10

```
//*****************************************************************************
// @Function:   void U1C1_SSC_Example10(void)
// DESCRIPTION: this function initializes a USIC module for SPI transfers
//              with a baudrate of 9600pbs from external osc with Fosc = 80MHz
//              write and read data from EEPROM
//              use End-of-frame handling is control by SCTRH.FLE
// @Date:       05.12.2008
//*****************************************************************************
void U1C1_SSC_Example10(void)// Test1: FEM=0, interrupt is disabled, polling system
{
  uword uwTmp;
  U1C1_SSC_MasterMode_init();
            //SCLKCFG=2(EEPROM) MCLKCFG=0   PDIV=3
  U1C1_BRGH = (uword)((2 << 14)|(0 << 13)|(3 << 0));   // for EEPROM SCLKCFG=10B
  // for EEPROM the 'read data' instrcution must be not interrupted (MSEL must be kepted low),
otherwise the received data will be always 0xFF
  // use FEM=1 The MSLS signal is kept active also while no new data is available and no other end
of frame conditionis reached.
  // in this case the debug tool single step can be used
            //DCTQ1   PCTQ1  CTQSEL1     FEM     SELINV  SELCTR   MSLSEN
  U1C1_PCRL=((15 << 8)|(1 << 6)|(0 << 4)|(1 << 3)|(1 << 2)|(1 << 1)|(1 << 0));

  // write 'write enable'
  U1C1_SCTRH = (7 << 8) | (7 << 0); // WLE=8, FLE=8;
  U1C1_TBUF00=EEPROM_CM_WREN; // write enable
  while(!(U1C1_PSR & 0xC000));    // AIF is not defined in XC2000
  U1C1_PSCR     |= 0xC000;          // clear receive indication flag
  uwTmp=U1C1_RBUF;

  // write 'read status'
  U1C1_SCTRH = (15 << 8) | (15 << 0);//WLE=15, FLE=15;
  U1C1_TBUF00=(EEPROM_CM_READ_STATUS << 8 | EEPROM_WRITE_DUMMY);
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;          // clear receive indication flag

  uwEEPROM_readStatus=(U1C1_RBUF & 0x00FF);
  while  (uwEEPROM_readStatus !=0x0002); // if 'write not enabled'


  // write 'write 0x55 into address 0x0000'
```

```
  U1C1_SCTRH = (7 << 8) | (31 << 0); //WLE=8, FLE=32;
  U1C1_TBUF00=EEPROM_CM_WRITE;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag

  U1C1_TBUF00=EEPROM_ADDRESS_HIGH;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag

  U1C1_TBUF00=EEPROM_ADDRESS_LOW;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag

  U1C1_TBUF00=ubEEPROM_Write[3];
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag

  // write 'read data from address 0x0000'
  U1C1_SCTRH = (7 << 8) | (31 << 0); //WLE=8, FLE=32;
  U1C1_TBUF00=EEPROM_CM_READ;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag

  U1C1_TBUF00=EEPROM_ADDRESS_HIGH;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag

  U1C1_TBUF00=EEPROM_ADDRESS_LOW;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag

  U1C1_TBUF00=EEPROM_WRITE_DUMMY;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag

  // USIC has a double receive buffer structure and
  // to simplify the use of this feature register RBUF is implemented.
  // a read action from RBUF delivers the first received data
  uwTmp=U1C1_RBUF;
  uwEEPROM_readData=U1C1_RBUF; // 0x55 can only be read out here!
  _nop();
  ubEEPROM_Write[3]++;
}
```

5.9 Example 11

```
//*****************************************************************************
// @Function:   void U1C1_SSC_Example11(void)
// DESCRIPTION: this function initializes a USIC module for SPI transfers
//              with a baudrate of 9600pbs from external osc with Fosc = 80MHz
//              write and read data from EEPROM
//              use SW-based start of frame indication SOF
// @Date:       05.12.2008
//*****************************************************************************
void U1C1_SSC_Example11(void)
{
  uword uwTmp;
  U1C1_SSC_MasterMode_init();
            //SCLKCFG=2(EEPROM) MCLKCFG=0   PDIV=3
  U1C1_BRGH = (uword)((2 << 14)|(0 << 13)|(3 << 0));   // for EEPROM SCLKCFG=10B

  // write 'write enable'
  U1C1_SCTRH = ((7 << 8) | (63 << 0));
  U1C1_TCSRL |= 0x0020; // SOF is set
  U1C1_TBUF00=EEPROM_CM_WREN;
```

```
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;
  uwTmp=U1C1_RBUF;

  // write 'read status'
  U1C1_SCTRH = ((15 << 8) | (63 << 0));
  U1C1_TCSRL |= 0x0020; // SOF is set
  U1C1_TBUF00=(EEPROM_CM_READ_STATUS << 8 | EEPROM_WRITE_DUMMY);
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;

  uwEEPROM_readStatus=(U1C1_RBUF & 0x00FF);
  while  (uwEEPROM_readStatus !=0x0002); // if 'write not enabled'

  // write 'write 0x55 into address 0x0000'
  U1C1_SCTRH = ((7 << 8) | (63 << 0));
  U1C1_TCSRL |= 0x0020; // SOF is set
  U1C1_TBUF00=EEPROM_CM_WRITE;
  while(!(U1C1_PSR & 0xC000));
  //U1C1_TCSRL &= ~0x0020; // due to the bug USIC_AI.003 bit SOF must be cleared in Xc2000 MR
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag
  U1C1_TBUF00=EEPROM_ADDRESS_HIGH;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag
  U1C1_TBUF00=EEPROM_ADDRESS_LOW;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag
  U1C1_TBUF00=ubEEPROM_Write[3];
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag

  // write 'read data from address 0x0000'
  U1C1_TCSRL |= 0x0020; // SOF is set
  U1C1_TBUF00=EEPROM_CM_READ;
  while(!(U1C1_PSR & 0xC000));
  //U1C1_TCSRL &= ~0x0020; // due to the bug USIC_AI.003 bit SOF must be cleared in Xc2000 MR
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag

  U1C1_TBUF00=EEPROM_ADDRESS_HIGH;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag

  U1C1_TBUF00=EEPROM_ADDRESS_LOW;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag

  U1C1_TBUF00=EEPROM_WRITE_DUMMY;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag

  // USIC has a double receive buffer structure and
  // to simplify the use of this feature register RBUF is implemented.
  // a read action from RBUF delivers the first received data
  uwTmp=U1C1_RBUF;          // break point before uwTmp, RBUFSR.DS=1 (also from RBUF1)
  uwEEPROM_readData=U1C1_RBUF; // 0x55 can only be read out here!
  _nop();
  ubEEPROM_Write[3]++;
}
```

5.10 Example 12

```
//*****************************************************************************
// @Function:   void U1C1_SSC_Example12(void)
// DESCRIPTION: this function initializes a USIC module for SPI transfers
//              with a baudrate of 9600pbs from external osc with Fosc = 80MHz
```

```
//              write and read data from EEPROM
//              use explicit end of frame by software (reset bit_0 of register PSCR)
// @Date:       05.12.2008
//*****************************************************************************
void U1C1_SSC_Example12(void)
{
  uword uwTmp;
  U1C1_SSC_MasterMode_init();
             //SCLKCFG=2(EEPROM) MCLKCFG=0   PDIV=3
  U1C1_BRGH = (uword)((2 << 14)|(0 << 13)|(3 << 0));   // for EEPROM SCLKCFG=10B

  // write 'write enable'
  U1C1_SCTRH = ((7 << 8) | (63 << 0));
  U1C1_TBUF00=EEPROM_CM_WREN;
  U1C1_PSCR |= 0x0001; // resetMSEL
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag
  uwTmp=U1C1_RBUF;

  while((U1C1_PSR & 0x0001));  //wait untill SPI in idle (MSEL is deactivated) before to configure
WLE and FLE

  // write 'read status'
  U1C1_SCTRH = ((15 << 8) | (63 << 0));
  U1C1_TBUF00=(EEPROM_CM_READ_STATUS << 8 | EEPROM_WRITE_DUMMY);
  U1C1_PSCR |= 0x0001; // resetMSEL
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag

  uwEEPROM_readStatus=(U1C1_RBUF & 0x00FF);
  while  (uwEEPROM_readStatus !=0x0002); // if 'write not enabled'

 while((U1C1_PSR & 0x0001));  // ait untill SPI in idle (MSEL is deactivated), before to configure
WLE and FLE

  // write 'write 0x55 into address 0x0000'
  U1C1_SCTRH = ((7 << 8) | (31 << 0));
  U1C1_TBUF00=EEPROM_CM_WRITE;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag
  U1C1_TBUF00=EEPROM_ADDRESS_HIGH;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag
  U1C1_TBUF00=EEPROM_ADDRESS_LOW;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag
  U1C1_TBUF00=ubEEPROM_Write[3];
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag

  // write 'read data from address 0x0000'
  U1C1_TBUF00=EEPROM_CM_READ;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag
  U1C1_TBUF00=EEPROM_ADDRESS_HIGH;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag
  U1C1_TBUF00=EEPROM_ADDRESS_LOW;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag
  U1C1_TBUF00=EEPROM_WRITE_DUMMY;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag

  // USIC has a double receive buffer structure and
  // to simplify the use of this feature register RBUF is implemented.
  // a read action from RBUF delivers the first received data
```

```
  uwTmp=U1C1_RBUF;
  uwEEPROM_readData=U1C1_RBUF; // 0x55 can only be read out here!
  _nop();
  ubEEPROM_Write[3]++;
}
```

5.11 Example 13

```
//*****************************************************************************
// @Function:   void U1C1_SSC_Example13(void)
// DESCRIPTION: this function initializes a USIC module for SPI transfers
//              with a baudrate of 9600pbs from external osc with Fosc = 80MHz
//              write and read data from EEPROM
//              use automatic update by data handler via WLEMD=1
// @Date:       05.12.2008
//*****************************************************************************
void U1C1_SSC_Example13(void)
{
  uword uwTmp;
  U1C1_SSC_MasterMode_init();
            //SCLKCFG=2(EEPROM) MCLKCFG=0   PDIV=3
  U1C1_BRGH = (uword)((2 << 14)|(0 << 13)|(3 << 0));   // for EEPROM SCLKCFG=10B
              // TDEN=1   TDSSM=1  WLEMD=1
  U1C1_TCSRL = (1 << 10) | (1 << 8) | (1 << 0);
  U1C1_SCTRH = (0 << 8) | (63 << 0);  //for WLEMD=1, FLW must be set to max.

  // write 'write enable'
  U1C1_TBUF23=EEPROM_CM_WREN; // 8 bit word and then frame will be finished
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag
  uwTmp=U1C1_RBUF;

  // write 'read status'
  U1C1_TBUF31=(EEPROM_CM_READ_STATUS << 8 | EEPROM_WRITE_DUMMY);// 16 bit word and then frame will
be finished
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag

  uwEEPROM_readStatus=(U1C1_RBUF & 0x00FF);
  while  (uwEEPROM_readStatus !=0x0002); // if 'write not enabled'

  // 'write 0x55 into address 0x0000'
  U1C1_TBUF07=EEPROM_CM_WRITE; // 8 bit word and then frame will not be finished
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag
  U1C1_TBUF07=EEPROM_ADDRESS_HIGH;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag
  U1C1_TBUF07=EEPROM_ADDRESS_LOW;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag
  U1C1_TBUF23=ubEEPROM_Write[3];
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag

  // 'read data from address 0x0000'
  U1C1_TBUF07=EEPROM_CM_READ;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag
  U1C1_TBUF07=EEPROM_ADDRESS_HIGH;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag
  U1C1_TBUF07=EEPROM_ADDRESS_LOW;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag
```

```
  U1C1_TBUF23=EEPROM_WRITE_DUMMY;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xC000;   //  clear receive indication flag

  // USIC has a double receive buffer structure and
  // to simplify the use of this feature register RBUF is implemented.
  // a read action from RBUF delivers the first received data
  uwTmp=U1C1_RBUF;
  uwEEPROM_readData=U1C1_RBUF; // 0x55 can only be read out here!
  _nop();
  ubEEPROM_Write[3]++;
}
```

5.12 Example 14

```
//*****************************************************************************
// @Function:   void U1C1_SSC_Example14(void)
// DESCRIPTION: this function initializes a USIC module for SPI transfers
//              with a baudrate of 9600pbs from external osc with Fosc = 80MHz
//              write and read data from EEPROM
//              use PECx in EEPROM data transfer process.
// @Date:       05.12.2008
//*****************************************************************************
void U1C1_SSC_Example14(void)
{
  U1C1_SSC_MasterMode_init();
            //SCLKCFG=2(EEPROM) MCLKCFG=0   PDIV=3
  U1C1_BRGH = (uword)((2 << 14)|(0 << 13)|(3 << 0));

  U1C1_1IC = 0;
  U1C1_2IC = 0;

  //---- write 'write enable'
  U1C1_SCTRH = (7 << 8) | (7 << 0);
  U1C1_TBUF00=EEPROM_CM_WREN;
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xFC00;   //  clear receive indication flag
  uwTmpTest[0]=U1C1_RBUF;

  //---- write 'read status'
  U1C1_SCTRH = (15 << 8) | (15 << 0);
  U1C1_TBUF00=(EEPROM_CM_READ_STATUS << 8 | EEPROM_WRITE_DUMMY);
  while(!(U1C1_PSR & 0xC000));
  U1C1_PSCR     |= 0xFC00;   //  clear receive indication flag

  uwEEPROM_readStatus=(U1C1_RBUF & 0x00FF);
  while  (uwEEPROM_readStatus !=0x0002); // if 'write not enabled'

  //---- 'write data into address 0x0000' use PEC3 and PEC2
  uwU1C1_1IR_End = 0;
  uwU1C1_2IR_End = 0;
  U1C1_SCTRH = (7 << 8) | (31 << 0);

  U1C1_1IC = (0x80 * 0 + 0x40 * 1 + ((14 << 2)|2));  //IR=0, IE=0, ILVL=14, GPX=2  --> PEC2
(SRIEN)
  U1C1_2IC = (0x80 * 0 + 0x40 * 1 + ((14 << 2)|3));  //IR=0, IE=0, ILVL=14, GPX=3  --> PEC3
(RIEN)

  SRCP2  =  _sof(&ubEEPROM_Write[0]);       // PEC2: write EEPROM_Write data to TBUF
  DSTP2  =  _sof(&U1C1_TBUF00);
  PECSEG2  =  ((_seg(&ubEEPROM_Write[0])<<8) | _seg(&U1C1_TBUF00)) ; //set destination pointer
segment   S|D
  PECC2 = ((INC_10 << 9) | (WT_BYTE << 8) | (COUNT_4 << 0));

  DSTP3  =  _sof(&ubEEPROM_Write_Temp[0]);
```

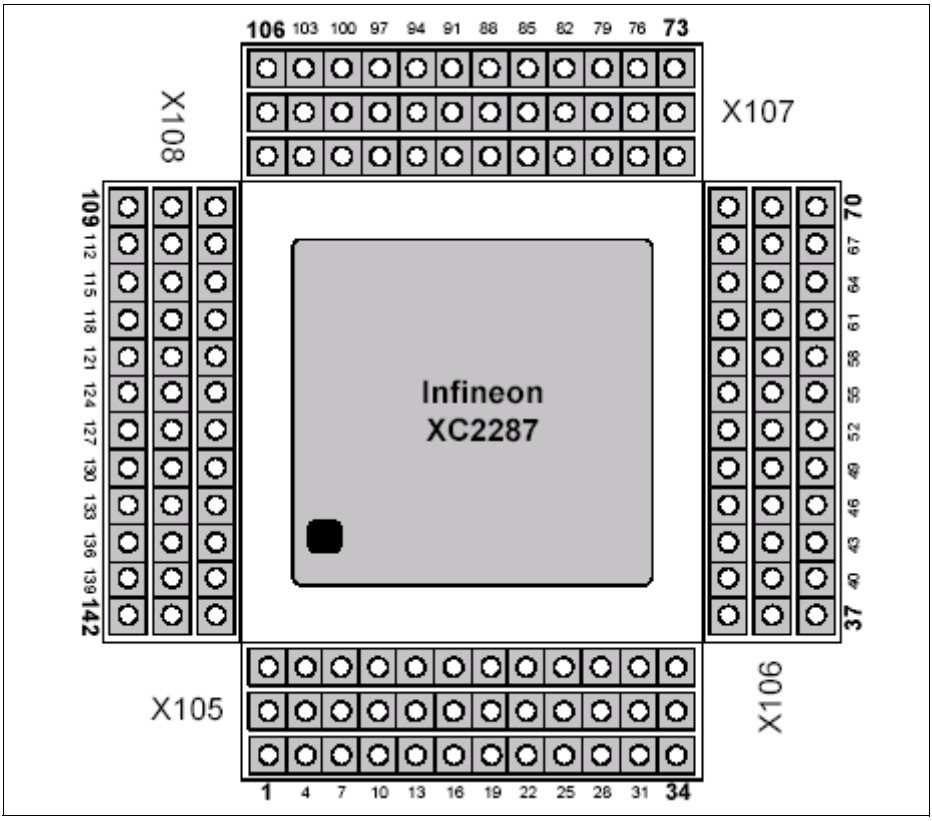
```
  SRCP3  =  _sof(&U1C1_RBUF);
  PECSEG3  =  (_seg(&U1C1_RBUF) <<8) | (_seg(&ubEEPROM_Write_Temp[0])) ; //set destination pointer
segment S|D
  PECC3 = ((INC_01 << 9) | (WT_BYTE << 8) | (COUNT_4 << 0));

  U1C1_1IC_IR=1;// generate interrupt request, data is sent out via PEC
  while(!uwU1C1_1IR_End);
  uwU1C1_1IR_End=0;

  while(!uwU1C1_2IR_End);
  uwU1C1_2IR_End=0;

  //---- 'read data from address 0x0000' use PEC3 and PEC2
  U1C1_1IC = (0x80 * 0 + 0x40 * 1 + ((14 << 2)|2));  //IR=0, IE=0, ILVL=14, GPX=2  --> PEC2
(SRIEN)
  U1C1_2IC = (0x80 * 0 + 0x40 * 1 + ((14 << 2)|3));  //IR=0, IE=0, ILVL=14, GPX=3  --> PEC3
(RIEN)

  SRCP2  =  _sof(&ubEEPROM_Read[0]);       // PEC2: write EEPROM_Read data to TBUF
  DSTP2  =  _sof(&U1C1_TBUF00);
  PECSEG2  =  ((_seg(&ubEEPROM_Read[0])<<8) | _seg(&U1C1_TBUF00)) ;
  PECC2 = ((INC_10 << 9) | (WT_BYTE << 8) | (COUNT_4 << 0));

  DSTP3  =  _sof(&ubEEPROM_Read_Temp[0]);
  SRCP3  =  _sof(&U1C1_RBUF);
  PECSEG3  =  (_seg(&U1C1_RBUF) <<8) | (_seg(&ubEEPROM_Read_Temp[0])) ; //set destination pointer
segment S|D
  PECC3 = ((INC_01 << 9) | (WT_BYTE << 8) | (COUNT_4 << 0));

  U1C1_1IC_IR=1;       // generate interrupt request, data is sent out via PEC
  while(!uwU1C1_1IR_End);
  uwU1C1_1IR_End=0;

  while(!uwU1C1_2IR_End);
  uwU1C1_2IR_End=0;

  uwEEPROM_readData=ubEEPROM_Read_Temp[3];
  _nop();
  ubEEPROM_Write[3]++;
  U1C1_PSCR     |= 0xFC00;   //  clear receive indication flag

}

#define U1C1_SRN1               0x5A
#define U1C1_SRN2               0x5B

interrupt (U1C1_SRN1) void SPI_viRSIENx(void) // defined for FEC transfer test3 and test4 (RSIEN
uses TBINP)
{
  while(!U1C1_1IC_IR);// wait untill the 2. word has been received
  U1C1_1IC_IE =0;       // must be cleared, so that no further PEC transfer can be generated
  uwSRN1_EEPROM++;
  uwU1C1_1IR_End=1;
}

interrupt (U1C1_SRN2) void SPI_viRIENx(void) // defined for FEC transfer test4 (RIEN uses RBINP)
{
  uwSRN2_EEPROM++;
  uwU1C1_2IR_End=1;
}
```

5.13 Example 15

```
//*****************************************************************************
// @Function:   void U1C1_SSC_Example15(void)
// DESCRIPTION: this function initializes a USIC module for SPI transfers
```

```
//              with a baudrate of 9600pbs from external osc with Fosc = 80MHz
//              demo SW for Multiple MSLS Output Signals Generation
//              automatic MSLS signals update control
// @Date:       05.12.2008
//*****************************************************************************
void U1C1_SSC_Example15(void)
{
  uword uwTmp;
  U1C1_SSC_MasterMode_init();
              //DCTQ1   PCTQ1  CTQSEL1     FEM     SELINV  SELCTR   MSLSEN
  U1C1_PCRL =((15 << 8)|(1 << 6)|(0 << 4)|(0 << 3)|(1 << 2)|(1 << 1)|(1 << 0));
              //MCLK   TIMEN=0    SELO[7..0]
  U1C1_PCRH =(0 << 15)| (0 << 8) | (0 << 0); //no MCLK, no delay Tiw, SELOx don't care
             // TDEN=1   TDSSM=1   SELMOD WLEMOD
  U1C1_TCSRL = (1 << 10) | (1 << 8) | (1 << 1) | (0 << 0);

  U1C1_TBCTRH    =  0x0100; // TBUF size must be set to >0, so that the U1C1_INx can be used


  P0_IOCR04  = 0x0090;  //ALT1 output(SELO0 U1C1)    --> SELO0 ALT1
  P0_IOCR05  = 0x0090;  //ALT1 output(SCLKOUT U1C1)
  P0_IOCR06  = 0x0090;  //ALT1 output for MTSR(DOUT U1C1)
  P0_IOCR07  = 0x0020;  //initiate pin as direct input, pull-up
  P0_IOCR03  = 0x00A0;  //ALT2 output(SELO1 U1C1) -->SELO1  ALT2

  U1C1_SCTRH = (7 << 8) | (7 << 0);
  //U1C1_TBUF01 = 0x55;// SELO0 is activated
  U1C1_IN01 = 0x55;// SELO0 is activated
  for (uwTmp=0; uwTmp<0xFFF; uwTmp++);

  //U1C1_TBUF02 = 0x55;// SELO1 is activated
  U1C1_IN02 = 0x55;// SELO1 is activated
  for (uwTmp=0; uwTmp<0xFFF; uwTmp++);

  //U1C1_TBUF03 = 0x55;// SELO0 and SELO1 are activated
  U1C1_IN03 = 0x55;// SELO0 and SELO1 are activated
  for (uwTmp=0; uwTmp<0xFFF; uwTmp++);

}
```

6 Appendix: Pin Connection of the XC2200 Starter Kit



w w w . i n f i n e o n . c o m

Published by Infineon Technologies AG