OMAP5912 Multimedia Processor Serial Interfaces Reference Guide

Literature Number: SPRU760C
February 2006

OMAP™
TEXAS INSTRUMENTS TECHNOLOGY

TEXAS INSTRUMENTS

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

Preface

Read This First

About This Manual

This document describes the serial interfaces of the OMAP5912 multimedia processor.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

Related Documentation From Texas Instruments

Documentation that describes the OMAP5912 device, related peripherals, and other technical collateral, is available in the OMAP5912 Product Folder on TI's website: www.ti.com/omap5912.

Trademarks

OMAP and the OMAP symbol are trademarks of Texas Instruments.

Contents

Figures

Tables

This page is intentionally left blank.

Serial Interfaces

This document describes the serial interfaces of the OMAP5912 multimedia processor.

1 SPI Master/Slave

1.1 Functional Description

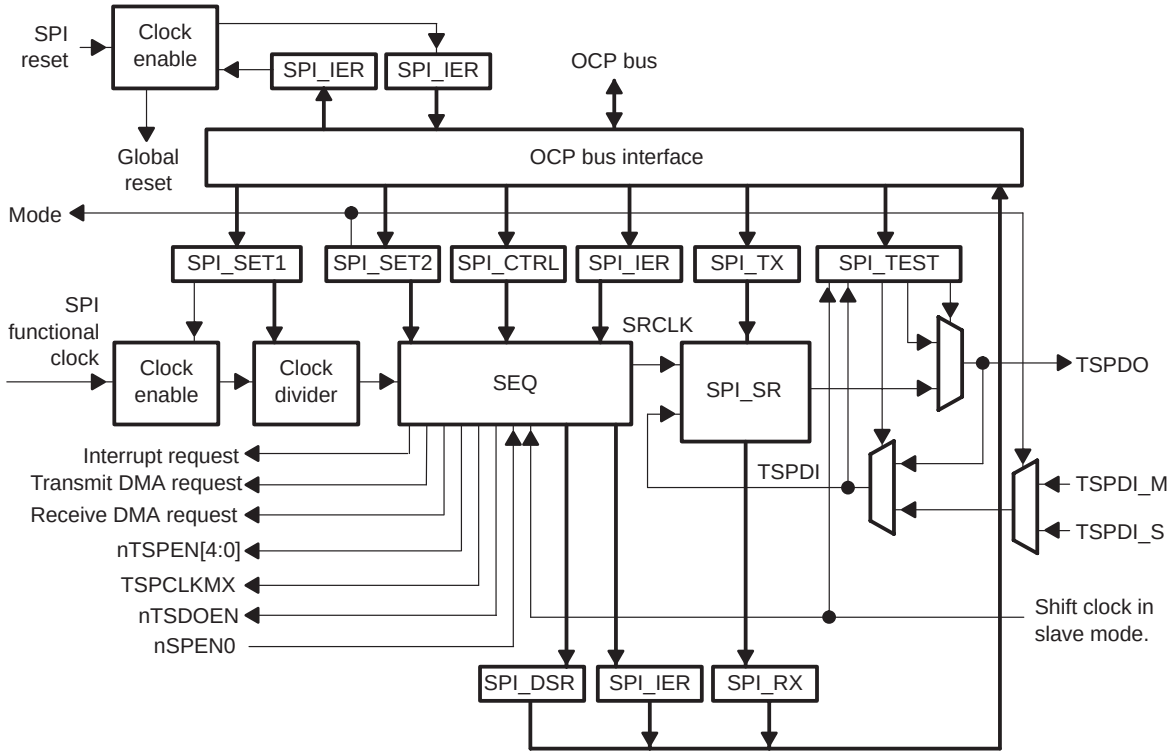
The serial interface is a bidirectional, four-line interface dedicated to the transfer of data to and from external devices offering a four-line serial interface. The four-line interface consists of:

- The clock used to shift data in and out
- The device enable
- The data input
- The data output

This serial port is based on a looped shift register, thus allowing both transmit (PISO) and receive (SIPO) modes. It can operate in master or in slave mode using MCU-DSP or DMA protocol. It supports up to five serial devices. The serial data is sent MSB first.

The serial port is fully controlled by the OCP bus (data write, data read, and activation of serialization operations).

Figure 1. SPI Master/Slave Block Diagram



1.2 Interface

Table 1 describes the SPI interface.

Table 1. SPI Interface

SPI Integration		
SPIF.SCK	Shift register clock Output in master mode Input in slave mode	I/O
SPIF.DIN	Serial data Input in master mode Serial data output in slave mode	I/O
SPIF.DOUT	Serial data output in master mode Serial data Input in slave mode Note: this signal is always driven by OMAP5912 regardless of master or slave mode.	I/O
$\overline{\text{SPIF.CS[0]}}$	SPI chip-select 0 Configured in output in master mode Configured in input in slave mode	I/O
$\overline{\text{SPIF.CS[3:1]}}$	External SPI chip-selects in master mode	O

1.3 SPI Registers

Start address: FFFB 0C00

Address of one register: Start address + offset address

Access supported: 16 or 32 bits

SPI uses little-endian addressing scheme.

The SPI offers input and output registers for loading data to serialize (transmit) and reading received data (receive).

Table 2. SPI Registers

Base Address = 0xFFFB 0C00				
Mnemonic	**Register Name**	**Size (Bits)**	**Access**	**Offset**
SPI_REV	Identification register	32	R	0x000
	Reserved			0x004
	Reserved			0x008
	Reserved			0x00C
SPI_SCR	System configuration register	32	R/W	0x010
SPI_SSR	System status register	32	R	0x014
SPI_ISR	Interrupt status register	32	R/W	0x018
SPI_IER	Interrupt enable register	32	R/W	0x01C
	Reserved			0x020
SPI_SET1	Set up 1 register	32	R/W	0x024
SPI_SET2	Set up 2 register	32	R/W	0x028
SPI_CTRL	Control register	32	R/W	0x02C
SPI_DSR	Data status register	32	R	0x030
SPI_TX	Transmit register	32	R/W	0x034
SPI_RX	Receive register	32	R	0x038
SPI_TEST	Test register	32	R/W	0x03C

Table 3. Identification Register Bit Description (SPI_REV—0x000)

Base Address = 0xFFFB 0C00, Offset = 0x00			
Bit	**Name**	**Function**	**Access**
31:8	Reserved	A read access returns 0.	R
7:0	REV	Revision number The 4-bit LSB indicates a minor revision. The 4-bit MSB indicates a major revision. Ex: 0x10 → version 1.0.	R

A write to this register has no effect.

A reset has no effect on the value returned.

Table 4. System Configuration Register Bit Description (SPI_SCR—0x010)

Base Address = 0xFFFB 0C00, Offset = 0x10				
Bit	**Name**	**Function**	**Access**	**Reset**
31:5	Reserved	A read access returns 0.	R	0x0000000
4:3	IDLEMODE	Power management, req/ack control 00: Force-idle. An idle request is acknowledged unconditionally. 01: No idle. An idle request is never acknowledged. 10: Smart idle. An idle request is acknowledged based on the internal activity of the module. 11: Reserved: Do not use.	R/W	00
2	EN AWAKEUP	Wake-up feature control 0: Wake-up is disabled. 1: Wake-up capability is enabled.	R/W	0
1	SOFTRESET	Software reset. Set this bit to 1 to trigger an SPI reset. This bit is automatically reset by hardware. Writing a 0 has no effect. During reads, it always returns 0. 0: Normal mode. 1: The module is reset.	R/W	0
0	AUTOIDLE	Internal OCP clock gating strategy 0: OCP clock is free-running. 1: Automatic OCP clock gating strategy is applied, based on the OCP interface activity.	R/W	0

This register allows control of the OCP interface parameters.

Table 5. System Status Register Bit Description (SPI_SSR—0x014)

This register provides status information about the reset.

Base Address = 0xFFFB 0C00, Offset = 0x14				
Bit	**Name**	**Function**	**Access**	**Reset**
31:1	Reserved	A read access returns 0.	R	0x00000000
0	RESETDONE	Internal reset monitoring 0: Internal module reset is ongoing. 1: Reset completed.	R	0

Note: Before accessing or using the module the local host must ensure that internal reset is released by reading the system status register (SPI_SSR).

Table 6. Interrupt Status Register Bit Description (SPI_ISR—0x018)

Base Address = 0xFFFB 0C00, Offset = 0x18				
Bit	**Name**	**Function**	**Access**	**Reset**
31:5	Reserved	A read access returns 0.	R	0x0000000
4	WAKEUP	Wake-up: Active high.	R/W	0
3	TX_UNDERFLOW	Transmit underflow: Active high.	R/W	0
2	RX_OVERFLOW	Receive overflow: Active high.	R/W	0
1	WE	Write end: Active high. Serialization complete.	R/W	0
0	RE	Read end: Active high. Receive register updated.	R/W	0

The interrupt status register is used to qualify the interrupt. Writing a 1 to the corresponding status bit releases the interrupt. Writing a 0 has no effect.

Table 7. Interrupt Enable Register Bit Description (SPI_IER—0x01C)

Base Address = 0xFFFB 0C00, Offset = 0x1C				
Bit	**Name**	**Function**	**Access**	**Reset**
31:5	Reserved	A read access returns 0.	R	0x0000000
4	MSK4	Enable interrupt when wake up 0: Interrupt disabled. 1: Interrupt active.	R/W	0
3	MSK3	Enable interrupt when TX underflow 0: Interrupt disabled. 1: Interrupt active.	R/W	0
2	MSK2	Enable interrupt when RX overflow 0: Interrupt disabled. 1: Interrupt active.	R/W	0
1	MSK1	Enable interrupt for write cycle 0: Interrupt disabled. 1: Interrupt active.	R/W	0
0	MSK0	Enable interrupt for read cycle 0: Interrupt disabled. 1: Interrupt active.	R/W	0

Table 8. Set Up SPI 1 Register Bit Description (SPI_SET1—0x024)

Base Address = 0xFFFB 0C00, Offset = 0x24				
Bit	**Name**	**Function**	**Access**	**Reset**
31:6	Reserved	A read access returns 0.	R	0x0000000
5	DMA_EN	Defines the protocol 0: MCU-DSP protocol. 1: DMA protocol.	R/W	0
4:1	PTV	The prescale time value (2^{PTV}) generates the shift register clock in master mode. (TSPCLKMX). The following scale values are supported: 0000: 1 0001: 2 0010: 4 0011: 8 0100: 16 0101: 32 0110: 64 0111: 128 1000: 256 1001: 512 1010: 1024 1011: 2048 Others: 4096	R/W	0000
0	EN_CLK	SPI functional clock enable 0: Clock is shut off. 1: Clock is running.	R/W	0

Note: A write access to this register during a transaction does not affect the register and activates an OCP error.

SPI_SET1 is dedicated to the configuration of the serial port interface.

SPI_SET2 is dedicated to the configuration of the serial port interface.

Table 9. Set Up SPI 2 Register Bit Description (SPI_SET2—0x028)

Base Address = 0xFFFB 0C00, Offset = 0x28				
Bit	**Name**	**Function**	**Access**	**Reset**
31:16	Reserved	A read access returns 0.	R	0x0000
15	MODE	0: Slave mode 1: Master mode	R/W	0
14:10	CP	Clock phase The shift register clock begins toggling at: 0: The middle of the data transfer 1: The beginning of the data transfer Bit 10 qualifies the access on device 0. Bit 11 qualifies the access on device 1. Bit 12 qualifies the access on device 2. Bit 13 qualifies the access on device 3. Bit 14 qualifies the access on device 4.	R/W	00000
9:5	CE	Shift register enable Active level of the shift register enable 0: The active level is low. 1: The active level is high. Bit 5 qualifies the access on device 0. Bit 6 qualifies the access on device 1. Bit 7 qualifies the access on device 2. Bit 8 qualifies the access on device 3. Bit 9 qualifies the access on device 4.	R/W	00000
4:0	CI	Clock invert Inactive edge of the shift register clock 0: The inactive state of the clock is low. 1: The inactive edge of the clock is high. Bit 0 qualifies the access on device 0. Bit 1 qualifies the access on device 1. Bit 2 qualifies the access on device 2. Bit 3 qualifies the access on device 3. Bit 4 qualifies the access on device 4.	R/W	00000

Notes:
1) In slave mode, the end of a transaction is detected based on CE field configuration. If CEi = 0, the end of a transaction is detected on the rising edge of nSPEN0. If CEi = 1, the end of a transaction is detected on the falling edge of nSPEN0. For more information on the different configurations, see Section 14.1.4.4.
2) In slave mode, all CIi bits must have the same value. This value depends on the inactive edge of the master clock.
3) In slave mode, all CEi bits must have the same value. This value depends on the active level of the master enable.
4) In slave mode, all CPi bits must have the same value. This value depends on the master clock phase.
5) A write access to this register during a transaction does not affect the register and activates an OCP error. The delay between a write in the SET2 register and the CI/CE bits taking effect is 2 x ARMXOR_CK cycles.

Table 10. Control SPI Register Bit Description (SPI_CTRL—0x02C)

Base Address = 0xFFFB 0C00, Offset = 0x2C				
Bit	**Name**	**Function**	**Access**	**Reset**
31:10	Reserved	A read access returns 0.	R	0x000000
9:7	AD	Index of the addressed device 000: Enable device 0 001: Enable device 1 010: Enable device 2 011: Enable device 3 100: Enable device 4 Others: Undefined	R/W	000
6:2	NB	Number of bits to receive/transmit 00000: 1 bit receive/transmit 11111: 32 bits receive/transmit	R/W	00000
1	WR	Write process activation− active high	R/W	0
0	RD	Read and write process activation− active high	R/W	0

Note: A write access to this register during a transaction does not affect the register and activates an OCP error.

This register is dedicated to the activation of the serial port interface. It defines:

- ❏ Read activation of the serial port
- ❏ Write activation of the serial port
- ❏ Number of bits to transfer (in the range 1 to 32)
- ❏ External device address (up to 5)

Table 11. Shift Status Register Bit Description (SPI_DSR—0x030)

Base Address = 0xFFFB 0C00, Offset = 0x30				
Bit	**Name**	**Function**	**Access**	**Reset**
31:2	Reserved	A read access returns 0.	R	0x0000000
1	TX_EMPTY	Shift register is empty: Active high This bit is cleared when the transmit register (SPI_TX) has been written (in functional or emulation mode).	R	1
0	RX_FULL	Receive register is full: Active high This bit is cleared when the receive register (SPI_RX) has been read (not cleared from debugger read).	R	0

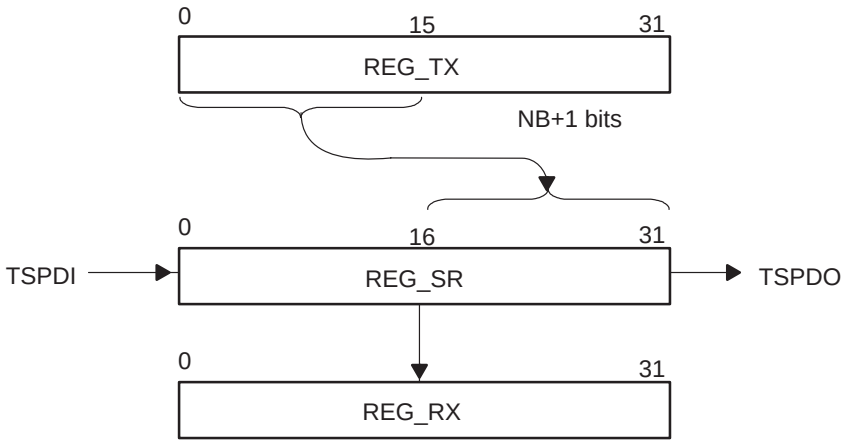
The data to transmit are loaded to the SPI_TX register.

Table 12. Transmit Register Bit Description (SPI_TX—0x034)

Base Address = 0xFFFB 0C00, Offset = 0x34				
Bit	**Name**	**Function**	**Access**	**Reset**
31:0	SPI_TX	Data to transmit	R/W	0x00000000

Notes: 1) The bits to transmit have to be aligned on the LSB of the SPI_TX register.
2) If the number of bits to transmit is 8, SPI_TX [7] is transmitted first, then SPI_TX [6], and so on.
3) If the number of bits to transmit is 32, SPI_TX [31] is transmitted first, then SPI_TX [30], and so on.

Figure 2. Transmission Example



Note: The number of bits of the word to be transmitted is programmed through NB bit field in the SPI_CTRL register. The transmit register (SPI_TX) data loading must be completed according to the transmitted word bit length and in the proper sequence register access. SPI_TX register is considered as updated (transmit register full with new data) when the most-significant byte part of the transmitted word has been written. If the number of bits of the transmitted word is not aligned on a byte boundary, the value of the unused bits is considered as *don't care*.

- 0 ≤ NB ≤ 7: MSB = SPI_TX[7:0]
- 8 ≤ NB ≤ 15: MSB = SPI_TX[15:8]
- 16 ≤ NB ≤ 23: MSB = SPI_TX[23:16]
- 24 ≤ NB ≤ 31: MSB = SPI_TX[31:24]

The SPI_TX register is a 32-bit wide register that is 16-bit, or 32-bit addressable. Partial register update with successive 16-bit accesses can be used to load the transmit register. In that case, the LSB must be updated before the MSB part of the transmitted word.

The received data are accessible on the OCP bus through SPI_RX register.

Table 13. Receive Register Bit Description (SPI_RX—0x038)

Base Address = 0xFFFB 0C00, Offset = 0x38				
Bit	**Name**	**Function**	**Access**	**Reset**
31:0	SPI_RX	Receive data	R	0x00000000

Note: The number of bits of the word to be received is programmed through the NB bit field in SPI_CTRL register. Receive register (SPI_RX) data read must be completed according to the received word bit length and in the proper sequence register access.

The SPI_RX register is considered to be empty if the most-significant byte part of the received word has been read (in functional mode only and not in emulation mode). If the number of bits of the received word is not aligned on a byte boundary, the unused bits are read as undefined value.

The SPI_RX register is a 32-bit register that is 16-bit, or 32-bit addressable. Partial register reads with successive 16-bit accesses can be used to read the receive register. In this case, the LSB must be read before the MSB part of the received word.

Table 14. Test Register Bit Description (SPI_TEST—0x03C)

Base Address = 0xFFFB 0C00, Offset = 0x3C				
Bit	**Name**	**Function**	**Access**	**Reset**
31:11	Reserved	A read access returns 0.	R	0x000000
10:6	RTSPEN	Read value of TSPEN	R	0
5	RCV	Read clock value	R	0
4	WCV	Write clock value	R/W	0
3	RTV	Read test value (spy TSPDI)	R	0
2	WTV	Write test value (force TSPDO)	R/W	0
1	FDO	Force TSPDO to read value from WTV bit: Active high	R/W	0
0	TMODE	Test mode enable: Active high	R/W	0

When the test mode is selected by setting the TMODE configuration bit in the SPI_TEST register, it enables the following features:

- TSPDO is fed back to TSPDI.
- It is possible to control and monitor the TSPDO, TSPDI, and CLK_S pins.

When the test mode is not active, a read to the SPI_TEST register always returns 0.

- FDO: This control bit enables forcing TSPDO to read the value of the WTV bit, allowing control of TSPDO.
- WTV: This bit enables the forcing the TSPDO value (test purposes).
- RTV: This bit is directly connected to TSPDI. It assumes that the input signal is static.
- WCV: This bit enables forcing the CLK output (in master mode only) to provide control of the CLK output.
- RCV: This bit is directly connected to CLK input. It enables testing connectivity of CLK_S in feedback mode only.
- RTSPEN: These bits are directly connected to TSPEN outputs when the test mode is enabled.

1.4 Protocol Description

The serial port interface must be configured via the setup registers.

A read process is always simultaneous with a write process, because the internal shift register is based on a loop (FIFO principle). However, the concurrent write process can be a dummy write if there is no data to transmit.

Depending on the mode selection (master or slave mode), the shift register clock can be derived internally from the CLK_M, or it can come directly from the CLK_S input.

The external transfer of a packet starts as soon as one of the transmit clocks is generated.

The received/transmitted data packet is shifted in/shifted out on the rising or falling edge of the shift register clock (SRCLK).

The loading of the packet is then validated on the deactivation of the enable signal (rising or falling edge).

SPI_ISR is updated at the end of a transaction in MCU-DSP and DMA modes (master or slave), and an interrupt request can be generated depending on SPI_IER bits.

Sections 1.4.1 to 1.4.3 present the steps describing the MCU-DSP and DMA protocols. These steps must be followed in order to have a correct behavior.

In MCU-DSP protocol, the interrupt request must not be masked. Thus, the SPI can issue an interrupt to inform the host that the transaction is finished.

1.4.1 MCU-DSP Protocol

The host is either the MCU or the DSP.

MCU-DSP Transmit Protocol in Master Mode

The protocol has several steps:

Step 1: MCU-DSP writes to the setup registers (SPI_SET1 and SPI_SET2).

Step 2: MCU-DSP writes to the transmit register (SPI_TX).

Step 3: MCU-DSP writes to the control register (SPI_CTRL).

Once the WR bit is set:

- ❏ The transmit register (SPI_TX) is copied into the shift register (SPI_SR).
- ❏ The device enable goes low (nTSPENi), if CEi = 0 in SPI_SET2.
- ❏ The shift register clock is activated (SRCLK) and the transmission starts.

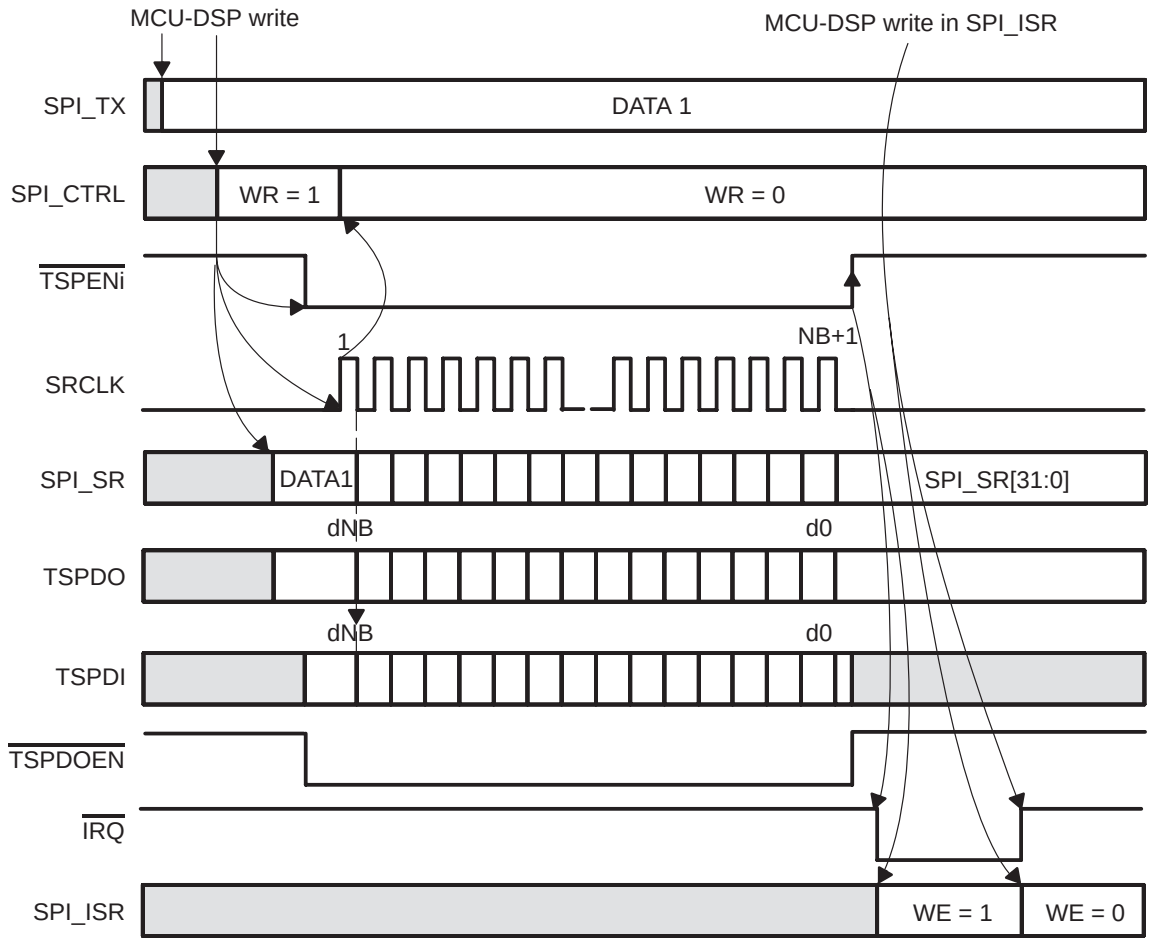
One SRCLK cycle later, the WR bit is reset.

Step 4: When the transmission is completed:

- ❏ The device enable (nTSPENi) goes high if CEi = 0 in SPI_SET2.
- ❏ The WE bit is set in the interrupt status register (SPI_ISR).
- ❏ An interrupt is generated, if MSK1 is set in the interrupt enable register (SPI_IER).

Step 5: Once the MCU-DSP clears the WE status bit (SPI_ISR), the interrupt request is released.

Figure 3. MCU-DSP Transmit Protocol in Master Mode With CIi = 0, CEi = 0 and CPi = 0



MCU-DSP Receive/Transmit Protocol in Master Mode

The protocol has several steps:

Step 1: MCU-DSP writes to the setup registers (SPI_SET1 and SPI_SET2).

Step 2: MCU-DSP writes to the transmit register (SPI_TX) (optional). This is necessary only when you want to perform a transmission at the same time.

Step 3: MCU-DSP writes to the control register (SPI_CTRL).

Once the RD bit is set:

- ❏ The transmit register is copied into the shift register (SPI_SR).
- ❏ The device enable goes low (nTSPEN[i]), if CEi = 0 in SPI_SET2.
- ❏ The shift register clock is activated (SRCLK) and the transmission and reception start.
- ❏ One SRCLK cycle later, the RD bit is reset.

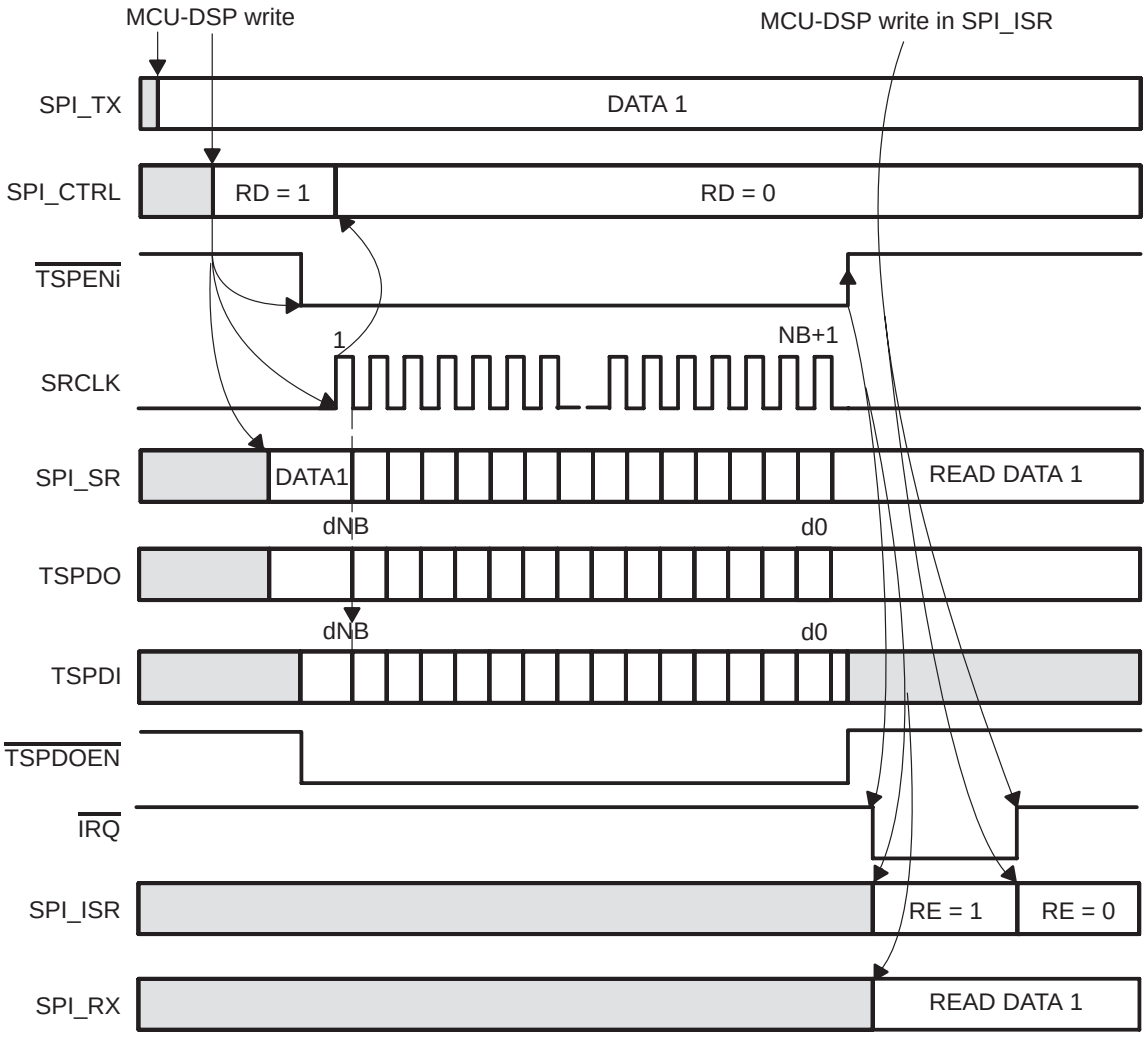
The WR bit has no effect on behavior. This bit is reset like RD, if it has been set.

When the reception is completed:

- ❏ The device enable goes high (nTSPEN[i]), if CEi = 0 in SPI_SET2.
- ❏ The RE bit is set in the interrupt status register (SPI_ISR).
- ❏ The shift register (SPI_SR) is copied into the receive register (SPI_RX).
- ❏ An interrupt is generated if MSK0 is set in the interrupt enable register (SPI_IER).

Step 4: Once the MCU-DSP has read the receive register (SPI_RX) and has cleared the RE status bit (SPI_ISR), the interrupt request is released.

Figure 4. MCU-DSP Receive Transmit Protocol in Master Mode With CIi = 0, CEi = 0, and CPi = 0



MCU-DSP Transmit Protocol in Slave Mode

The protocol has several steps:

Step 1: MCU-DSP writes to the setup registers (SPI_SET1 and SPI_SET2).

Step 2: MCU-DSP writes to the transmit register (SPI_TX).

Step 3: MCU-DSP writes to the control register (SPI_CTRL).

Once the WR bit is set, the transmit register (SPI_TX) is copied into the shift register (SPI_SR).

If CEi = 0 in SPI_SET2, the transmission starts as soon as the slave device enable goes low (nSPEN0) and the shift register clock is activated (SRCLK).

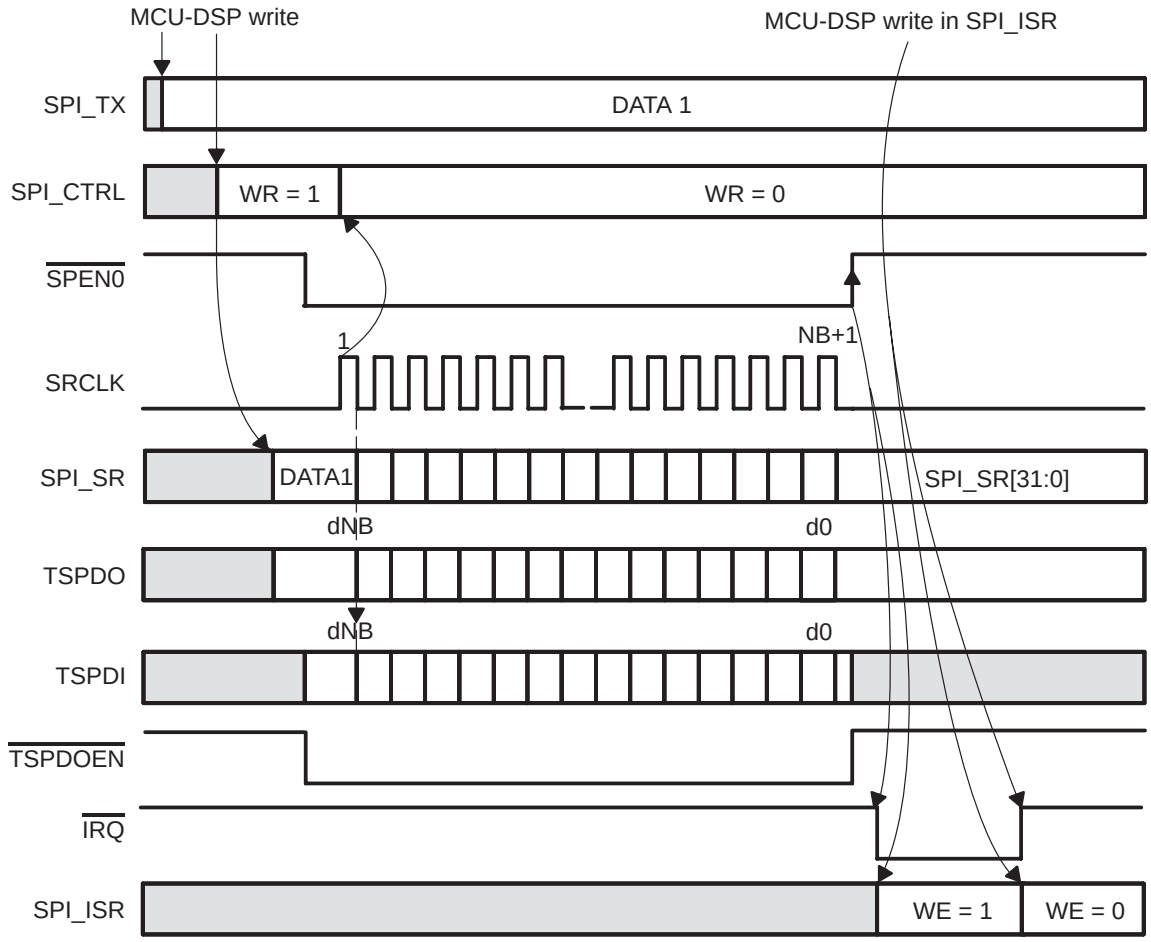
One SRCLK cycle later, the WR bit is reset.

Step 4: When the transmission is completed:

- ❏ The device enable goes high (nSPEN0) if CEi = 0 in SPI_SET2.
- ❏ The WE bit is set in the interrupt status register (SPI_ISR).
- ❏ An interrupt is generated if MSK1 is set in the interrupt enable register (SPI_IER).

Step 5: Once the MCU-DSP clears the WE status bit (SPI_ISR), the interrupt request is released.

Figure 5. MCU-DSP Transmit Protocol in Slave Mode With CIi = 0, CEi = 0 and CPi = 0



MCU-DSP Receive/Transmit Protocol in Slave Mode

The protocol is made up of several steps:

Step 1: MCU-DSP writes to the setup registers (SPI_SET1 and SPI_SET2).

Step 2: MCU-DSP writes to the transmit register (SPI_TX) (optional).

This is necessary only when you want to perform a transmission at the same time.

Step 3: MCU-DSP writes to the control register (SPI_CTRL).

Once the RD bit is set, the transmit register (SPI_TX) is copied into the shift register (SPI_SR).

If CEi = 0 in SPI_SET2, the transmission and reception start as soon as the slave device enable goes low (nSPEN0) and the shift register clock is activated (SRCLK).

One cycle later, the RD bit is reset.

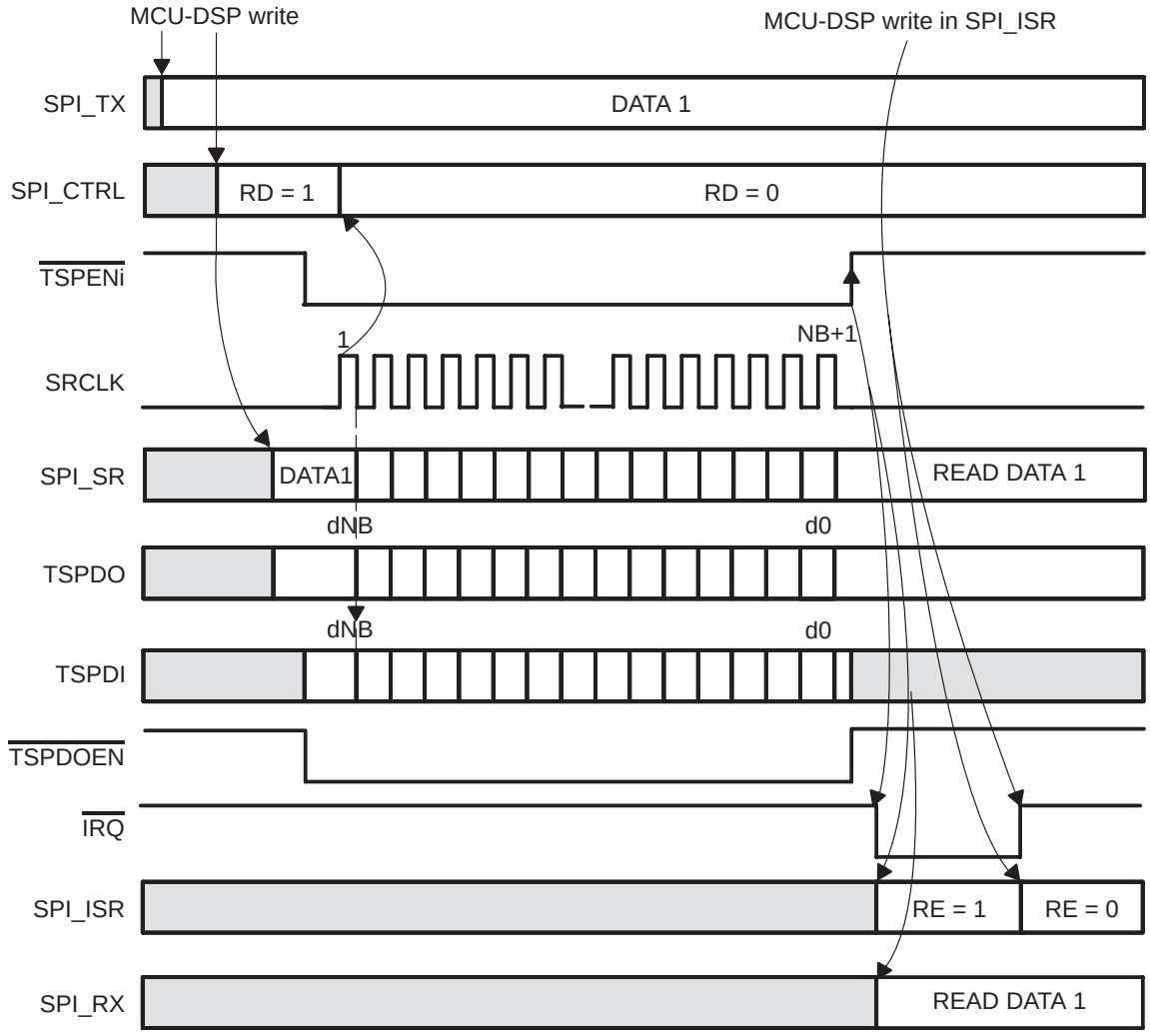
The WR bit has no effect on the behavior. This bit is reset like RD, if it has been set.

Step 4: When the reception is completed:

- ❏ The device enable goes high (nSPEN0) if CEi = 0 in SPI_SET2.
- ❏ RE status bit is set in the interrupt status register (SPI_ISR).
- ❏ The shift register (SPI_SR) is copied into the receive register (SPI_RX).
- ❏ An interrupt is generated if MSK0 is set in the interrupt enable register (SPI_IER).

Step 5: Once the MCU-DSP has read the receive register (SPI_RX) and has cleared the RE status bit (SPI_ISR), the interrupt request is released.

Figure 6. MCU-DSP RX/TX Protocol in Slave Mode With CIi = 0, CEi = 0 and CPi = 0



1.4.2 DMA Protocol

The interrupt (nIRQ) waveform is the same as the one in MCU-DSP protocol.

DMA Transmit Protocol in Master Mode

The protocol has several steps:

Step 1: MCU-DSP writes to the setup and control registers (SPI_SET1, SPI_SET2, and SPI_CTRL).

When DMA_EN is set, a transmit DMA request is generated, once the WR bit is set.

Step 2: The DMA writes the data to the transmit register (SPI_TX). Once the data is written:

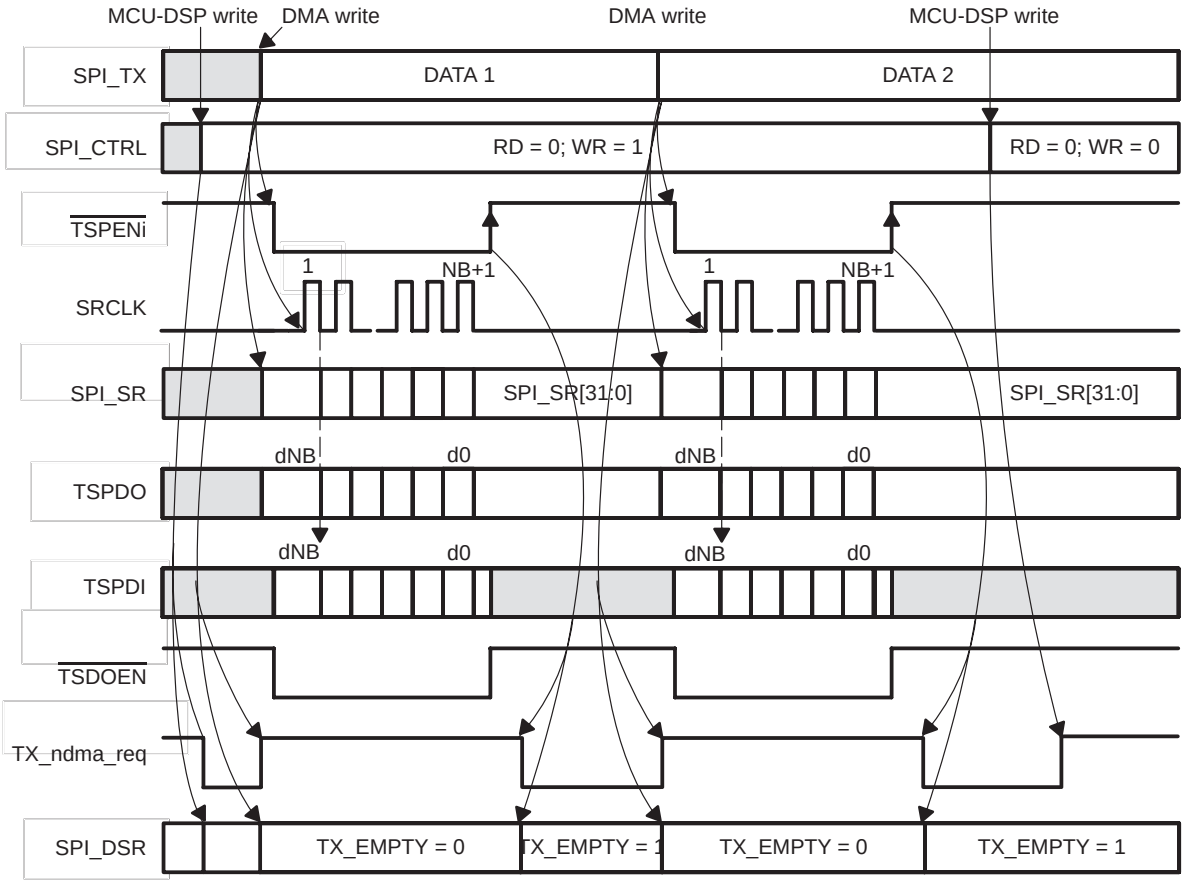
- ❏ The transmit DMA request is cleared.
- ❏ The TX_EMPTY status bit is reset in the data status register (SPI_DSR).
- ❏ The transmit register (SPI_TX) is copied into the shift register (SPI_SR).
- ❏ The device enable goes low (nTSPENi), if CEi = 0 in SPI_SET2.
- ❏ The shift register clock is activated (SRCLK) and the transmission starts.

Step 3: When the transmission is completed:

- ❏ The device enable goes high (nTSPENi) if CEi = 0 in SPI_SET2.
- ❏ TX_EMPTY is set in the data status register (SPI_DSR).
- ❏ A transmit DMA request is generated.
- ❏ Another transmission is able to start (Step2 → Step3 → Step2 → Step3…).

To stop the process, the MCU-DSP must reset the WR bit in the control register (SPI_CTRL).

Figure 7. DMA TX Protocol in Master Mode With CIi = 0, CEi = 0 and CPi = 0



DMA Receive Protocol in Master Mode

The protocol has several steps:

Step 1: MCU-DSP writes to the setup registers (SPI_SET1 and SPI_SET2).

Step 2: MCU-DSP writes to the control register (SPI_CTRL). Once the RD bit is set:

- ❏ The device enable goes low (nTSPENi) if CEi = 0 in SPI_SET2.
- ❏ The shift register clock is activated (SRCLK) and the reception starts.

Step 3: When the reception is completed:

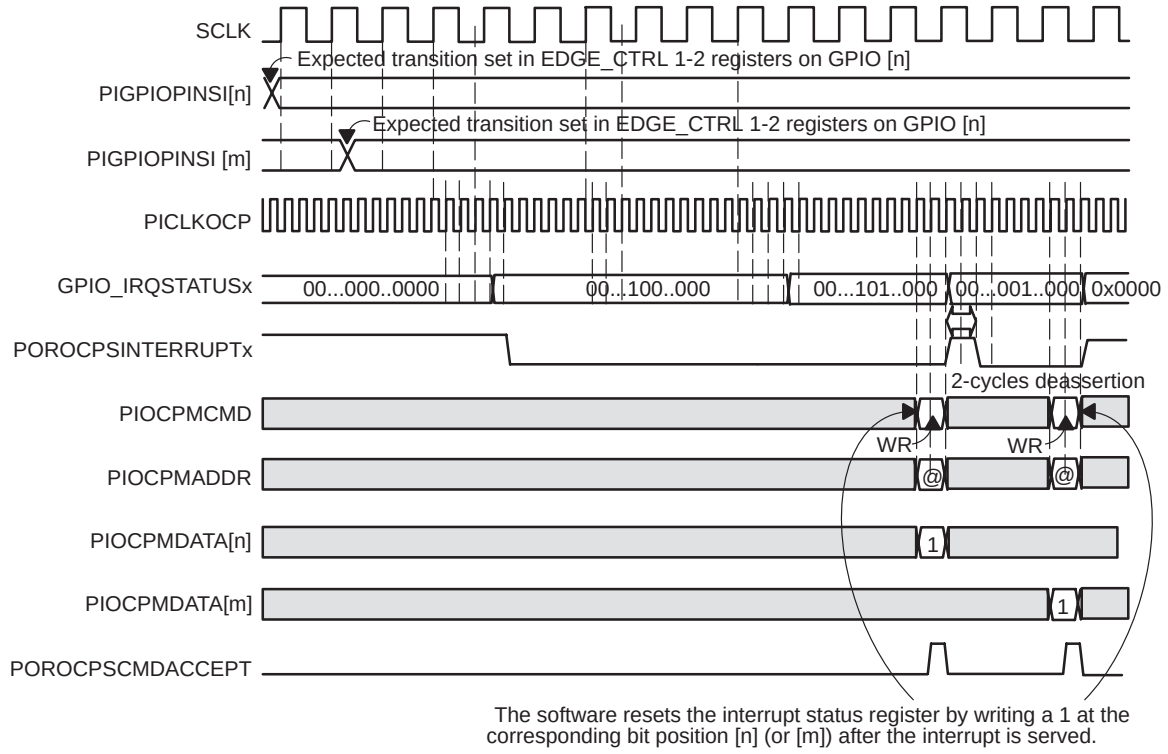
- ❏ The device enable goes high (nTSPENi) if CEi = 0 in SPI_SET2.
- ❏ The RX_FULL status bit is set in the data status register (SPI_DSR).
- ❏ The shift register (SPI_SR) is copied into the receive register (SPI_RX).
- ❏ A receive DMA request is generated.

Step 4: Once the DMA reads the receive register (SPI_RX):

- ❏ The device enable goes low (nTSPENi) if CEi = 0 in SPI_SET2.
- ❏ The RX_FULL status bit is reset in the data status register (SPI_DSR).
- ❏ The receive DMA request is released.
- ❏ Another reception starts (Step3 → Step4 → Step3 → Step4…).

To stop the process, the MCU-DSP must reset the RD bit in the control register (SPI_CTRL).

Figure 8. DMA Receive Protocol in Master Mode With CIi = 0, CEi = 0 and CPi = 0



DMA Transmit and Receive Protocol in Master Mode

The protocol has several steps:

Step 1: MCU-DSP writes to the setup and control registers (SPI_SET1, SPI_SET2, and SPI_CTRL).

When DMA_EN is set, a transmit DMA request is generated, once the RD and WR bits are set.

Step 2: The DMA writes the data in the transmit register (SPI_TX). Once the data is written:

- ❏ The transmit DMA request is released.
- ❏ TX_EMPTY is reset in the data status register (SPI_DSR).
- ❏ The transmit register (SPI_TX) is copied into the shift register (SPI_SR).
- ❏ The device enable goes low (nTSPENi), if CEi = 0 in SPI_SET2.
- ❏ The shift register clock (SRCLK) is activated and the transmission and reception start.

Step 3: When the transmission and the reception are completed:

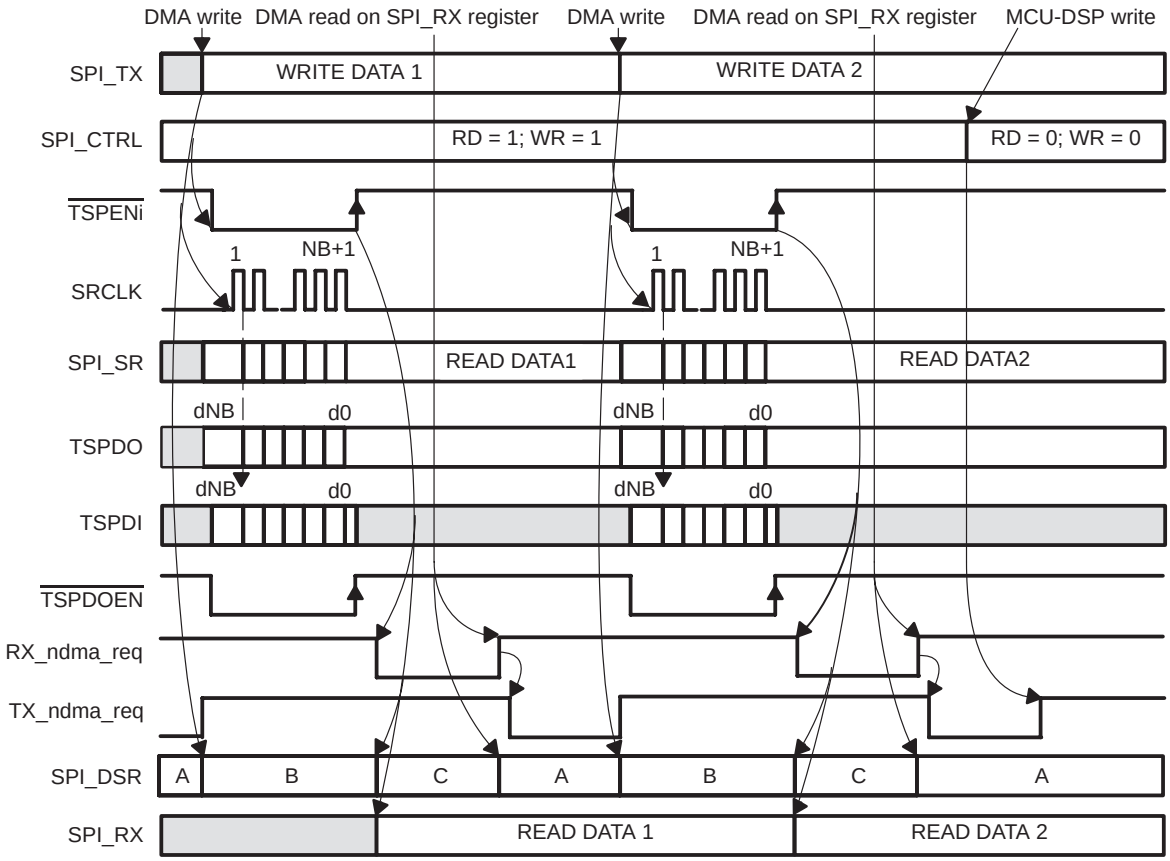
- ❏ The device enable goes high (nTSPENi), if CEi = 0 in SPI_SET2.
- ❏ The shift register (SPI_SR) is copied into the receive register (SPI_RX).
- ❏ RX_FULL and TX_EMPTY are set in the data status register (SPI_DSR).
- ❏ A receive DMA request is generated.

Step 4: Once the DMA reads the receive register (SPI_RX):

- ❏ The RX_FULL status bit is reset in the data status register (SPI_DSR).
- ❏ The receive DMA request is released.
- ❏ A transmit DMA request is generated.
- ❏ Another transmission and reception can start (Step 2 → Step 3 → Step 4 → Step 2 → Step 3 → Step4 →…).

To stop the process, the MCU-DSP has to reset the RD and WR bits in the control register (SPI_CTRL).

Figure 9. DMA Transmit and Receive Protocol in Master Mode With CIi = 0, CEi = 0 and CPi = 0



A: TX_EMPTY = 1 and RX_FULL = 0

B: TX_EMPTY = 0 and RX_FULL = 0

C: TX_EMPTY = 1 and RX_FULL = 1

DMA Transmit Protocol in Slave Mode

The protocol has several steps:

Step 1: MCU-DSP writes to the setup and control registers (SPI_SET1, SPI_SET2, and SPI_CTRL).

When DMA_EN is set, a transmit DMA request is generated, once the WR bit is set.

Step 2: The DMA writes the data in the transmit register (SPI_TX). Once the data is written:

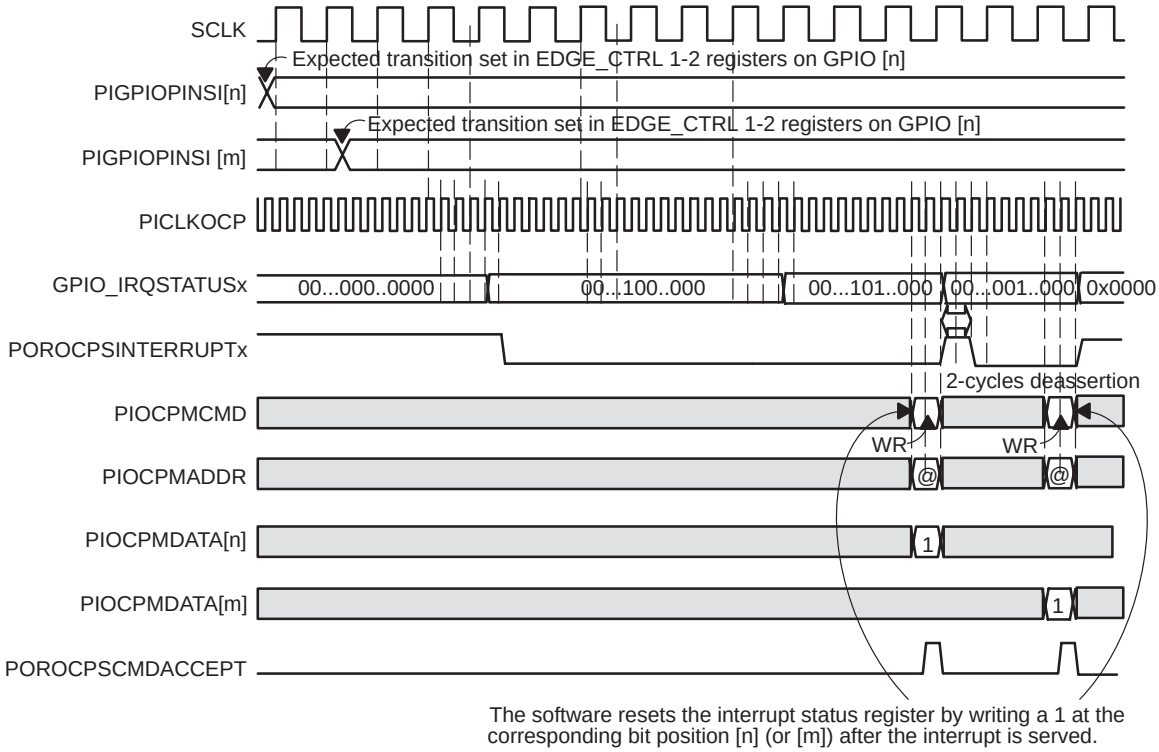
- ❏ The transmit DMA request is cleared (TX_NDMA_REQ goes high).
- ❏ TX_EMPTY is reset in the data status register (SPI_DSR).
- ❏ The transmit register (SPI_TX) is copied into the shift register (SPI_SR).
- ❏ If CEi = 0 in SPI_SET2, the transmission starts as soon as the slave device enable goes low (nSPEN0) and the shift register clock is activated (SRCLK).

Step 3: When the transmission is completed:

- ❏ TX_EMPTY is set in the data status register (SPI_DSR).
- ❏ A transmit DMA request is generated.
- ❏ Another transmission can start (Step2 → Step3 → Step2 → Step3…).

To stop the process, the MCU-DSP has to reset the WR bit in the control register (SPI_CTRL).

Figure 10. DMA Transmit Protocol in Slave Mode With CIi = 0, CEi = 0 and CPi = 0



DMA Receive Protocol in Slave Mode

The protocol has several steps:

Step 1: MCU-DSP writes to the setup and control registers (SPI_SET1, SPI_SET2, and SPI_CTRL).

- ❑ If CEi = 0 in SPI_SET2 and the RD bit is set, the reception starts as soon as the slave device enable goes low (nSPEN0) and the shift register clock is activated (SRCLK).

Step 2: When the reception is completed:

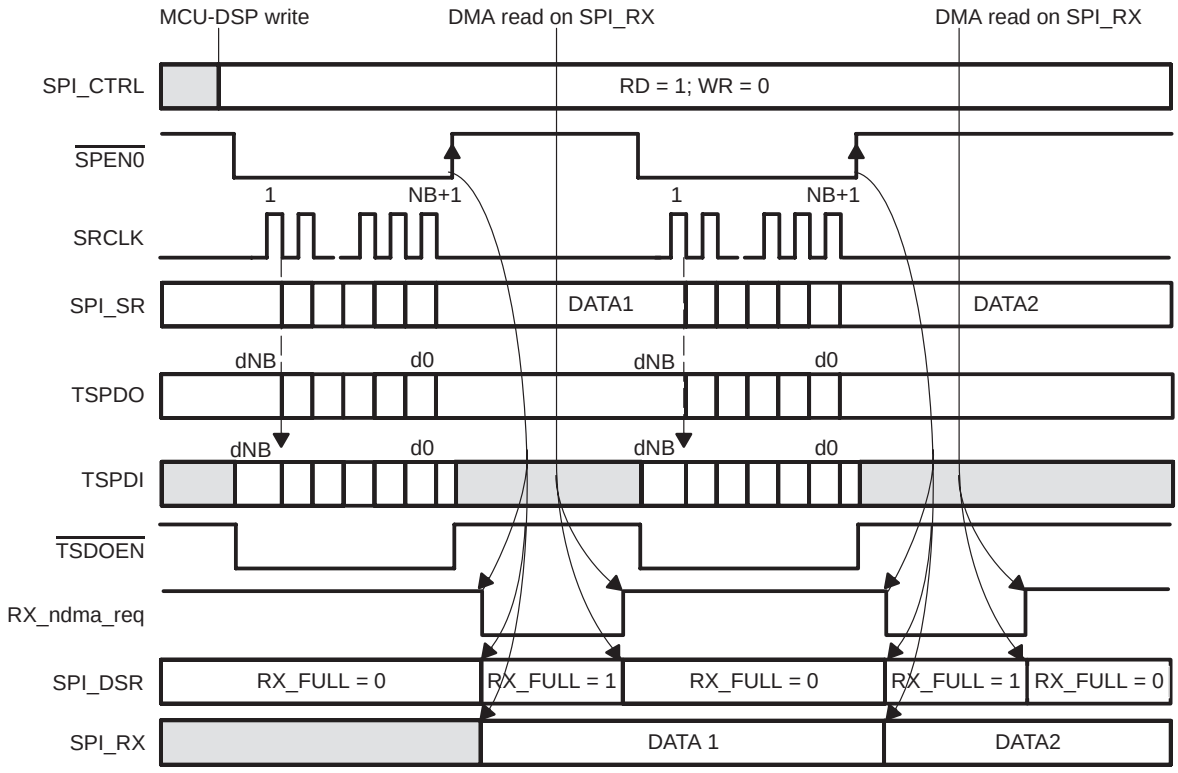
- ❑ The device enable goes high (nSPEN0), if CEi = 0 in SPI_SET2.
- ❑ The RX_FULL bit is set in the data status register (SPI_DSR).
- ❑ The shift register (SPI_SR) is copied into the receive register (SPI_RX).
- ❑ A receive DMA request is generated.

Step 3: Once the DMA reads the receive register (SPI_RX):

- ❑ The RX_FULL status bit is reset in the data status register (SPI_DSR).
- ❑ The receive DMA request is released.
- ❑ Another reception can start.

To stop the process, the MCU-DSP must reset the RD bit in the control register (SPI_CTRL).

Figure 11. DMA Receive Protocol in Slave Mode With CIi = 0, CEi = 0 and CPi = 0



DMA Transmit and Receive Protocol in Slave Mode

The protocol has several steps:

Step 1: MCU-DSP writes to the setup and control registers (SPI_SET1, SPI_SET2, and SPI_CTRL).

When the DMA_EN and WR bits are set, a transmit DMA request is generated.

Step 2: The DMA writes the data in the transmit register (SPI_TX). Once the data is written:

- ❏ The transmit DMA request is released.
- ❏ The TX_EMPTY status bit is reset in the data status register (SPI_DSR).
- ❏ The transmit register (SPI_TX) is copied into the shift register (SPI_SR).

If CEi = 0 in SPI_SET2, the transmission and reception start as soon as the slave device enable goes low (nSPEN0) and the shift register clock is activated (SRCLK).

Step 3: When the transmission and the reception are completed:

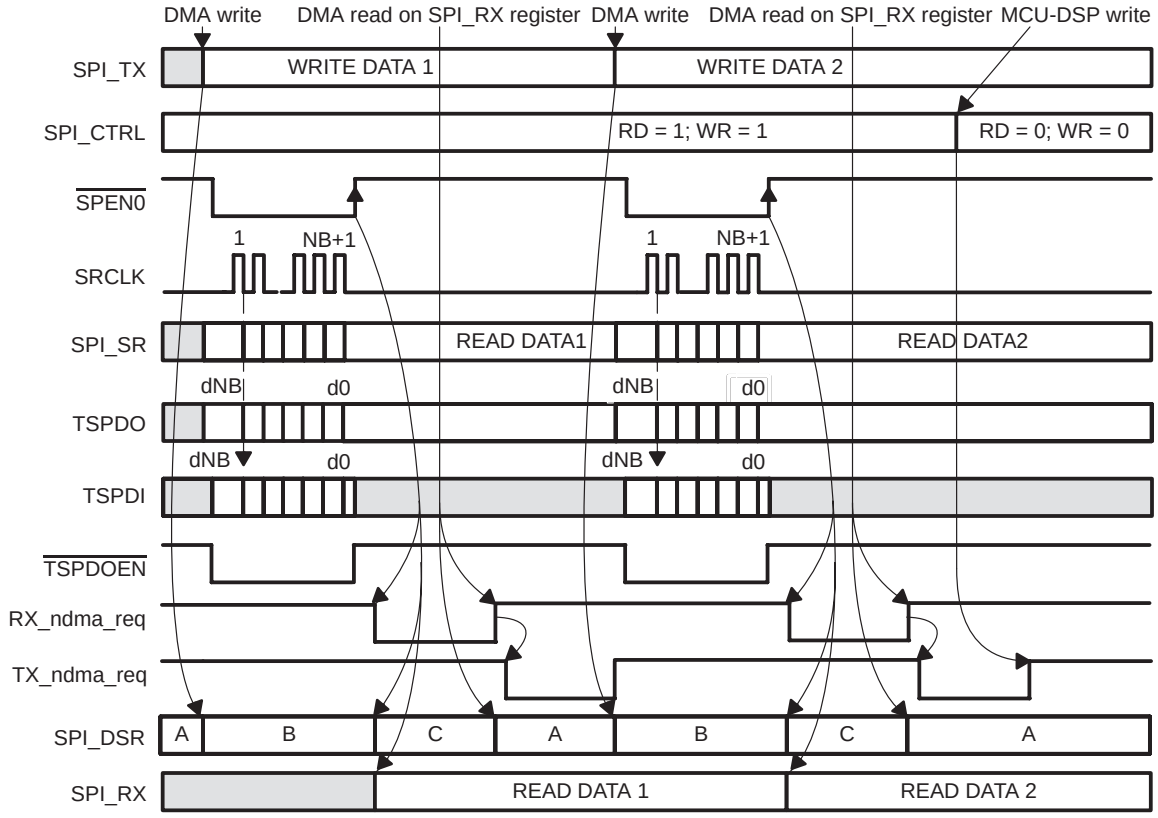
- ❏ The shift register (SPI_SR) is copied into the receive register (SPI_RX).
- ❏ The RX_FULL and TX_EMPTY bits are set in the data status register (SPI_DSR).
- ❏ A receive DMA request is generated.

Step 4: Once the DMA reads the receive register (SPI_RX):

- ❏ The RX_FULL status bit is reset in the data status register (SPI_DSR).
- ❏ The receive DMA request is released.
- ❏ A transmit DMA request is generated.
- ❏ Another transmission and reception can start.

To stop the process, the MCU-DSP must reset the RD and WR bits in the control register (SPI_CTRL).

Figure 12. DMA Transmit and Receive Protocol in Slave Mode With CIi = 0, CEi = 0 and CPi = 0



A: TX_EMPTY = 1 and RX_FULL = 0
B: TX_EMPTY = 0 and RX_FULL = 0
C: TX_EMPTY = 1 and RX_FULL = 1

1.4.3 Overflow/Underflow Interrupts

In slave mode, whether the functional mode is MCU DSP or DMA, the possibility exists that the receive register (SPI_RX) is overflowing or/and the transmit register (SPI_TX) is underflowing. In either case, an interrupt is generated and sent to the host. It is up to the host to take the right action, according to the received interrupt.

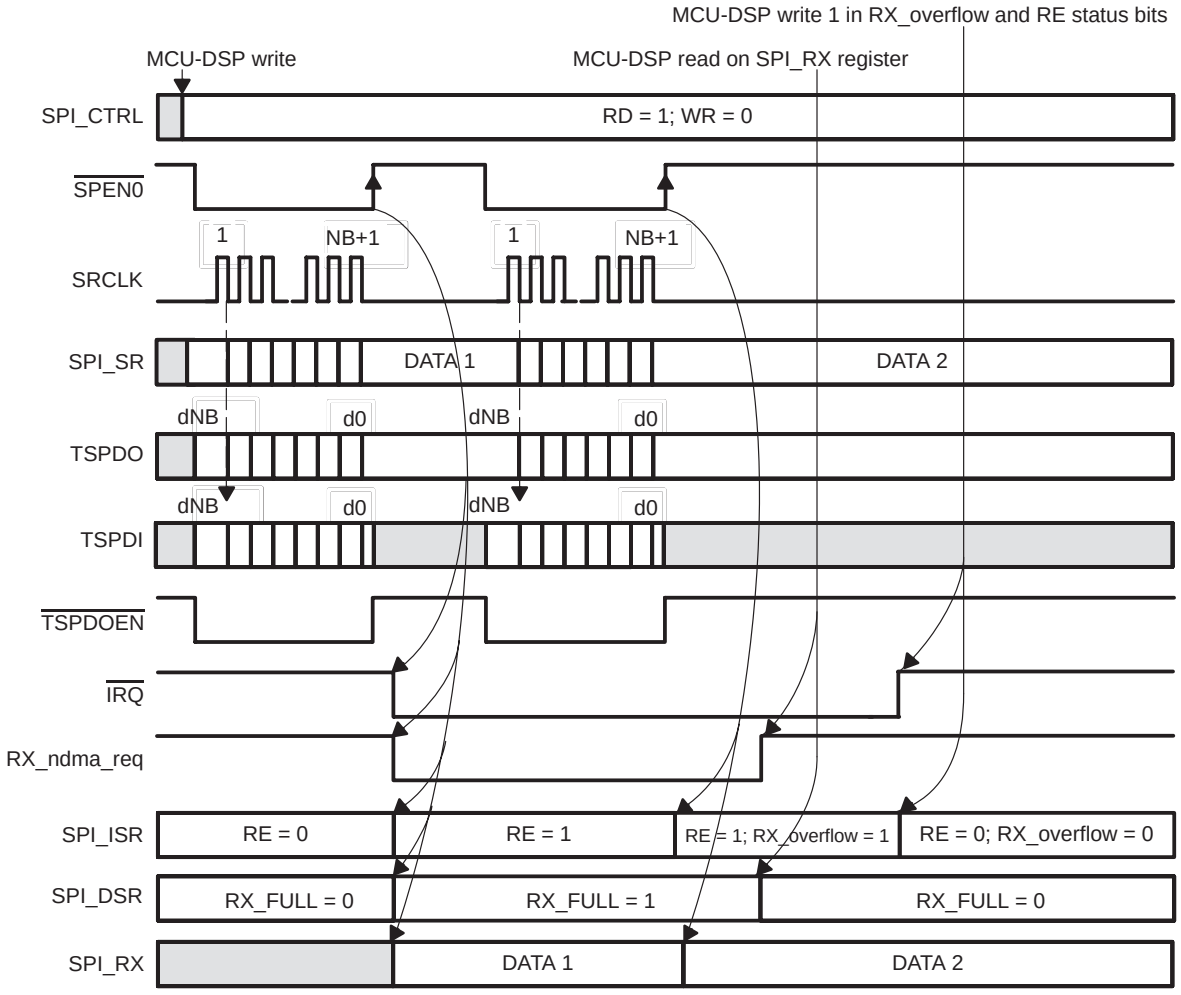
Overflow Interrupt Generation

To generate an overflow interrupt, the SPI must be in the following state:

- SPI is configured in slave mode (SPI_SET2 [15] = 0).
- Enable for overflow interrupt is active (SPI_IER [2] = 1).
- The receive register (SPI_RX) has not been read between two receptions.

To release the interrupt (nIRQ) activated by the RX overflow bit (SPI_ISR [2]), the user has to clear the RX overflow status bit by writing a 1 in SPI_ISR [2].

Figure 13. Example of an Overflow Generation With CIi = 0, CEi = 0 and CPi = 0



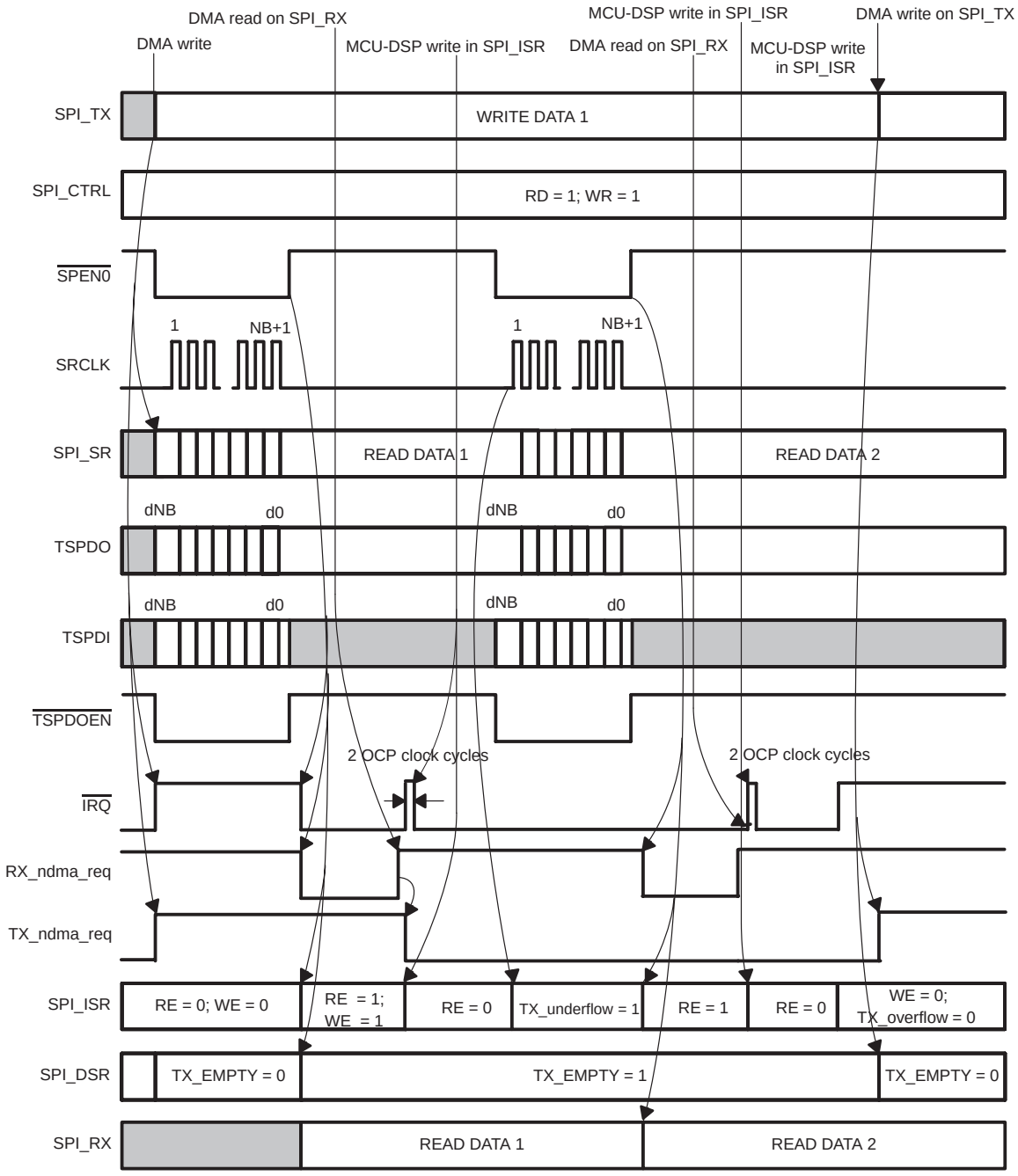
Underflow Interrupt Generation

To generate an underflow interrupt, the SPI has to be in the following state:

- ❏ SPI is configured in slave mode (SPI_SET2 [15] = 0).
- ❏ Enable for underflow interrupt is active (SPI_IER [3] = 1).
- ❏ The transmit register (SPI_TX) has not been updated between two transmissions.

To release the interrupt (nIRQ) activated by the TX underflow bit (SPI_ISR [3]), the user has to clear the Tx_underflow status bit by writing a 1 to SPI_ISR [3].

Figure 14. Example of an Underflow Generation With CIi = 0, CEi = 0 and CPi = 0



1.4.4 Transmission Modes

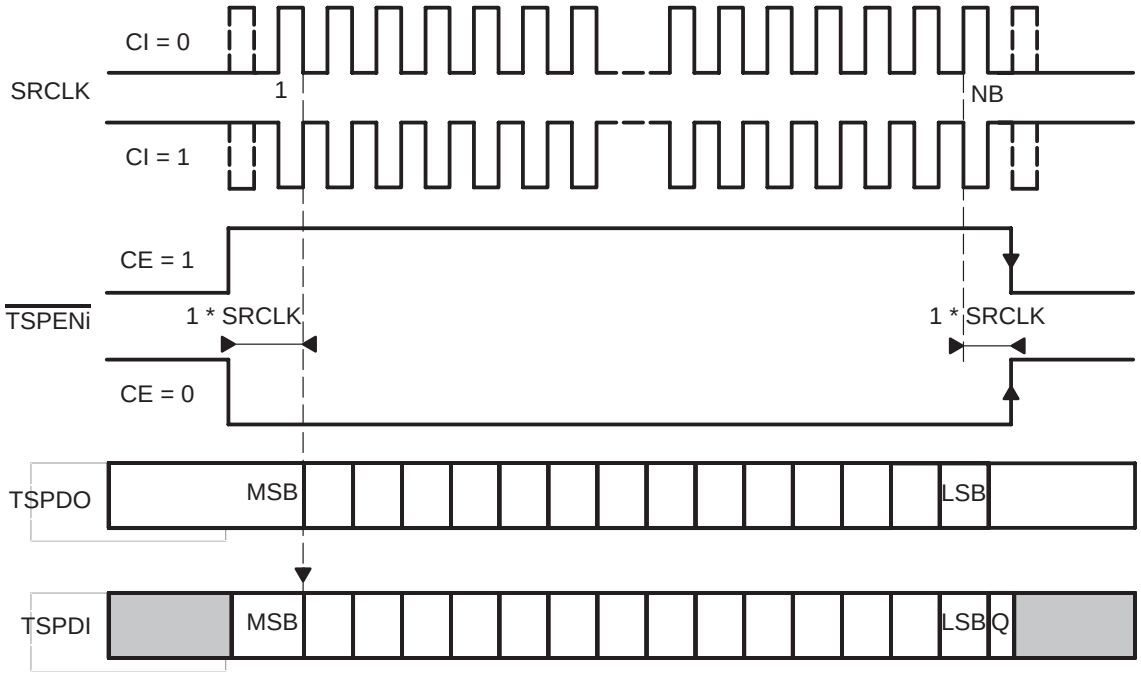
The serial interface is active as soon as the shift register clock is activated.

nSPEN0 behaves the same as nTSPENi, as shown in Figure 15 and Figure 16.

The transmitted data packet is shifted out on the rising or falling edge of SRCLK, whereas the received data packet is captured on the falling or rising edge of SRCLK (complementary edge).

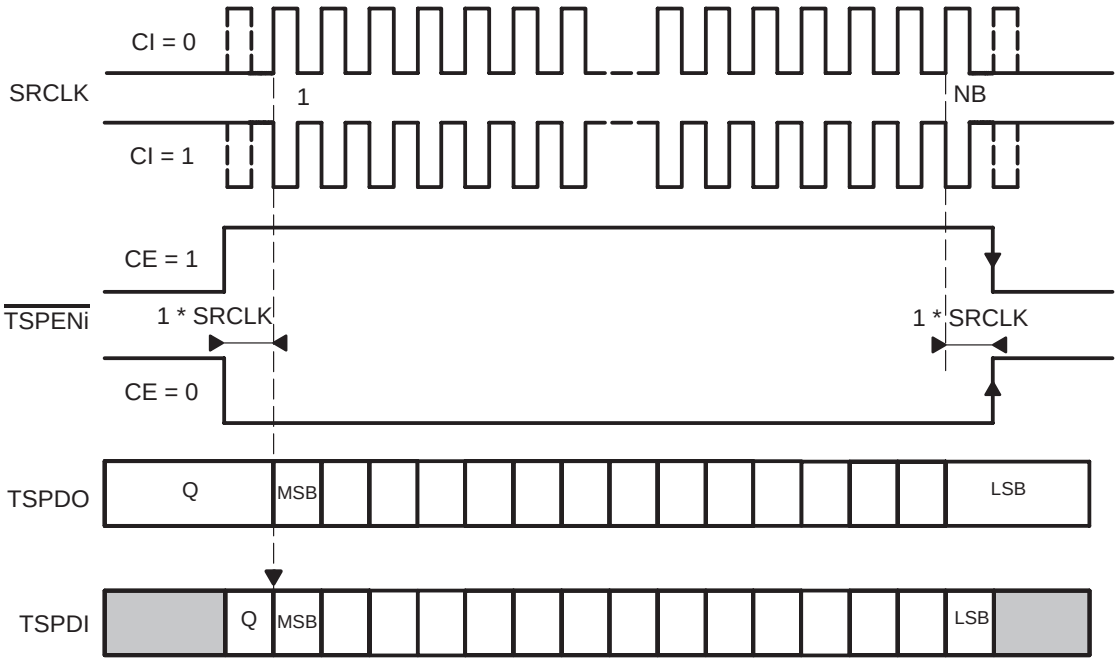
When CPi = 0, the first edge (rising or falling) of SRCLK is used to capture the data and the second edge (falling or rising) is used to shift the data.

Figure 15. Example of a Transmission With CPi = 0



When CPi = 1, the first edge (rising or falling) of SRCLK is used to shift the data and the second edge (falling or rising) is used to capture the data.

Figure 16. Example of a Transmission With CPi = 1



1.5 Idle and Wake-Up Feature

When the host processor issues an idle request, the SPI goes into idle mode (in this mode, no clock is provided to the SPI), according to the IDLEMODE field of the system configuration register SPI_SCR as described in Table 4.

If the IDLEMODE field is set to no-idle, the SPI does not go to idle mode.

If the IDLEMODE field is set to force-idle, the SPI goes into idle mode and acknowledges the request unconditionally. In this mode, SPI does not execute any transaction.

If the IDLEMODE field is set to smart-idle, the SPI module evaluates its internal capability to have the functional and interface clocks switched off. Once there is no more internal activity, the request signal is acknowledged and the SPI enters the sleep mode.

In this mode, the module issues a wake-up request when the following conditions are met:

- ❏ SPI is configured in slave mode (SPI_SET2 [15] = 0).
- ❏ The slave device enable (nSPEN0) becomes inactive after a transaction.

The ENAWAKEUP bit of the system configuration register (SPI_SCR) controls this wake-up request.

When the system wakes up, the following actions are executed:

- The idle request goes low (inactive).
- The wake-up request is deasserted (in smart-idle mode only).
- The WAKEUP status bit is set in the interrupt status register (SPI_ISR).
- The idle acknowledge goes low (inactive).
- An interrupt is generated if MSK4 is set in the interrupt enable register (SPI_IER).

Once the SPI acknowledges the idle request, the functional and interface clocks can be stopped one cycle later.

Some restrictions apply on the module functionality when a wake-up request is generated (in Smart-Idle mode only). The following conditions must be met to ensure the right behavior of the requested transaction:

- The SPI must be in DMA mode: DMA_EN set in the set up register (SPI_SET1 [5] = 1).
- The requested transaction that wakes up the module must be a receive: RD set and WR reset in the control register (SPI_CTRL [1:0] = 01).
- The functional clock (CLK_M) must be active if the bit ENAWAKEUP (SPI_SCR [2]) is set, in order to generate a wake-up request and to copy the received data into the SPI_RX register. If the bit ENAWAKEUP is reset, SPI does not work when receiving a read transaction.

The AUTOIDLE bit of the system configuration register (SPI_SCR [0]) can be set in order to save power. This bit controls the internal OCP clock activity:

- When this bit is cleared, the internal OCP clock is free-running.
- When this bit is set, the internal OCP clock becomes inactive if the OCP command is in IDLE state.

1.6 Emulation Mode

In emulation mode, SPI has a slightly different functionality: a read of SPI_RX does not clear the bit RX_FULL (SPI_DSR [0]). Thus, SPI_RX is still considered as not read for SPI.

Otherwise, SPI behavior in emulation mode is the same as in functional mode.

A read of the bits EMUSOFT and EMUFREE (SPI_SCR [6:5]) always gives 01. A write has no effect on these bits.

1.7 Reset

Before accessing or using the module, the local host must ensure that internal reset is released by reading the system status register (SPI_SSR).

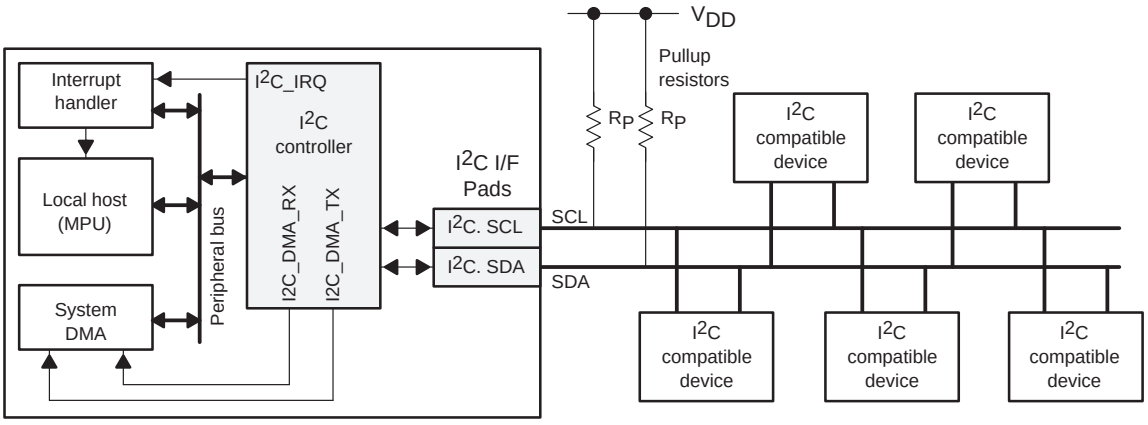
2 I2C Multimaster Peripheral

2.1 Overview

The multimaster I2C peripheral provides an interface between a local host (LH) such as an MPU, MIPS, or DSP processor and any I2C-bus-compatible device that connects via the I2C serial bus. External components attached to the I2C bus can serially transmit/receive up to 8-bit data to/from the LH device through the two-wire I2C interface.

This I2C peripheral supports any slave or master I2C-compatible device. Figure 17 shows the example of a system with multiple I2C compatible devices in which the I2C serial ports are connected together for a two-way transfer from one device to other devices.

Figure 17. I2C System Overview



2.2 Functional Overview

The I2C bus is a multimaster bus. The I2C controller supports the multimaster mode that allows more than one device capable of controlling the bus to be connected to it. Each I2C device, including the DSP, is recognized by a unique address and can operate as either transmitter or receiver, according to the function of the device. In addition to being a transmitter or receiver, a device connected to the I2C bus can also be considered as master or slave when performing data transfers. A master device is a device that initiates a data transfer on the bus and generates the clock signals to permit that transfer. During this transfer, any device addressed by this master is considered a slave.

2.3 I2C Controller Features

The main features of the I2C controller are:

- ❑ Compliance with Philips I2C specification version 2.1, January 2000
- ❑ Standard mode (up to 100K bits/s) and fast mode (up to 400K bits/s) support
- ❑ 7-bit and 10-bit device addressing modes
- ❑ General call
- ❑ Start/restart/stop
- ❑ Multimaster transmitter/slave receiver mode
- ❑ Multimaster receiver/slave transmitter mode
- ❑ Combined master transmit/receive and receive/transmit mode
- ❑ Built-in FIFO for buffered read or write
- ❑ Module enable/disable capability
- ❑ Programmable clock generation
- ❑ 16-bit wide access to maximize bus throughput
- ❑ Low-power design
- ❑ Two DMA channels
- ❑ Wide-interrupt capability

The current I2C does not support:

- ❑ High-speed (HS) mode for transfer up to 3.4M bits/s
- ❑ C-bus-compatibility mode

2.4 I2C Master/Slave Controller Signal Pads

Data are communicated to devices interfacing with the I2C via the serial data line (SDA) and the serial clock line (SCL). These two wires carry information between the DSP or MPU device and other devices connected to the I2C bus. The I2C is a shared peripheral that can be allocated to the MCU or the DSP. Both the SDA and SCL are bidirectional pins. They must be connected to a positive supply voltage via a pullup resistor. When the bus is free, both pins are high. The driver of these two pins has an open drain to perform the required wired-AND function.

Table 15. Signal Pads

Name	Type	Reset Value	Description
I2C_SCL	In/ Out(OD)	Input	I2C serial CLK line. Open-drain output buffer. Requires external pull-up resistor (Rp).
I2C_SDA	In/ Out(OD)	Input	I2C serial data line. Open-drain output buffer. Requires external pull-up resistor (Rp).

2.5 Operational Details

2.5.1 I2C Reset

The I2C module can be reset in the following three ways:

- A system bus reset (RESET_ = 0). A device reset causes the system bus reset.
- A software reset by setting the SRST bit in the I2C_SYSC register. This bit has the same action on the module logic as the system bus reset.
- The I2C_EN bit in the I2C_CON register can also reset a part of the I2C module. When the system bus reset is released (RESET_ = 1), I2C_EN = 0 keeps the functional part of the I2C module in reset state and all configuration registers can be accessed.

Table 16. Reset State of I2C Signals

Pin	I/O/Z	System Reset	I2C Reset (I2C_EN = 0)
SDA	I/O/Z	High impedance	High impedance
SCL	I/O/Z	High impedance	High impedance

2.5.2 I2C Bit Transfer

The master device generates one clock pulse for each data bit transferred. Because of the variety of technology devices (CMOS, NMOS, bipolar) that can be connected to the I2C bus, the levels of logical 0 (low) and 1 (high) are not fixed and depend on the associated level of VDD. See Table 17 for electrical specifications.

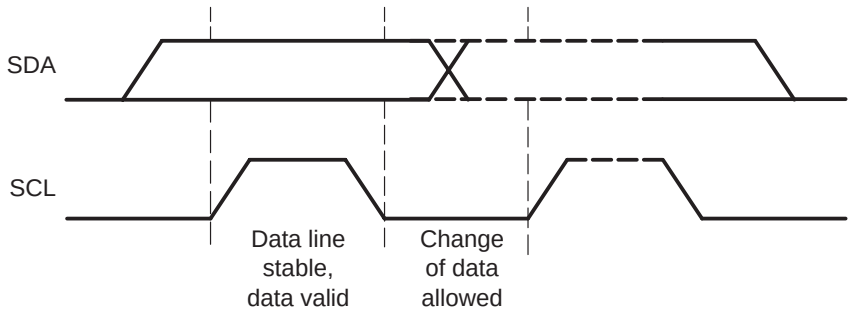
Table 17. Electrical Specification of the Input/Output

	Parameter	Standard Mode Devices		Fast-Mode Devices		Unit
		Min	Max	Min	Max	
VIL	Low-level input voltage:					
	Fixed input levels	−0.5	1.5	n/a	n/a	V
	VDD-related input levels	−0.5	0.3VDD	−0.5	0.3VDD	
VIH	High-level input voltage:					
	Fixed input levels	3.0	VDDmax+0.5	n/a	n/a	V
	VDD-related input levels	0.7VDD	VDDmax+0.5	0.7VDD	VDDmax+0.5	
	Low-level output voltage:					
	VDD>2V					
VOL1	At 3mA sink current	0	0.4	0	0.4	V
VOL2	At 6mA sink current	n/a	n/a	0	0.6	
	VDD<2V					
VOL3	At 3mA sink current	n/a	n/a	0	0.2VDD	

2.5.3 Data Validity

The data on the SDA line must be stable during the high period of the clock. The high and low states of the data line can change only when the clock signal on the SCL line is low.

Figure 18. Bit Transfer on the I^2C Bus



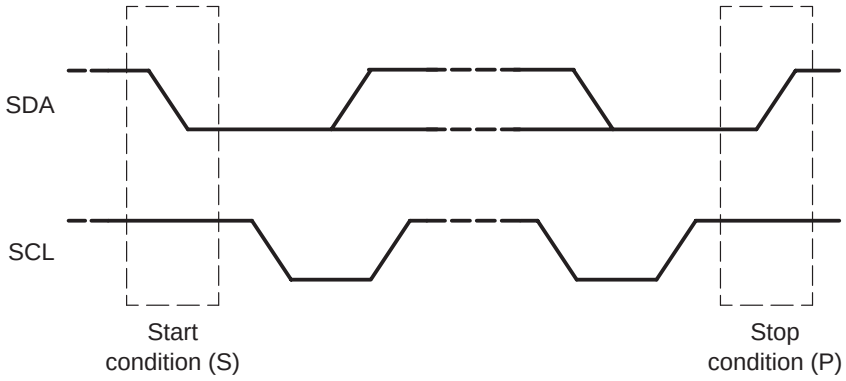
2.5.4 START and STOP Conditions

The I2C module generates start and stop conditions when it is configured as a master:

- START condition is a high-to-low transition on the SDA line while SCL is high.
- STOP condition is a low-to-high transition on the SDA line while SCL is high.

The bus is considered to be busy after the START condition (BB = 1) and free after the STOP condition (BB = 0).

Figure 19. Start and Stop Condition Events

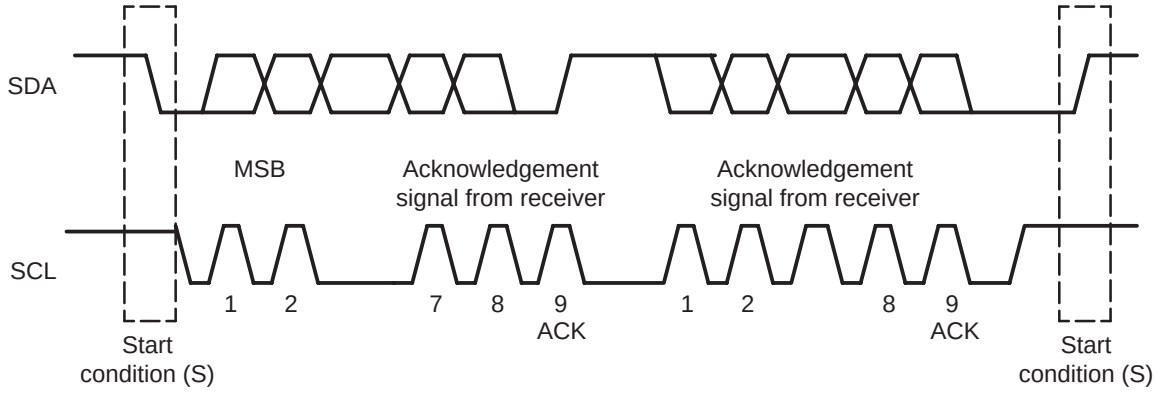


2.6 I2C Operation

Serial Data Formats

The I2C controller operates in 16-bit word data format (byte write access supported for the last access). Each byte put on the SDA line is 8 bits long. The number of bytes that can be transmitted or received is unrestricted. The data are transferred with the most-significant bit (MSB) first. Each byte is followed by an acknowledge bit from the I2C module, if it is in receiver mode. The I2C controller supports endianism.

Figure 20. I2C Data Transfer



The I2C module supports two data formats, as shown in Figure 21:

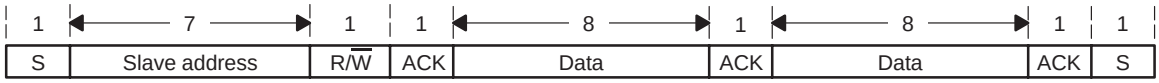
- 7-bit/10-bit addressing format
- 7-bit/10-bit addressing format with repeated start condition

The first byte after a start condition (S) always consists of 8 bits. In the acknowledge mode, an extra bit dedicated for acknowledgement is inserted after each byte.

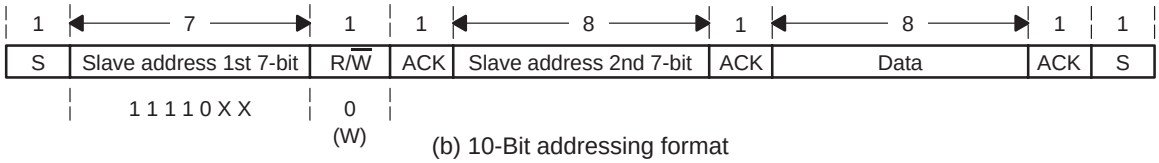
In the addressing formats with 7-bit addresses, the first byte is composed of 7 MSB slave-address bits and 1 LSB R/W_ bit. In the addressing formats with 10-bit addresses, the first byte has 7 MSB slave address bits, such as 11110XX where XX is the two MSB of the 10-bit addresses and 1 LSB R/W_ bit, which is 0 in this case.

The least-significant R/W_ of the address byte indicates the direction of the transmission of the following data bytes. If R/W_ is 0, the master writes data into the selected slave; if it is 1, the master reads data out of the slave.

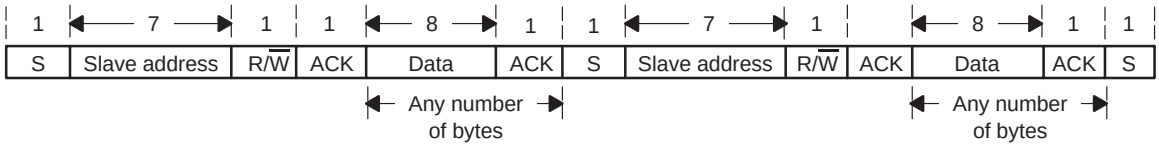
Figure 21. I^2C Data Transfer Formats



(a) 7-Bit addressing format



(b) 10-Bit addressing format



(c) Addressing Format With Repeated Start Condition

Master Transmitter

In this mode, data assembled in one of the previously described data formats is shifted out on the serial data line SDA in synch with the self-generated clock pulses on the serial clock line SCL. The clock pulses are inhibited and SCL is held low when the intervention of the processor is required (XUDF) after a byte has been transmitted.

Master Receiver

This mode can be entered only from the master transmitter mode. With any of the address formats (Figure 21 (a), (b), and (c)), the master receiver is entered after the slave address byte and bit R/W_ have been transmitted, if R/W_ is high. Serial data bits received on bus line SDA are shifted in synch with the self-generated clock pulses on SCL. The clock pulses are inhibited and SCL held low when the intervention of the processor is required (ROVR) after a byte has been transmitted. At the end of a transfer, it generates the stop condition.

Slave Transmitter

This mode can be entered only from the slave receiver mode. With any of the address formats (Figure 21 (a), (b), and (c)), the slave transmitter is entered if the slave address byte is the same as its own address and bit R/W_ has been transmitted, if R/W_ is high. The slave transmitter shifts the serial data out on the data line SDA in synch with the clock pulses that are generated by the master device. It does not generate the clock, but it can hold clock line SCL low while the local host is required to intervene (XUDF).

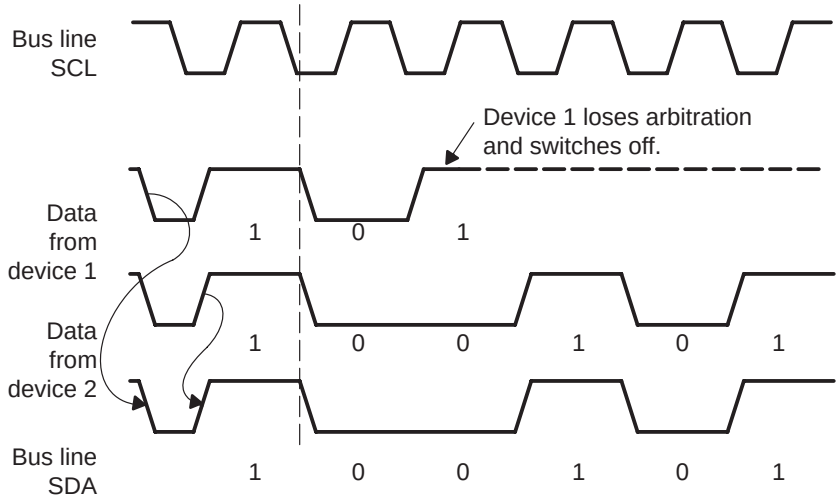
Slave Receiver

In this mode, serial data bits received on the bus line SDA are shifted-in in synch with the clock pulses on SCL that are generated by the master device. It does not generate the clock, but it can hold the clock line SCL low while the local host is required to intervene (ROVR) following the reception of a byte.

2.6.1 Arbitration

If two or more master transmitters start a transmission on the same bus almost simultaneously, an arbitration procedure is invoked. The arbitration procedure uses the data presented on the serial bus by the competing transmitters. When a transmitter senses that a high signal it has presented on the bus has been overruled by a low signal, it switches to the slave receiver mode, sets the arbitration lost (AL) flag, and generates the arbitration lost interrupt. Figure 22 shows the arbitration procedure between two devices. The arbitration procedure gives priority to the device that transmits the serial data stream with the lowest binary value. Should two or more devices send identical first bytes, arbitration continues on the subsequent bytes.

Figure 22. Arbitration Procedure Between Two Master Transmitters

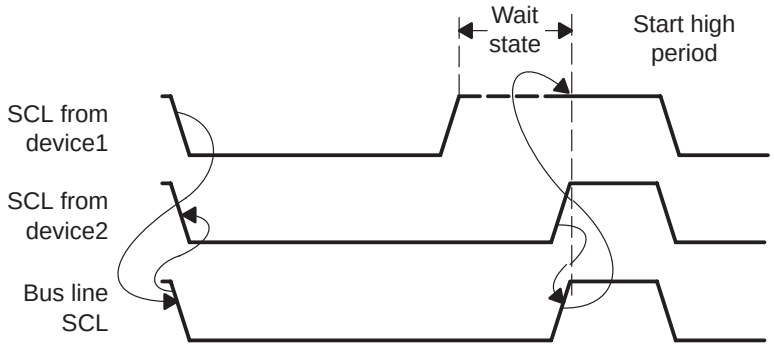


2.6.2 I²C Clock Generation and I²C Clock Synchronization

Under normal conditions, only one master device generates the clock signal, SCL. During the arbitration procedure, however, two or more master devices and the clock must be synchronized so that the data output can be compared. The wired-AND property of the clock line means that a device that first generates a low period of the clock line overrules the other devices. At this high/low transition, the clock generators of the other devices are forced to start generation of their own low period. The clock line is then held low by the device with the longest low period, while the other devices that finish their low periods must wait for the clock line to be released before starting their high periods. A synchronized signal on the clock line is thus achieved, where the slowest device determines the length of the low period and the fastest the length of the high period.

If a device pulls down the clock line for a longer time, the result is that all clock generators must enter the WAIT state. In this way, a slave can slow down a fast master, and the slow device can create enough time to store a received byte or prepare a byte to be transmitted. Figure 23 illustrates the clock synchronization.

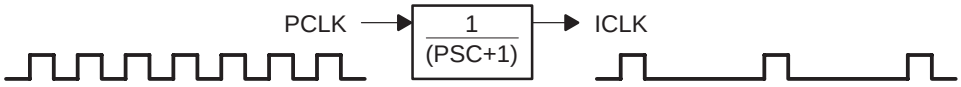
Figure 23. Synchronization of Two I2C Clock Generators



2.6.3 Prescaler (SCLK/ICLK)

The I2C module is operated with an internal approximately 12-MHz clock (ICLK). This clock is generated via the I2C prescaler block. The prescaler consists of an 8-bit register. I2C _PSC is used for dividing down the system clock (SCLK) to obtain an approximately 12-MHz clock for the I2C module.

Figure 24. Synchronization of Two I2C Clock Generators



0x0:	Divide by 1
0x1:	Divide by 2
↓	↓
0xFF:	Divide by 256

Values after reset are low (All 8 bits).

2.6.4 Noise Filter

The noise filter suppresses any noise that is 50 ns or less. It is designed to suppress noise with 1 ICLK, assuming the lower and upper limits of ICLK are 8 MHz and 16 MHz, respectively.

2.6.5 I2C Interrupts

The I2C module generates six types of interrupts: Arbitration-lost, no-acknowledge, general call, registers-ready-for-access, receive, and transmit. These six interrupts are accompanied by six interrupt masks and flags defined in the I2C_IE and I2C_STAT registers, respectively.

- Arbitration lost interrupt (AL): Generated when the I2C arbitration procedure is lost.

- No-acknowledge interrupt (NACK): Generated when the master I2C does not receive an acknowledge from the receiver.
- General call interrupt (GC): Generated when the device detects the address of all zeros (8 bits).
- Registers-ready-for-access interrupt (ARDY): Generated by the I2C when the previously programmed address, data, and command have been performed and the status bits have been updated. This interrupt is used to let the LH know that the I2C registers are ready for access.
- Receive interrupt/status (RRDY): Generated when there is received data ready to be read by the LH from the I2C_DATA register. The LH can poll this bit to read the received data from the I2C_DATA register.
- Transmit interrupt/status (XRDY): Generated when the LH needs to put more data in the I2C_DATA register after the transmitted data has been shifted out on the SDA pin. The LH can poll this bit to write the next transmitted data into the I2C_DATA register.

When the interrupt signal is activated, the local host must read the I2C_STAT register to define the type of interrupt, process the request, and then write into this register the right value to clear the interrupt flag.

2.6.6 DMA Events

The I2C module can generate two DMA requests events, read (I2C_DMA_RX) and write (I2C_DMA_TX), that the DMA controller can use to synchronously read received data from the I2C_DATA and write transmitted data to the I2C_DATA register. The DMA read and write requests are generated in a similar manner as RRDY and XRDY, respectively.

The I2C DMA request signals (I2C_DMA_TX and I2C_DMA_RX) are activated for every new 16-bit word to be read or written in the FIFOs.

2.7 Register Map

Table 18 lists the revision, interrupt enable, I2C status, system status, buffer configuration, data counter, and data access registers. Table 19 through Table 26 describe the register bits.

Start address: FFFB 3800

Table 18. Register Map

Name	Addr Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2C_REV	0x00	REV															
I2C_IE	0x04											GC_IE	XRDY_IE	RRDY_IE	ARDY_IE	NACK_IE	AL_IE
I2C_STAT	0x08	SBD			BB	ROVR	XUDF	AAS				GC	XRDY	RRDY	ARDY	NACK	AL
Reserved	0x0C																
I2C_SYSS	0x10																RDONE
I2C_BUF	0x14	RDMA_EN								XDMA_EN							
I2C_CNT	0x18	DCOUNT															
I2C_DATA	0x1C	DATA															
I2C_SYSC	0x20															SRST	
I2C_CON	0x24	I2C_EN	BE			STB	MST	TRX	XA							STP	STT
I2C_OA	0x28	OA															
I2C_SA	0x2C	SA															
I2C_PSC	Ox30	PSC															
I2C_SCLL	0x34	SCLL															
I2C_SCLH	0x38	SCLH															
I2C_SYSTEST	0x3C	ST_EN	FREE	TMODE		SSB								SCL_I	SCL_O	SDA_I	SDA_O

All bits defined as reserved must be written by software with zeros to preserve future compatibility. When read, any reserved bit returns 0. Also, note that it is good software practice to use complete mask patterns for setting or testing bit fields individually within a register.

Table 19. Module Revision Register(I2C_REV)

Bit	Name	Description
15:8	–	Reserved
7:0	REV	Module version number

This read-only register contains the hard-coded revision number of the module. A write to this register has no effect.

Module Revision Number (REV)

This 8-bit field indicates the revision number of the current I2C controller module. Its value is fixed by hardware and corresponds to the RTL revision of this module.

The 4 LSBs indicate a minor revision. The 4 MSBs indicate a major revision.

For example:

- 0x20: Revision 2.0
- 0x21: Revision 2.1

A reset has no effect on the value returned.

Note:

I2C controller with interrupt using interrupt vector register (I2C_IV) is revision 1.x.

I2C controller with interrupt using status register bits (I2C_STAT) is revision 2.x.

Table 20. Interrupt Enable Register(I2C_IE)

Bit	Name	Description
15:6	–	Reserved
5	GC_IE	General call interrupt enable
4	XRDY_IE	Transmit data ready interrupt enable
3	RRDY_IE	Receive data ready interrupt enable
2	ARDY_IE	Register access ready interrupt enable
1	NACK_IE	No acknowledgment interrupt enable
0	AL_IE	Arbitration lost interrupt enable

This R/W register controls the interrupts mask/unmask function.

The following are common to all bits:

When the local host sets a bit location to 1 , an interrupt is signaled to the local host if the corresponding bit location in I2C_STAT (status register) is asserted to 1 by the core of the I2C controller. If it is set to 0, the interrupt is masked and is not signaled to the local host.

- 0: Interrupt disabled
- 1: Interrupt enabled

Value after reset is low (all bits).

Table 21. Status Register(I2C_STAT)

Bit	Name	Description
15	SBD	Single byte data
14:13	–	Reserved
12	BB	Bus busy
11	ROVR	Receive overrun
10	XUDF	Transmit underflow
9	AAS	Address as slave
8:6	–	Reserved
5	GC	General call
4	XRDY	Transmit data ready
3	RRDY	Receive data ready
2	ARDY	Register access ready
1	NACK	No acknowledgment interrupt enable
0	AL	Arbitration lost interrupt enable

This register is composed of read-only and read-/clear-only registers. It provides core status information for interrupt handling and other I^2C control management.

Single Byte Data (SBD)

This read-only bit is set to 1 in slave-receive or master-receive modes when the last byte that was read from the I2C_DATA register contains a single valid byte.

The core clears this bit to 0 when the local host clears the register access ready interrupt flag.

Note:

When SBD = 1, in little-endian data format (I2C_CON:BE = 0), the MSB reads as 0x00, and in big-endian format (I2C_CON:BE = 1), the LSB reads as 0x00.

Whenever the number of bytes to be received is unknown (for example, slave receiver), the LH must poll this bit before clearing the register access ready interrupt flag.

- 0: No action
- 1: Single valid byte in last 16-bit data read

Value after reset is low.

Bus Busy (BB)

This read-only bit indicates the state of the serial bus.

In slave mode, on reception of a start condition, the device sets BB to 1. BB is cleared to 0 after reception of a stop condition.

In master mode, the software controls BB. To start a transmission with a start condition, MST, TRX, and STT must be set to 1 in the I2C_CON register. To end a transmission with a stop condition, STP must be set to 1 in the I2C_CON register. When BB = 1, and STT is set to a 1, a restart condition is generated.

- 0: Bus is free
- 1: Bus is occupied

Value after reset is low.

Receive Overrun (ROVR)

Receive mode only.

This read-only bit indicates whether the receiver has experienced overrun. Overrun occurs when the shift register is full and the receive FIFO is full. An overrun condition does not result in a data loss; the peripheral is just holding the bus (low on SCL) to prevent others bytes from being received.

ROVR is set to 1 when the I^2C recognizes an overrun.

ROVR is clear when reading the I2C_DATA register, or when resetting the I^2C (I2C_CON:I2C_EN = 0).

- 0: Normal operation
- 1: Receiver overrun

Value after reset is low.

Transmit Underflow (XUDF)

Transmit mode only.

This read-only bit indicates whether the transmitter has experienced underflow.

In the master transmit mode, underflow occurs when the shift register is empty, the transmit FIFO is empty, and there are still some bytes to transmit (DCOUNT ≠ 0).

In the slave transmit mode, underflow occurs when the shift register is empty, the transmit FIFO is empty, and there are still some bytes to transmit (read request from external I^2C master).

XUDF is set to 1 when the I^2C recognizes an underflow. The core holds the line till the underflow cause disappears.

XUDF is clear when writing the I2C_DATA register or resetting the I^2C (I2C_CON:I2C_EN = 0).

- 0: Normal operation
- 1: Transmit underflow

Value after reset is low.

Address As Slave (AAS)

The device sets this bit to 1 when it recognizes its own slave address or an address of all zeros (8 bits). The AAS bit is reset to 0 by restart or stop.

- 0: No action
- 1: Address as slave

Value after reset is low.

General Call (GC)

The device sets this read-/clear-only bit to1 if it detects the address of all zeros (8 bits), that is, general call.

When the core sets this bit to 1, an interrupt is signaled to the local host if the interrupt was enabled.

The LH is able to clear this bit only by writing a 1 into this register. Writing 0 has no effect.

When this bit is set to 1, AAS also reads as 1.

- 0: No action
- 1: General call

Value after reset is low.

Transmit Data Ry (XRDY)

Transmit mode only.

This read-/clear-only bit (XRDY) is set to 1 when the I^2C peripheral is a master or slave transmitter, the LH is able to write new data into the I2C_DATA register, and the transmitter still requires new data. A master transmitter requests new data if DCOUNT $\neq 0$, and a slave transmitter requests new data if a read request is received from external master.

The transmitter requests two bytes to be written even if only a single byte is needed. In this case, the other byte needs to be filled with a dummy 0x00 value that is not transmitted over the I^2C line.

When the core sets this bit to 1, an interrupt is signaled to the local host if the interrupt was enabled. The LH can also poll this bit to write new transmitted data into the I2C_DATA register.

The LH is able to clear this bit only by writing a 1 into this register. Writing 0 has no effect.

If the DMA transmit mode is enabled, this bit is not set. Instead, a DMA TX request to the main DMA controller of the system is generated.

- 0: Transmit buffer full (or receiver mode)
- 1: Transmit data ready (for write) and byte is needed

Value after reset is low.

Receive Data Ry (RRDY)

This read-/clear-only RRDY is set to 1 when the LH is able to read new data from the I2C_DATA register. When the core sets this bit to 1, an interrupt is signaled to the local host if the interrupt was enabled. The LH can also poll this bit to read the received data in I2C_DATA register.

The LH is able to clear this bit only by writing a 1 into this register. Writing 0 has no effect.

In interrupt mode, the LH needs to poll this bit after each read to I2C_DATA to ensure that there is no other DATA on the FIFO waiting to be read. Indeed, the RRDY needs to be cleared to 0 in order to receive a new RRDY interrupt.

If the DMA receive mode is enabled, this bit is not set. Instead, a DMA RX request to the main DMA controller of the system is generated.

- 0: Receive buffer empty
- 1: Receive data ready (for read)

Value after reset is low.

Register Access Ry (ARDY)

When set to 1, this read-/clear-only bit indicates that the previously programmed data and command (receive or transmit, master or slave) has been performed and the status bit has been updated. The LH uses this flag to let it know that the I^2C registers are ready to be accessed again (see Table 22).

Table 22. ARDY Set Conditions

Mode	Others	ARDY Set Conditions
Master transmit	STP = 1	DCOUNT = 0
Master receive	STP = 1	DCOUNT = 0 and receiver FIFO empty
Master transmit or receive	STP = 0	DCOUNT passed 0
Slave transmit	–	Stop or restart condition received from master
Slave receive	–	Stop or restart condition and receiver FIFO empty

The LH is able to clear this bit only by writing a 1 into this register. Writing 0 has no effect.

- 0: No action
- 1: Access ready

Value after reset is low.

No Acknowledgment (NACK)

The read-/clear-only NO_ACKNOWLEDGE flag bit is set when the hardware detects NO_ACKNOWLEDGE has been received.

The LH is able to clear this bit only by writing a 1 into this register. Writing 0 has no effect.

- 0: Normal/no action required
- 1: NACK

Value after reset is low.

Arbitration_Lost (AL)

The read-/clear-only ARBITRATION_LOST flag bit is set to 1 when the device in the master transmitter mode senses it has lost an arbitration. This occurs when two or more transmitters start a transmission almost simultaneously, or when the I^2C attempts to start a transfer while BB (bus busy) is 1.

When this is set to 1 due to arbitration lost, the core automatically clears the MST/STP bits in the I2C_CON register and the I^2C becomes a slave receiver.

The LH is able to clear this bit only by writing a 1 to this register. Writing 0 has no effect.

- 0: Normal/no action required
- 1: Arbitration lost

Value after reset is low.

Table 23. System Status Register(I2C_SYSS)

Bit	Name	Description
15:1	−	Reserved
0	RDONE−	Reset Done

Reset Done (RDONE)

This read-only bit indicates the state of the reset in case of hardware reset, global software reset (I2C_SYSC.SRST), or partial software reset (I2C_CON.I2C_EN).

The module must receive all its clocks before it can grant a reset-completed status.

- 0: Internal module reset is ongoing or partially held in reset
- 1: Reset completed

Value after reset is low.

Table 24. Buffer Configuration Register(I2C_BUF)

Bit	Name	Description
15	RDMA_EN	Receive DMA channel enable
14:8	−	Reserved
7	XDMA_EN	Transmit DMA channel enable
6:0	−	Reserved

This R/W register enables DMA transfers.

Receive DMA Channel Enable (RDMA_EN)

When this bit is set to 1, the receive DMA channel is enabled and the core forces the receive data ready status bit (I2C_STAT:RRDY) to 0.

- ❏ 0: Receive DMA channel disabled
- ❏ 1: Receive DMA channel enabled

Value after reset is low.

Transmit DMA Channel Enable (XDMA_EN)

When this bit is set to 1, the transmit DMA channel is enabled and the core forces the transmit data ready status (I2C_STAT:XRDY) bit to 0.

- ❏ 0: Transmit DMA channel disabled
- ❏ 1: Transmit DMA channel enabled

Value after reset is low.

Table 25. Data Counter Register(I2C_CNT)

Bit	Name	Description
15:0	DCOUNT	Data count

This R/W register controls the numbers of bytes in the I^2C data payload.

Data Count (DCOUNT)

Master modes only (receive or transmit).

This 16-bit countdown counter decrements by 1 for every byte received or sent. A write initializes DCOUNT to a saved initial value. A read returns the number of bytes that are yet to be received or sent. A read into DCOUNT returns the initial value only before a start condition and after a stop condition.

When DCOUNT reaches 0, the core generates a stop condition if a stop condition was specified (I2C_CON:STP = 1), and the ARDY status flag is set to 1 in the I2C_STAT register.

If I2C_CON:STP = 0, then the I^2C asserts SCL = 0 when DCOUNT reaches 0. The LH can then reprogram DCOUNT to a new value and resume sending or receiving data with a new start condition (restart). This process repeats until the LH sets the STP to 1 in the I2C_CON register.

The ARDY flag is set each time DCOUNT reaches 0 and DCOUNT is reloaded to its initial value.

In slave mode (receive or transmit), DCOUNT is not used.

0x0: Data counter = 65536 bytes (2^{16})

0x1: Data counter = 1 bytes

↓ ↓

0xFFFF: Data counter = 65535 bytes ($2^{16} - 1$)

DCOUNT value after reset is 0x0000.

Table 26. Data Access Register(I2C_DATA)

Bit	Name	Description
15:0	DATA	Transmit/Receive FIFO data

This register is the entry point for the local host to read data from or write data to the FIFO buffer. The FIFO size is 2 x 16 bits (4 bytes). Bytes within a word are stored and read in little-endian format (I2C_CON:BE = 0) or big-endian format (I2C_CON:BE = 1).

Transmit/Receive FIFO Data Value (DATA)

When read, this register contains the received I^2C data packet (1 or 2 bytes). This register must be accessed 16-bit-wise by the LH. In the case of an odd number of bytes received to read, the upper byte (with I2C_CON.BE = 0) or the lower byte (with I2C_CON.BE = 1) of the last access always reads as 0x00. The LH must check the SBD status bit in I2C_STAT register in order to flush this null byte.

When written, this register contains the byte(s) value(s) to transmit over the I^2C data line (1or 2 bytes). This register must be accessed 16-bit-wise, except for the last byte in case of an odd number of bytes to transmit. The last byte of the data packet can be written using a byte write access or a 16-bit-wise access. In the 16-bit-wise access, the module transmits only the relevant byte, based on the byte counter (I2C_CNT). This feature is useful for DMA access, which supports only one word size per channel.

In SYSTEST loopback mode (I2C_SYSTEST:TMODE = 11), this register is also the entry/receive point for the data.

Values after reset are low (all 16-bits).

A read access when the buffer is empty returns the previous read data value. A write access when the buffer is full is ignored. In both events, the FIFO pointers are not updated and a remote access error (hardware error) is generated (access qualifier). No remote error is generated if the local host performs a 16-bit access if the buffer contains a single byte.

Table 27. I^2C System Configuration Register(I2C_SYSC)

Bit	Name	Description
15:2	–	Reserved
1	SRST	Soft Reset
0	–	Reserved

Soft Reset (SRST)

When this bit is set to 1, all of the module is reset, as for the hardware reset. The core automatically sets this bit to 0, and it is only reset by the hardware reset. During reads, it always returns 0.

- 0: Normal mode
- 1: The module is reset

Values after reset is low.

Table 28. I2C Configuration Register(I2C_CON)

Bit	Name	Description
15	I2C_EN	I2C module enable
14	BE	Big endian mode
13:12	–	Reserved
11	STB	Start byte mode (master mode only)
10	MST	Master/slave mode
9	TRX	Transmitter/Receiver mode (master mode only)
8	XA	Expand address
7:2	–	Reserved
1	STP	Stop condition (master mode only)
0	STT	Start condition (master mode only)

Active Transfer Phase

During an active transfer phase (STT has been set to 1), no modification must be done in this register. Changing it may result in an unpredictable behavior.

I2C Module Enable (I2C_EN)

When this bit is set to 0, the I2C controller is not enabled and reset. When 0, the receive and transmit FIFOs are cleared and all status bits are set to their default values. None of the configuration registers (I2C_IE, I2C_BUF, I2C_CNT, I2C_CON, I2C_OA, I2C_SA, I2C_PSC, I2C_SCLL and I2C_SCLH) are reset, all keep their initial values, and all can be accessed. The LH must set this bit to 1 for normal operation.

- 0: I2C controller in reset
- 1: I2C module enabled

Value after reset is low.

Big Endian (BE)

When this bit is 0 (default), the FIFO is accessed in little-endian format. In transmit mode, the LSB (I2C_DATA [7:0]) is transmitted first, and the MSB (I2C_DATA [15:8]) is transmitted in second position over the I^2C line. Conversely, in receive mode, the first, or odd, byte received (1,3, 5…) is stored in the LSB position, and the second, or even, byte received is stored in the MSB position.

When the LH sets this bit to a 1, the FIFO is accessed in big-endian format. In transmit mode, the MS byte (I2C_DATA [15:8]) is transmitted first and the LSB (I2C_DATA [7:0]) is transmitted in second position over the I^2C line. Conversely, in receive mode, the first, or odd, byte received (1,3, 5…) is stored in the MS byte position, and the second, or even, byte received is stored in the LSB position.

- ❏ 0: Little-endian mode
- ❏ 1: Big-endian mode

Value after reset is low.

Start Byte (STB)

Master mode only.

The local host sets the start-byte mode bit to 1 to configure the I^2C in start byte mode (I2C_SA = 00000001). See the Philips I^2C specification Version 2.1 for more details.

- ❏ 0: Normal mode
- ❏ 1: Start byte mode

Value after reset is low.

Master/Slave Mode (MST)

When this bit is cleared, the I^2C controller is in the slave mode and the serial clock (SCL) is received from the master device.

When this bit is set, the I^2C controller is in the master mode and it generates the serial clock.

This bit is automatically cleared at the end of the transfer on a detected stop condition and in case of arbitration lost.

- ❏ 0: Slave mode
- ❏ 1: Master mode

Value after reset is low.

Transmitter/Receiver Mode (TRX)

Master mode only.

When this bit is cleared, the I2C controller is in the receiver mode, and data on data line SDA is shifted into the receiver FIFO and can be read from the I2C_DATA register.

When this bit is set, the I2C controller is in the transmitter mode, and the data written in the transmitter FIFO via I2C_DATA is shifted out on data line SDA.

- 0: Receiver mode
- 1: Transmitter mode

Value after reset is low. The operating modes are defined as shown in Table 29.

Table 29. I2C Controller Transmitter/Receiver Operating Modes

MST	TRX	Operating Modes
0	x	Slave receiver
0	x	Slave transmitter
1	0	Master receiver
1	1	Master transmitter

Expand Address (XA)

When set, this bit expands the address to 10-bit.

- 0: 7-bit address mode
- 1: 10-bit address mode

Value after reset is low.

Stop Condition (STP)

Master mode only.

The local host is able to set this bit to a 1 to generate a stop condition. The hardware resets it to 0 after the stop condition has been generated. The stop condition is generated when DCOUNT passes 0.

When this bit is not set to 1 before the end of the transfer (DCOUNT = 0), the stop condition is not generated and the SCL line is held to 0 by the master, which can restart a new transfer by setting the STT bit to 1.

- 0: No action or stop condition detected
- 1: Stop condition queried

Value after reset is low.

Start Condition (STT)

Master mode only.

The local host is able to set this bit to a 1 to generate a start condition. The hardware resets it to 0 after the start condition has been generated. The start/stop bits can be configured to generate different transfer formats.

- 0: No action or start condition generated
- 1: Start

Value after reset is low.

Table 30. STT Register Values

STT	STP	Conditions	Bus Activities
1	0	Start	S−A−D
0	1	Stop	P
1	1	Start−Stop (DCOUNT = n)	S−A−D..(n)..D−P
1	0	Start (DCOUNT = n)	S−A−D..(n)..D

DCOUNT is the data count value in the I2C_CNT register.

Table 31. I^2C Own Address Register(I2C_OA)

Bit	Name	Description
15:10	−	Reserved
9:0	OA	Own address

This register specifies the module I^2C 7-bit or 10-bit address (own address).

2.7.1 Own Address (OA)

This field specifies either:

- A 10-bit address coded on OA [9:0] when XA (expand address, I2C_CON [8]) is set to 1.
- A 7-bit address coded on OA [6:0] when XA (expand address, I2C_CON [8]) is set to 0. In this case, application software must set OA [9:7] bits to 000.

Values after reset is low (all 10 bits).

Table 32. I2C_SA: I2C Slave Address Register

Bit	Name	Description
15:10	−	Reserved
9:0	SA	Slave address

This register specifies the addressed I2C module 7-bit or 10-bit address (slave address).

Slave Address (SA)

This field specifies either:

- A 10-bit address coded on SA [9:0] when XA (expand address, I2C_CON [8]) is set to 1.
- A 7-bit address coded on SA [6:0] when XA (expand address, I2C_CON [8]) is set to 0. In this case, application software must set SA [9:7] bits to 000.

Values after reset are high (all 10 bits).

Table 33. I2C Clock Prescaler Register(I2C_PSC)

Bit	Name	Description
15:8	−	Reserved
7:0	PSC	Prescale sampling clock divider value

This register specifies the internal clocking of the I2C peripheral core.

I2C_PSC [7−0]: Prescale sampling clock divider value (PSC)

The core uses this 8-bit value to divide the system clock (SCLK) and generate its own internal sampling clock (ICLK). The core logic is sampled at the clock rate of the system clock for the module, divided by (PSC+1).

0x0: Divide by 1

0x1: Divide by 2

↓ ↓

0xFF: Divide by 256

Values after reset are low (all 8 bits).

Table 34. I2C SCL Low Time Control Register(I2C_SCLL)

Bit	Name	Description
15:8	–	Reserved
7:0	SCLL	SCL low time

This register determines the SCL low time value when master.

SCL Low Time (SCLL)

Master mode only.

This 8-bit value generates the SCL low time value (t_{LOW}) when the peripheral is operated in master mode.

The SCL low time depends on the I2C_PSC value and the ICLK time period (internal sampling clock rate):

- When I2C_PSC = 0, t_{LOW} = (SCLL+7) * ICLK time period
- When I2C_PSC = 1, t_{LOW} = (SCLL+6) * ICLK time period
- When I2C_PSC > 1, t_{LOW} = (SCLL+5) * ICLK time period

The different values to compute the SCL low time are due to the synchronization stage and noise filter on the SCL line.

Values after reset are low (all 8 bits).

Table 35. I2C SCL High Time Control Register(I2C_SCLH)

Bit	Name	Description
15:8	–	Reserved
7:0	SCLH	SCL high time

This register determines the SCL high time value when master.

SCL High Time (SCLH)

Master mode only.

This 8-bit value generates the SCL high-time value (t_{HIGH}) when the peripheral is operated in master mode.

The SCL high time depends on the I2C_PSC value and the ICLK time period (internal sampling clock rate):

- When I2C_PSC = 0, t_{HIGH} = (SCLH+7) * ICLK time period
- When I2C_PSC = 1, t_{HIGH} = (SCLH+6) * ICLK time period
- When I2C_PSC > 1, t_{HIGH} = (SCLH+5) * ICLK time period

The different values to compute the SCL high time are due to the synchronization stage and noise filter on the SCL line.

Values after reset are low (all 8 bits).

Table 36. System Test Register (I2C_SYSTEST)

Bit	Name	Description
15	ST_EN	System test enable
14	FREE	Free running mode (on breakpoint)
13:12	TMODE	Test mode select
11	SSB	Set status bits
10:4	–	Reserved
3	SCL_I	SCL line sense input value
2	SCL_O	SCL line drive output value
1	SDA_I	SDA line sense input value
0	SDA_O	SDA line drive output value

System Test Register

Never set this register for normal I^2C operation.

CAUTION

This register facilitates system-level tests by overriding some of the standard functional features of the peripheral. It can permit the test of SCL counters, control the signals that connect to I/O pins, or create digital loopback for self-test when the module is configured in system test (SYSTEST) mode. It also provides stop/no-stop functionality in debug mode.

System Test Enable (ST_EN)

This bit must be set to 1 to permit other system test registers bits to be set.

- 0: Normal mode
- 1: System test enabled

Value after reset is low.

Free Running Mode After Breakpoint (FREE)

This bit determines the state of the I^2C controller when a breakpoint is encountered in the HLL debugger.

This bit can be set independently of the ST_EN value.

FREE = 0: If the I^2C controller is a master, it stops immediately after the completion of the ongoing bit transfer. If the I^2C controller is a slave, it stops during the data phase transfer when one byte is completely transmitted/ received.

FREE = 1: The I^2C runs free.

- 0: Stop mode (on breakpoint condition)
- 1: Free running mode

Value after reset is low.

Test Mode Select (TMODE)

In normal functional mode (ST_EN = 0), these bits are *don't care*. They always read as 00 and a write is ignored.

In system test mode (ST_EN = 1), these bits can be set according to the following table to permit various system tests (see Table 37).

Table 37. Test Mode Select

TMODE	Mode
00	Functional mode (default)
01	Reserved
10	Test of SCL counters (SCLL, SCLH, PSC)
11	Loop back mode select + SDA/SCL IO mode select

Values after reset is low (2 bits).

- SCL counter test mode: In this mode, the SCL pin is driven with a permanent clock as if mastered, with the parameters set in the I2C_PSC, I2C_SCLL, and I2C_SCLH registers.
- Loopback mode: In the master transmit mode only, data transmitted out of the I2C_DATA register (write action) is received in the same I2C_DATA register via an internal path through the 1-deep FIFO buffers. The DMA and interrupt requests are normally generated if enabled.
- SDA/SCL IO mode: In this mode, the SCL IO and SDA IO are controlled via the I2C_SYSTEST [3:0] register bits.

Set Status Bits (SSB)

Writing 1 into this bit also sets the six read/clear-only status bits contained in the I2C_STAT register (bits 5:0) to 1.

Writing 0 into this bit does not clear status bits that are already set; only writing 1 into a set status bit can clear it. This bit must be cleared before attempting to clear a status bit.

- 0: No action
- 1: Set all six bits in I2C_STAT [5:0] to 1

Value after reset is low.

SCL Line Sense Input Value (SCL_I)

In normal functional mode (ST_EN = 0), this read-only bit always reads 0.

In system test mode (ST_EN = 1 & TMODE = 11), this read-only bit returns the logical state taken by the SCL line (either 1 or 0).

Value after reset is low.

SCL Line Drive Output Value (SCL_O)

In normal functional mode (ST_EN = 0), this bit is *don't care*. It always reads 0 and a write is ignored.

In system test mode (ST_EN = 1 & TMODE = 11), a 0 forces a low level on the SCL line, and a 1 puts the I^2C output driver to a high impedance state.

- 0: Forces 0 on the SCL data line
- 1: SCL output driver in HiZ state

Value after reset is low.

SDA Line Sense Input Value (SDA_I)

In normal functional mode (ST_EN = 0), this read-only bit always reads 0.

In system test mode (ST_EN = 1 & TMODE = 11), this read-only bit returns the logical state taken by the SDA line (either 1 or 0).

Value after reset is low.

SDA Line Drive Output Value (SDA_O)

In normal functional mode (ST_EN = 0), this bit is *don't care*. It reads as 0 and a write is ignored.

In system test mode (ST_EN = 1 & TMODE = 11), a 0 forces a low level on the SDA line and a 1 puts the I^2C output driver to a high impedance state.

- 0: Forces 0 on the SDA data line
- 1: SDA output driver in Hi-Z state

Value after reset is low.

2.8 Programming Guidelines

2.8.1 Main Program

Module Configuration Before Enabling the Module

Step 1: Program the prescaler to obtain an approximately 12-MHz I^2C module clock (I2C_PSC = x; this value is to be calculated and is dependent on the system clock frequency).

Step 2: Program the I^2C clock to obtain 100K bps or 400K bps (I2C_SCLL = x and I2C_SCLH = x; these values are to be calculated and are dependent on the system clock frequency).

Step 3: Configure its own address (I2C_OA = x).

Step 4: Take the I^2C module out of reset (I2C_CON:I2C_EN = 1).

Initialization Procedure

Step 1: Configure the I^2C mode register (I2C_CON) bits.

Step 2: If using interrupt for transmit/receive data, enable interrupt masks (I2C_IE).

Step 3: If using DMA for transmit/receive data, enable the DMA (I2C_BUF) and program the DMA controller.

Configure Slave Address and Data Counter Registers

In master mode, configure the slave address (I2C_SA = x) and the number of bytes associated with the transfer (I2C_CNT = x).

Initiate a Transfer

Poll the bus busy (BB) bit in the I^2C status register (I2C_STAT). If it is cleared to 0 (bus not busy), configure START/STOP (I2C_CON:STT / I2C_CON:STP) condition to initiate a transfer.

Poll Receive Data

Poll the receive data ready interrupt flag bit (RRDY) in the I^2C status register (I2C_STAT). Use the RRDY interrupt or the DMA to read the receive data in the data receive register (I2C_DATA).

Poll Transmit Data

Poll the transmit data ready interrupt flag bit (XRDY) in the I2C status register (I2C_STAT). Use the XRDY interrupt or the DMA to write data into the data transmit register (I2C_DATA).

2.9 Interrupt Subroutines

Step 1: Test for arbitration lost and resolve accordingly.

Step 2: Test for no-acknowledge and resolve accordingly.

Step 3: Test for register access ready and resolve accordingly.

Step 4: Test for receive data and resolve accordingly.

Step 5: Test for transmit data and resolve accordingly.

2.10 Flow Diagrams

Figure 25. Setup Procedure

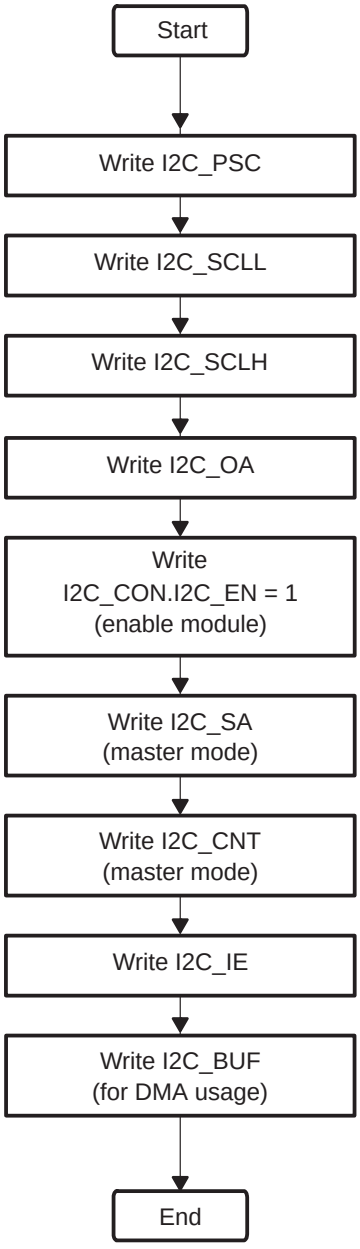


Figure 26. Master Transmitter Mode, Polling

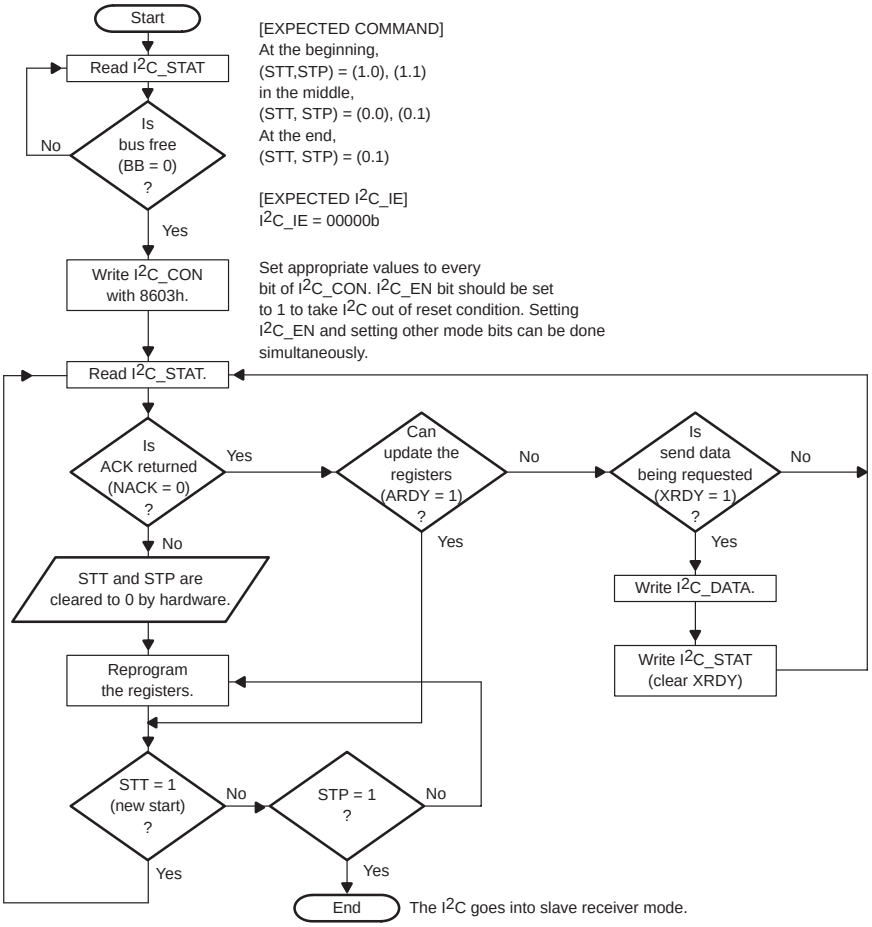


Figure 27. Master Receiver Mode, Polling

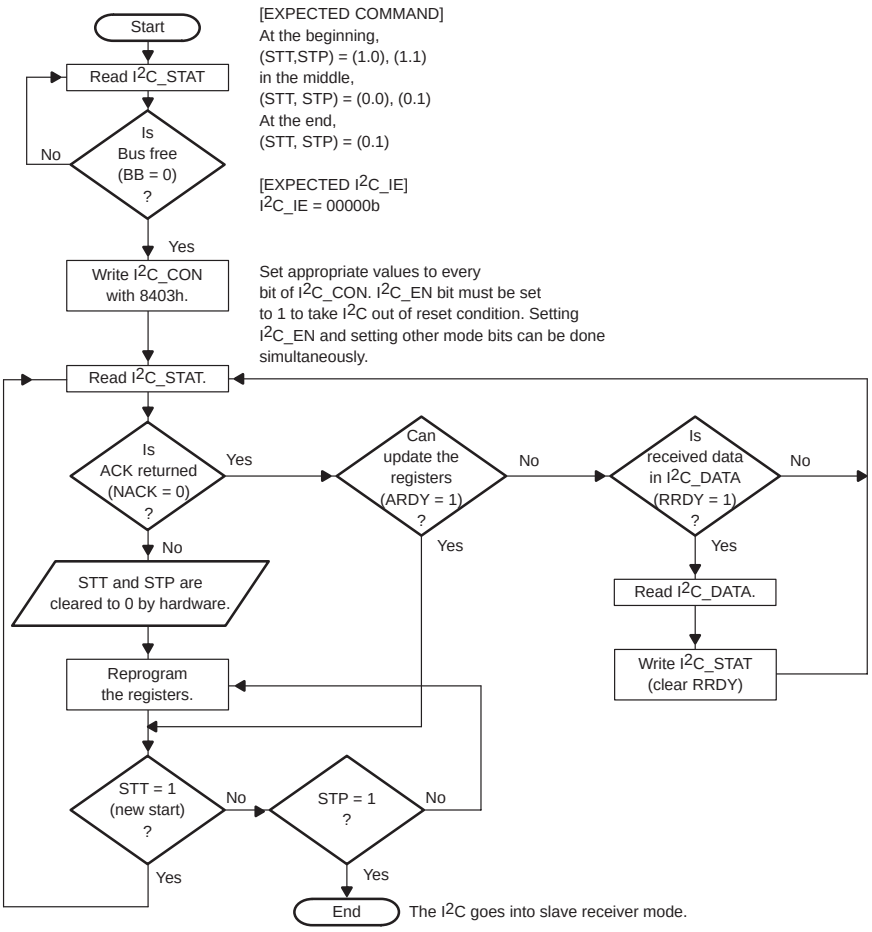


Figure 28. Master Transmitter Mode, Interrupt

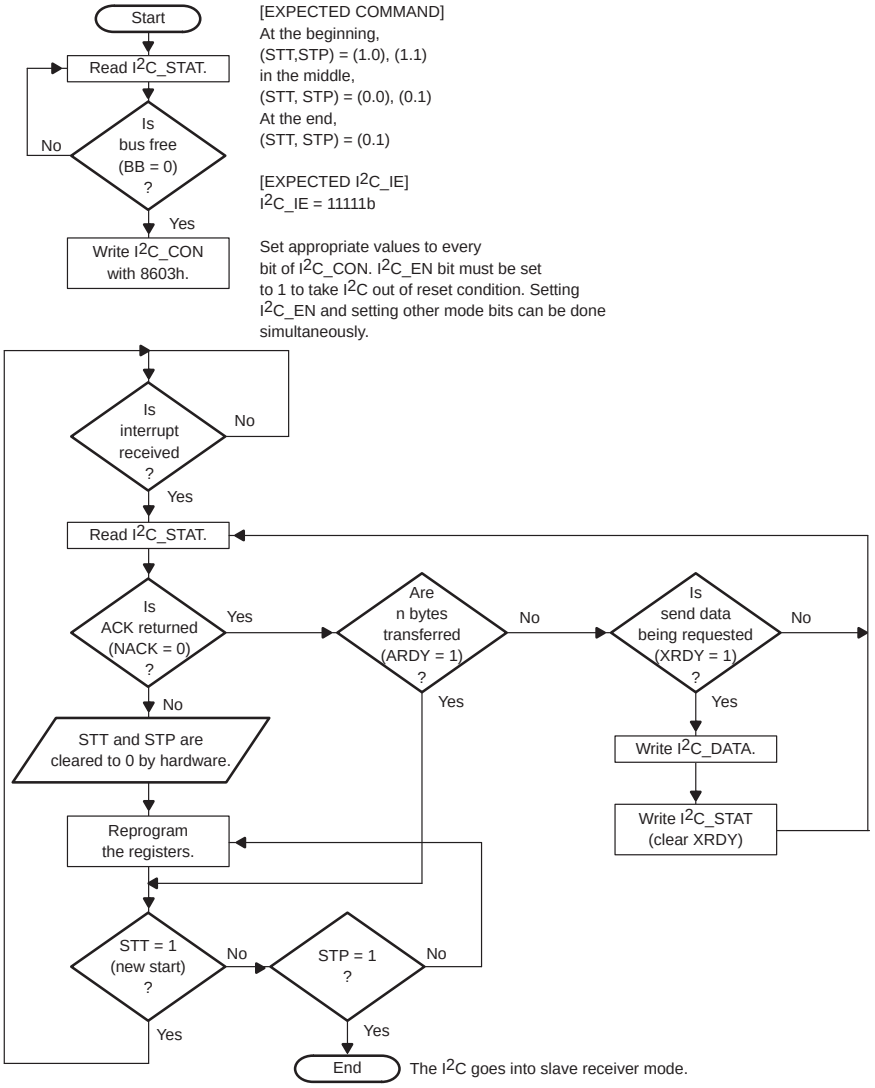


Figure 29. Master Receiver Mode, Interrupt

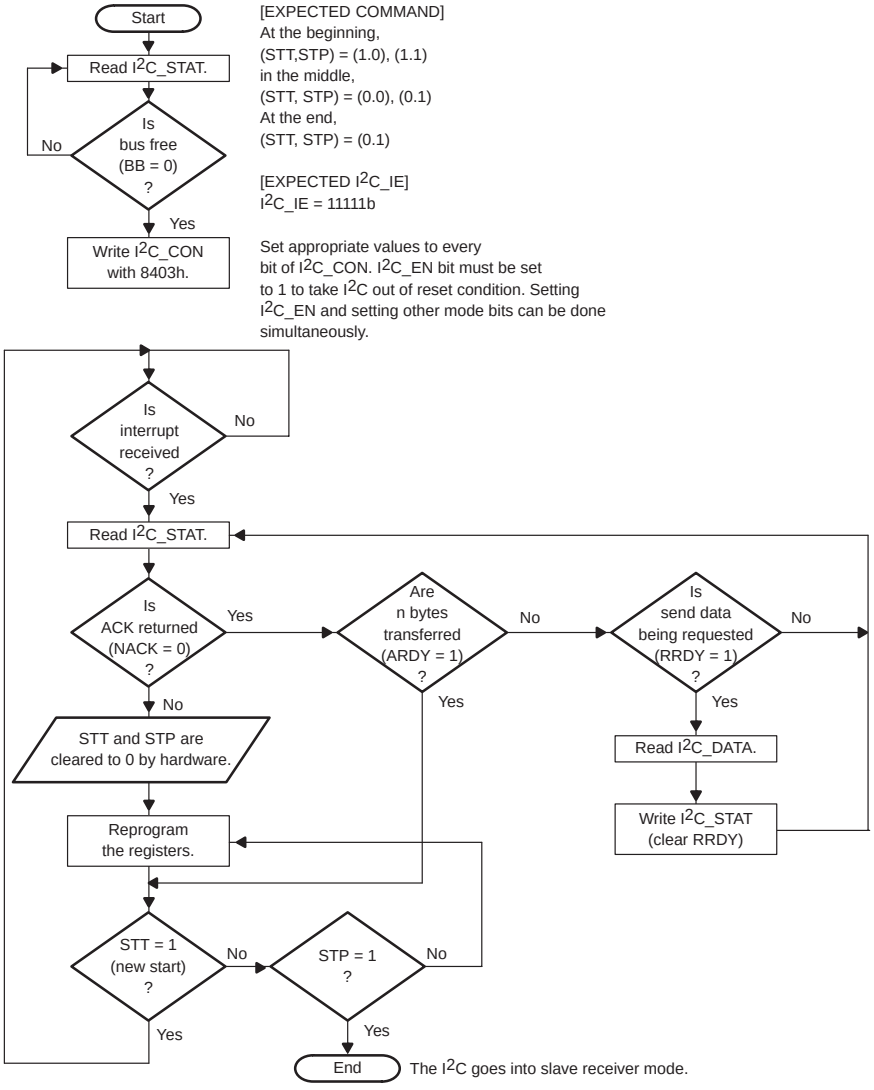


Figure 30. Master Transmitter Mode, DMA

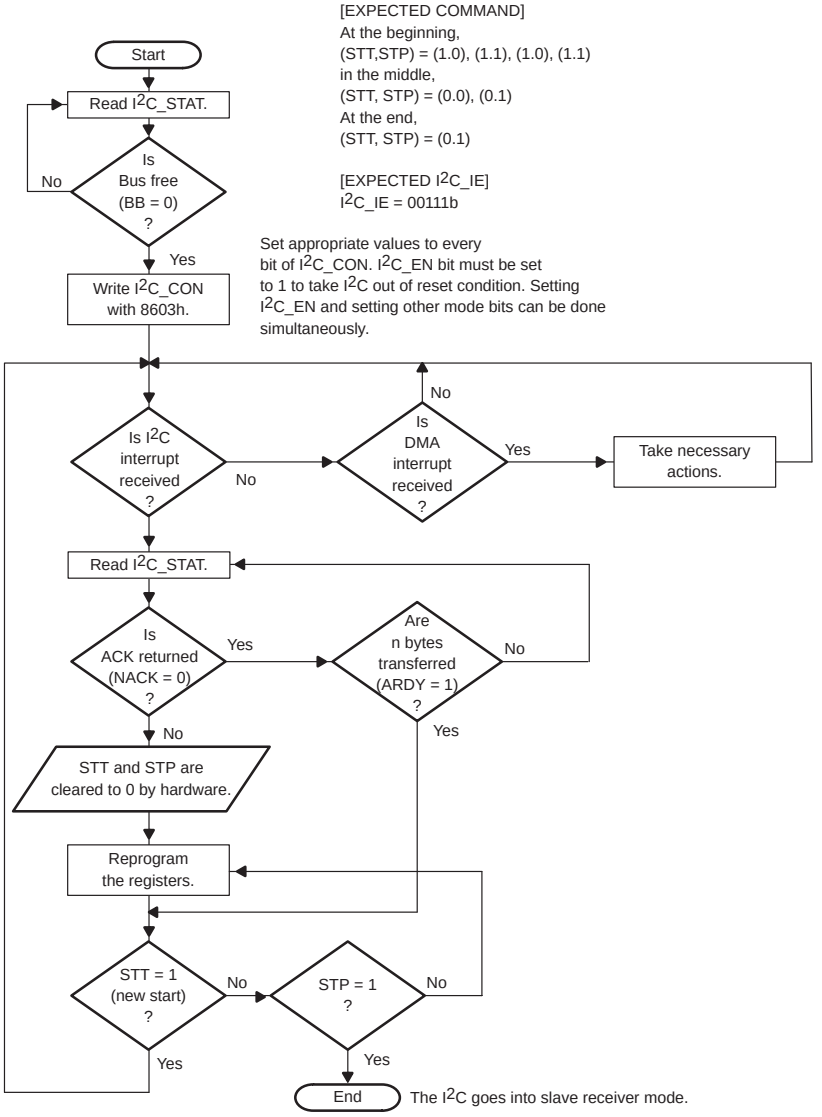


Figure 31. Master Receiver Mode, DMA

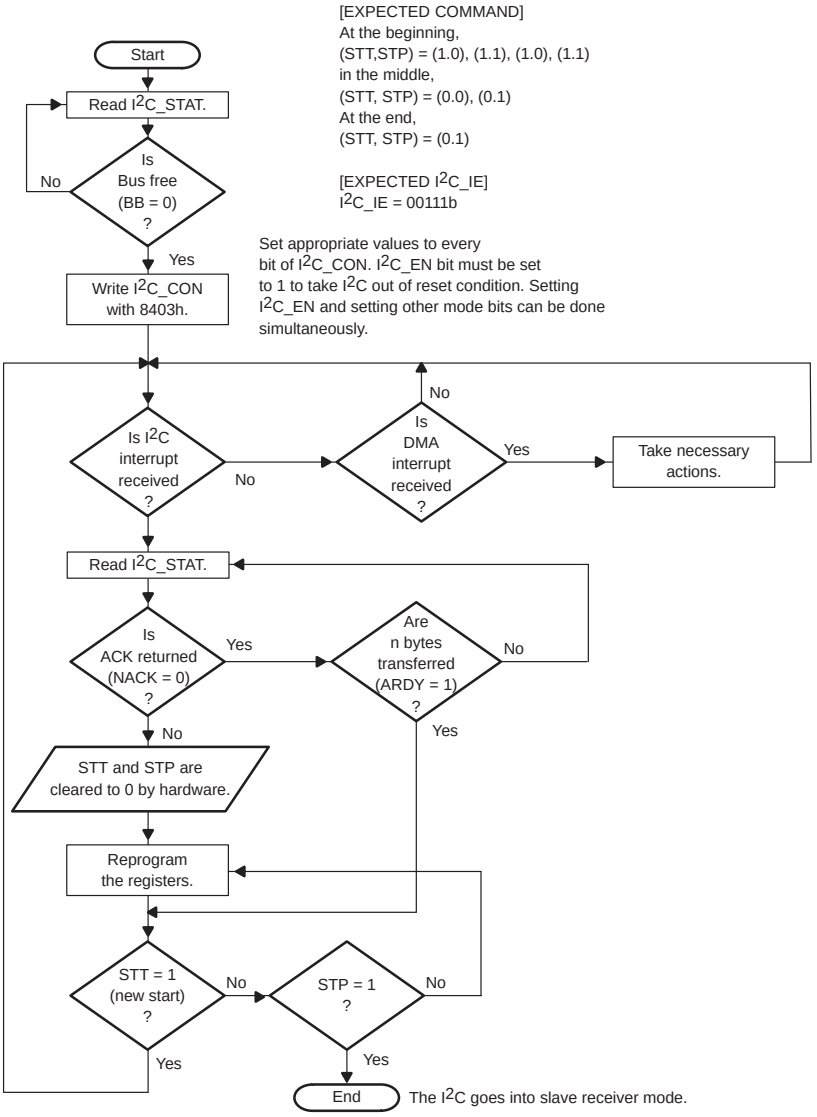


Figure 32. Slave Transmitter/Receiver Mode, Polling

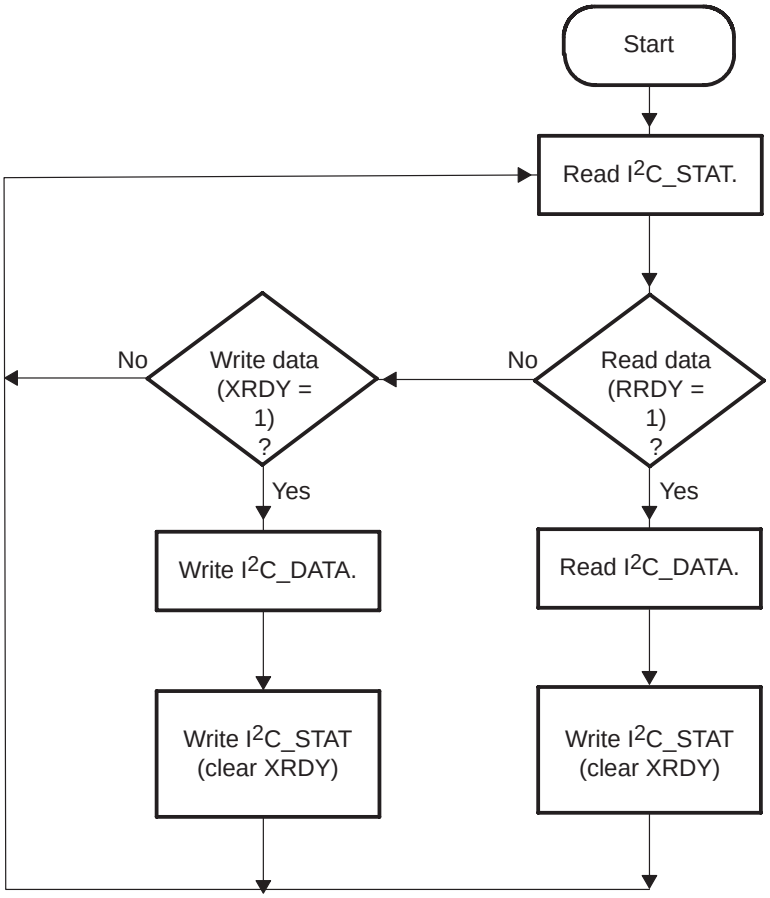
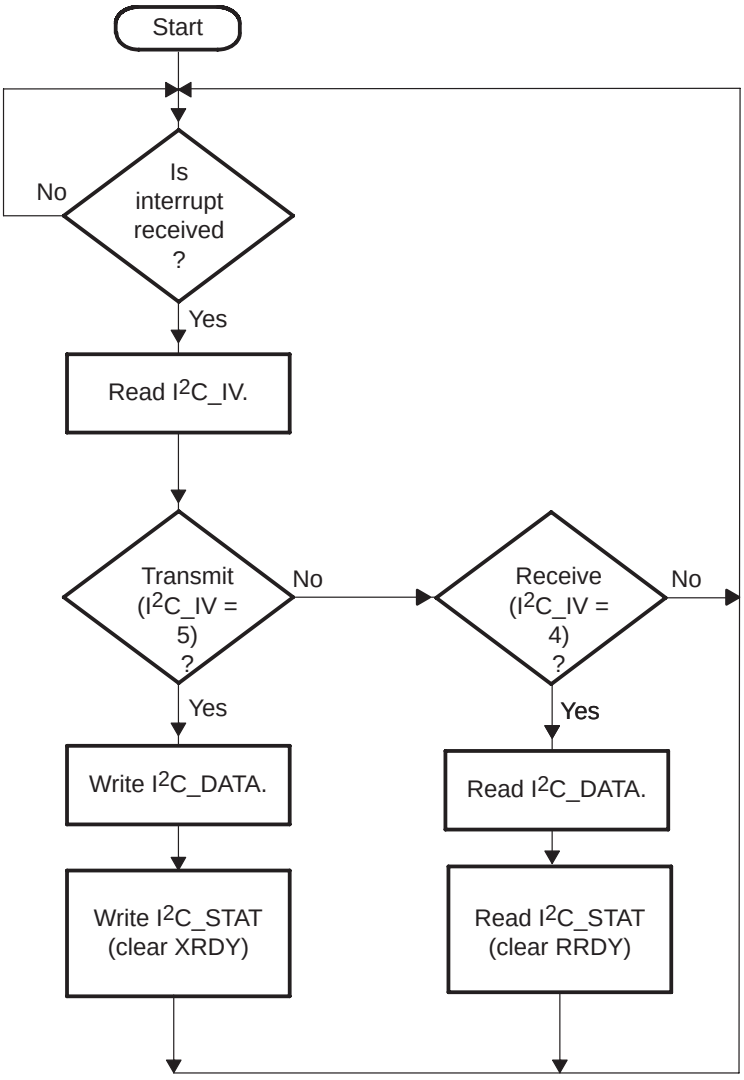


Figure 33. Slave Transmitter/Receiver Mode, Interrupt

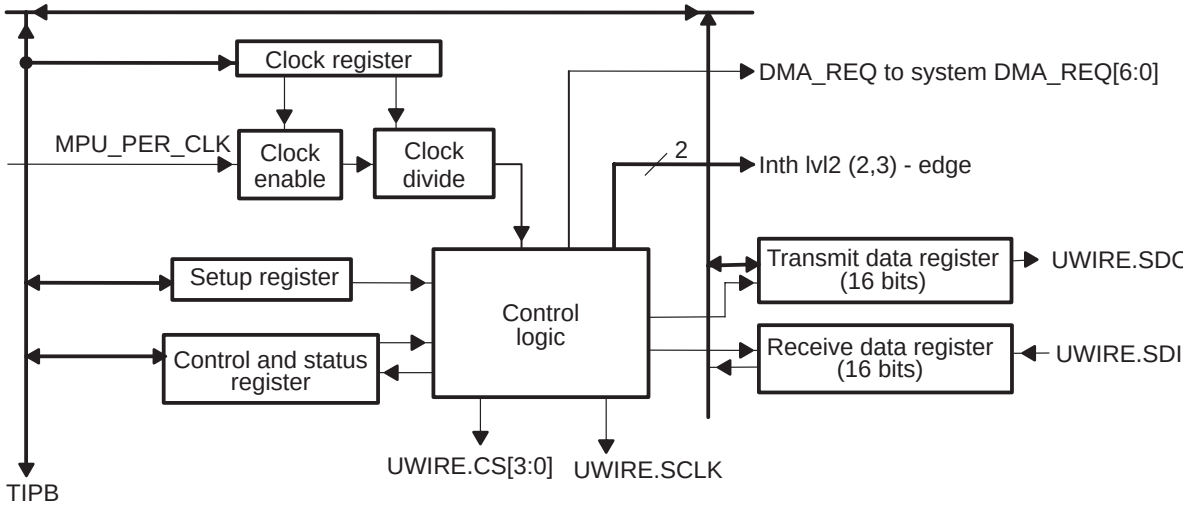


3 MicroWire Interface

This serial synchronous interface can drive up to four serial external components. For the external devices, this interface is compatible with the µWire standard and is seen as the master (see Figure 34).

A transmit DMA mode is available.

Figure 34. Block Diagram



3.1 MicroWire Registers

Start address in the peripheral range (hex): FFFB:3000

Table 38 lists the MicroWire registers. Table 39 through Table 46 describe the individual registers.

Table 38. MicroWire Registers

Register	Description	R/W	Size	Address	Offset
TDR	Transmit data	W	16 bits	FFFB:3000	0x00
RDR	Receive data	R	16 bits	FFFB:3000	0x00
CSR	Control and status	R/W	16 bits	FFFB:3000	0x04
SR1	Setup 1	R/W	16 bits	FFFB:3000	0x08
SR2	Setup 2	R/W	16 bits	FFFB:3000	0x0C
SR3	Setup 3	R/W	16 bits	FFFB:3000	0x10
SR4	Setup 4	R/W	16 bits	FFFB:3000	0x14
SR5	Setup 5	R/W	16 bits	FFFB:3000	0x18

Table 39. Transmit Data Register (TDR)

Base Address = 0xFFFB 3000, Offset = 0x00			
Bit	**Name**	**Function**	**Reset**
15:0	TD	Data to transmit	Undefined

Note: MSB (bit 15) is the first transmitted bit.

Whatever its size, the word is aligned on the most-significant bit (MSB) side.

Table 40. Receive Data Register (RDR)

Base Address = 0xFFFB 3000, Offset = 0x00			
Bit	**Name**	**Function**	**Reset**
15:0	RD	Received data	Undefined

Note: LSB (bit 0) is the last received bit.

Whatever its size, the word is aligned on the least-significant bit (LSB) side.

Table 41. Control and Status Register (CSR)

Base Address = 0xFFFB 3000, Offset = 0x04			
Bit	**Name**	**Function**	**Reset**
15	RDRB	RDRB bit at 1 indicates that the receive (RDR) is full. When the controller reads the content of the RDR, this bit is cleared. This bit is read only.	0
14	CSRB	CSRB bit at 0 indicates that the control and status (CSR) is ready to receive new data. After starting a µWire transfer with the CSR, this bit is set to 1. When the corresponding action has been done, CSRB is reset. This bit is controlled by a µWire internal state machine running on the F_INT internal clock (12 MHz/N). If the CSR is read just after being written, and the MPU is running at high frequency (60 MHz or 120 MHz, for instance) compared to the internal clock, the CSRB status bit may still be low for the first read access. The CSRB latency is 0 if the transfer was initiated by modifying the CS_CMD bit, but it can be 0−3 cycles if initiated by the START bit. Suggested workarounds are a) to have a few NOPs between initiating a µWire transfer and checking CSRB status or, b) to check that CSRB first has a high value on an initial read before it goes low on a subsequent read. This bit is read only.	0

Table 41. Control and Status Register (CSR) (Continued)

Base Address = 0xFFFB 3000, Offset = 0x04			
Bit	**Name**	**Function**	**Reset**
13	START	1: Start a write and/or a read process. This bit is automatically reset by internal logic when a write or a read process is activated. Send NB_BITS_WR bits (contained in TDR) to the serial output DO. If NB_BITS_WR is equal to zero, then the write process is not started. Receive NB_BITS_RD bits from the serial input DI and store them in RDR.	0
12	CS_CMD	1: Set the chip-select of the selected device to its active level.	0
11:10	INDEX	Index of the external device 00: CS0 01: CS1 10: CS2 11: CS3	Undefined
9:5	NB_BITS_WR	Number of bits to transmit	Undefined
4:0	NB_BITS_RD	Number of bits to receive	Undefined

Table 42. Setup Register 1 (SR1)

Base Address = 0xFFFB 3000, Offset = 0x08			
Bit	**Name**	**Function**	**Reset**
11	CS1_CHK	Idem CS0_CHK. Used when the CS1 is selected.	0xX (undefined)
10:9	CS1_FRQ	Defines the frequency of the serial clock SCLK when CS1 is selected 00 : F_INT/2 01 : F_INT/4 10 : F_INT/8 11 : undefined	0xX (undefined)
8	CS1CS_LVL	Defines the active level of the CS1 chip-select.	0x0
7	CS1_EDGE_WR	Idem CS0_EDGE_WR when CS1 is selected.	0xX (undefined)
6	CS1_EDGE_RD	Idem CS0_EDGE_RD when CS1 is selected.	0xX (undefined)

Note: Content of this register must not be changed when a read or write process is running.

Table 42. Setup Register 1 (SR1) (Continued)

Base Address = 0xFFFB 3000, Offset = 0x08			
Bit	**Name**	**Function**	**Reset**
5	CS0_CHK	Before activating a write process, checks if external device is ready. 0: No check is done and the write process is immediately executed. 1: If DI signal is low, the interface considers the external component busy; if DI is high, the interface considers the first external component ready and starts the write process. Used when CS0 is selected.	Undefined
4:3	CS0_FRQ	Defines the frequency of the serial clock SCLK when CS0 is selected (F_INT is the frequency of the internal clock). 00: F_INT/2 01: F_INT/4 10: F_INT/8 11: Undefined	Undefined
2	CS0CS_LVL	Defines the active level of the chip-select by CS0	0
1	CS0_EDGE_WR	When CS0 is selected, defines the active edge of the serial clock SCLK used to write data to the serial input D0. (Output data is generated on this edge) 0: Falling (serial clock not inverted) 0: Rising (when serial clock inverted) 1: Rising (serial clock not inverted) 1: Falling (when serial clock inverted)	Undefined
0	CS0_EDGE_RD	When CS0 is selected, defines the active edge of the serial clock SCLK used to read data from the serial input DI. (Input data is strobed on this edge) 0: Falling (serial clock not inverted) 0: Rising (when serial clock inverted) 1: Rising (serial clock not inverted) 1: Falling (when serial clock inverted)	Undefined

Note: Content of this register must not be changed when a read or write process is running.

Table 42 sets up the serial interface for the first and second external components.

Table 43. Setup Register 2 (SR2)

Base Address = 0xFFFB 3000, Offset = 0x0C			
Bit	**Name**	**Function**	**Reset**
11	CS3_CHK	Same as CS0_CHK. Used when CS3 is selected.	Undefined
10:9	CS3_FRQ	Defines the frequency of the serial clock SCLK when CS3 is selected 00: F_INT/2 01: F_INT/4 10: F_INT/8 11: Undefined	Undefined
8	CS3CS_LVL	Defines the active level of the CS3 chip-select	0
7	CS3_EDGE_WR	Same as CS0_EDGE_WR when CS3 is selected	Undefined
6	CS3_EDGE_RD	Same as CS0_EDGE_RD when CS3 is selected	Undefined
5	CS2_CHK	Idem CS0_CHK. Used when the CS2 is selected.	0xX (undefined)
4:3	CS2_FRQ	Defines the frequency of the serial clock SCLK when CS2 is selected (F_INT is the frequency of the internal clock): 00 : F_INT/2 01 : F_INT/4 10 : F_INT/8 11 : Undefined	0xX (undefined)
2	CS2CS_LVL	Defines the active level of the CS2 chip-select.	0x0
1	CS2_EDGE_WR	Idem CS0_EDGE_WR when CS2 is selected.	0xX (undefined)
0	CS2_EDGE_RD	Idem CS0_EDGE_RD when CS2 is selected.	0xX (undefined)

Note: Content of this register must not be changed when a read or write process is running.

Table 43 sets up the serial interface for the first and second external components.

Table 44. Setup Register 3 (SR3)

This register sets up the serial interface for the internal clock.

Base Address = 0xFFFB 3000, Offset = 0x10			
Bit	**Name**	**Function**	**Reset**
2:1	CK_FREQ	Defines the frequency of the internal clock, F_INT, when CLK_EN = 1. All the internal logic is controlled by F_INT (F is the frequency of the external input clock). 00: F/2 01: F/4 10: F/7 11: F/10	00
0	CLK_EN	Switch off the clock if 0. Switch on the clock if 1.	0

Note: Content of this register must not be changed when a read or write process is running.

Table 45. Setup Register 4 (SR4) (R/W)

This register sets up the serial clock polarity.

Base Address = 0xFFFB 3000, Offset = 0x14			
Bit	**Name**	**Function**	**Reset**
0	CLK_IN	Serial clock is not inverted if 0. Serial clock is inverted if 1.	0

Note: Content of this register must not be changed when a read or write process is running.

Table 46. Setup Register 5 (SR5) (R/W)

Base Address = 0xFFFB 3000, Offset = 0x18			
Bit	**Name**	**Function**	**Reset Value**
3	CS_TOGGLE_TX_EN	CS_TOGGLE_TX_EN is possible only in autotransmit mode. When in autotransmit mode with CS_TOGGLE_TX_EN inactive, the CS does not go to its active level automatically. Control the CS with the CS CMD bit of the control and status register (CSR) in the software. CS_toggle transmit mode is disabled if 0. CS_toggle transmit mode is enabled if 1.	0
2	AUTO_TX_EN	In autotransmit mode, the CS_CMD and START bits of the control and status register (CSR) are not used. A hardware state machine detects a TXD write and automatically sets the programmed CS to its active value, and then starts the transmission. The CS CMD and the START bits in the control and status register (CSR) are not updated during autotransmit. Autotransmit mode is disabled if 0. Autotransmit mode is enabled if 1.	0
1	IT_EN	In IT mode, an interrupt is generated each time a word has been transferred or received. This interrupt is a low-level interrupt. A status register (IST) allows the CPU to know which interrupt (receive and/or transmit) occurred. IT mode is disabled if 0. IT mode is enabled if 1.	0
0	DMA_TX_EN	DMA transmit mode is disabled if 0. DMA transmit mode is enabled if 1.	0

Note: Content of this register must not be changed when a read or write process is running.

Set up the DMA, IT, AUTO_TX, and CS_TOGGLE modes in this register.

In DMA mode, a DMA request is initiated each time a transmission slot is available.

The maximum word size in DMA mode is 16 bits.

Note:

You cannot use another CS in normal or DMA modes when a DMA mode is active on one specific CS.

Note:

To use the µWire in DMA transmit mode, DMA_EN and AUTO_TX_EN must be enabled, and IT_EN is best disabled. The AUTO_TX_EN can be active when DMA_EN is disabled.

3.2 Protocol Description

The serial port must be configured in order to use the setup registers.

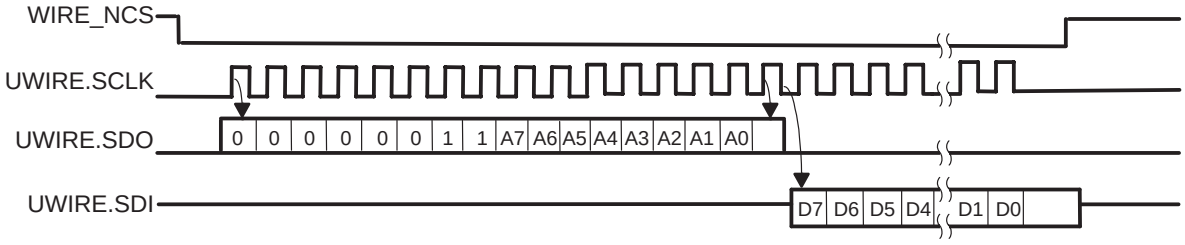
This interface drives only one device at a given time. Therefore, the chip-select of the selected device must be set to its active level before starting any read or write process.

After loading the transmit data register (TDR), the write process is activated by setting the START bit to 1 and by writing a value different from zero to the NB_BITS_WR field.

A read process is always simultaneous with a write process, which means that at every serial clock (SCLK) cycle, data is read. After having finished a write process (if necessary), a number (defined by NB_BITS_RD) of SCLK cycles is generated to allow storage of data from the serial input DI.

The transmitted data word is shifted out on the rising or falling edge of the serial clock, according to the value of the *_EDGE_WR bits of the setup registers. The received data word is shifted in on the falling or rising edge of the serial clock, according to the value of the *_EDGE_RD bits of the setup registers. When *_EDGE_WR and *_EDGE_RD bits have the same value, it is assumed that the device behavior is the one shown in Figure 35. Otherwise, the required behavior of the external device is as shown in Figure 36.

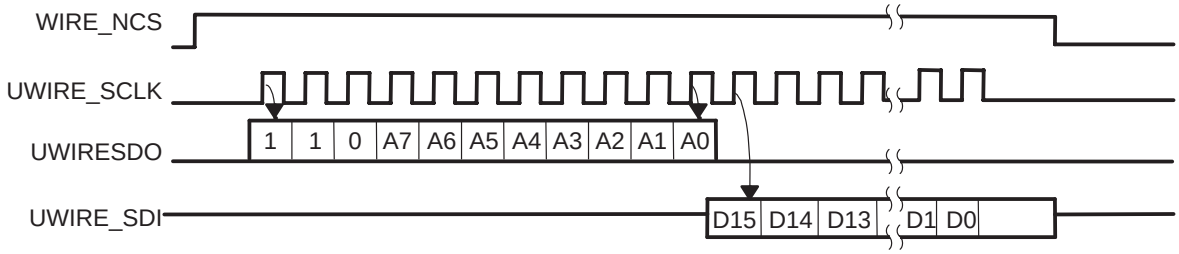
Figure 35. Behavior of an X25C02 EEPROM Read Cycle



On the DO line, data is generated from the µWire interface on SLCK falling edge and read by the EEPROM interface on SCLK rising edge.

On the DI line, data is generated from the EEPROM interface on SCLK falling edge and read by the µWire interface on SCLK falling edge.

Figure 36. Behavior of an XL93LC66 EEPROM Read Cycle



On the DO line, data is generated from the µWire interface on SLCK falling edge and read by the EEPROM interface on SCLK rising edge.

On the DI line, data is generated from the EEPROM interface on SCLK rising edge and read by the µWire interface on SCLK rising edge.

3.3 Example of Protocol Using a Serial EEPROM (XL93LC66)

Set up the interface by writing the following values in setup 1 register (SR1):

- ❏ CS_EDGE_RD = 1
- ❏ CS_EDGE_WR = 0
- ❏ CSCS_LVL = 1
- ❏ CS_FRQ = 00
- ❏ CS_CHK = 1

In this example, only two cycles (read and write) are described.

3.3.1 Read Cycle

1) Set the following fields of the control and status register (CSR):
   - NB_BITS_RD: 0
   - NB_BITS_WR: 0
   - INDEX: 00
   - CS_CMD: 1
   - START: 0
2) Load the transmit data register (TDR) with:
   - 1 1 0 A7 A6 A5 A4 A3 A2 A1 A0 x x x x x: *Don't care*
   - A7 ... A0: Address of the selected memory register

3) Wait for the CSRB bit of CSR to be reset.

4) Set the following fields of the control and status register (CSR):

   - NB_BITS_RD: 16 (decimal)
   - NB_BITS_WR: 11 (decimal)
   - INDEX: 00
   - CS_CMD: 1
   - START: 1

5) Wait until CSRB = 0 and RDRB = 1 (status bits of CSR).

6) Read the content of receive data register (RDR).

7) To continue reading data external component, the EEPROM, go to step 8. Else go to step 9.

8) Set the following fields of the control and status register (CSR):

   - NB_BITS_RD: 16 (decimal)
   - NB_BITS_WR: 0 (decimal)
   - INDEX: 00
   - CS_CMD: 1
   - START: 1
   - Go to step 5.

9) Set the following fields of the control and status register (CSR):

❑ INDEX: 00

   - CS_CMD: 0
   - START: 0

3.3.2 Write Cycle

1) Set the following fields of the control and status register (CSR):

   - NB_BITS_RD: 0
   - NB_BITS_WR: 0
   - INDEX: 00
   - CS_CMD: 1
   - START: 0

2) Load the transmit data register (TDR) with:
   - 1 0 1 A7 A6 A5 A4 A3 A2 A1 A0 x x x x x: *Don't care*
   - A7 ... A0: Address of the selected memory register
3) Wait for the CSRB bit of the control and status register (CSR) to be reset.
4) Set the following fields of the control and status register (CSR):
   - NB_BITS_RD: 0
   - NB_BITS_WR: 11 (decimal)
   - INDEX: 00
   - CS_CMD: 1
   - START: 1
5) Wait for the CSRB bit of the control and status register (CSR) to be reset.
6) Load the transmit data register (TDR) with:
   - D15 D14 ... D0
   - D15 ... D0: Data
7) Set the following fields of the control and status register (CSR):
   - NB_BITS_RD: 0
   - NB_BITS_WR: 16 (decimal)
   - INDEX: 00
   - CS_CMD 1
   - START: 1
8) Wait for the CSRB bit of CSR to be reset.
9) Set the following fields of the control and status register (CSR):
   - INDEX: 00
   - CS_CMD: 0
   - START: 0

3.4 Example of Protocol Using an LCD Controller (COP472-3)

Set up the interface by writing in setup 1 register (SR1) the following value:

- ❑ CS_EDGE_RD = 1
- ❑ CS_EDGE_WR = 0
- ❑ CSCS_LVL = 0
- ❑ CS_FRQ = 10
- ❑ CS_CHK = 0

This example describes a loading sequence to drive a four-digit display.

3.4.1 Loading Sequence

1) Set the following fields of the control and status register (CSR):
   - NB_BITS_RD: 0
   - NB_BITS_WR: 0
   - INDEX: 01
   - CS_CMD: 1
   - START: 0
2) Wait for the CSRB bit of the control and status register (CSR) to be reset.
3) Load the transmit data register (TDR) with:
   - D7d1...D0d1 D7d2...D0d2 D7d1...D0d1: Data for digit 1
   - D7d2...D0d2: Data for digit 2
4) Set the following fields of the control and status register (CSR):
   - NB_BITS_RD: 0
   - NB_BITS_WR: 16 (decimal)
   - INDEX: 01
   - CS_CMD: 1
   - START: 1
5) Wait for the CSRB bit of the control and status register (CSR) to be reset.
6) Load the transmit data register (TDR) with:
   - D7d3...D0d3 D7d4...D0d4 D7d3...D0d3: Data for digit 3
   - D7d4...D0d4: Data for digit 4
7) Set the following fields of the control and status register (CSR):
   - NB_BITS_RD: 0
   - NB_BITS_WR: 16 (decimal)
   - INDEX: 01
   - CS_CMD: 1
   - START: 1
8) Wait for the CSRB bit of the control and status register (CSR) to be reset.
9) Load the transmit data register (TDR) with:
   - D7...D0 x x x x x x x x: *Don't care*
   - D7...D0: Data for special segment and control function

10) Set the following fields of the control and status register (CSR):
    - NB_BITS_RD: 0
    - NB_BITS_WR: 8 (decimal)
    - INDEX: 01
    - CS_CMD: 1
    - START: 1
11) Wait for CSRB to go low, which indicates the CSR is ready to receive new data. It is advised that you read the bit before and after every write access to CSR to check the status.
12) Set the following fields of the control and status register (CSR):
    - INDEX: 01
    - CS_CMD: 0
    - START: 0

3.5 Example of Protocol Using Autotransmit Mode

The setup 5 register (SR5) controls the autotransmit mode. The following example configures µWire for a read access on CS0 with serial clock out inverted, CS autotoggle enabled, DMA request disabled, and interrupt enabled:

1) SR5 = DMA_TX_EN: 0

   IT_EN: 1

   AUTO_TX_EN: 1

   CS_TOGGLE_TX_EN: 1

2) SR1 = CS0_EDGE_RD: 0

   CS0_EDGE_WR: 1

   CS0CS_LVL: 0

   CS0_FREQ: 00

   CS0_CHK: 1

Note:

Data out is latched on the falling edge of the serial clock. Data in is sampled on the rising edge.

3) SR3 = CLK_EN: 1

   CK_FREQ: 00 (must wait for 1 ARMXOR_CK + 1 F_INT cycle before any other register access)

4) SR4 = CLK_IN: 1

5) Set the following fields of the control and status register (CSR):

- ❑ NB_BITS_RD: 5
- ❑ NB_BITS_WR: 7
- ❑ INDEX: 00
- ❑ CS_CMD: 0
- ❑ START: 0

6) Wait for the CSRB = 0 of the control and status register (CSR).

7) Load the transmit data register (TDR) with:

   - A6 A5 A4 A3 A2 A1 A0 x x x x x x x x x: *Don't care*
   - A6 ... A0: Address of the selected memory register

   Transfer is automatically started.

8) Wait until CSRB = 0 and RDRB = 1 (status bits of CSR).

9) Read the content of receive data register (RDR).

10) To continue reading data external component, go to 5 else go to 11.

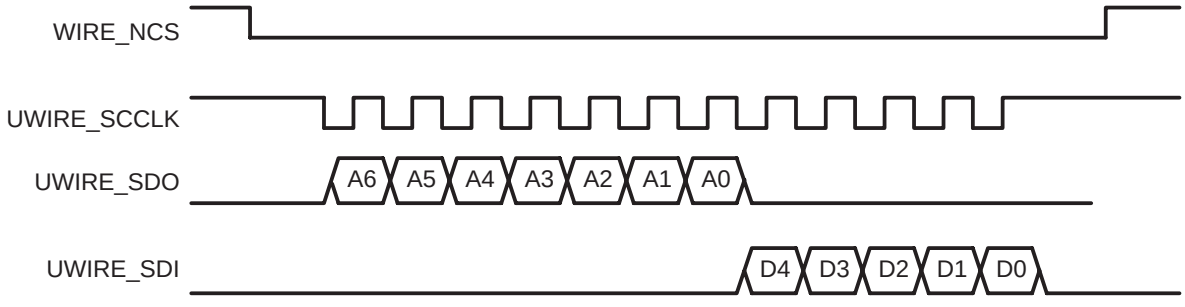
11) 20

12) Release autotransmit mode: SR5 = AUTO_TX_EN: 0.

13) END

The corresponding behavior of the serial interface is described in Figure 37.

Figure 37. Read Cycle in Autotransmit Mode



3.6 Example of Autotransmit Mode With DMA Support

The setup 5 register (SR5) controls the autotransmit mode and DMA mode. The following example configures μWire for a 16-bit write access on CS1 with serial clock out not inverted, CS autotoggle enabled, DMA request enabled, and interrupt disabled:

1) Set up and enable the DMA channel.

2) Program the configuration registers SR1, SR3, and SR4.

3) Check CSRB status to ensure that the peripheral is ready to receive (low).

4) Program the control and status register (CSR) as follows:

- ❏ NB_BITS_RD = 0
- ❏ NB_BITS_WR = 16
- ❏ INDEX = 00
- ❏ CS_CMD: = 1
- ❏ START = 0

5) Write to the setup register (SR5) to configure and initiate the transfer:

- ❏ DMA_TX_EN = 1
- ❏ IT_EN = 0
- ❏ AUTO_TX_EN = 1
- ❏ CS_TOGGLE_TX_EN = 1 (Note that in AUTO TX mode, setting the DMA_TX_EN bit to 1 starts the transfer)

6) When the DMA transfer is complete, check the status of CSRB to determine whether μWire has finished the serial data transfer.

7) Write to the setup register (SR5) to disable DMA and AUTO TX mode:

- ❏ DMA_TX_EN = 0
- ❏ IT_EN = 0
- ❏ AUTO_TX_EN = 0
- ❏ CS_TOGGLE_TX_EN = 0

Using Autostart and Autotoggle CS Mode

You must wait for a minimum of 3 x F_INT clock cycles after the end of transfer (transition 1 to 0 detected on CSRB) before setting the SR3 register to turn off the internal clock.

4 Multichannel Serial Interfaces

The Multichannel Serial Interface (MCSI) is a general-purpose serial port used to interface the OMAP5912 to external devices such as Codes and other serail devices that require a continuous clock.

The MSCI peipheral is a serial interface with multichannel transmission capability. It is intended for voice/data transfer between OMAP5912 and communication and Bluetooth chipsets. The MCSI requires a continuous clock to operate and therefore does not support protocols such as I2C, SPI, Microwire or any other protocol in which a gated clock is required.

The two public MCSIs on the device provide full duplex transmission and master or slave clock control. All transmission parameters are configurable to cover the maximum number of operating conditions:

- Master or slave clock control (transmission clock and frame synchronization pulse)
- Programmable transmission clock frequency
- Single-channel or multichannel (x16) frame structure
- Programmable word length: 3 to 16 bits
- Full-duplex transmission
- Programmable frame configuration
  - Continuous or burst transmission
  - Normal or alternate framing
  - Normal or inverted frame polarity
  - Short or long frame pulse
  - Programmable oversize frame length
  - Programmable frame length
- Programmable interrupt occurrence time (TX and RX)
- Error detection with interrupt generation on wrong frame length
- DMA support for both TX and RX data transfers

4.1 Communication Protocol

4.1.1 Configuration Parameters

The configuration parameters can be modified only if the MCSI is disabled (CONTROL_REG[0] = 0).

Slave/Master Control

Using the control bit, the interface can be configured in one of two ways:

- In master mode, with the transmission clock and the frame synchronization pulse generated by the interface
- In slave mode, with the transmission clock and the frame synchronization pulse generated from an external device

Control bit:
MAIN_ PARAMETERS _REG(6) = MCSI_MODE
1: Master
0: Slave

Single-Channel/Multichannel

The frame structure can be either single-channel-based (one channel per frame), or multichannel-based with the number of channels fixed at 16.

Control bit:
MAIN_ PARAMETERS _REG(7) = MULTI
1: Multichannel
0: Single-channel

Short/Long Framing

The frame-synchronization pulse duration can be either short, with a pulse duration equal to the bit duration, or long, with a pulse duration equal to the channel duration.

The long frame is active only during transmission on channel 0.

Control bit:
MAIN_ PARAMETERS _REG(8) = FRAME_SIZE
1: Long
0: Short

Normal/Alternate Frame Synchronization

The frame-synchronization pulse position is either normal, with the frame-synchronization pulse starting one bit before channel 0, or alternates with the frame-synchronization pulse starting with the first bit of channel 0.

Control bit:
MAIN_ PARAMETERS _REG(9) = FRAME_POSITION
1: Alternate
0: Normal

Continuous/Burst Mode

The frame mode is either continuous with one frame-synchronization pulse at the first frame, or bursts with one frame-synchronization pulse at each frame.

Control bit:
MAIN_ PARAMETERS _REG(5) = FRAME_MODE
1: Continuous
0: Burst

Normal/Inverted Clock

The polarity of the clock can be either normal, with writing on the positive edge clock and reading on the negative edge clock , or inverted, with writing on the negative edge clock and reading on the positive edge clock.

Control bit:
MAIN_ PARAMETERS _REG(4) = CLOCK_POLARITY
1: Inverted
0: Normal

Normal/Inverted Frame Synchronization

The polarity of the frame-synchronization pulse can be either normal, with a positive pulse, or inverted, with a negative pulse.

Control bit:
MAIN_ PARAMETERS _REG(10) = FRAME_POLARITY
1: Inverted
0: Normal

Channel Used

To enable a channel in multimode, set bit n for the desired channel n.

Word Size

To choose the size of the word, set its size minus one into the main parameters registers.

Control bit:
MAIN_PARAMETERS_REG(3:0) = WORD_ SIZE
(2 <= WORD_ SIZE <= 15)

The MCSI transmits and receives the most significant bit first. For example, if the word_size equals 11, the upper 12 bits of the TX registers are transmitted, the upper 12 bits of the RX registers contain the received data, and the lower 4 bits are zeros.

Frame Size

To add any overhead bits at the end of each frame, set the number of desired overhead bits in the over_size_register.

Control bit:
OVER_CLOCK_REG(9:0) = OVER_CLK (0<=OVER_CLK <=1023)

Transmission Clock Frequency

In master mode, the clock frequency is derived from the 12-MHz master clock and can be programmed from 5.8 kHz to 6 MHz in increments of 83 ns.

Control bit:
CLOCK_FREQUENCY_REG(10:0) = CLK_FREQ
(2<=CLK_FREQ <= 2047)

with
($t_{CLK} = t_{12MHz} * CLK\_FREQ$)

4.1.2 Sample Setup for Communication μ-Law Interface Using Interrupts

MCSI Configuration

An example of communication μ-law interface setup using interrupts follows.

- ❏ DSP_Write(0x0000) = CONTROL_REG (disable MCSI for setup)
- ❏ DSP_Write(0x0007) = MAIN_PARAMETERS_REG (set up MCSI per configuration below)
  - ■ Bit 15-14 (00b): No DMA
  - ■ Bit 10 (0b): Positive polarity for frame
  - ■ Bit 9 (0b): Normal synchronization mode
  - ■ Bit 8 (0b): Short framing
  - ■ Bit 7 (0b): Single channel
  - ■ Bit 6 (0b): Slave mode
  - ■ Bit 5 (0b): Burst mode
  - ■ Bit 4 (0b): Positive edge for clock
  - ■ Bit 3-0 (0111b): 8-bit data
- ❏ DSP_Write(0x0700) = INTERRUPTS_REG (all interrupts are enabled)
- ❏ DSP_Write(0x0000) = OVER_CLOCK_REG
- ❏ DSP_Write(0x0001) = CONTROL_REG (start MCSI)

Transmit Data Loading (TX_INT ISR)

- ❏ DSP_Write = TX_REG

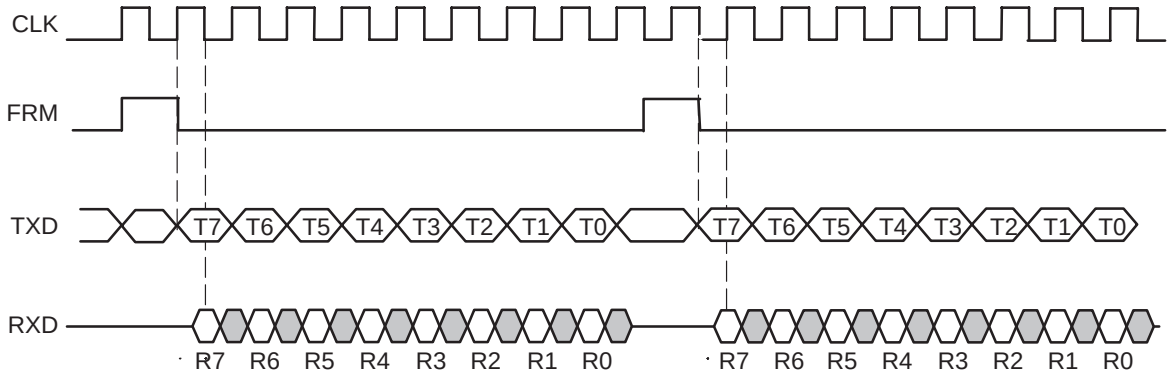
Received Data Loading (RX_INT ISR)

- ❏ DSP_Read = RX_REG

Stop MCSI

- ❏ DSP_Write(0x0000) = CONTROL_REG (disable MCSI clock)
- ❏ DSP_Write(0x0002) = CONTROL_REG (reset MCSI registers)

Figure 38. Communication μ-Law Interface Interrupts Waveform Example



4.1.3 Interface Management

Interrupts Generation

Three physical interrupts are available for real-time management of the MCSI by the DSP:

- RX_INT (data receive interrupt)
- TX_INT (data transmit interrupt)
- FERR_INT (frame duration error interrupt)

RX_INT, TX_INT, and FERR_INT are maskable with dedicated programmable control bits of the interrupt register INTERRUPTS_REG.

- RX_INT is masked when MASK_IT_RX = 0.
- TX_INT is masked when MASK_IT_TX = 0.
- FERR_INT is masked when MASK_IT_ERROR = 0.

Each interrupt is associated with a flag bit in the STATUS_REG register that is set to 1 when the interrupt is generated. To acknowledge the interrupt and release the corresponding physical signal, the DSP must write a 1 at the bit location in the status register. The following list provides interrupt/flag bit associations:

- RX_INT (RX_READY flag and acknowledge bit)
- TX_INT (TX_READY flag and acknowledge bit)
- FERR_INT (FRAME _ERROR flag and acknowledge bit)

Receive Interrupt

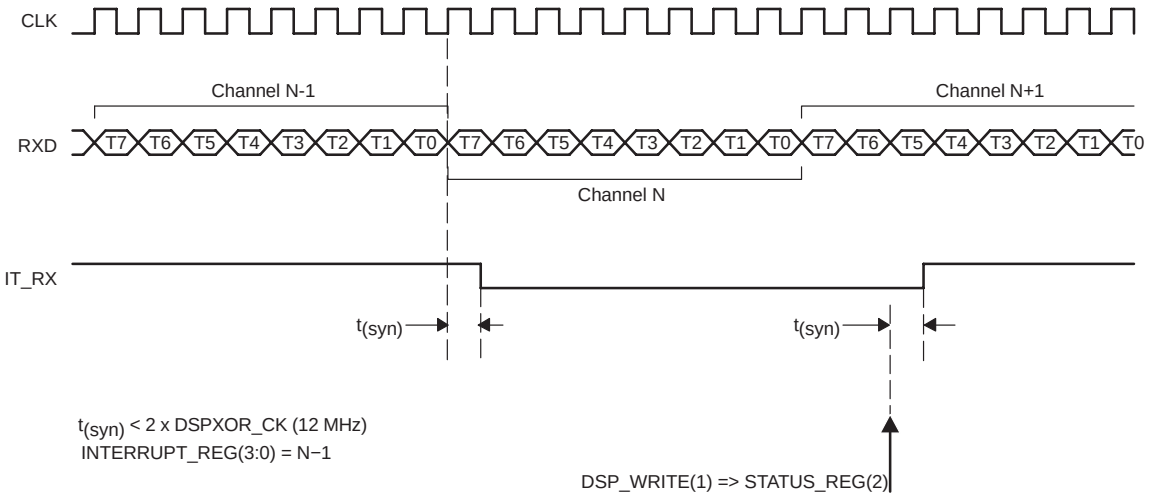
The receive interrupt is generated every frame after the completion of the reception of a data word:

- In single-channel mode, the interrupt is generated one half-clock period (plus a synchronization delay) after the reception of the word.
- In multichannel mode, the interrupt is generated one half-clock period (plus a synchronization delay) after the reception of the word of the channel whose number is defined by the NB_CHAN_IT_RX parameter of INTERRUPTS_REG register.

Note:

If MCSI is in slave mode, the clock must be driven after valid data reception until the interrupt is generated and must not be gated before then, because the interrupt is generated on the MCSI interface clock.

Figure 39. Receive Interrupt Timing Diagram

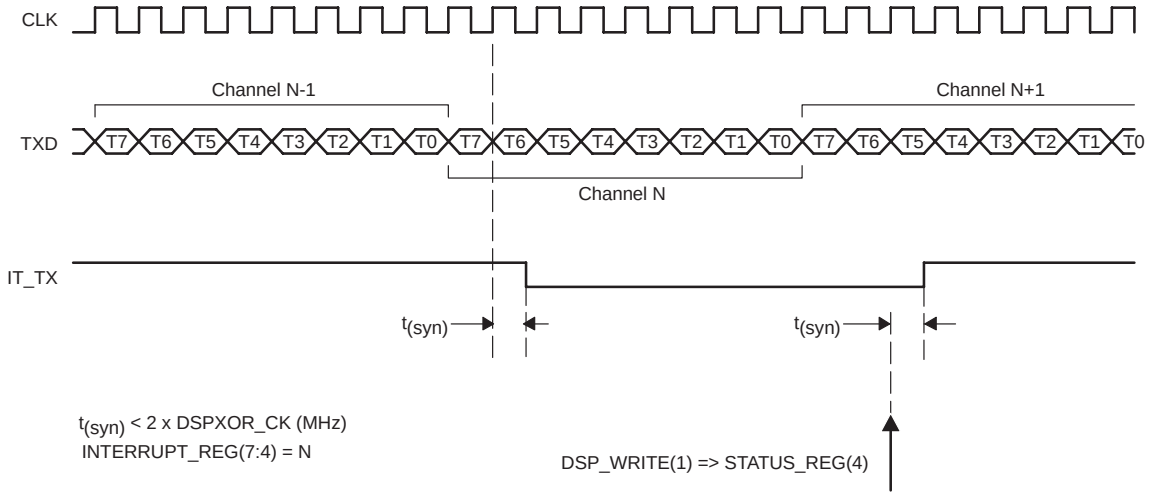


Transmit Interrupt

The transmit interrupt is generated every frame after the start of the transmission of a data word.

- In single-channel mode, the interrupt is generated one clock period after the beginning of the transmission of the word.
- In multichannel mode, the interrupt is generated one clock period after the transmission of the word of the channel whose number is defined by the NB_CHAN_IT_RX parameter of INTERRUPTS_REG register.

Figure 40. Transmit Interrupt Timing Diagram



Frame Duration Error Interrupt

The frame duration error interrupt is only generated when:

- ❏ The interface is configured in burst mode (CONTINUOUS = 0).
- ❏ The frame duration is smaller or longer than the expected value.

Namely, expected frame duration = [(channels number) * (word size)] + (over-size number) in clock periods units with over-size number defined in OVER_SIZE_REG register.

If the frame duration is longer than the expected value, then the interrupt is generated one clock period after the number of the over_size clock periods, as defined in OVER_CLOCK parameter.

If the frame duration is smaller than the expected value, then the interrupt is generated one clock period after the occurrence of the next frame pulse (first active edge).

Figure 41. Frame Duration Error—Too Many (Long)

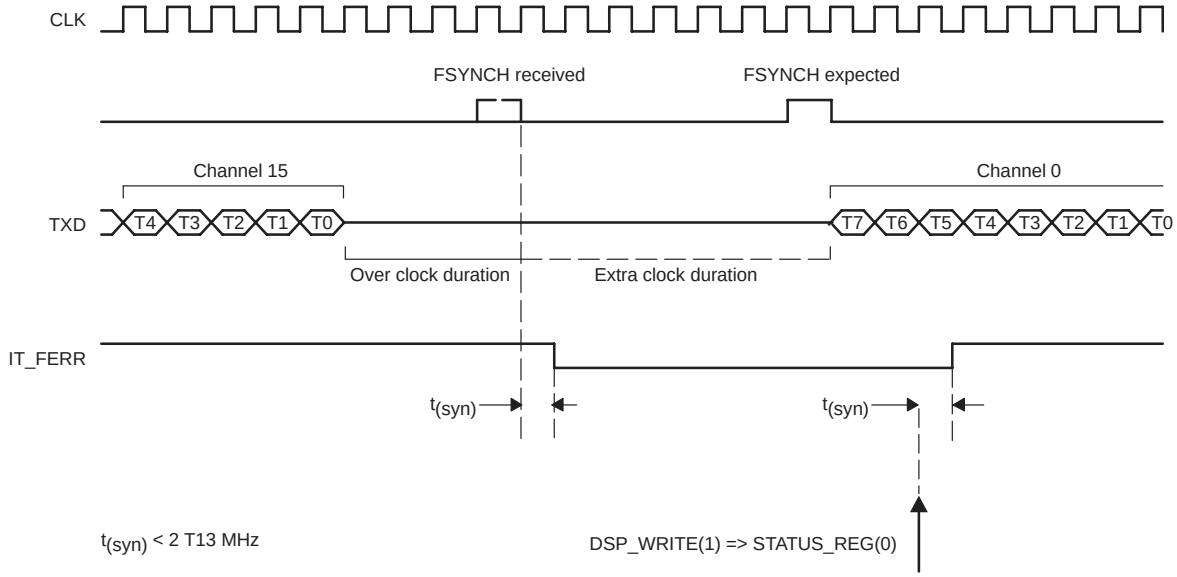
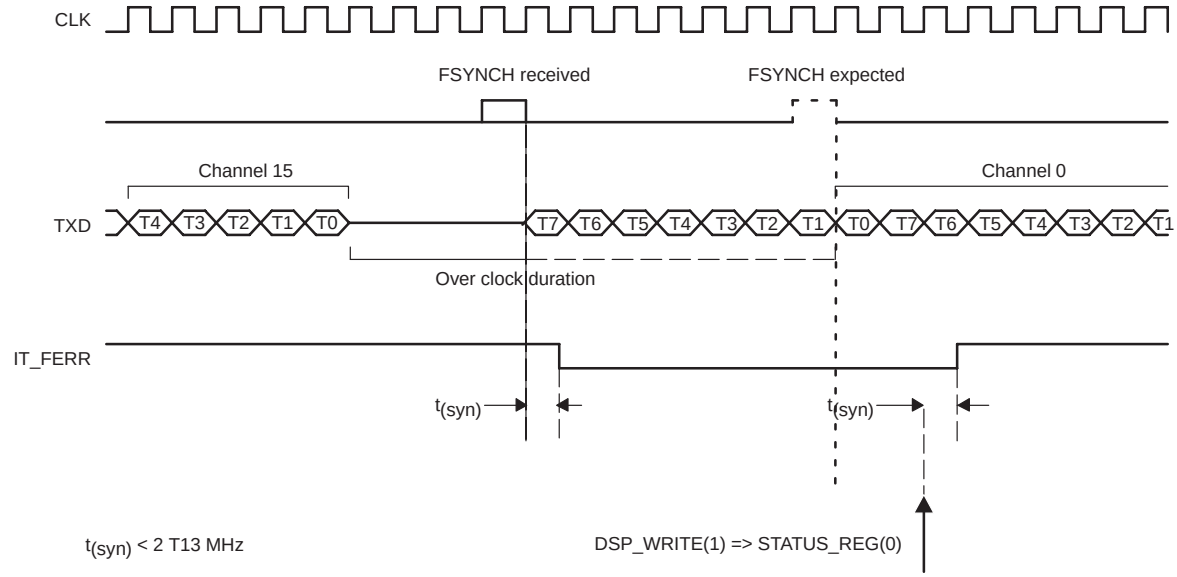


Figure 42. Frame Duration Error—Too Few (Short)



4.1.4 Interrupt Programming

At module reset, RX_INT, TX_INT, and FERR_INT are masked.

To validate an interrupt:

If in multichannel mode, the RX and TX interrupts can be configured to occur in a dedicated channel of the frame [1-16].

- DSP_WRITE(channel_nb) = INTERRUPTS_REG(3:0) for RX_INT
- INTERRUPTS_REG(7:4) for TX_INT

Unmask the interrupt:

- DSP_WRITE(1) =
  - INTERRUPTS_REG(8) for RX_INT
  - INTERRUPTS_REG(9) for TX_INT
  - INTERRUPTS_REG(10) for FERR_INT

On interrupt occurrence:

- DSP_READ =
  - STATUS_REG(0) for FERR_INT occurrence
  - STATUS_REG(2) for RX_INT occurrence
  - STATUS_REG(3) for RX character overflow
  - STATUS_REG(4) for TX_INT occurrence
  - STATUS_REG(5) for TX character underflow

Then, to release the interrupt signal and reset the corresponding status bits:

- DSP_WRITE(1) =
  - STATUS_REG(0) for FERR_INT release
  - STATUS_REG(2) for RX_INT release
  - STATUS_REG(4) for TX_INT release

4.1.5 DMA Channel Operation

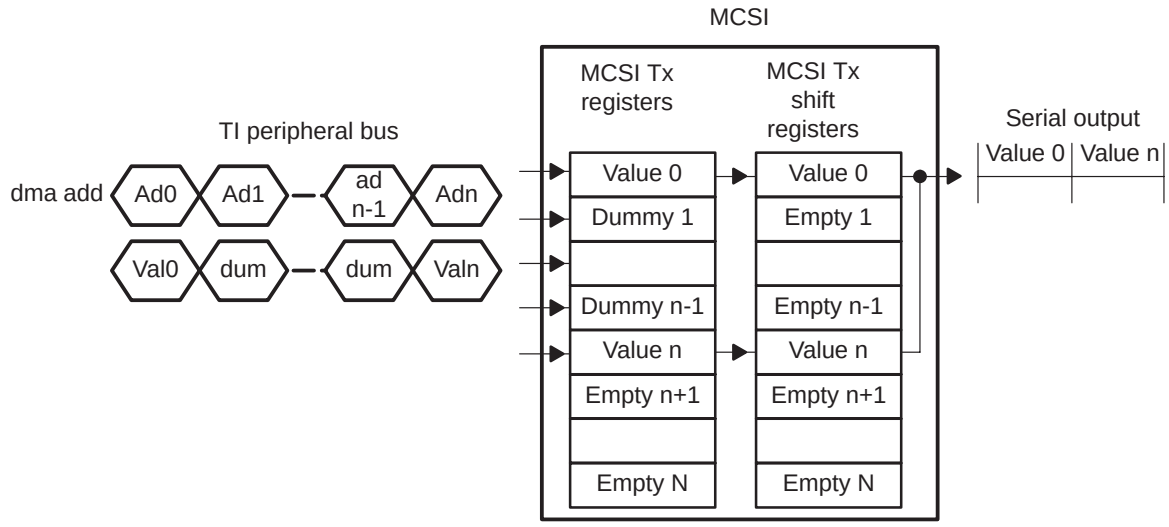
Both transmit and receive operations can be supported by DMA. DMA support is enabled by control bits in the MAIN_PARAMETERS_REG:

- MAIN_PARAMETERS_REG(15:14) = DMA_ENABLE(1:0)
  - TX_DMA_REQ enabled when DMA_ENABLE(0) = 1
  - TX_DMA_REQ disabled when DMA_ENABLE(0) = 0
  - RX_DMA_REQ enabled when DMA_ENABLE(1) = 1
  - RX_DMA_REQ disabled when DMA_ENABLE(1) = 0

Transmit DMA Transfers

A new transmit DMA transfer is initiated during the transmission of the last channel of a frame, at which time all data in the transmit registers (TX_REGs) has been moved to shift registers; the TX_REGs are now ready to be rewritten. If N channels are used, the DMA controller successively accesses all consecutive registers between TX_REG(0) and TX_REG(N-1). If some channels between TX_REG(0) and TX_REG(N-1) are not used, the DMA controller writes dummy values when addressing these unused registers (see Figure 43).

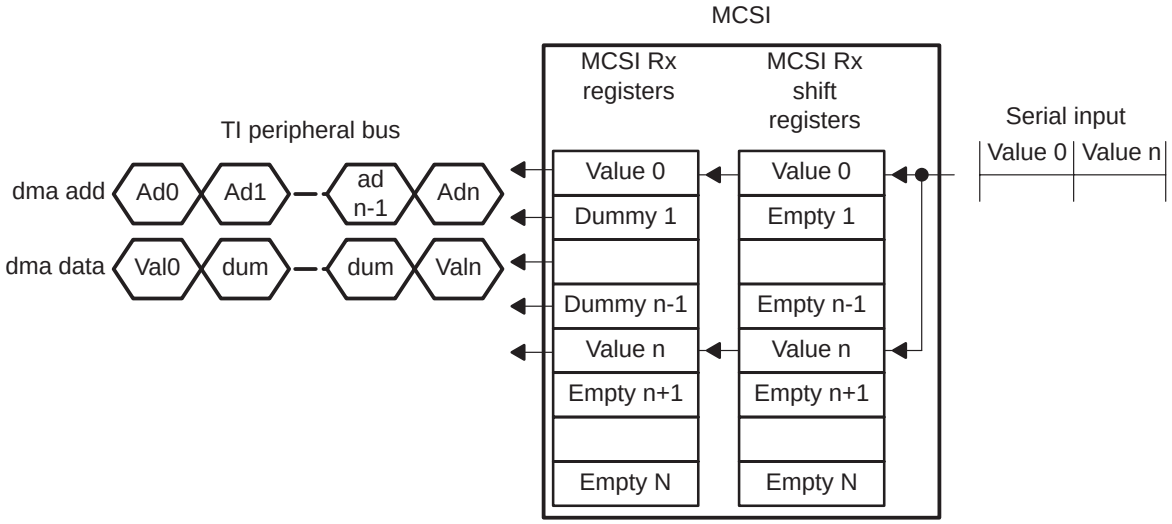
Figure 43. Transmit DMA Transfers



Receive DMA Transfers

A receive DMA transfer is initiated after the reception of the last channel of a frame, at which time all receive registers RX_REG have been updated and are ready to be read. If N channels are used, the DMA controller successively accesses all consecutive registers between RX_REG(0) and RX_REG(N-1). If some channels between RX_REG(0) and RX_REG(N-1) are not used, the DMA controller reads dummy values when addressing these unused registers (see Figure 44).

Figure 44. Receive DMA Transfers



A multichannel application cannot use DMA for some channels and interrupt servicing for others. RX/TX interrupts are not generated when DMA RX/TX transfers are enabled.

4.1.6 Interface Activation

Start Sequence

A typical sequence to start the interface is:

1) MCSI configuration:
   a) DSP_WRITE(0x0000)= CONTROL_REG in order to remove the write protection on the control registers
   b) DSP_WRITE(0x….)= MAIN_PARAMETERS_REG
   c) DSP_WRITE(0x….)= INTERRUPTS_REG
   d) DSP_WRITE(0x….)= CHANNEL_USED_REG
   e) DSP_WRITE(0x….)= CLOCK_FREQUENCY_REG
   f) DSP_WRITE(0x….)= OVER_CLOCK_REG
2) Transmit data loading for selected channels:
   a) DSP_WRITE(0x….)= TX_REG[channel index]
3) Enable MCSI clock:
   a) DSP_WRITE(0x0001)= CONTROL_REG

Stop Sequence

A typical sequence to stop the interface is:

1) Disable MCSI clock: DSP_WRITE(0x0000) = CONTROL_REG

   The status register keeps its content even after the stop of the transmission. The control registers can now be modified.

2) Software reset: DSP_WRITE(0x0002) = CONTROL_REG

   The software reset initializes the status register.

Software Reset

The MCSI software reset is activated with the SW_RESET bit of the control register (CONTROL_REG) (see Table 52, *Activity Control Register*).

This reset is limited to the control and status registers, the internal state machine, and the PISO and SIPO logic. The parameters registers are not affected by this software reset.

On the software reset, the MCSI reference clock is disabled, thus halting the execution of any current operating mode.

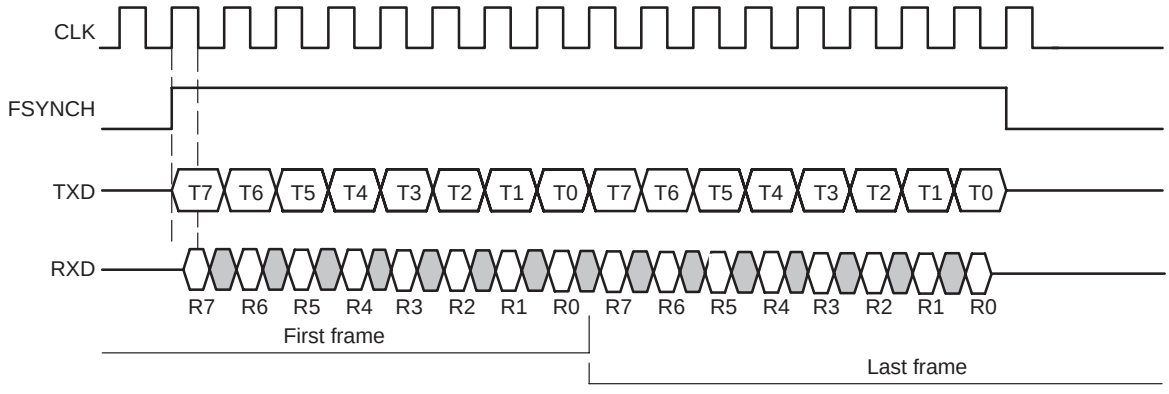
4.1.7 Functional Mode Timing Diagrams

The following timing diagrams are based on a positive clock polarity with parameter CLOCK_POL = 0.

(Transmit on rising edge/receive on falling edge.)

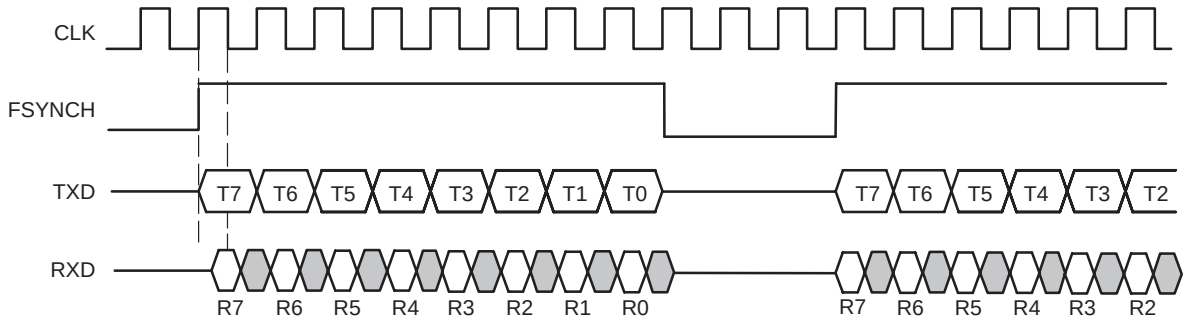
Single-Channel/Alternate Long Framing

Figure 45. Single-Channel/Alternate Long Framing



Single-Channel/Alternate Long Framing/Burst

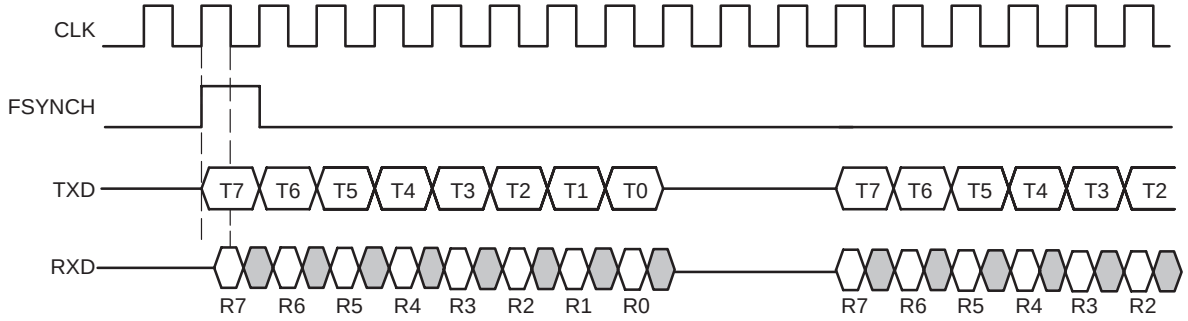
Figure 46. Single-Channel/Alternate Long Framing/Burst



OVER_CLOCK_REG = 0x0003

Single-Channel/Alternate Short Framing/Continuous/Burst

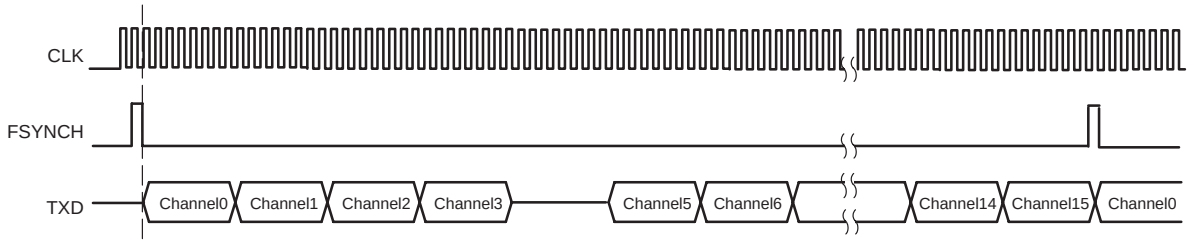
Figure 47. Single-Channel/Alternate Short Framing/Continuous/Burst



OVER_CLOCK_REG = 0x0003

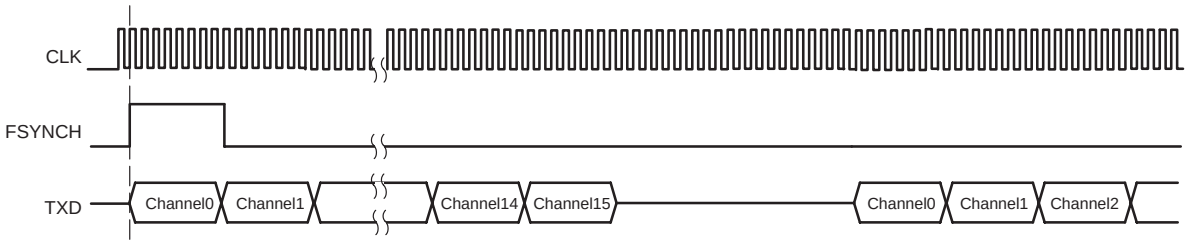
Multichannel/Normal Short Framing/Channel4 Disable

Figure 48. Multichannel/Normal Short Framing/Channel4 Disable



Multichannel/Alternate Long Framing/Continuous/Burst

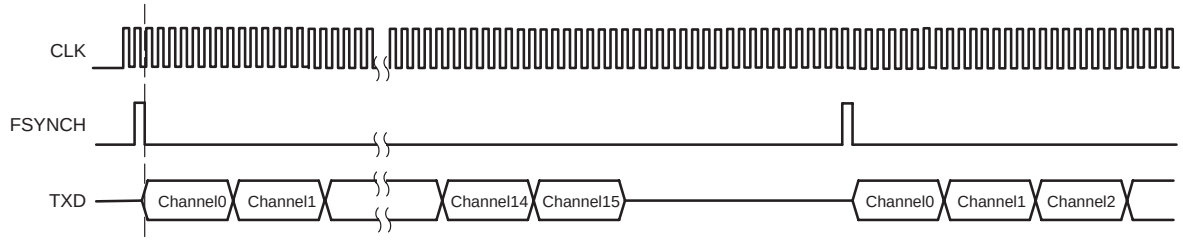
Figure 49. Multichannel/Alternate Long Framing/Continuous/Burst



OVER_CLOCK_REG = 0x0013

Multichannel/Normal Short Framing/Burst

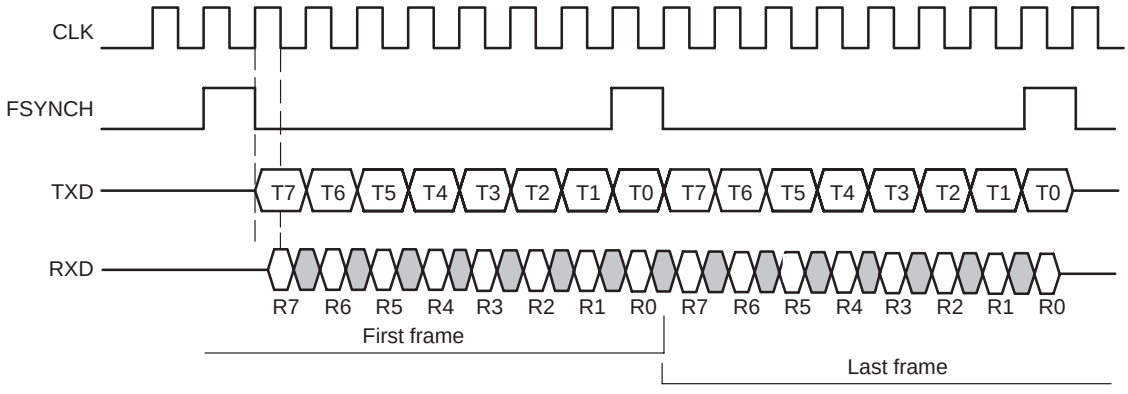
Figure 50. Multichannel/Normal Short Framing/Burst



OVER_CLOCK_REG = 0x0013

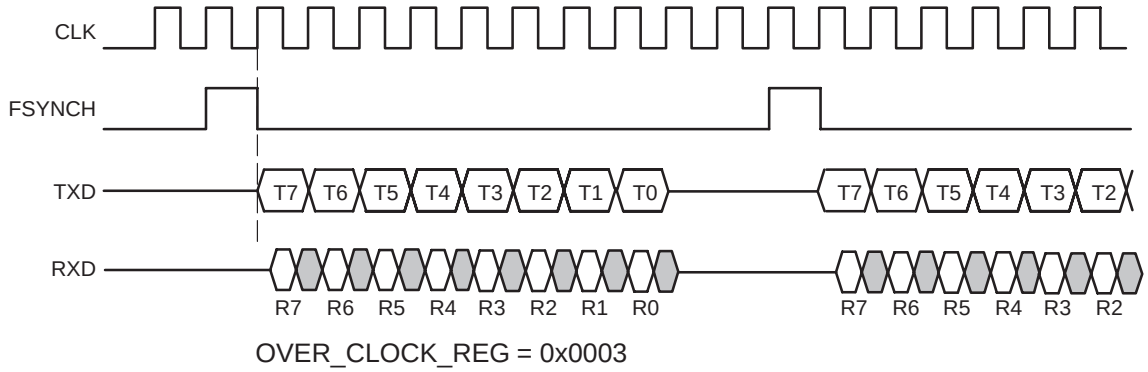
Single-Channel/Normal Short Framing

Figure 51. Single-Channel/Normal Short Framing



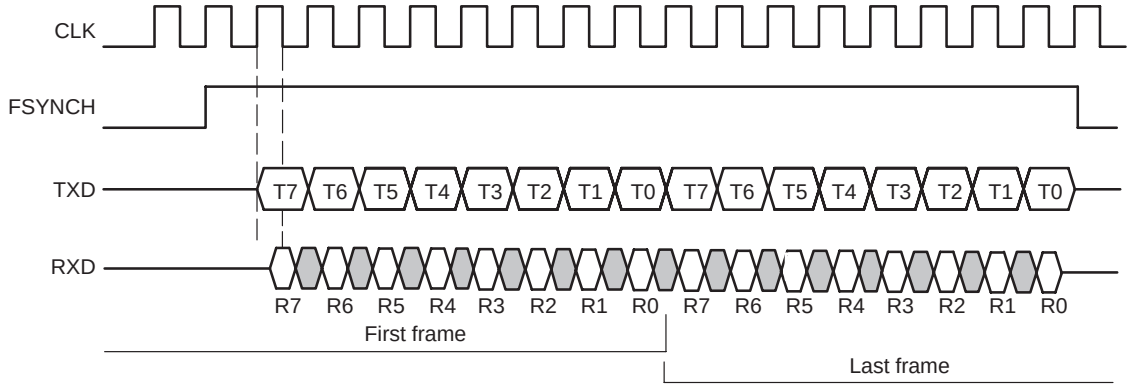
Single-Channel/Normal Short Framing/Burst

Figure 52. Single-Channel/Normal Short Framing/Burst



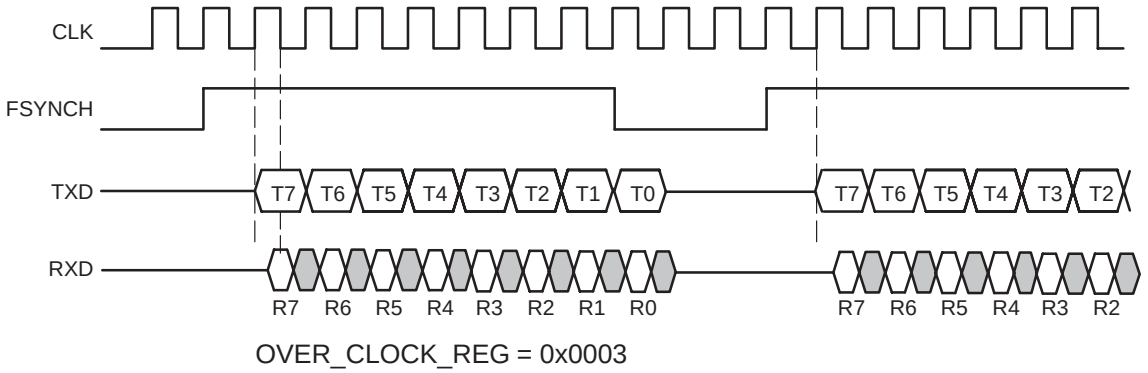
Single-Channel/Normal Long Framing

Figure 53. Single-Channel/Normal Long Framing



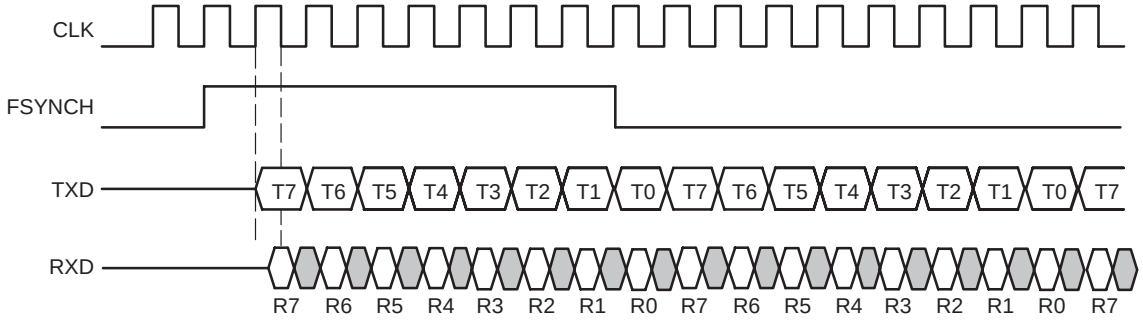
Single-Channel/Normal Long Framing/Burst

Figure 54. Single-Channel/Normal Long Framing/Burst



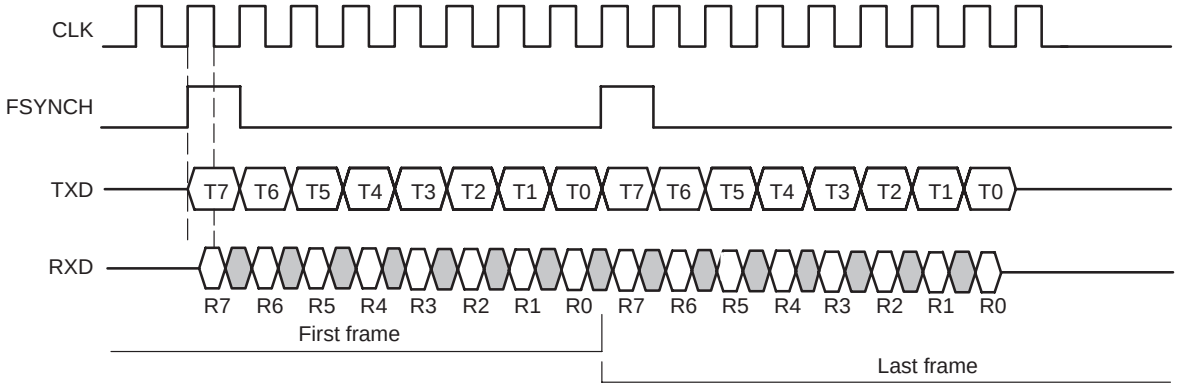
Single-Channel/Normal Long Framing/Continuous

Figure 55. Single-Channel/Normal Long/Continuous



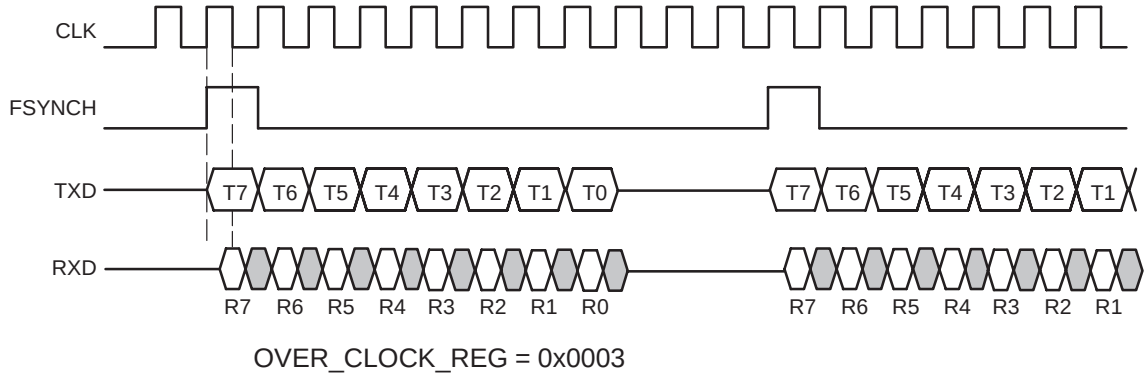
Single-Channel/Alternate Short Framing

Figure 56. Single-Channel/Alternate Short Framing



Single-Channel/Alternate Short Framing/Burst

Figure 57. Single-Channel/Alternate Short Framing/Burst



4.2 MCSI Register Descriptions

Table 47 through Table 55 describe the MCSI registers. The CHANNEL_USED_REG, CLOCK_FREQUENCY_REG, OVER_CLOCK_REG, INTERRUPTS_REG, and MAIN_PARAMETERS_REG registers are write protected if the MCSI is enabled (CONTROL_REG[0] = 1).

The channel selection register is only used in multichannel mode.

Table 47. Channel Selection Register (CHANNEL_USED_REG)

Bit	Name	Access	Hardware Reset
15	USE_CH15	R/W	0
14	USE_CH14	R/W	0
13	USE_CH13	R/W	0
12	USE_CH12	R/W	0
11	USE_CH11	R/W	0
10	USE_CH10	R/W	0
9	USE_CH9	R/W	0
8	USE_CH8	R/W	0
7	USE_CH7	R/W	0
6	USE_CH6	R/W	0
5	USE_CH5	R/W	0
4	USE_CH4	R/W	0
3	USE_CH3	R/W	0
2	USE_CH2	R/W	0
1	USE_CH1	R/W	0
0	USE_CH0	R/W	0

USE_CH[i] selects channel [i] for data transmission (active high).

Table 48. Clock Frequency Register (CLOCK_FREQUENCY_REG)

The clock frequency register is used only in master mode when the interface generates the serial clock.

Bit	Name	Description	Access	Hardware Reset
15:11	Unused		R	0000 0
10:0	CLK_FREQ	Division factor of 12-MHz reference clock (2<=CLK_FREQ<= 2047) In master mode, this register defines the transmission baud rate from a frequency ratio based on a 12-MHz reference clock. The transmission clock frequency can be programmed from 5.8 kHz to 6 MHz in steps or increments of 83 ns. Clock frequency = 12 MHz/CLK_FREQ with 2 <= CLK_FREQ <= 2047.	R/W	000 0000 0000

CLK_FREQ: division factor of 12-MHz reference clock (2<=CLK_FREQ<= 2047)

In master mode, this register defines the transmission baud rate from a frequency ratio based on a 12-MHz reference clock. The transmission clock frequency can be programmed from 5.8 kHz to 6 MHz in steps or increments of 83 ns.

Clock frequency = 12 MHz / CLK_FREQ with 2 <= CLK_FREQ <= 2047.

Table 49. Oversized Frame Dimension Register (OVER_CLOCK_REG)

Bit	Name	Description	Access	Hardware Reset
15:10	Unused		R	0000 00
9:0	OVER_CLOCK	Overhead clock periods in frame duration (0 = OVER_CLOCK = 1023)	R/W	00 0000 0000

Table 50. Interrupt Masks Register (INTERRUPTS_REG)

Bit	Name	Description	Access	Hardware Reset
15:11	Unused		R	0000 0
10	MASK_IT_ERROR	Mask of frame duration error interrupt (active at 0)	R/W	0
9	MASK_IT_TX	Mask of transmit interrupt (active at 0)	R/W	0

Table 50. Interrupt Masks Register (INTERRUPTS_REG) (Continued)

Bit	Name	Description	Access	Hardware Reset
8	MASK_IT_RX	Mask of receive interrupt (active at 0)	R/W	0
7:4	Number channel for IT_TX	Channel number for transmit interrupt generation (0 <=NB_CHAN <= 15)	R/W	0000
3−0	Number channel for IT_RX	Channel number for receive interrupt generation (0 <= NB_CHAN <=15)	R/W	0000

Table 51. Main Parameters Register (MAIN_PARAMETERS__REG)

Bit	Name	Value	Description	Access	Hardware Reset
15:14	DMA enable		Enable bits for DMA:	R/W	00
		00	Normal mode (No DMA)		
		01	DMA transmit mode, normal receive mode		
		10	Normal transmit mode, DMA receive mode		
		11	DMA transmit and receive mode		
13:11	Reserved		Reserved bits. These bits must always be written as 0.	R/W	000
10	FSYNCH_POLARITY		Frame-synchronization pulse polarity	R/W	0
		0	Positive		
		1	Negative		
9	FSYNCH_MODE		Frame-synchronization pulse position	R/W	0
		0	Normal		
		1	Alternate		
8	FSYNCH_SIZE		Frame-synchronization pulse shape	R/W	0
		0	Short		
		1	Long		
7	Multi/single		Frame structure	R/W	0

Table 51. Main Parameters Register (MAIN_PARAMETERS__REG) (Continued)

Bit	Name	Value	Description	Access	Hardware Reset
		0	Single		
		1	Multi		
6	MCSI mode		Interface transmission mode	R/W	0
		0	Slave		
		1	Master		
5	Continuous/burst		Frame mode	R/W	0
		0	Burst		
		1	Continuous		
4	CLOCK_POLARITY		Clock edge selection	R/W	0
		0	Positive		
		1	Negative		
3:0	Word size		Word size in bits number (2 <= size <= 15) with 2 for 3 bits and 15 for 16 bits.	R/W	0000

Table 52. Activity Control Register (CONTROL_REG)

Bit	Name	Value	Description	Access	Hardware Reset	Software Reset
15:3	Reserved		Reserved bits. These bits must always be written as 0.	R	0000 0000 0000 0	0000 0000 0000 0
2	Reserved		Reserved bits. These bits must always be written as 0.	R/W	0	0
1	MCSI software reset		Asynchronous reset of MCSI module	R/W	0	1
		0	Disable			
		1	Enable			

Table 52. Activity Control Register (CONTROL_REG) (Continued)

Bit	Name	Value	Description	Access	Hardware Reset	Software Reset
0	MCSI clock enable		Enable clock of MCSI module	R/W	0	0
		0	Disable			
		1	Enable			

Note:

The software reset is applied as long as the MCSI software reset bit is set to 1. A software reset disables the MSCI (the MCSI clk enable bit is cleared) and initializes the status register. It does not modify the other registers.

To clear an interrupt on the MCSI, the DSP must write to the MCSI status register with the bit corresponding to the interrupt set to 1. The MCSI status register has a two-cycle latency when writing into it, so the interrupt line is cleared two cycles after a write. To prevent clearing the interrupt handler before the interrupt line is cleared, the interrupt routine must be at least two cycles long.

Table 53. Interface Status Register (STATUS_REG)

Bit	Name	Value	Description	Access	Hardware Reset	Software Reset
15:7	Reserved		Reserved bits. These bits must always be written as 0.	R	0000 0000 0	0000 0000 0
6	Reserved		Reserved bits. These bits must always be written as 0.	R/W	0	0
5	TX underflow		Transmit underflow	R	0	0
		0	No under			
		1	Under			
4	TX ready		Flag for transmit interrupt occurrence	R/W	0	0
		0	No interrupt			
		1	Interrupt			

Table 53. Interface Status Register (STATUS_REG) (Continued)

Bit	Name	Value	Description	Access	Hardware Reset	Software Reset
3	RX overflow		Receive overflow	R	0	0
		0	No over			
		1	Over			
2	RX ready		Flag for receive interrupt occurrence	R/W	0	0
		0	No interrupt			
		1	Interrupt			
1	Error type few/many		Too short (few) or too long frame (many) status	R	0	0
		0	Short			
		1	Long			
0	Frame error		Error flag when wrong frame duration	R/W	0	0
		0	Correct			
		1	Bad			

This register is cleared by a software reset.

Table 54. Receive Word Register (RX_REG[15:0])

Bit	Name	Access	Hardware Reset
15	b15	R	U
14	b14	R	U
13	b13	R	U
12	b12	R	U
11	b11	R	U
10	b10	R	U
9	b9	R	U

Note: The MCSI receives the most significant bit first. For example, if the word_size equals 11, the upper 12 bits of the RX registers contain the received data, and the lower 4 bits are zeros.

Table 54. Receive Word Register (RX_REG[15:0]) (Continued)

Bit	Name	Access	Hardware Reset
8	b8	R	U
7	b7	R	U
6	b6	R	U
5	b5	R	U
4	b4	R	U
3	b3	R	U
2	b2	R	U
1	b1	R	U
0	b0	R	U

Note: The MCSI receives the most significant bit first. For example, if the word_size equals 11, the upper 12 bits of the RX registers contain the received data, and the lower 4 bits are zeros.

Table 55. Transmit Word Register (TX_REG[15:0])

Bit	Name	Access	Hardware Reset
15	b15	R/W	U
14	b14	R/W	U
13	b13	R/W	U
12	b12	R/W	U
11	b11	R/W	U
10	b10	R/W	U
9	b9	R/W	U
8	b8	R/W	U
7	b7	R/W	U
6	b6	R/W	U
5	b5	R/W	U
4	b4	R/W	U

Note: The MCSI transmits the most significant bit first. For example, if the word_size equals 11, the upper 12 bits of the TX registers are transmitted.

Table 55. Transmit Word Register (TX_REG[15:0]) (Continued)

Bit	Name	Access	Hardware Reset
3	b3	R/W	U
2	b2	R/W	U
1	b1	R/W	U
0	b0	R/W	U

Note: The MCSI transmits the most significant bit first. For example, if the word_size equals 11, the upper 12 bits of the TX registers are transmitted.

5 MCSI1 and MCSI2

This section provides information specific to MCSI1 and MCSI2 on the device.

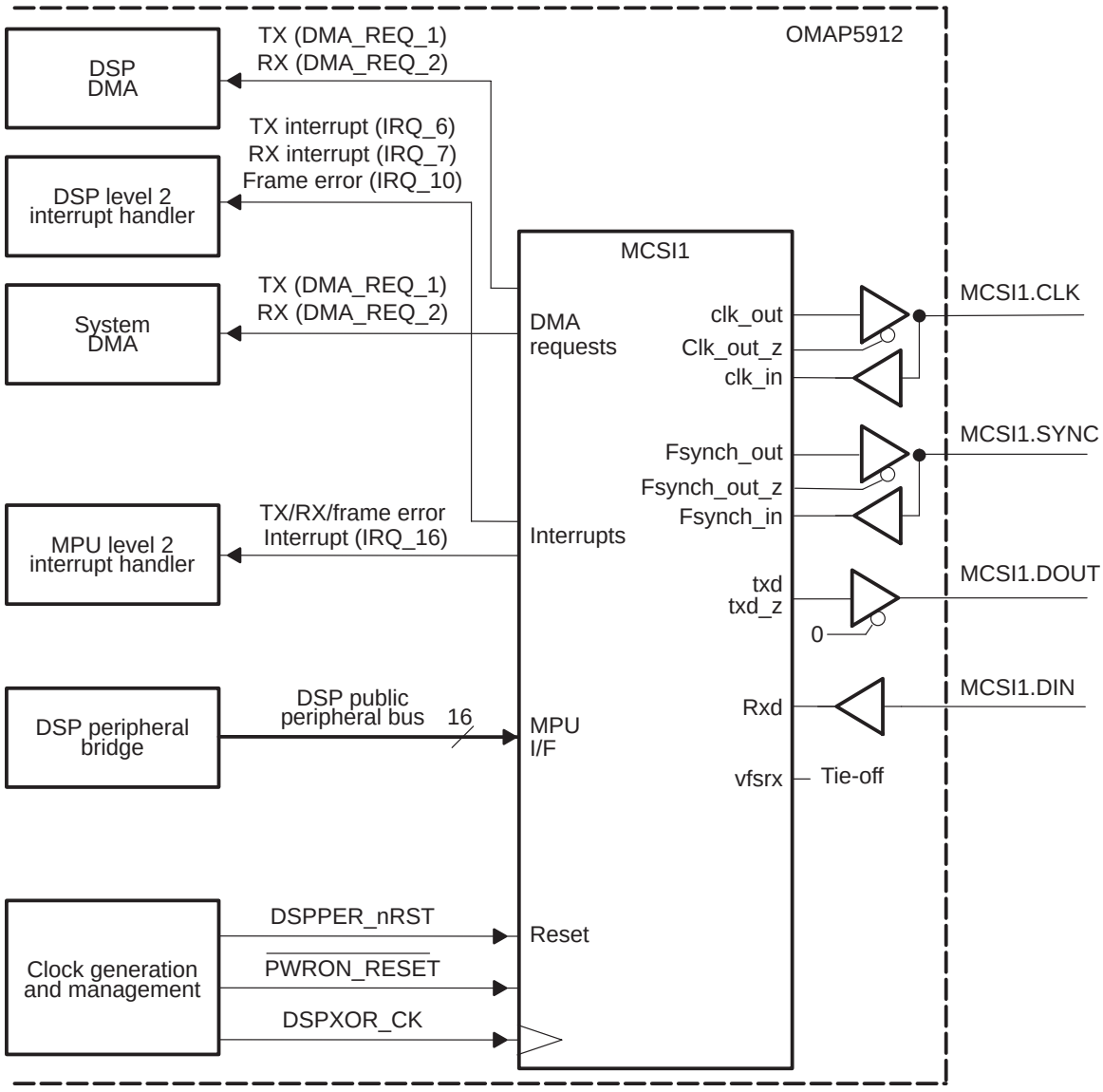
5.1 MCSI1 Pin Description

Table 56 identifies the MCSI1 I/O pins. Figure 58 shows the MCSI1 interface.

Table 56. MCSI1 Pin Descriptions

Pin	I/O Direction	Description
MCSI1.DIN	In	Data input
MCSI1.DOUT	Out	Data output
MCSI1.CLK	In/out	Bit clock
MCSI1.SYNC	In/out	Frame synchronization

Figure 58. MCSI1 Interface



5.2 MCSI1 Interrupt Mapping

Table 57 identifies the MCSI1 interrupt mappings. MCSI1 generates level-2 interrupts for both the DSP and the MPU. Only one MPU MCSI1 interrupt covers TX, RX, and frame error conditions; software must check the MCSI1 status register to determine the interrupt source.

Table 57. MCSI1 Interrupt Mapping

Incoming Interrupts	Level 2 DSP Interrupt	Level 2 MPU Interrupt
MCSI1 TX interrupt	IRQ_06	IRQ_16
MCSI1 RX interrupt	IRQ_07	IRQ_16
MCSI1 frame error	IRQ_10	IRQ_16

5.3 MCSI1 DMA Request Mapping

Table 58 identifies MCSI1 DMA request lines.

Table 58. TDMA Request Mapping—MCSI1

DMA Request Source	DMA Request Line—DSP	DMA Request Line—MPU
MCSI1 TX	DMA_REQ_01	DMA_REQ_01
MCSI1 RX	DMA_REQ_02	DMA_REQ_02

5.4 MCSI2 Pin Description

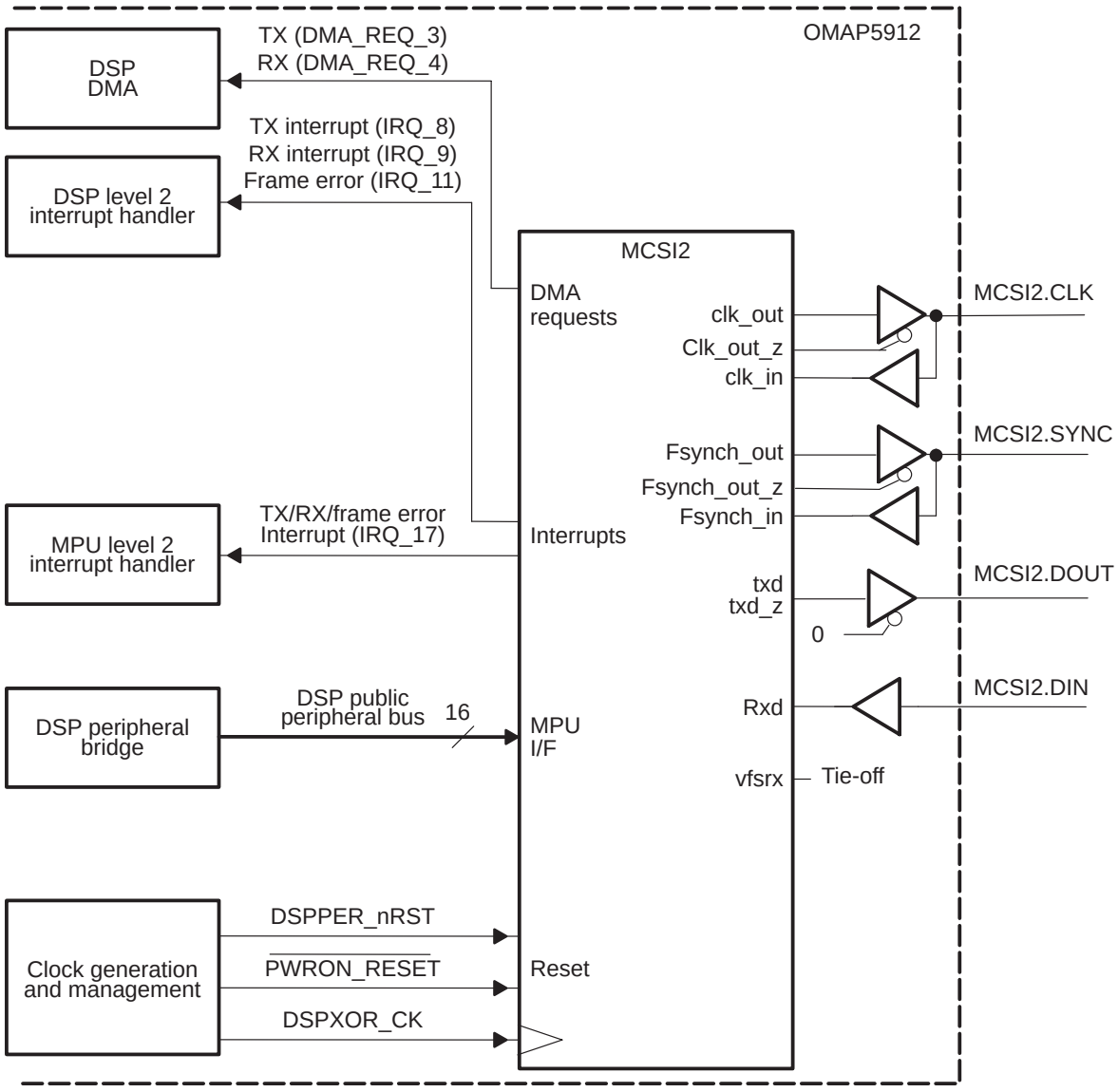
This section provides information specific to MCSI2 on the device.

Table 59 identifies the MCSI2 I/O pins. Figure 59 shows the MCSI2 interface.

Table 59. MCSI2 Pin Descriptions

Pin	I/O Direction	Description
MCSI2.DIN	In	Data input
MCSI2.DOUT	Out	Data output
MCSI2.CLK	In/out	Bit clock
MCSI2.SYNC	In/out	Frame synchronization

Figure 59. MCSI2 Interface



5.5 MCSI2 Interrupt Mapping

Table 60 identifies the MCSI2 interrupts. MCSI2 generates level-2 interrupts for both the DSP and the MPU. Only one MPU MCSI2 interrupt covers TX, RX, and frame error conditions; software must check the MCSI2 status register to determine the interrupt source.

Table 60. MCSI2 Interrupt Mapping

Incoming Interrupts	Level 2 DSP Interrupt	Level 2 MPU Interrupt
MCSI2 TX interrupt	IRQ_08	IRQ_17
MCSI2 RX interrupt	IRQ_09	IRQ_17
MCSI2 frame error	IRQ_11	IRQ_17

5.6 MCSI2 DMA Request Mapping

Table 61 identifies MCSI2 DMA request lines. Only the DSP DMA controller can transfer MCSI2 data; there is no MPU DMA capability.

Table 61. DMA Request Mapping—MCSI2

DMA Request Source	DMA Request Line—DSP	DMA Request Line—MPU
MCSI2 TX	DMA_REQ_03	–
MCSI2 RX	DMA_REQ_04	–

6 UARTs

There are three nearly identical UART modules on this device. The modules are identical except for the fact that UART2 does not support infrared data association (IrDA). UART1 and UART3 support IrDA. This section describes all 3 UART modules. Note that discussions regarding IrDA apply only to UART1 and UART3. This section includes a register description and a module configuration example. It also shows the the basic UART IrDA module pins.

Figure 60. UART IrDA Signals

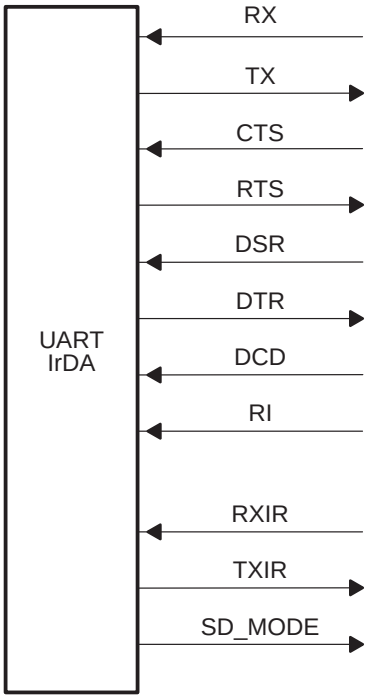
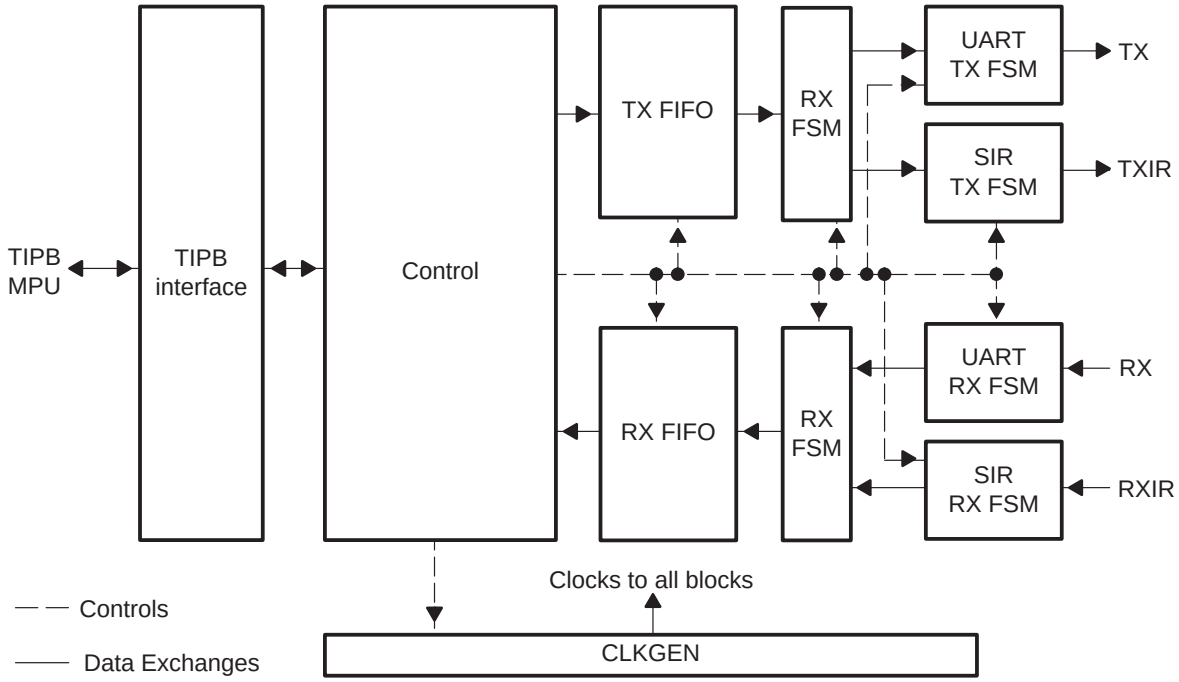


Figure 61. Functional Block Diagram



6.1 Main Features

- ❑ Selectable UART/IrDA modes
- ❑ Dual 64 entry FIFOs for received and transmitted data
- ❑ Programmable and selectable transmit and receive FIFO trigger levels for DMA and interrupt generation
- ❑ Programmable sleep mode
- ❑ Complete status reporting capabilities in both normal and sleep modes
- ❑ Frequency prescaler values from 0 to 16383 to generate the appropriate baud rates
- ❑ Single 48-MHz clock reference for baud setting
- ❑ Two DMA requests and one interrupt request to the system

6.1.1 UART/Modem Functions

- ❑ Baud-rate from 300 bits/s up to 3.6864M bits/s
- ❑ Autobaud between 1200 bits/s and 115.2K bits/s
- ❑ Software/hardware flow control
  - ■ Programmable Xon/Xoff characters
  - ■ Programmable auto-$\overline{\text{RTS}}$ and auto-$\overline{\text{CTS}}$
- ❑ Programmable serial interface characteristics
  - ■ 5-, 6-, 7-, or 8-bit characters
  - ■ Even, odd, mark (always = 1), space (always = 0) or no parity (non parity bit frame) bit generation and detection
  - ■ 1, 1.5, or 2 stop-bit generation
- ❑ False start bit detection
- ❑ Line break generation and detection
- ❑ Fully prioritized interrupt system controls
- ❑ Internal test and loopback capabilities
- ❑ Modem control functions ($\overline{\text{CTS}}$, $\overline{\text{RTS}}$, $\overline{\text{DSR}}$, $\overline{\text{DTR}}$, $\overline{\text{RI}}$ and $\overline{\text{DCD}}$)

6.1.2 IrDA Functions

- ❑ Slow infrared (SIR 115.2 KBAUD), medium infrared (MIR 0.576 MBAUD) and fast infrared (FIR 4.0 MBAUD) operations. Very fast infrared (VFIR) is not supported.
- ❑ Framing error, cyclic redundancy check (CRC) error, illegal symbol (FIR), abort pattern (SIR, MIR) detection.
- ❑ 8-entry status FIFO (with selectable trigger levels) available to monitor frame length and frame errors.

In Table 62, the module I/O description is at the module level.

Table 62. I/O Description

Signal	I/O	Description	Reset
		UART/MODEM Signals (UART 1,2,3)	
RX	I	Serial data input.	Unknown
TX	O	Serial data output.	1
$\overline{\text{CTS}}$	I	Clear to send. Active-low modem status signal. Reading bit 4 of the modem status register checks the condition of $\overline{\text{CTS}}$. Reading bit 0 of that register checks a change of state of $\overline{\text{CTS}}$ since the last read of the modem status register. $\overline{\text{CTS}}$ is used in auto-$\overline{\text{CTS}}$ mode to control the transmitter.	Unknown
$\overline{\text{RTS}}$	O	Request to send. When active (low), the module is ready to receive data. Setting modem control register bit 1 activates $\overline{\text{RTS}}$. It becomes inactive as a result of a module reset, loopback mode, or by clearing the MCR[1]. In auto-$\overline{\text{RTS}}$ mode, it becomes inactive as a result of the receiver threshold logic.	1
$\overline{\text{DSR}}$	I	Data set ready. Active-low modem status signal. Reading bit 5 of the modem status register checks the condition of $\overline{\text{DSR}}$. Reading bit 1 of that register checks a change of state of $\overline{\text{DSR}}$ since the last read of the modem status register.	Unknown
$\overline{\text{DTR}}$	O	Data terminal ready. Active-low modem control signal. Reading bit 0 of the modem control register checks the condition of $\overline{\text{DTR}}$.	1
$\overline{\text{DCD}}$	I	Data carrier detect. Active-low modem status signal. The condition of $\overline{\text{DCD}}$ is checked by reading bit 7 of the modem status register, and any change in its state can be detected by reading bit 3 of that register.	Unknown
$\overline{\text{RI}}$	I	Reading indicator. Active-low modem status signal. The condition of $\overline{\text{RI}}$ is checked by reading bit 6 of the modem status register, and any change in its state is detected by reading bit 2 of that register.	Unknown

Table 62. I/O Description (Continued)

Signal	I/O	Description	Reset
		IrDA Signals (UART1 and UART3 only)	
RXIR	I	Serial data input.	Unknown
TXIR	O	Serial data output.	0
SD	O	Signal used to configure transceivers.	1

6.2 Control and Status Registers Description

Each register is selected using a combination of address and some LCR register bit(s) settings as shown in the following Table 63.

6.2.1 UART IrDA Registers Mapping

The local host can access the following registers at address = module base address + address offset. The module base address is the module start address. Register address offsets depend on the module address alignment at the system top level. The address offsets (0x13 x S) and (0x18 x S to 0x31 x S (inclusive)) are reserved and must be read as 0x00 at all times.

- S = 1 for 8-bit aligned addresses
- 2 for 16-bit aligned addresses
- 4 for 32-bit aligned addresses

All UART registers are 8-bit. Start addresses:

- UART1: FFFB 0000
- UART2: FFFB 0800
- UART3: FFFB 9800

Table 63. UART IrDA Registers

Address Offset	Registers					
	LCR[7] = 0		LCR[7] = 1 and LCR[7:0] is not 0xBF		LCR[7:0] = 0xBF	
	READ	WRITE	READ	WRITE	READ	WRITE
0x00 x S	RHR	THR	DLL	DLL	DLL	DLL
0x01 x S	IER §	IER §	DLH	DLH	DLH	DLH
0x02 x S	IIR	FCR‡	IIR	FCR‡	EFR	EFR
0x03 x S	LCR	LCR	LCR	LCR	LCR	LCR
0x04 x S	MCR‡	MCR‡	MCR‡	MCR‡	XON1/ADDR1	XON1/ADDR1
0x05 x S	LSR	–	LSR	–	XON2/ADDR2	XON2/ADDR2
0x06 x S	MSR/TCR†	TCR†	MSR/TCR†	TCR†	XOFF1/TCR†	XOFF1/TCR†
0x07 x S	SPR/TLR†	SPR/TLR†	SPR/TLR†	SPR/TLR†	XOFF2/TLR†	XOFF2/TLR†
0x08 x S	MDR1	MDR1	MDR1	MDR1	MDR1	MDR1
0x09 x S	MDR2	MDR2	MDR2	MDR2	MDR2	MDR2
0x0A x S	SFLSR	TXFLL	SFLSR	TXFLL	SFLSR	TXFLL
0x0B x S	RESUME	TXFLH	RESUME	TXFLH	RESUME	TXFLH
0x0C x S	SFREGL	RXFLL	SFREGL	RXFLL	SFREGL	RXFLL
0x0D x S	SFREGH	RXFLH	SFREGH	RXFLH	SFREGH	RXFLH
0x0E x S	BLR	BLR	UASR	–	UASR	–
0x0F x S	ACREG	ACREG	–	–	–	–
0x10 x S	SCR	SCR	SCR	SCR	SCR	SCR
0x11 x S	SSR	–	SSR	–	SSR	–
0x12 x S	EBLR	EBLR	–	–	–	–
0x14 x S	MVR	–	MVR	–	MVR	–
0x15 x S	SYSC	SYSC	SYSC	SYSC	SYSC	SYSC

† In UART modes, IER[7:4] can only be written when EFR[4] = 1. In IrDA modes, EFR[4] has no effect on access to IER[7:4].
‡ MCR[7:5] and FCR[5:4] can only be written when EFR[4] = 1.
§ Transmission control register (TCR) and trigger level register (TLR) are accessible only when EFR[4] = 1 and MCR[6] = 1.

Table 63. UART IrDA Registers (Continued)

Address Offset	Registers					
	LCR[7] = 0		LCR[7] = 1 and LCR[7:0] is not 0xBF		LCR[7:0] = 0xBF	
	READ	WRITE	READ	WRITE	READ	WRITE
0x16 x S	SYSS	SYSS	SYSS	SYSS	SYSS	SYSS
0x17 x S	WER	WER	WER	WER	WER	WER

† In UART modes, IER[7:4] can only be written when EFR[4] = 1. In IrDA modes, EFR[4] has no effect on access to IER[7:4].
‡ MCR[7:5] and FCR[5:4] can only be written when EFR[4] = 1.
§ Transmission control register (TCR) and trigger level register (TLR) are accessible only when EFR[4] = 1 and MCR[6] = 1.

Offset Address (hex): 0x00 x S and LCR[7] = 0 and read

The receiver section consists of the receiver holding register (RHR) and the receiver shift register. The RHR is actually a 64-byte FIFO. The receiver shift register receives serial data from RX input. The data is converted to parallel data and moved to the RHR. If the FIFO is disabled, location 0 of the FIFO is used to store the single data character.

If an overflow occurs the data in the RHR is not overwritten.

Table 64. Receive Holding Register (RHR)

Bit	Name	Function	R/W	Reset
7:0	RHR	Receive holding register	R	Unknown

Offset Address (hex): 0x00 x S and LCR[7] = 0 and write

The transmitter section consists of the transmit holding register (THR) and the transmit shift register. The transmit holding register is actually a 64-byte FIFO. The LH writes data to the THR. The data is placed into the transmit shift register where it is shifted out serially on the TX output. If the FIFO is disabled, location 0 of the FIFO is used to store the data.

Table 65. Transmit Holding Register (THR)

Bit	Name	Function	R/W	Reset
7:0	THR	Transmit holding register	W	Unknown

Offset Address (hex): 0x02 x S and LCR not 0xBF (and EFR[4] = 1 for FCR[5:4]) and write.

Table 66. FIFO Control Register (FCR)

Bit	Name	Function	R/W	Reset
7:6	RX_FIFO_TRIG	Sets the trigger level for the RX FIFO: If SCR[7] = 0 and TLR[7:4] = 0000: 00: 8 characters 01: 16 characters 10: 56 characters 11: 60 characters If SCR[7] = 0 and TLR[7:4] non-0, RX_FIFO_TRIG is not considered. If SCR[7] = 1, RX_FIFO_TRIG is 2 LSB of the trigger level (1−63 on 6 bits) with the granularity 1.	W	00
5:4	TX_FIFO_TRIG	Sets the trigger level for the TX FIFO: If SCR[6] = 0 and TLR[3:0] = 0000: 00: 8 spaces 01: 16 spaces 10: 32 spaces 11: 56 spaces If SCR[6] = 0 and TLR[3:0] non-0, TX_FIFO_TRIG is not considered. If SCR[6] = 1, TX_FIFO_TRIG is 2 LSB of the trigger level (1−63 on 6 bits) with the granularity 1.	W	00
3	DMA_MODE	0: DMA_MODE 0 (No DMA) 1: DMA_MODE 1 (UART_nDMA_REQ[0] in TX, UART_nDMA_REQ[1] in RX) This register is considered if SCR[0] = 0.	W	0
2	TX_FIFO_CLEAR	0: No change. 1: Clears the transmit FIFO and resets its counter logic to 0. Returns to 0 after clearing FIFO.	W	0
1	RX_FIFO_CLEAR	0: No change. 1: Clears the receive FIFO and resets its counter logic to 0. Returns to 0 after clearing FIFO.	W	0
0	FIFO_EN	0: Disables the transmit and receive FIFOs. 1: Enables the transmit and receive FIFOs.	W	0

Notes: 1) Bits 4 and 5 can only be written to when EFR[4] = 1.

2) Bits 0 to 3 can be changed only when the baud clock is not running (DLL and DLH set to 0).

3) See for FCR[5:4] setting restriction when SCR[6] = 1.

4) See for FCR[7:6] setting restriction when SCR[7] = 1.

Offset Address (hex): 0x10 x S

Table 67. Supplementary Control Register (SCR)

Bit	Name	Function	R/W	Reset
7	RX_TRIG_GRANU1	0: Disables the granularity of 1 for trigger RX level. 1: Enables the granularity of 1 for trigger RX level.	R/W	0
6	TX_TRIG_GRANU1	0: Disables the granularity of 1 for trigger TX level. 1: Enables the granularity of 1 for trigger TX level.	R/W	0
5	DSR_IT	0: Disables $\overline{\text{DSR}}$ interrupt. 1: Enables $\overline{\text{DSR}}$ interrupt.	R/W	0
4	RX_CTS_DSR_ WAKE_UP_ENABLE	0: Disables the wake-up interrupt and clears SSR[1]. 1: Waits for a falling edge of pins RX, $\overline{\text{CTS}}$ or $\overline{\text{DSR}}$ to generate an interrupt.	R/W	0
3	TX_EMPTY_CTL_IT	0: Normal mode for THR interrupt (See UART mode interrupts table.) 1: The THR interrupt is generated when TX FIFO and TX shift register are empty.	R/W	0
2:1	DMA_MODE_2	Used to specify the DMA mode valid if SCR[0] = 1 00: DMA mode 0 (no DMA) 01: DMA mode 1 (UART_nDMA_REQ[0] in TX, UART_nDMA_REQ[1] in RX) 10: DMA mode 2 (UART_nDMA_REQ[0] in RX) 11: DMA mode 3 (UART_nDMA_REQ[0] in TX)	R/W	00
0	DMA_MODE_CTL	0: The DMA_MODE is set with FCR[3]. 1: The DMA_MODE is set with SCR[2:1].	R/W	0

Bit 4 enables the wake-up interrupt, but this interrupt is not mapped into the IIR register. Therefore, when an interrupt occurs and there is no interrupt pending in the IIR register, the SSR[1] bit must be checked. To clear the wake-up interrupt, bit SCR[4] must be reset to 0.

Offset Address (hex): 0x03 x S

LCR[6:0] defines parameters of the transmission and reception.

Table 68. Line Control Register (LCR)

Bit	Name	Function	R/W	Reset
7	DIV_EN	0: Normal operating condition. 1: Divisor latch enable. Allows access to DLL, DLH, and other registers (refer to the registers' mapping).	R/W	0
6	BREAK_EN	Break control bit 0: Normal operating condition. 1: Forces the transmitter output to go low to alert the communication terminal.	R/W	0
5	PARITY_TYPE2		R/W	0
4	PARITY_TYPE1	0: Odd parity is generated (if LCR[3] = 1). 1: Even parity is generated (if LCR[3] = 1).	R/W	0
3	PARITY_EN	0: No parity. 1: A parity bit is generated during transmission and the receiver checks for received parity.	R/W	0
2	NB_STOP	Specifies the number of stop bits: 0: 1 stop bits (word length = 5, 6, 7, 8) 1: 1.5 stop bits (word length = 5) 1−2 stop bits (word length = 6, 7, 8)	R/W	0
1:0	CHAR_LENGTH	Specifies the word length to be transmitted or received 00: 5 bits 01: 6 bits 10: 7 bits 11: 8 bits	R/W	00

Offset Address (hex): 0x05 x S and LCR is not 0xBF and read

Table 69. Line Status Register (LSR) (UART Mode)

Bit	Name	Function	R/W	Reset
7	RX_FIFO_STS	0: Normal operation. 1: At least one parity error, framing error, or break indication in the receiver FIFO. Bit 7 is cleared when no more errors are present in the FIFO.	R	0
6	TX_SR_E	0: Transmitter hold and shift registers are not empty. 1: Transmitter hold and shift registers are empty.	R	1
5	TX_FIFO_E	0: Transmit hold register is not empty. 1: Transmit hold register is empty. The processor can now load up to 64 bytes of data into the THR if the TX FIFO is enabled.	R	1
4	RX_BI	0: No break condition. 1: A break was detected while the data being read from the RX FIFO was being received (that is, RX input was low for one character time frame).	R	0
3	RX_FE	0: No framing error in data being read from RX FIFO. 1: Framing error occurred in data being read from RX FIFO (received data did not have a valid stop bit).	R	0
2	RX_PE	0: No parity error in data being read from RX FIFO. 1: Parity error in data being read from RX FIFO.	R	0
1	RX_OE	0: No overrun error. 1: Overrun error has occurred. Set when the character held in the receive shift register is not transferred to the RX FIFO. This case occurs only when receive FIFO is full.	R	0
0	RX_FIFO_E	0: No data in the receive FIFO. 1: At least one data character in the RX_FIFO.	R	0

When the LSR is read, LSR[4:2] reflects the error bits [BI, FE, PE] of the character at the top of the RX FIFO (next character to be read). Therefore, reading the LSR, and then reading the RHR, identifies errors in a character.

Reading RHR updates [BI, FE, PE] (See Table 109, *UART Mode Interrupts*.)

LSR [7] is set when there is an error anywhere in the RX FIFO, and is cleared only when there are no more errors remaining in the FIFO.

Reading the LSR does not cause an increment of the RX FIFO read pointer. The RX FIFO read pointer is incremented by reading the RHR.

Reading LSR clears [OE] if set (See Table 109, UART Mode Interrupts.)

Table 70. Line Status Register (LSR) (IR Mode)

Bit	Name	Function	R/W	Reset
7	THR_EMPTY	0: Transmit holding register is not empty. 1: Transmit holding register is empty. The processor can now load up to 64 bytes of data into the THR if the TX FIFO is enabled.	R	1
6	STS_FIFO_FULL	0: Status FIFO not full. 1: Status FIFO full.	R	0
5	RX_LAST_BYTE	0: The RX FIFO does not contain the last byte of the frame to be read. 1: The RX FIFO contains the last byte of the frame to be read. This bit is set only when the last byte of a frame is available to be read. It is used to determine the frame boundary. It is cleared on a single read of the LSR register.	R	0
4	FRAME_TOO_LONG	0: No frame-too-long error in frame. 1: Frame-too-long error in the frame at the top of the STATUS FIFO, [next character to be read]. This bit is set to 1 when a frame exceeding the maximum length (set by RXFLH and RXFLL registers) has been received. When this error is detected, current frame reception is terminated. Reception is stopped until the next START flag is detected.	R	0
3	ABORT	0: No abort pattern error in frame. 1: Abort pattern is received. SIR & MIR: Abort pattern FIR: Illegal symbol 0000	R	0
2	CRC	0: No CRC error in frame. 1: CRC error in the frame at the top of the STATUS FIFO (next character to be read).	R	0

Table 70. Line Status Register (LSR) (IR Mode) (Continued)

Bit	Name	Function	R/W	Reset
1	STS_FIFO_E	0: Status FIFO not empty. 1: Status FIFO empty.	R	1
0	RX_FIFO_E	0: At least one data character in the RX_FIFO. 1: No data in the receive FIFO.	R	1

When the LSR is read, LSR[4:2] reflects the error bits [FL, CRC, ABORT] of the frame at the top of the STATUS FIFO (next frame status to be read). See Table 110, *IrDA Mode Interrupts*.

Offset Address (hex): 0x11 x S and read

Table 71. Supplementary Status Register (SSR)

Bit	Name	Function	R/W	Reset
7:2	–	Reserved	R	000000
1	RX_CTS_DSR_WAKE_UP_STS	0: No falling edge event on RX, $\overline{\text{CTS}}$ and $\overline{\text{DSR}}$. 1: A falling edge occurred on RX, $\overline{\text{CTS}}$ or $\overline{\text{DSR}}$.	R	0
0	TX_FIFO_FULL	0: TX FIFO not full. 1: TX FIFO full.	R	0

Bit 1 is reset only when SCR[4] is reset to 0.

Offset Address (hex): 0x04 x S and LCR is not 0xBF (and EFR[4] = 1 for MCR[7:5])

MCR[3:0] controls the interface with the modem, data set, or peripheral device that is emulating the modem.

Table 72. Modem Control Register (MCR)

Bit	Name	Function	R/W	Reset
7	–	Reserved.	R	0
6	TCR_TLR	0: No action. 1: Enables access to the TCR and TLR registers.	R/W	0
5	XON_EN	0: Disable the *XON any* function. 1: Enable the *XON any* function.	R/W	0

Table 72. Modem Control Register (MCR) (Continued)

Bit	Name	Function	R/W	Reset
4	LOOPBACK_EN	0: Normal operating mode. 1: Enable local loopback mode (internal). In this mode, the MCR[3:0] signals are looped back into MSR[7:4]. The transmit output is looped back to the receive input internally.	R/W	0
3	CD_STS_CH	0: In loopback, forces $\overline{\text{DCD}}$ input high and IRQ outputs to inactive state. 1: In loopback, forces $\overline{\text{DCD}}$ input low and IRQ outputs to inactive state.	R/W	0
2	RI_STS_CH	0: In loopback, forces $\overline{\text{RI}}$ input high. 1: In loopback, forces $\overline{\text{RI}}$ input low.	R/W	0
1	RTS	0: Force $\overline{\text{RTS}}$ output to inactive (high). 1: Force $\overline{\text{RTS}}$ output to active (low). In loopback, controls MSR[4] If auto-RTS is enabled, the $\overline{\text{RTS}}$ output is controlled by hardware flow control.	R/W	0
0	DTR	0: Force $\overline{\text{DTR}}$ output to inactive (high). 1: Force $\overline{\text{DTR}}$ output to active (low).	R/W	0

Bits 5 and 6 can be written only when EFR[4] = 1.

Offset Address (hex): 0x06 x S and LCR is not 0xBF and (EFR[4] = 0 or MCR[6] = 0) and read.

This register provides information about the current state of the control lines from the modem, data set, or peripheral device to the LH. It also indicates when a control input from the modem changes state.

Table 73. Modem Status Register (MSR)

Bit	Name	Function	R/W	Reset
7	NCD_STS	This bit is the complement of the $\overline{\text{DCD}}$ input. In loopback mode it is equivalent to MCR[3].	R	Unknown
6	NRI_STS	This bit is the complement of the $\overline{\text{RI}}$ input. In loopback mode it is equivalent to MCR[2].	R	Unknown
5	NDSR_STS	This bit is the complement of the $\overline{\text{DSR}}$ input. In loopback mode, it is equivalent to MCR[0].	R	Unknown

Table 73. Modem Status Register (MSR) (Continued)

Bit	Name	Function	R/W	Reset
4	NCTS_STS	This bit is the complement of the $\overline{CTS}$ input. In loopback mode it is equivalent to MCR[1].	R	Unknown
3	DCD_STS	Indicates that $\overline{DCD}$ input (or MCR[3] in loopback) has changed. Cleared on a read.	R	0
2	RI_STS	Indicates that $\overline{RI}$ input (or MCR[2] in loopback) has changed state from low to high. Cleared on a read.	R	0
1	DSR_STS	1: Indicates that $\overline{DSR}$ input (or MCR[0] in loop-back) has changed state. Cleared on a read.	R	0
0	CTS_STS	1: Indicates that $\overline{CTS}$ input (or MCR[1] in loopback) has changed state. Cleared on a read.	R	0

6.3 Interrupt Enable Register (IER)

Offset Address (hex): 0x01 x S and LCR[7] = 0 (and EFR[4] = 1 for IER[7:4]—UART modes only)

UART Modes IER

The interrupt enable register (IER) can be programmed to enable/disable any interrupt. This mode has seven types of interrupt: Receiver error, RHR interrupt, THR interrupt, XOFF received, and $\overline{CTS}$/$\overline{RTS}$ change of state from low to high. Each interrupt can be enabled/disabled individually. The IER also has a sleep-mode enable bit.

Table 74. Interrupt Enable Register (IER) (UART Mode)

Bit	Name	Function	R/W	Reset
7	CTS_IT	0: Disables the $\overline{CTS}$ interrupt. 1: Enables the $\overline{CTS}$ interrupt.	R/W	0
6	RTS_IT	0: Disables the $\overline{RTS}$ interrupt. 1: Enables the $\overline{RTS}$ interrupt.	R/W	0
5	XOFF_IT	0: Disables the XOFF interrupt. 1: Enables the XOFF interrupt.	R/W	0
4	SLEEP_MODE	0: Disables sleep mode. 1: Enables sleep mode (stop baud rate clock when the module is inactive).	R/W	0

Table 74. Interrupt Enable Register (IER) (UART Mode) (Continued)

Bit	Name	Function	R/W	Reset
3	MODEM_STS_IT	0: Disables the modem status register interrupt. 1: Enables the modem status register interrupt.	R/W	0
2	LINE_STS_IT	0: Disables the receiver line status interrupt. 1: Enables the receiver line status interrupt.	R/W	0
1	THR_IT	0: Disables the THR interrupt. 1: Enables the THR interrupt.	R/W	0
0	RHR_IT	0: Disables the RHR interrupt and time-out interrupt. 1: Enables the RHR interrupt and time-out interrupt.	R/W	0

Bits 4, 5, 6, and 7 can only be written when EFR[4] = 1.

IrDA Modes IE

These modes have eight types of interrupts: received EOF, LSR interrupt, TX status, status FIFO interrupt, RX overrun, last byte in RX FIFO, THR interrupt, and RHR interrupt. All can be enabled/disabled individually.

Table 75. Interrupt Enable Register (IER) (IrDA Mode)

Bit	Name	Function	R/W	Reset
7	EOF_IT	0: Disables the received EOF interrupt. 1: Enables the received EOF interrupt.	R/W	0
6	LINE_STS_IT	0: Disables the receiver line status interrupt. 1: Enables the receiver line status interrupt.	R/W	0
5	TX_STATUS_IT	0: Disables the TX status interrupt. 1: Enables the TX status interrupt.	R/W	0
4	STS_FIFO_TRIG_IT	0: Disables status FIFO trigger level interrupt. 1: Enables status FIFO trigger level interrupt.	R/W	0
3	RX_OVERRUN_IT	0: Disables the RX overrun interrupt. 1: Enables the RX overrun interrupt.	R/W	0
2	LAST_RX_BYTE_IT	0: Disables the last byte of frame in RX FIFO interrupt. 1: Enables the last byte of frame in RX FIFO interrupt.	R/W	0

Table 75. Interrupt Enable Register (IER) (IrDA Mode) (Continued)

Bit	Name	Function	R/W	Reset
1	THR_IT	0: Disables the THR interrupt. 1: Enables the THR interrupt.	R/W	0
0	RHR_IT	0: Disables the RHR interrupt. 1: Enables the RHR interrupt.	R/W	0

The TX_STATUS_IT interrupt reflects two possible conditions. The MDR2[0] must be read to determine the status in the event of this interrupt.

Offset Address (hex): 0x02 x S and LCR is not 0xBF and read.

The IIR is a read-only register that provides the source of the interrupt in a prioritized manner.

Table 76. Interrupt Identification Register (IIR) (UART Mode)

Bit	Name	Function	R/W	Reset
7:6	FCR_MIRROR	Mirror the contents of FCR[0] on both bits	R	00
5:1	IT_TYPE		R	00000
0	IT_PENDING	0: An interrupt is pending (UART_nIRQ active). 1: No interrupt is pending (UART_nIRQ inactive).	R	1

The UART_nIRQ output is activated whenever one of the eight interrupts is active.

Table 77. IrDA Mode Register (IIR)

Bit	Name	Function	R/W	Reset
7	EOF_IT	0: Received EOF interrupt inactive. 1: Received EOF interrupt active.	R	0
6	LINE_STS_IT	0: Receiver line status interrupt inactive. 1: Receiver line status interrupt active.	R	0
5	TX_STATUS_IT	0: TX status interrupt inactive. 1: TX status interrupt active.	R	0
4	STS_FIFO_IT	0: Status FIFO trigger level interrupt inactive. 1: Status FIFO trigger level interrupt active.	R	0

Table 77. IrDA Mode Register (IIR) (Continued)

Bit	Name	Function	R/W	Reset
3	RX_OE_IT	0: RX overrun interrupt inactive. 1: RX overrun interrupt active.	R	0
2	RX_FIFO_LAST_BYTE_IT	0: Last byte of frame in RX FIFO interrupt inactive. 1: Last byte of frame in RX FIFO interrupt active.	R	0
1	THR_IT	0: THR interrupt inactive. 1: THR interrupt active.	R	0
0	RHR_IT	0: RHR interrupt inactive. 1: RHR interrupt active.	R	0

Offset Address (hex): 0x02 x S and LCR = 0xBF

This register enables or disables enhanced features. Most of the enhanced functions apply only to UART modes, but EFR[4] enables write access to FCR[5:4], the TX trigger level, which is also used in IrDA modes.

Table 78. Enhanced Feature Register (EFR)

Bit	Name	Function	R/W	Reset
7	AUTO_CTS_EN	Auto-CTS enable bit 0: Normal operation. 1: Auto-CTS flow control is enabled, that is, transmission is halted when the $\overline{\text{CTS}}$ pin is high (inactive).	R/W	0
6	AUTO_RTS_EN	Auto-RTS enable bit 0: Normal operation. 1: Auto-RTS flow control is enabled. $\overline{\text{RTS}}$ pin goes high (inactive) when the receiver FIFO HALT trigger level, TCR[3:0], is reached, and goes low (active) when the receiver FIFO RESTORE transmission trigger level is reached.	R/W	0
5	SPECIAL_CHAR_DETECT	0: Normal operation. 1: Special character detect enable. Received data is compared with XOFF2 data. If a match occurs the received data is transferred to FIFO, and IIR bit 4 is set to 1 to indicate that a special character has been detected.	R/W	0

Table 78. Enhanced Feature Register (EFR) (Continued)

Bit	Name	Function	R/W	Reset
4	ENHANCED_EN	Enhanced functions write enable bit 0: Disables writing to IER bits 4−7, FCR bits 4−5, and MCR bits 5−7. 1: Enables writing to IER bits 4−7, FCR bits 4−5, and MCR bits 5−7.	R/W	0
3:0	SW_FLOW_ CONTROL	Combinations of software flow control can be selected by programming bits 3−0. See Table 79.	R/W	0000

Table 79. Software Flow Control Options(EFR[0−3])

Bit 3	Bit 2	Bit 1	Bit 0	TX, RX Software Flow Controls
0	0	X	X	No transmit flow control
1	0	X	X	Transmit XON1, XOFF1
0	1	X	X	Transmit XON2, XOFF2
1	1	X	X	Transmit XON1, XON2: XOFF1, XOFF2
X	X	0	0	No receive flow control
X	X	1	0	Receiver compares XON1, XOFF1
X	X	1	1	Receiver compares XON2, XOFF2
X	X	1	1	Receiver compares XON1, XON2: XOFF1, XOFF2†

† XON1 and XON2 must be set to different values if software flow control is enabled.

Offset Address (hex): 0x04 x S and LCR = 0xBF

Table 80. XON1/ADDR1 Register

Bit	Name	Function	R/W	Reset
7:0	XON_WORD1	Used to store the 8-bit XON1 character in UART modes and ADDR1 address 1 for IrDA modes	R/W	0x00

Offset Address (hex): 0x05 x S and LCR = 0xBF

Table 81. XON2/ADDR2 Register

Bit	Name	Function	R/W	Reset
7:0	XON_WORD2	Used to store the 8-bit XON2 character in UART modes and ADDR2 address 2 for IrDA modes	R/W	0x00

Offset Address (hex): 0x06 x S and LCR = 0xBF and (EFR[4] = 0 or MCR[6] = 0)

Table 82. XOFF1 Register

Bit	Name	Function	R/W	Reset
7:0	XOFF_WORD1	Used to store the 8-bit XOFF1 character used in UART modes	R/W	0x00

Offset Address (hex): 0x07 x S and LCR = 0xBF and (EFR[4] = 0 or MCR[6] = 0)

Table 83. XOFF2 Register

Bit	Name	Function	R/W	Reset
7:0	XOFF_WORD2	Used to store the 8-bit XOFF2 character used in UART modes	R/W	0x00

Offset Address (hex): 0x07 x S and LCR is not 0xBF and (EFR[4] = 0 or MCR[6] = 0)

This R/W register does not control the module in any way. It is a scratchpad register the programmer can use to hold temporary data.

Table 84. Scratchpad Register (SPR)

Bit	Name	Function	R/W	Reset
7:0	SPR_WORD	Scratchpad register	R/W	0x00

6.3.1 Divisor Latches (DLL, DLH)

These two registers store the 14-bit divisor for generation of the baud clock in the baud rate generator. DLH stores the most-significant part of the divisor. DLL stores the least-significant part of the divisor.

DLL and DLH can only be written to before sleep mode is enabled, that is, before IER[4] is set.

Offset Address (hex): 0x00 x S and LCR[7] = 1

Table 85. Divisor Latches LowRegister (DLL)

Bit	Name	Function	R/W	Reset
7:0	CLOCK_LSB	Used to store the 8-bit LSB divisor value	R/W	0x00

Offset Address (hex): 0x01 x S and LCR[7] = 1

Table 86. Divisor Latches High Register (DLH)

Bit	Name	Function	R/W	Reset
7:6	−	Reserved	R	00
5:0	CLOCK_MSB	Used to store the 6-bit MSB divisor value	R/W	000000

Offset Address (hex): 0x06 x S and EFR[4] = 1 and MCR[6] = 1

This register stores the receive FIFO threshold levels to start/stop transmission during hardware/software flow control.

Table 87. Transmission Control Register (TCR)

Bit	Name	Function	R/W	Reset
7:4	RX_FIFO_TRIG_START	RCV FIFO trigger level to RESTORE transmission (0 − 60)	R/W	0x0
3:0	RX_FIFO_TRIG_HALT	RCV FIFO trigger level to HALT transmission (0 − 60)	R/W	0xF

Notes: 1) Trigger levels from 0 − 60 bytes are available with a granularity of 4. (Trigger level = 4 x [4-bit register value])

2) The programmer must ensure that TCR[3:0] > TCR[7:4] whenever auto-RTS or software flow control is enabled, to prevent device malfunction.

3) In FIFO interrupt mode with flow control, the programmer also must ensure that the trigger level to HALT transmission is greater than or equal to the receive FIFO trigger level (either TLR[7:4] or FCR[7:6]). Otherwise, the FIFO operation stalls. This problem does not exist in FIFO DMA mode with flow control because a DMA request is sent each time a byte is received.

Offset Address (hex): 0x07 x S and EFR[4] = 1 and MCR[6] = 1

This register stores the programmable transmit and receive FIFO trigger levels used for DMA and IRQ generation.

Table 88. Trigger Level Register (TLR)

Bit	Name	Function	R/W	Reset
7:4	RX_FIFO_TRIG_DMA	RCV FIFO trigger level	R/W	0x0
3:0	TX_FIFO_TRIG_DMA	Transmit FIFO trigger level	R/W	0x0

Table 89 and Table 90 summarize the different ways to set the trigger levels for the transmit FIFO and the receive FIFO, respectively.

Table 89. TX FIFO Trigger Level Setting Summary

SCR[6]	TLR[3:0]	TX FIFO Trigger Level
0	0000	Defined by FCR[5:4] (either 8,16,32, 56 spaces)
0	Non-zero	Defined by TLR[3:0] (from 4 to 60 spaces with a granularity of 4 spaces)
1	Value	Defined by the concatenated value of TLR[3:0] and FCR [5:4] (from 1 to 63 spaces with a granularity of 1 space). Note: The combination of TLR [3:0] = 0000 and FCR [5:4] = 00 (all zeros) is not supported (min 1 space required). All zeros result in unpredictable behavior.

Table 90. RX FIFO Trigger Level Setting Summary

SCR[7]	TLR[7:4]	RX FIFO Trigger Level
0	0000	Defined by FCR[7:6] (either 8,16,56, 60 characters)
0	Non-zero	Defined by TLR[7:4] (from 4 to 60 characters with a granularity of 4 characters)
1	Value	Defined by the concatenated value of TLR[7:4] and FCR [7:6] (from 1 to 63 characters with a granularity of 1 character) Note: The combination of TLR[7:4] = 0000 and FCR [7:6] = 00 (all zeros) is not supported (min 1 character required). All zeros result in unpredictable behavior.

Offset Address (hex): 0x08 x S

The mode of operation is programmed by writing to MDR1[2:0]. Therefore, the MDR1 must be programmed on start-up after configuring registers DLL, DLH, and LCR. The value of MDR1[2:0] must not be changed again during normal operation.

Table 91. Mode Definition Register 1 (MDR1)

Bit	Name	Function	R/W	Reset
7	FRAME_END_MODE	0: Frame-length method.	R/W	0
		1: Set EOT bit method.		
6	SIP_MODE	MIR/FIR modes only	R/W	0
		0: Manual SIP mode: SIP is generated with the control of ACREG[3].		
		1: Automatic SIP mode: SIP is generated after each transmission.		

Table 91. Mode Definition Register 1 (MDR1) (Continued)

Bit	Name	Function	R/W	Reset
5	SCT	Store and control the transmission	R/W	0
		0: Starts the IrDA transmission as soon as a value is written to THR.		
		1: Starts the IrDA transmission with the control of ACREG[2].		
4	SET_TXIR	Used to configure the IrDA transceiver	R/W	0
		0: No action.		
		1: TXIR pin output is forced high.		
3	IR_SLEEP	0: IrDA sleep mode disabled.	R/W	0
		1: IrDA sleep mode enabled.		
2:0	MODE_SELECT	000: UART 16x mode	R/W	111
		001: SIR mode		
		010: UART 16x autobaud		
		011: UART 13x mode		
		100: MIR mode		
		101: FIR mode		
		110: Reserved		
		111: Disable (default state)		

Offset Address (hex): 0x09 x S

IrDA modes only.

MDR2[0] describes the status of the interrupt in IIR[5]. The IRTX_UNDERRUN bit should be read after an IIR[5] TX_STATUS_IT interrupt has occurred. The bits [2:1] of this register set the trigger level for the frame status FIFO (8 entries) and must be programmed before the mode is programmed in MDR1[2:0].

Table 92. Mode Definition Register 2 (MDR2)

Bit	Name	Function	R/W	Reset
7:3	–	Reserved	R	00000
2:1	STS_FIFO_TRIG	Frame status FIFO threshold select: 00: 1 entry 01: 4 entries 10: 7 entries 11: 8 entries	R/W	00
0	IRTX_UNDERRUN	IRDA transmission status interrupt When the IIR[5] interrupt occurs, the meaning of the interrupt is: 0: IRTX last bit of the frame has been transmitted successfully without error. 1: IRTX underrun has occurred. The last bit of the frame has been transmitted but with an underrun error present. The bit is reset to 0 when the RESUME register is read.	R	0

Offset Address (hex): 0x0E x S and LCR[7] = 1 and read

UART autobauding mode only.

This status register returns the speed, the number of bits by characters, and the type of the parity in UART autobauding mode.

In autobauding mode the input frequency of the UART modem must be fixed to 48 MHz. Any other module clock frequency results in incorrect baud rate recognition.

Table 93. UART Autobauding Status Register (UASR)

Bit	Name	Function	R/W	Reset
7:6	PARITY_TYPE	00: No parity identified 01: Parity space 10: Even parity 11: Odd parity	R	00
5	BIT_BY_CHAR	0: 7 bits character identified. 1: 8 bits character identified.	R	0
4:0	SPEED	Used to report the speed identified 00000: No speed identified 00001: 115 200 bauds 00010: 57 600 bauds 00011: 38 400 bauds 00100: 28 800 bauds 00101: 19 200 bauds 00110: 14 400 bauds 00111: 9 600 bauds 01000: 4 800 bauds 01001: 2 400 bauds 01010: 1 200 bauds	R	00000

This register sets up transmission according to characteristics of previous reception instead of the LCR, DLL, and DLH registers used when UART is in autobauding mode.

To reset the autobauding hardware (to start a new AT detection) or to set the UART in standard mode (no autobaud), MDR1[2:0] must be set to reset state 111, and then to the UART in autobaud mode 010 or UART in standard mode 000.

Usage limitation:

- Only 7- and 8-bit character (5 and 6 bits not supported)
- 7-bit character with space parity not supported
- Baud rate between 1200 and 115200 bp/s (10 possibilities)

6.4 Transmit Frame Length Register (TXFLL, TXFLH)

IrDA modes only.

The registers TXFLL and TXFLH hold the 13-bit transmit frame length (expressed in bytes). TXFLL holds the least-significant bits, and TXFLH holds the most-significant bits. The frame length value is used if the frame length method of frame closing is used.

Offset Address (hex): 0x0A x S and write

Table 94. Transmit Frame Length Low Register (TXFLL)

Bit	Name	Function	R/W	Reset
7:0	TXFLL	LSB register used to specify the frame length.	W	0x00

Offset Address (hex): 0x0B x S and write

Table 95. Transmit Frame Length High Register (TXFLH)

Bit	Name	Function	R/W	Reset
7:5	–	Reserved	R	000
4:0	TXFLH	MSB register used to specify the frame length.	W	00000

6.4.1 Received Frame Length Register (RXFLL, RXFLH)

IrDA modes only.

The registers RXFLL and RXFLH hold the 12-bit receive maximum frame length. RXFLL holds the least-significant bits, and RXFLH holds the most-significant bits. If the intended maximum-receive frame length is n bytes, program RXFLL and RXFLH to be *n* + 3 in SIR or MIR modes and *n* + 6 in FIR mode (+3 and +6 are due to frame format with CRC and stop flag; there are two bytes associated with the FIR stop flag).

Offset Address (hex): 0x0C x S and write

Table 96. Received Frame Length Low Register (RXFLL)

Bit	Name	Function	R/W	Reset
7:0	RXFLL	LSB register used to specify the frame length in reception	W	0x00

Offset Address (hex): 0x0D x S and write

Table 97. Received Frame Length High Register (RXFLH)

Bit	Name	Function	R/W	Reset
7:4	−	Reserved	R	0x0
3:0	RXFLH	MSB register used to specify the frame length in reception	W	0x0

Offset Address (hex): 0x0A x S and read

IrDA modes only.

Reading this register in effect reads frame-status information from the status FIFO. This register does not physically exist. Reading this register increments the status FIFO read pointer (SFREGL and SFREGH must be read first).

Table 98. Status FIFO Line Status Register (SFLSR)

Bit	Name	Function	R/W	Reset
7:5	−	Reserved	R	000
4	OE_ERROR	1: Overrun error in RX FIFO when frame at top of FIFO was received.	R	Unknown
3	FRAME_TOO_LONG_ERROR	1: Frame-length too long error in frame at top of FIFO.	R	Unknown
2	ABORT_DETECT	1: Abort pattern detected in frame at top of FIFO.	R	Unknown
1	CRC_ERROR	1: CRC error in frame at top of FIFO.	R	Unknown
0	−	Reserved	R	0

Offset Address (hex): 0x0B x S and read

IrDA modes only.

This register clears internal flags that halt transmission/reception when an underrun/overrun error occurs. Reading this register resumes the halted operation. This register does not physically exist and reads always as 0x00.

Table 99. Resume Register (RESUME)

Bit	Name	Function	R/W	Reset
7:0	RESUME	Dummy read to restart the TX or RX	R	0x00

6.4.2 Status FIFO Register (SFREGL, SFREGH)

IrDA modes only. The frame lengths of received frames are written into the status FIFO. This information can be read from the SFREGL and SFREGH registers. These registers do not physically exist. The least-significant bits are read from SFREGL, and the most-significant bits are read from SFREGH. Reading these registers does not alter the status FIFO read pointer. These registers must be read before the pointer is incremented by reading the SFLSR.

Offset Address (hex): 0x0C x S and read

Table 100. Status FIFO Register Low (SFREGL)

Bit	Name	Function	R/W	Reset
7:0	SFREGL	LSB part of the frame length	R	Unknown

Offset Address (hex): 0x0D x S and read

Table 101. Status FIFO Register High (SFREGH)

Bit	Name	Function	R/W	Reset
7:4	−	Reserved	R	0x0
3:0	SFREGH	MSB part of the frame length	R	Unknown

Offset Address (hex): 0x0E x S and LCR[7] = 0

IrDA modes only. BLR[6] is used to select whether 0xC0 or 0xFF start patterns are to be used when multiple start flags are required in SIR mode. If only one start flag is required, the start pattern is always 0xC0. If *n* start flags are required, either (*n*−1) 0xC0 or (*n*−1) 0xFF flags are sent, followed by a single 0xC0 flag immediately preceding the first data byte.

Table 102. BOF Control Register (BLR)

Bit	Name	Function	R/W	Reset
7	STS_FIFO_RESET	Status FIFO reset. This bit is self-clearing.	R/W	0
6	XBOF_TYPE	SIR xBOF select 0: 0xFF. 1: 0xC0.	R/W	1
5:0	−	Reserved	R	000000

Offset Address (hex): 0x12 x S and LCR[7] = 0

IrDA modes only.

This register specifies the number of BOF + xBOFs to transmit in IrDA SIR operation. The value set into this register must take into account the BOF character. To send only one BOF with no XBOF, this register must be set to 1. To send one BOF with N XBOF, this register must be set to *n*+1. Furthermore, the value 0 sends 1 BOF plus 255 XBOF.

In IrDA MIR mode, this register specifies the number of additional start flags (MIR protocol mandates a minimum of two start flags).

Table 103. BOF Length Register (EBLR)

Bit	Name	Function	R/W	Reset
7:0	EBLR	This register allows definition up to 176 xBOFs, the maximum required by IrDA specification.	R/W	0x00

Offset Address (hex): 0x0F x S and LCR[7] = 0

IrDA modes only.

Table 104. Auxiliary Control Register (ACREG)

Bit	Name	Function	R/W	Reset
7	PULSE_TYPE	SIR pulse width select 0: 3/16 of baud-rate pulse width. 1: 1.6 µs.	R/W	0
6	SD_MOD	Primary output used to configure transceivers. Connected to the SD/MODE input pin of IrDA transceivers 0: SD pin is set to high. 1: SD pin is set to low.	R/W	0
5	DIS_IR_RX	0: Normal operation (Note: RXIR input automatically disabled during transmit but enabled outside of transmit operation.) 1: Disables RXIR input (permanent state−independent of transmit). Hence, RX_IR is disabled when either TX is active or ACREG[5] = 1	R/W	0
4	DIS_TX_UNDERRUN	0: Long stop bits cannot be transmitted, and TX underrun is enabled. 1: Long stop bits can be transmitted, and TX underrun is disabled.	R/W	0

Table 104. Auxiliary Control Register (ACREG) (Continued)

Bit	Name	Function	R/W	Reset
3	SEND_SIP	MIR/FIR modes only Send serial infrared interaction pulse (SIP) 0: No action. 1: Send SIP pulse. If this bit is set during a MIR/FIR transmission, the SIP is sent at the end of it. This bit is automatically cleared at the end of the SIP transmission.	R/W	0
2	SCTX_EN	Store and controlled TX start When MDR1[5] = 1 and the LH writes 1 to this bit, the TX state machine starts frame transmission. This bit is self-clearing.	R/W	0
1	ABORT_EN	Frame abort The LH can intentionally abort transmission of a frame by writing 1 to this bit. Neither the end flag nor the CRC bits are appended to the frame.	R/W	0
0	EOT_EN	EOT (end of transmission) bit The LH writes 1 to this bit just before it writes the last byte to the TX FIFO in set-EOT bit frame closing method. This bit is automatically cleared when the LH writes to the THR (TX FIFO).	R/W	0

Offset Address (hex): 0x14 x S and read

Table 105. Module Version Register (MVR)

Bit	Name	Function	R/W	Reset
7:4	MAJOR_REV	Major revision number of the module	R	1
3:0	MINOR_REV	Minor revision number of the module	R	–

Note: UART/IRDA SIR only module is revision 1.x (WMU_012_1 specification). UART/IRDA with SIR, MIR, and FIR support is revision 2.x (**this specification)**.

Offset Address (hex):0x15 x S

The autoidle bit controls a power-saving technique to reduce the logic power consumption of the OCP interface. That is, when the feature is enabled, the clock is gated off until an OCP command for this device has been detected.

When the software reset bit is set high, it causes a full device reset.

Table 106. System Configuration Register (SYSC)

Bit	Name	Function	R/W	Reset
7:5	–	Reserved	R	000
4:3	IdleMode	Power management request/acknowledge control 00: Force idle. An idle request is acknowledged unconditionally. 01: No idle. An idle request is never acknowledged. 10: Smart idle. Acknowledgement to an idle request is given based on the internal activity of the module. 11: Reserved. Ref: OCP design guidelines version 1.1	R/W	00
2	EnaWakeUp	Wake-up feature control 0: Wake up is disabled. 1: Wake-up capability is enabled.	R/W	0
1	SoftReset	Software reset Set this bit to 1 to trigger a module reset. This bit is automatically reset by the hardware. During reads it always returns a 0. 0: Normal mode. 1: The module is reset.	R/W	0
0	AutoIdle	Internal OCP clock gating strategy 0: Clock is running. 1: Automatic OCP clock gating strategy is applied, based on the OCP interface activity.	R/W	0

Offset Address (hex): 0x16 x S

Table 107. System Status Register (SYSS)

Bit	Name	Function	R/W	Reset
7:1	–	Reserved	R	0000000
0	ResetDone	Internal reset monitoring 0: Internal module reset is ongoing. 1: Reset completed.	R	0

Offset Address (hex): 0x17 x S

The UART wake-up enable register masks and unmasks a UART event that would subsequently notify the system. Such events are any activity in the logic that can cause an interrupt and/or any that require the system to wake up. Even if the wake-up is disabled for certain events, if these events also interrupt the UART, the UART still registers the interrupt as such.

Table 108. Wake-Up Enable Register (WER)

Bit	Name	Function	R/W	Reset
7	–	Reserved	R	0
6	Event 6 Receiver line status interrupt	0: Event is not allowed to wake up the system. 1: Event can wake up the system.	W/R	1
5	Event 5 RHR interrupt	0: Event is not allowed to wake up the system. 1: Event can wake up the system.	W/R	1
4	Event 4 RX/ RXIR activity	0: Event is not allowed to wake up the system. 1: Event can wake up the system.	W/R	1
3	Event 3 DCD (CD) activity	0: Event is not allowed to wake up the system. 1: Event can wake up the system.	W/R	1
2	Event 2 RI activity	0: Event is not allowed to wake up the system. 1: Event can wake up the system.	W/R	1
1	Event 1 DSR activity	0: Event is not allowed to wake up the system. 1: Event can wake up the system.	W/R	1
0	Event 0 CTS activity	0: Event is not allowed to wake up the system. 1: Event can wake up the system.	W/R	1

6.5 Different Modes of Operation

The UART/IRDA module can operate in six different modes:

1) UART 16x mode (≤230.4K bits/s)
2) UART 16x mode with autobauding (≥1200 bits/s and ≤115.2K bits/s)
3) UART 13x mode (≥460.8K bits/s)
4) IrDA SIR mode (≤115.2K bits/s)
5) IrDA MIR mode (0.576 and 1.152M bits/s)
6) IrDA FIR mode (4M bits/s)

The module performs serial-to-parallel conversion on received data characters, and parallel-to-serial conversion on data characters the processor transmits. The complete status of each channel of the module and each received character/frame can be read at any time during functional operation via the line status register (LSR).

The module can be placed in an alternate mode (FIFO mode), relieving the processor of excessive software overhead by buffering received/transmitted characters. Both the receiver and transmitter FIFOs store up to 64 bytes of data (plus 3 additional bits of error status per byte for the receiver FIFO) and have selectable trigger levels.

Both interrupts and DMA are available to control the data flow between the LH and the module.

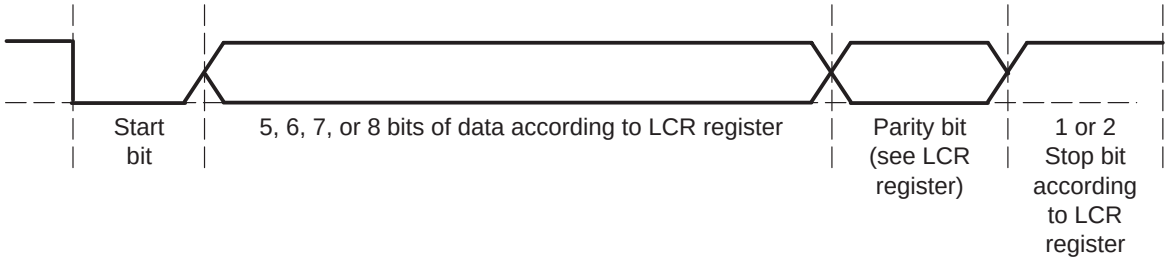
6.5.1 UART Modes

The UART uses a wired interface for serial communication with a remote device.

The module can use hardware or software flow control to manage transmission/ reception. Hardware flow control uses the RTS output and CTS input signals to automatically control serial data flow. This significantly reduces software overhead and increases system efficiency. Software flow control uses programmable XON/XOFF characters to automatically control data flow.

The UART modem module is enhanced with an autobauding functionality, which, in control mode, allows for automatic setting of the speed, number of bits per character, and parity.

Figure 62. UART Data Format



6.5.2 SIR Mode

In slow infrared (SIR) mode, data transfer takes place between the LH and peripheral devices at speeds up to 115200 bauds. A SIR transmit frame begins with start flags (either a single 0xC0, multiple 0xC0, or a single 0xC0 preceded by a number of 0xFF flags), followed by frame data, CRC−16, and ends with a stop flag (0xC1).

BLR[6] is used to select whether 0xC0 or 0xFF start patterns are to be used, when multiple start flags are required.

The SIR transmit state machine attaches start flags, CRC−16, and stop flags. It checks the outgoing data to determine whether data transparency is required.

SIR transparency is carried out if the outgoing data, between the start and stop flags, contains 0xC0, 0xC1, or 0x7D. If one of these is about to be transmitted, the SIR state machine sends an escape character (0x7D) first, then inverts the fifth bit of the real data to be sent, and sends this data immediately after the 0x7D character.

The SIR receive state machine recovers the receive clock, removes the start flags, removes any transparency from the incoming data, and determines frame boundary with reception of the stop flag. It also checks for errors such as frame abort (0x7D character followed immediately by a 0xC1 stop flag, without transparency), CRC error, and frame-length error. At the end of a frame reception, the LH reads the line status register (LSR) to find possible errors in the received frame.

The module can transfer data both ways, but when the device is transmitting, hardware automatically disables the IR RX circuitry. Refer to Table 104, Auxiliary Control Register Bit 5, for a description of the logical operation of all three modes, SIR, MIR, and FIR.

The infrared output in SIR mode can be either 1.6 µs or 3/16 encoding, selected by the PULSE_TYPE bit of the auxiliary control register (ACREG[7]). In 1.6 µs encoding, the infrared pulse width is 1.6 µs, and in 3/16 encoding the infrared pulse width is 3/16 of a bit duration (1/baud-rate).

The transmitting device must send at least 2 start flags at the start of each frame for back-to-back frames. Reception supports variable-length stop bits.

Frame Format

Figure 63. IrDA SIR Frame Format

The CRC is applied on the address (A), control (C), and information (I) bytes.

The two words of CRC are written in the FIFO in reception.

Asynchronous Transparency

Before transmitting a byte, the UART IrDA controller examines each byte of the payload and the CRC field (between BOF and EOF). For each byte equal to 0xC0 (BOF), 0xC1 (EOF), or 0x7D (control escape), it does the following.

In transmission:

- Inserts a control escape (CE) byte preceding the byte.
- Complements bit 5 of the byte (that is, exclusive ORs the byte with 0x20).

The byte sent for the CRC computation is the initial byte written in the TX FIFO (before the XOR with 0x20).

In reception:

For the A, C, I, and CRC fields:

- Compares the byte with the CE byte. If not equal, sends it to the CRC detector and stores it in the RX FIFO.
- If equal to CE, discards the CE byte.
- Complements bit 5 of the byte following the CE.
- Sends the complemented byte to the CRC detector and stores it in the RX FIFO.

Abort Sequence

The transmitter may decide to prematurely close a frame. The transmitter aborts by sending the 0x7DC1 sequence. The abort pattern closes the frame without a CRC field or an ending flag.

It is possible to abort a transmission frame by programming the ABORT_EN bit of the auxiliary control register (ACREG [1]).

When this bit is set to 1, 0x7D and 0xC1 are transmitted and the frame is not terminated with CRC or stop flags.

The receiver treats a frame as an aborted frame when a 0x7D character followed immediately by a 0xC1 character is received without transparency.

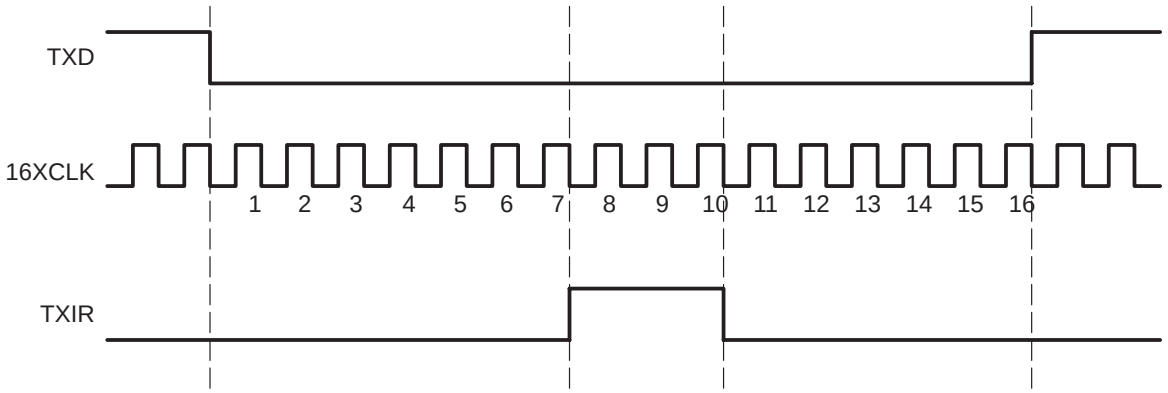
Pulse Shaping

In SIR mode both the 3/16 and the 1.6 ìs pulse duration methods are supported. ACREG[7] selects the pulse-width method in transmit mode.

Encoder

Serial data from the transmit state machine is encoded to transmit data to the optoelectronics. While the serial data input to the (TXD) is high, the output (TXIR) is always low, and the counter used to form a pulse on TXIR is continuously cleared. After TXD resets to 0, TXIR rises on the falling edge of the 7th 16XCLK. On the falling edge of the 10th 16XCLK pulse, TXIR falls, creating a 3-clock-wide pulse. While TXD stays low, a pulse is transmitted during the 7th to the 10th clocks of each 16-clock bit cycle.

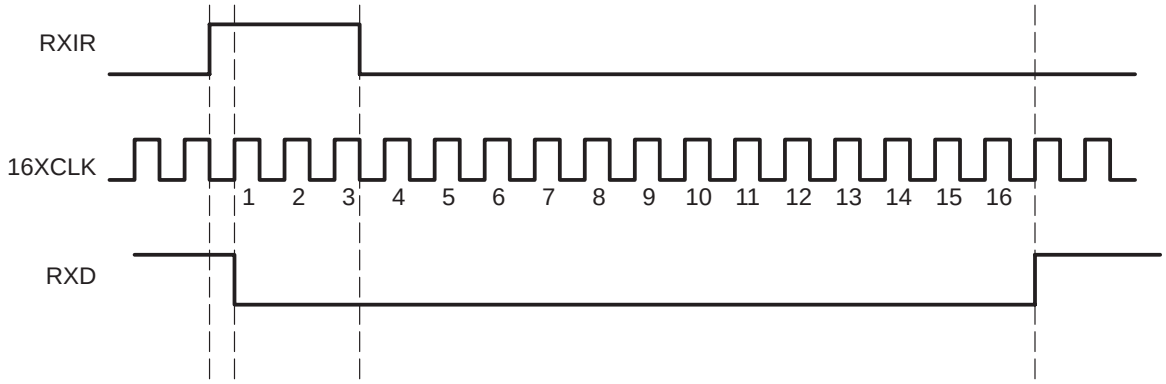
Figure 64. IrDA Encoder Mechanism



Decoder

After reset, RXD is high and the 4-bit counter is cleared. When a rising edge is detected on RXIR, RXD falls on the next rising edge of 16XCLK with sufficient setup time. RXD stays low for 16 cycles (16XCLK) and then returns to high as required by the IrDA specification. As long as no pulses (rising edges) are detected on the RXIR, RXD remains high.

Figure 65. IrDA Decoder Mechanism



The reception of RXIR input is disabled with DIS_IR_RX bits of the auxiliary control register (ACREG[5]).

IR Address Checking

In all IR modes, if address checking has been enabled, only frames intended for the device are written to the RX FIFO. This is to avoid receiving frames not meant for this device in a multi-point infrared environment. It is possible to program two frame addresses that the UART IrDA receives with XON1/ADDR1 and XON2/ADDR2 registers.

Selecting address1 checking is done by setting EFR[0] to 1, and address2 checking is done by setting EFR[1] to 1. Setting EFR[1:0] to 0 disables all address checking operations. If both bits are set, the incoming frame is checked for both private and public addresses.

If address checking is disabled, all received frames are written into the reception FIFO.

6.5.3 MIR Mode

In medium infrared (MIR) mode, data transfer takes place between the LH and peripheral devices at 0.576 or 1.152M bits/s speed. A MIR transmit frame begins with start flags (at least 2), followed by a frame data, CRC−16, and ends with a stop flag.

6.5.4 MIR Transmit Frame Format

On transmit, the MIR state machine attaches start flags, CRC−16, and stop flags. It also looks for 5 consecutive 1s in the frame data and automatically inserts 0 after them. (This is called bit stuffing.)

Figure 66. MIR Transmit Frame Format

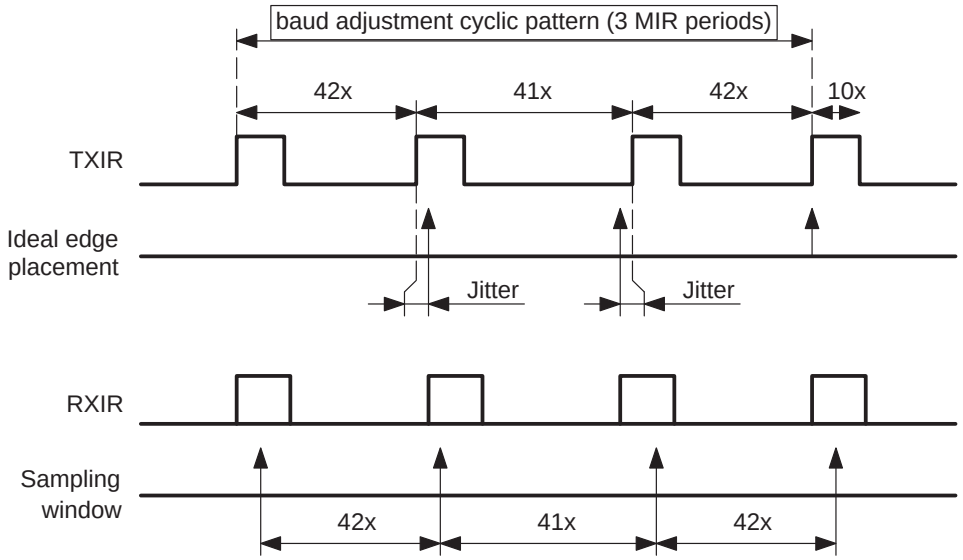
On receive, the MIR receive state machine recovers the receive clock, removes the start flags, destuffs the incoming data, and determines the frame boundary with reception of the stop flag. It also checks for errors such as frame abort, CRC error, or frame-length error. At the end of a frame reception, the LH reads the line status register (LSR) to discover possible errors in the received frame.

The module transfers data both ways, but when the device is transmitting, hardware automatically disables the IR RX circuitry. Refer to Table 104, Auxiliary Control Register Bit 5, for a description of the logical operation of all three modes, SIR, MIR, and FIR.

MIR Encoder/Decoder

In order to meet MIR baud-rate tolerance of ±0.1% with a 48-Mhz clock input, a 42−41−42 encoding/decoding adjustment is performed. The reference start point is 1st start flag, and the 42−41−42 cyclic pattern is repeated until the stop flag is sent or detected. The jitter created this way is within MIR tolerances. The pulse width is not exactly 1/4 but within tolerances defined by the IrDA specifications.

Figure 67. MIR Baud-Rate Adjustment Mechanism

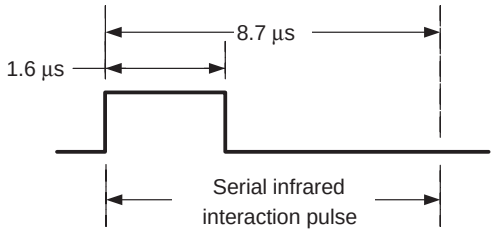


SIP Generation

In MIR and FIR operation modes, the transmitter needs to send a serial infrared interaction pulse (SIP) at least once every 500 ms. The purpose of the SIP is to let slow devices (operating in SIR mode) know that the medium is currently occupied.

The SIP pulse is shown below in Figure 68.

Figure 68. SIP Pulse

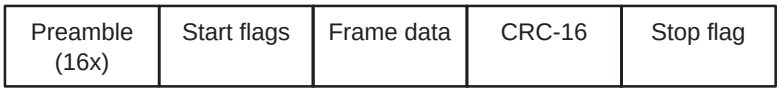


When the SIP_MODE bit of mode definition register 1 = 1 (MDR1[6]), the TX state machine always sends 1 SIP at the end of a transmission frame. But when MDR1[6] = 0, the transmission of the SIP depends on the SEND_SIP bit of the auxiliary control register (ACREG[3]). The local host can set ACREG[3] at least once every 500 ms. The advantage of this approach over the default is that the TX state machine does not need to send the SIP at the end of each frame, which may reduce the overhead required.

6.5.5 FIR Mode

In fast infrared mode (FIR), data transfer takes place between the LH and peripheral devices at 4M bits/s. A FIR transmit frame starts with a preamble, followed by a start flag, frame data, CRC−32, and ends with a stop flag.

Figure 69. FIR Transmit Frame Format



On transmit, the FIR transmit state machine attaches the preamble, start flag, CRC−32, and stop flag. It also encodes the transmit data into 4PPM format and generates the serial infrared interaction pulse (SIP).

On receive, the FIR receive state machine recovers the receive clock, removes the start flag, decodes the 4PPM incoming data, and determines the frame boundary with reception of the stop flag. It also checks for errors such as illegal symbol, CRC error, and frame-length error. At the end of a frame reception, the LH reads the line status register (LSR) to discover possible errors in the received frame.

The module transfers data both ways, but when the device is transmitting, hardware automatically disables the IR RX circuitry. Refer to Table 104, Auxiliary Control Register Bit 5, for a description of the logical operation for all three modes, SIR, MIR, and FIR.

6.6 Functional Description

6.6.1 Trigger Levels

The UART provides programmable trigger levels for both receiver and transmitter DMA and interrupt generation. After reset, both transmitter and receiver FIFOs are disabled, so in effect the trigger level is the default value of 1 byte. The programmable trigger levels are an enhanced feature available via the trigger level register (TLR).

6.6.2 Interrupts

The UART/IrDA module generates interrupts on the UART_nIRQ output pin. All interrupts are enabled/disabled by writing to the appropriate bit in the interrupt enable register (IER). The interrupt status of the device can be checked at any time by reading the interrupt identification register (IIR).

The UART and IrDA modes have different interrupts in the UART/IrDA module, and therefore different IER and IIR mappings according to the selected mode.

UART Mode Interrupts

The UART modes have seven possible interrupts. These interrupts are prioritized to six different levels.

When an interrupt is generated, the interrupt identification register (IIR) indicates that an interrupt is pending by bringing IIR[0] to 0, and provides the type of interrupt through IIR[5−1]. It also summarizes the interrupt control functions.

Table 109. UART Mode Interrupts

IIR[5−0]	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Method
0 0 0 0 0 1	None	None	None	None
0 0 0 1 1 0	1	Receiver line status	OE, FE, PE, or BI errors occur in characters in the RX FIFO	FE,PE,BI: Read RHR. OE: Read LSR
0 0 1 1 0 0	2	RX time-out	Stale data in RX FIFO	Read RHR
0 0 0 1 0 0	2	RHR interrupt	DRDY (data ready) (FIFO disable) RX FIFO above trigger level (FIFO enable)	Read RHR until interrupt condition disappears
0 0 0 0 1 0	3	THR interrupt	TFE (THR empty) (FIFO disable) TX FIFO below trigger level (FIFO enable)	Write to THR until interrupt condition disappears.
0 0 0 0 0 0	4	Modem status	MSR[1:0] / = 0	Read MSR
0 1 0 0 0 0	5	XOFF interrupt/special character interrupt	Receive XOFF characters(s)/special character	Receive XON character(s), if XOFF interrupt/read of IIR, if special character interrupt
1 0 0 0 0 0	6	CTS,RTS, DSR	RTS pin, CTS pin or DSR pin change state from active (low) to inactive (high).	Read IIR

It is important to note that for the receiver line status interrupt, RX_FIFO_STS bit (LSR[7]) generates the interrupt.

For the XOFF interrupt, if an XOFF flow character detection caused the interrupt, an XON flow character detection clears the interrupt. If special character detection caused the interrupt, a read of the IIR clears the interrupt.

IrDA Mode Interrupts

IrDA modes have eight possible interrupts. The UART_nIRQ output is activated when any of the eight interrupts is generated (there is no priority).

Table 110. IrDA Mode Interrupts

IIR Bit no.	Interrupt Type	Interrupt Source	Interrupt Reset Method
0	RHR interrupt	DRDY (data ready) (FIFO disable) RX FIFO above trigger level (FIFO enable)	Read RHR until interrupt condition disappears.
1	THR interrupt	TFE (THR empty) (FIFO disable) TX FIFO below trigger level (FIFO enable)	Write to THR until interrupt condition disappears.
2	Last byte in RX FIFO	Last byte of frame in RX FIFO	Read IIR
3	RX overrun	Write to RHR when RX FIFO full	Read RESUME register
4	Status FIFO interrupt	Status FIFO triggers level reached	Read STATUS FIFO
5	TX status	1. THR empty before EOF sent. Last bit of transmission of the IRDA frame has occurred but with an underrun error. OR 2. Transmission of the last bit of the IRDA frame is finished successfully.	1. Read RESUME register. OR 2. Read IIR
6	Receiver line status interrupt	CRC, ABORT or frame-length error is written into STATUS FIFO	Read STATUS FIFO [Read until empty—max 8 reads required]
7	Received EOF	Received end-of-frame.	Read IIR

For IIR[5] the interrupt source 1 is used with interrupt reset method 1. The interrupt source 2 is used with interrupt reset method 2.

Wake-Up Interrupt

Wake-up interrupt is a special interrupt, not designed the same as the previous ones. It is enabled when the RX_CTS_DSR_WAKE_UP_ENABLE bit of the supplementary control register (SCR[4]) is set to 1. The IIR register is not modified when it occurs. SSR[1] must be checked to detect a wake-up event. When wake-up interrupt occurs, the only way to clear it is to reset SCR[4] to 0.

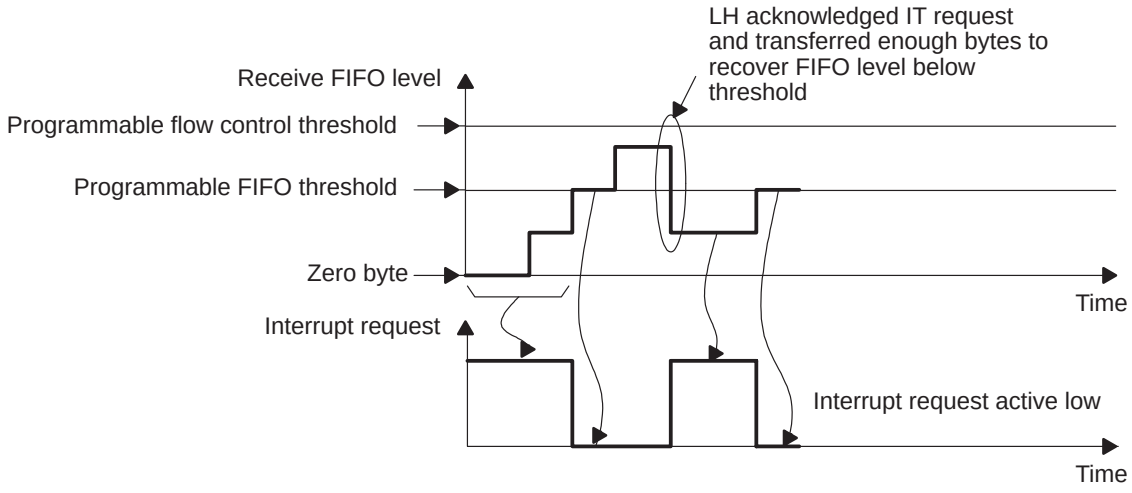
6.6.3 FIFO Interrupt Mode Operation

In FIFO interrupt mode (FIFO control register FCR[0] = 1, relevant interrupts enabled via IER), an interrupt signal (UART_nIRQ) informs the processor of the receiver and transmitter status. These interrupts are raised when receive/transmit FIFO thresholds (respectively, TLR[7:4] and TLR[3:0], or FCR[7:6] and FCR[5:4]) are reached. The interrupt signals instruct the local host to transfer data to the destination (from the UART module in receive mode and/or from any source to the UART FIFO in transmit mode).

Note that when the UART flow control is enabled along with the interrupt capabilities, the user must ensure that the UART flow control FIFO threshold (TCR[3:0]) is greater than, or equal to, the receive FIFO threshold.

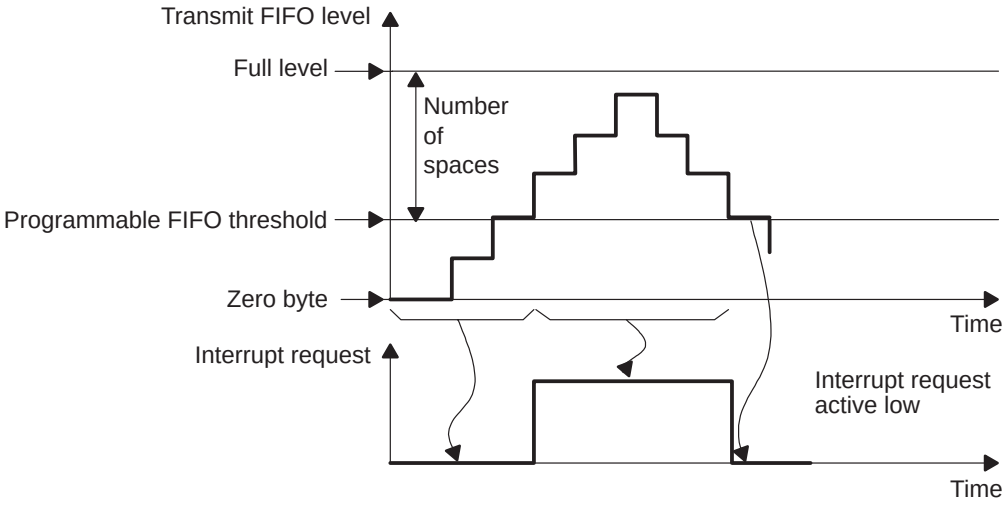
Figure 70 and Figure 71 respectively depict receive and transmit operations.

Figure 70. Receive FIFO IT Request Generation



In receive, no interrupt is generated until receive FIFO reaches its threshold. Once low, the interrupt can only be deasserted when the local host has handled enough bytes to make the FIFO level below threshold. The flow control threshold is set at a higher value than FIFO threshold.

Figure 71. Transmit FIFO IT Request Generation



In transmit mode, an interrupt request is automatically asserted when FIFO is empty. This request is deasserted when the FIFO crosses the threshold level. The interrupt line is deasserted until a sufficient number of elements has been transmitted to go below FIFO threshold.

6.6.4 FIFO Polled Mode Operation

In FIFO polled mode (FCR [0] = 0, relevant interrupts disabled via interrupt enable register (IER)), the status of the receiver and transmitter are checked by polling the line status register (LSR). This mode is an alternative to the FIFO interrupt mode of operation in which the status of the receiver and transmitter is automatically known by means of interrupts sent to the LH.

6.6.5 FIFO DMA Mode Operation

DMA Signaling

The four modes of DMA operation, DMA modes 0/1/2/3, are selected as follows:

When SCR[0] = 0: Setting FCR[3] to 0 enables DMA mode 0.

Setting FCR[3] to 1 enables DMA mode 1.

When SCR[0] = 1: SCR[2:1] determine DMA mode 0 to 3 according to supplementary control register (SCR) description.

For example:

- If no DMA operation is desired: Set SCR[0] to 1 and SCR[2:1] to 00 (FCR[3] is discarded).
- If DMA mode 1 is desired: Either set SCR[0] to 0 and FCR[3] to 1 or set SCR[0] to 1 and SCR[2:1] to 01 (FCR[3] is discarded).

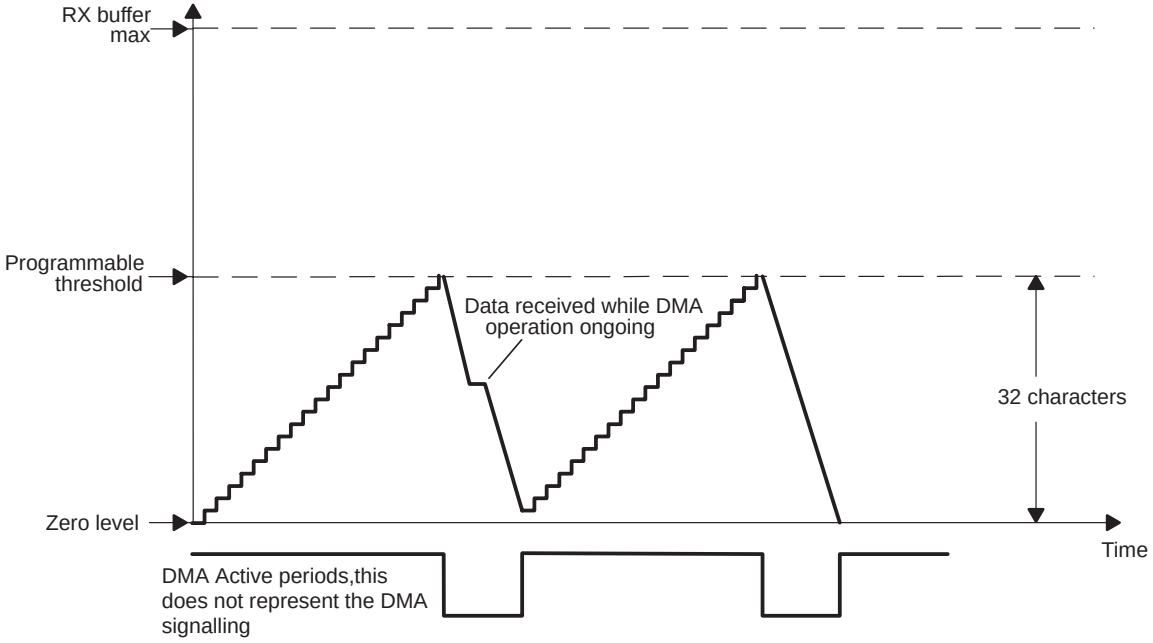
If the FIFOs are disabled (FCR[0] = 0), DMA occurs in single-character transfers.

When DMA mode 0 has been programmed, the signals associated with DMA operation are not active.

DMA Transfers (DMA Mode 1, 2, or 3)

Figure 72 through Figure 75 show the supported DMA operations.

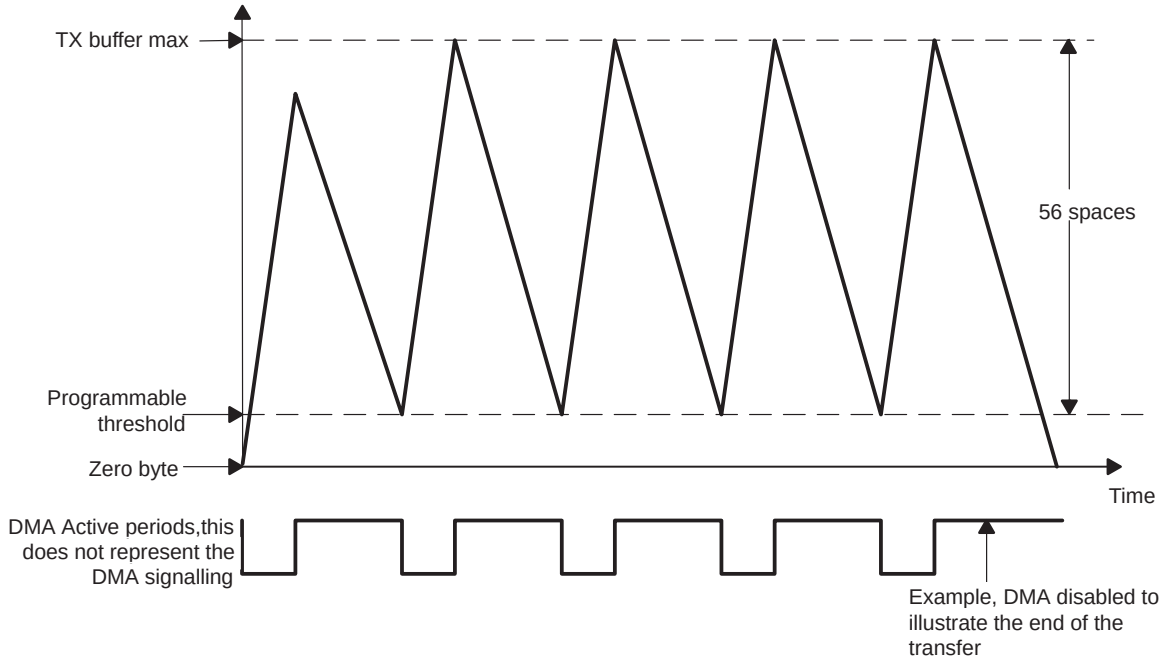
Figure 72. Receive FIFO DMA Request Generation (32 Characters)



In receive mode, a DMA request is generated as soon as the receive FIFO reaches its threshold level as defined in the trigger level register (TLR). (See Table 88.) This request is deasserted when the system DMA reads the number of bytes defined by the threshold level.

In transmit mode, a DMA request is automatically asserted when the FIFO is empty. This request is deasserted when the system DMA writes the number of bytes defined by the number of spaces in the trigger level register (TLR). If an insufficient number of characters is written, the DMA request remains active.

Figure 73. Transmit FIFO DMA Request Generation (56 Spaces)

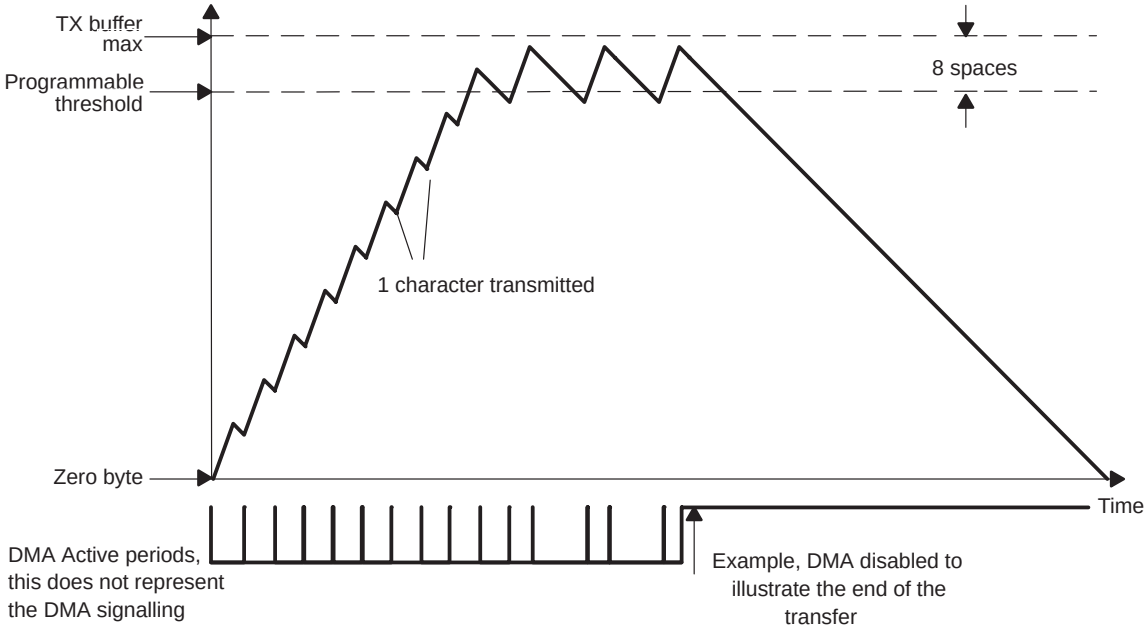


The DMA request is again asserted if the FIFO is able to receive the number of bytes defined by the TLR register. (See Table 88.)

The threshold can be programmed in a number of ways. See Figure 73 for an example of a DMA transfer that operates with a space setting of 56, which could arise from the use of the autosettings in the FCR[5:4] or the use of the TLR[3:0] concatenated with the FCR[5:4]. The setting of 56 spaces in the UART_IrDA should correlate with settings of the system DMA so that the buffer does not overflow (program the DMA request size of the local host controller to be equal to the number of spaces value in the UART).

Figure 74 shows another example with 8 spaces, to illustrate the buffer level crossing the space threshold. Again, the local host DMA controller settings must correspond to those of the UART_IrDA.

Figure 74. Transmit FIFO DMA Request Generation (8 Spaces)

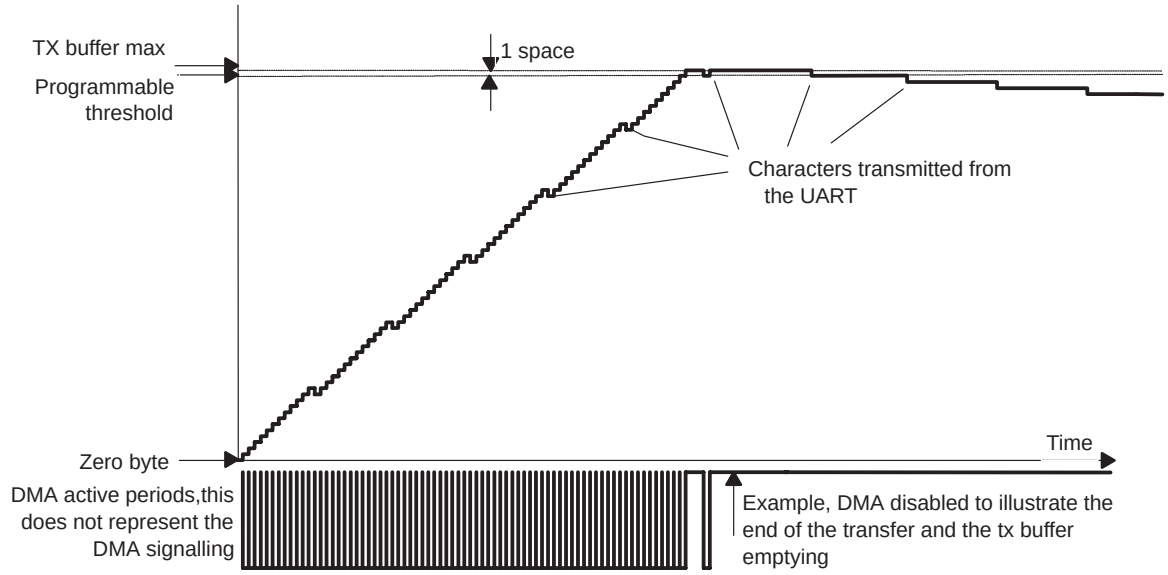


The final example in Figure 75 illustrates the setting of 1 space that uses the DMA for each transfer of 1 character to the transmit buffer. The buffer is filled at a faster rate than the BAUD rate transmits data to the TX pin. Eventually, the buffer is completely full and the DMA operation stops transferring data to the transmit buffer.

The buffer holds the maximum amount of data words on 2 occasions. Shortly after that the DMA is disabled to illustrate the slower transmission of the data words to the TX pin. Eventually, the buffer is emptied at the rate specified by the baud-rate settings of the DLL and DLH registers.

Again, the DMA settings must correspond to the local host DMA controller settings to ensure the correct operation of this logic.

Figure 75. Transmit FIFO DMA Request Generation (1 Space)



6.6.6 Sleep Mode

UART Modes

In UART modes, sleep mode is enabled by writing a 1 to IER[4] (when EFR[4] = 1).

Sleep mode is entered when

- The serial data input line, RX, is idle.
- The TX FIFO and TX shift register are empty.
- The RX FIFO is empty.
- There are no interrupts pending except THR interrupts.

Sleep mode is a good way to lower power consumption of the UART, but this state can be achieved only when the UART is set in modem mode. Therefore, even if the UART has no functional key role, it must be initialized in a functional mode to take advantage of sleep mode.

In sleep mode, the module clock and baud rate clock are stopped internally. Because most registers are clocked using these clocks, the power consumption is greatly reduced. The module wakes up when any change is detected on the RX line; if data is written to the TX FIFO, it occurs with any change in the state of the modem input pins. An interrupt is generated on a wake-up event by setting SCR[4] to 1.

Note:

Writing to the divisor latches, DLL and DLH, to set the baud clock, BCLK, must not be done during sleep mode. Therefore, it is advisable to disable sleep mode using IER[4] before writing to DLL or DLH.

IrDA Modes

In IrDA modes, sleep mode is enabled by writing a 1 to MDR1[3].

Sleep mode is entered when

- The serial data input line, RXIR, is idle.
- The TX FIFO and TX shift register are empty.
- The RX FIFO is empty.
- There are no interrupts pending except THR interrupts.

The module wakes up when any change is detected on the RXIR line, if data is written to the TX FIFO.

6.6.7 Idle Modes

Sleep and autoidle modes are embedded power-saving features. At the system level, power reduction techniques are applied by shutting down certain internal clock and power domains of the device.

The UART supports REQ_IDLE ACK handshaking protocol. This protocol is used at system level to shut down UART clocks in a clean and controlled manner, and to switch the UART from the interrupt generation mode to a wake-up generation mode for unmasked events (Refer to SYSC[2] and WER.)

For a software programming guide, refer to the *OCP Design Guidelines for Idle Mode Control*.

6.6.8 Break and Time-Out Conditions

Time-Out Counter

An RX idle condition is detected when the receiver line, RX, has been high for a time equivalent to 4X programmed word length+12 bits. The receiver line is sampled midway through each bit.

For sleep mode, the counter is reset when there is activity on the RX line.

For the timeout interrupt, the counter only counts when there is data in the RX FIFO. The count is reset when there is activity on the RX line or when the RHR is read.

Break Condition

When a break condition occurs, the TX line is pulled low. A break condition is activated by setting LCR[6]. Be aware that the break condition is not aligned on word stream, that is, a break condition can occur in the middle of a character. The only way to send a break condition on a full character is:

- ❏ Reset transmit FIFO (if enabled).
- ❏ Wait for transmit shift register to become empty (LSR[6] = 1).
- ❏ Take a guard time according to stop bit definition.
- ❏ Set LCR[6] to 1.

Break condition is asserted as long as LCR[6] is set to 1.

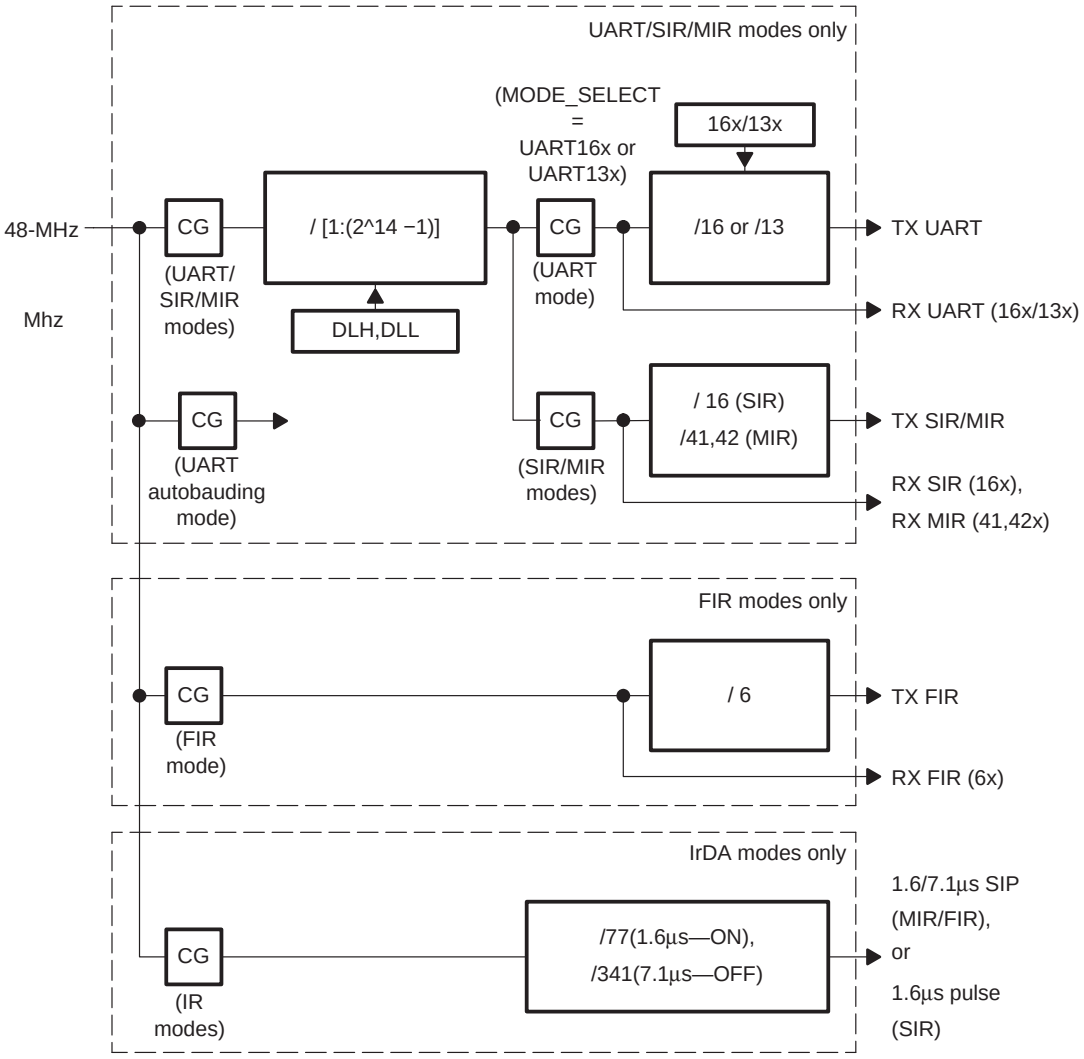
The above functionality (time-out counter and break condition) applies only to the UART modem operation and does not extend to the UART IrDA modes of operation.

6.6.9 Programmable Baud Rate Generator

The UART/IrDA module contains a programmable baud generator and a set of fixed dividers that take the 48-MHz clock input and divide it down to the expected baud rate.

The baud rate generator and associated controls are depicted in Figure 76.

Figure 76. Baud Rate Generator



Before Modifying Clock Parameters

It is recommended that MODE_SELECT = DISABLE (MDR1[2:0] = 111) be set before attempting to initialize or modify clock parameters controls (DLH, DLL). Nonobservance of this rule may result in an unpredictable behavior of the module.

Choosing the appropriate divisor value:

- ❑ UART 16x mode: Divisor value = Operating Freq/(16x baud rate)
- ❑ UART 13x mode: Divisor value = Operating Freq/(13x baud rate)
- ❑ SIR mode: Divisor value = Operating Freq/(16x baud rate)
- ❑ MIR mode: Divisor value = Operating Freq/(41x/42x baud rate)
- ❑ FIR mode: Divisor value = none.

Table 111. UART BAUD Rate Settings (48-MHz Clock)

Baud Rate (b/s)	Baud Multiple	DLH,DLL (Decimal)	DLH,DLL (Hex)	Actual Baud Rate (b/s)	Error (%)
0.3 K	16 x	10000	0x27, 0x10	0.3 K	0
0.6 K	16 x	5000	0x13, 0x88	0.6 K	0
1.2 K	16 x	2500	0x09, 0xC4	1.2 K	0
2.4 K	16 x	1250	0x04, 0xE2	2.4 K	0
4.8 K	16 x	625	0x02, 0x71	4.8 K	0
9.6 K	16 x	313	0x01, 0x39	9.6153 K	+0.16
14.4 K	16x	208	0x00, 0xD0	14.423 K	+0.16
19.2 K	16 x	156	0x00, 0x9C	19.231 K	+0.16
28.8 K	16 x	104	0x00, 0x68	28.846 K	+0.16
38.4 K	16 x	78	0x00, 0x4E	38.462 K	+0.16
57.6 K	16 x	52	0x00, 0x34	57.692 K	+0.16
115.2 K	16 x	26	0x00, 0x1A	115.38 K	+0.16
230.4 K	16 x	13	0x00, 0x0D	230.77 K	+0.16
460.8 K	13 x	8	0x00, 0x08	461.54 K	+0.16
921.6 K	13 x	4	0x00, 0x04	923.08 M	+0.16
1.8342 M	13 x	2	0x00, 0x02	1.1846 M	+0.16
3.6864 M	13 x	1	0x00, 0x01	3.6923 M	+0.16

Table 112. IrDA Baud Rate Settings (48-MHz Clock)

Baud Rate (b/s)	IR Mode	Baud Multi-ple	En-coding	DLH, DLL	Actual Baud Rate (* = Avg) (b/s)	Error (%)	Source Jitter (%)[1]	Pulse Duration
2.4 K	SIR	16x	3/16	1250	2.4 K	0	0	78.1 µs
9.6 K	SIR	16x	3/16	312	9.6153 K	+0.16	0	19.5 µs
19.2 K	SIR	16x	3/16	156	19.231 K	+0.16	0	9.75 µs
38.4 K	SIR	16x	3/16	78	38.462 K	+0.16	0	4.87 µs
57.6 K	SIR	16x	3/16	52	57.692 K	+0.16	0	3.25 µs
115.2 K	SIR	16x	3/16	26	115.38 K	+0.16	0	1.62 µs
0.576 M	MIR	41x/42 x	1/4	2	0.5756 M*	0	+1.63/ −0.80	416 ns
1.152 M	MIR	41x/42 x	1/4	1	1.1511 M*	0	+1.63/ −0.80	208 ns
4 M	FIR	6x	4 PPM	−	4 M	0	0	125 ns

6.6.10 Hardware Flow Control

Hardware flow control is composed of auto-CTS and auto-RTS. Auto-CTS and auto-RTS can be enabled/disabled independently by programming EFR[7:6].

With auto-CTS, $\overline{\text{CTS}}$ must be active before the module can transmit data.

Auto-RTS only activates the $\overline{\text{RTS}}$ output when there is enough room in the FIFO to receive data, and deactivates the $\overline{\text{RTS}}$ output when the RX FIFO is sufficiently full. The HALT and RESTORE trigger levels in the TCR determine the levels at which $\overline{\text{RTS}}$ is activated/deactivated.

If both auto-CTS and auto-RTS are enabled, data transmission does not occur unless the receiver FIFO has empty space. Thus, overrun errors are eliminated during hardware flow control. If they are not enabled, overrun errors occur when the transmit data rate exceeds the receive FIFO latency.

Auto-RTS

Auto-RTS data flow control originates in the receiver block (see Figure 61, Functional Block Diagram). The receiver FIFO trigger levels used in auto-RTS are stored in the TCR. $\overline{\text{RTS}}$ is active if the RX FIFO level is below the HALT trigger level in TCR[3:0]. When the receiver FIFO HALT trigger level is reached, $\overline{\text{RTS}}$ is deasserted. The sending device (for example, another UART) may send an additional byte after the trigger level is reached because it may not recognize the deassertion of $\overline{\text{RTS}}$ until it has begun sending the additional byte. $\overline{\text{RTS}}$ is automatically reasserted once the receiver FIFO reaches the RESUME trigger level programmed via TCR[7:4]. This reassertion requests the sending device to resume transmission.

Auto-CTS

The transmitter circuitry checks $\overline{\text{CTS}}$ before sending the next data byte. When $\overline{\text{CTS}}$ is active, the transmitter sends the next byte. To stop the transmitter from sending the following byte, $\overline{\text{CTS}}$ must be deasserted before the middle of the last stop bit that is currently being sent. The auto-CTS function reduces interrupts to the host system. When auto-CTS flow control is enabled, the $\overline{\text{CTS}}$ state changes need not trigger host interrupts because the device automatically controls its own transmitter. Without auto-CTS, the transmitter sends any data present in the transmit FIFO and a receiver overrun error can result.

6.6.11 Software Flow Control

The enhanced feature register (EFR) and the modem control register (MCR) enable software flow control. Different combinations of software flow control are enabled by setting different combinations of EFR[3:0].

Two other enhanced features relate to software flow control:

- XON any function (MCR[5]): The operation will resume after receiving any character, after recognizing the XOFF character.

  The XON-any character is written into the RX FIFO even if it is a software flow character.

- Special character (EFR[5]): Incoming data is compared to XOFF2. Detection of the special character sets the XOFF interrupt (IIR[4]) but does not halt transmission. A read of the IIR clears the XOFF interrupt. The special character is transferred to the RX FIFO.

Receive (RX)

When the software flow control operation is enabled, the UART compares incoming data with XOFF1/2 programmed characters (in certain cases, XOFF1 and XOFF2 must be received sequentially). When the correct XOFF characters are received, transmission is halted after completing transmission of the current character. XOFF detection also sets IIR[4] (if enabled via IER[5]) and causes UART_nIRQ to go low.

To resume transmission an XON1/2 character must be received (in certain cases XON1 and XON2 must be received sequentially). When the correct XON characters are received, IIR[4] is cleared and the XOFF interrupt disappears.

If a parity, framing, or break error occurs while receiving a software flow control character, the character is treated as normal data and is written to the RX FIFO.

When XON-any and special character detect are disabled and software flow control is enabled, no valid XON or XOFF characters are written to the RX FIFO. For example, in EFR[1:0] = 10, if the XON1 and XOFF1 characters are received they are not written to the RX FIFO.

In the case where pairs of software flow characters are programmed to be received sequentially (EFR[1:0] = 11), the software flow characters are not written to the RX FIFO if they are received sequentially. However, received XON1/XOFF1 characters must be written to the RX FIFO if the subsequent character is not XON2/XOFF2.

Transmit (TX)

XOFF1: Two characters are transmitted when the RX FIFO passes the programmed trigger level TCR[3:0].

XON1: Two characters are transmitted when the RX FIFO reaches the programmed trigger level via TCR[7:4].

If an XOFF character has been sent, software flow control is disabled and the module transmits XON characters automatically to enable normal transmission to proceed. The transmission of XOFF/XON (s) follows the same protocol as transmission of an ordinary byte from the FIFO. This means that even if the word length is set to be 5, 6, or 7 characters, then the 5, 6, or 7 least-significant bits of XOFF1,2/XON1,2 are transmitted. Note that the transmission of 5, 6, or 7 bits of a character is seldom done, but this functionality is included to maintain compatibility with earlier designs.

It is assumed that software flow control and hardware flow control will never be enabled simultaneously.

6.6.12 Autobauding Mode

In autobaud mode, UART extracts transfer characteristics (speed, length and parity) from an AT command. These characteristics are used to receive data following an *at* and to send data.

Here are valid AT commands:

AT DATA <CR>
AT DATA <CR>
A/
a

The line break during the acquisition of the sequence AT is not recognized and echo functionality is not implemented in hardware.

A/ and *a/* are not used to extract characteristics, but they have to be recognized because of their special meaning. They are used to instruct the software to repeat the last received AT command. Therefore, an *a/* always comes after an AT, and transfer characteristics are not expected to change between an AT and an *a/*.

As soon as a valid *at* (AT) is received, it and all subsequent data are saved into FIFO, including final <CR> (0x0D). Then, the autobaud state machine waits for the next valid AT command. If an *a/* (AI) is received, it is saved into FIFO and the state machine waits for next valid AT command.

Upon the first successful detection of the baud rate, the UART activates an interrupt to signify that the AT(upper- or lowercase) sequence has been detected. The UASR register reflects the correct settings for the baud rate that has been detected. The interrupt activity continues in this fashion each time a subsequent character is received. Therefore, it is recommended that the software enable the RHR interrupt when using the autobaud mode.

The following settings are detected in autobaud mode with a module clock of 48 MHz:

- Speed:115.2K bauds, 57.6K bauds, 38.4K bauds, 28.8K bauds, 19.2K bauds, 14.4K bauds, 9.6K bauds, 4.8K bauds, 2.4K bauds, or 1.2K bauds
- Length: 7 or 8 bits
- Parity: Odd, even, or space

Note:

The combination of 7-bit character + space parity is not supported.

The method used to identify the speed is:

1) Detect the transition 1 → 0 on the received data. This happens as soon as a stop-to-start bit transition occurs. The transition is valid after a majority vote on 3 sampling periods.

2) Sample the start bit duration with 115 200 *16-Hz clock frequency as long as there is no rising edge. A transition 0 → 1 is considered as valid after a majority vote on 3 sampling periods.

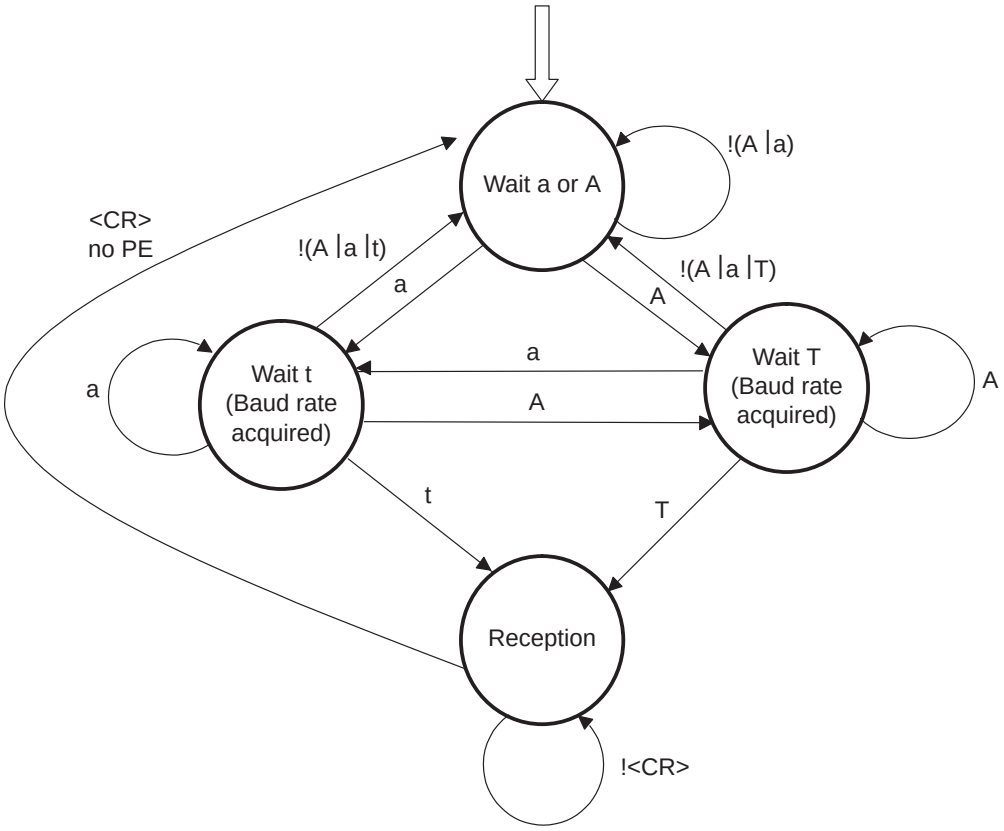
3) Compare the sampled value with a table. If the sampled value is outside a valid range, an error is reported (no speed identified) and the hardware goes back to the first state (1).

4) Else store the first data bit in the received register (for serial to parallel conversion) and go to frame format identification.

The next received bits are sampled according to the programmed baud rate. However, after 7 bits reception, the speed identification must be restarted because several *a* or *A* characters may have been received before a valid *t* or *T* character.

The autobauding mode is selected when MDR1[2:0] = 010. In the UART autobauding mode, DLL, DLH, and LCR[5:0] settings are not used. Instead, UASR is updated with the configuration detected by the autobauding logic.

Figure 77. Autobaud State Machine



6.6.13 Frame Closing

A transmission frame is properly terminated in one of two ways:

- Frame-length method: The frame-length method is selected when MDR1[7] = 0. The local host writes the frame-length value to the TXFLH and TXFLL registers. The device automatically attaches end flags to the frame once the number of bytes transmitted becomes equal to the frame-length value.
- Set-EOT bit method: The Set-EOT bit method is selected when MDR1[7] = 1. The local host writes 1 to ACREG[0] (EOT bit) just before it writes the last byte to the TX FIFO. When the local host writes the last byte to the TX FIFO, the device internally sets the tag bit for that particular character in the TX FIFO. As the TX state machine reads data from the TX FIFO, it uses this tag-bit information to attach end flags and properly terminate the frame.

6.6.14 Store and Controlled Transmission (SCT)

In SCT the local host first starts writing data into the TX FIFO. Then, after it writes a part of a frame for a bigger frame, or a whole frame (a small frame, that is, a supervisory frame), it writes a 1 to ACREG[2] (deferred TX start) to start transmission. SCT is enabled when MDR1[5] = 1. This method of transmission differs from the normal mode, where transmission of data starts immediately after data is written to the TX FIFO. SCT is useful for sending short frames without TX underrun.

6.6.15 Underrun During Transmission

Underrun in transmission occurs when the TX FIFO becomes empty before the end of the frame is transmitted. When underrun occurs, the device closes the frame with end flags but attaches an incorrect CRC value. The receiving device detects a CRC error and discards the frame; it can then ask for a retransmission. Underrun also causes an internal flag to be set, which disables further transmission. Before the next frame can be transmitted the system (LH) must:

- ❏ Reset the TX FIFO.
- ❏ Read the RESUME register. This clears the internal flag.

This functionality is disabled with ACREG[4], compensated by the extension of the stop bit in transmission in case the TX FIFO is empty.

6.6.16 Overrun During Receive

Overrun occurs during receive if the RX state machine tries to write data into the RX FIFO when it is already full. When overrun occurs, the device interrupts the local host with IIR[3] and discards the remaining portion of the frame. Overrun also causes an internal flag to be set, which disables further reception. Before the next frame can be received the system (LH) must:

- ❏ Reset the RX FIFO.
- ❏ Read the RESUME register. This clears the internal flag.

6.6.17 Status FIFO

In IrDA modes, a status FIFO is used to record the received frame status. When a complete frame is received, the length of the frame and the error bits associated with it are written into the status FIFO.

The frame length and error status can be determined by reading SFREGL/H and SFLSR. Reading the SFLSR causes the read pointer to be incremented. The status FIFO is eight entries deep and therefore can hold the status of eight frames.

The LH uses the frame-length information to locate the frame-boundary in the received frame data. The LH screens bad frames using the error-status information and later requests the sender to resend only the bad frames.

This status FIFO is used effectively in DMA, as the LH need not be interrupted each time a frame is received but only when the programmed status FIFO trigger level is reached.

6.7 UART Configuration Example

This section outlines the programming stages for operating one UART module with FIFO, interrupt, and no DMA capabilities. This is a three-step procedure that ensures a quick start of these modules (obviously, it does not cover every UART module feature). The first stage covers the software reset of the module (interrupts, status, and controls). The second stage deals with FIFO configuration and enable. The last stage with deals with the baud rate data and stop configuration. The procedure below is programming language agnostic.

6.7.1 UART Software Reset

Goal:

To clear IER and MCR registers, remove UART breaks (LCR[6] = 0) and put module in reset (MDR1[2:0] = 0x07).

Procedure:

To write into both the IER and MCR register EFR[4] must first be set to 1.

To be able to access the EFR register, 0xBF must be first be written to LCR register. Hence,

1) LCR = 0xBF; First write to the LCR register.
2) EFR[4] = 1; When LCR = 0xBF, enable the enhanced feature register.
3) LCR[7] = 0; Here, access to IER and MCR is allowed.
4) IER = 0x00; Disable interrupt.
5) MCR = 0x00; Force control signals inactive.
6) LCR[6] = 0; Here, remove UART breaks.
7) MDR1 = 0x07; Here, UART is in reset or disabled.

Alternatively, the SYSC[1] can be set to one to instigate a hardware reset from the generic synchronous reset module. The reset progress can be monitored via the SYSS[0]. Once complete, the above sequence should ensure that the UART is in the equivalent disabled mode with reference to MDR1[2:0].

6.7.2 UART FIFO Configuration

Goal:

To set trigger level for halt/restore (TCR register), set trigger level for transmit/receive (TLR register), and configure FIFO (FCR register).

Procedure:

To write into both the TLR and TCR registers, EFR[4] must be set to 1 and MCR[6] to 1. To write into FCR, EFR[4] must be set to 1. Note that EFR[4] = 1 was already done in the previous section. Therefore, a simple write to MCR[6] is necessary.

MCR[6] = 1; Sets TCR TLR and FCR to the desired value.

Here accesses to TCR, TLR, and FCR must be disabled to avoid any further undesired write to these registers. Hence,

- ❏ LCR = 0xBF; Provides access to EFR
- ❏ EFR[4] = 0
- ❏ LCR[7] = 0
- ❏ MCR[6] = 0

6.7.3 Baud Rate Data and Stop Configuration

Goal:

To configure UART data, stop (LCR register) baud rate (DLH and DLL registers), and enable UART operation. In case interrupt capability is added, configuration must be added right before UART enable.

Procedure:

- ❏ Set LCR to desired value.

  LCR[7] to 1; Gives access to DLH and DLL registers
- ❏ Set DLH and DLL;

  LCR[7] = 0; Removes access to DLH and DLL registers
- ❏ Set IER to desired value. Sets interrupts.

  MDR1[2:0] = 0; Enables UART without autobauding

The UART module is operational.

7 HDQ and 1-Wire Protocols

This module implements the hardware protocol of the master function of the HDQ™ and the 1-Wire™ protocol.

This module works off a command structure that is programmed into transmit command registers. The received data is in the receive data register. The firmware is responsible for performing correct sequencing in the command registers. The module only implements the hardware interface layer of the protocols.

The HDQ and the 1-Wire modes are selectable in software and must be chosen before any transmit and receive from the module is performed. The mode is assumed static during operation of the device. The 1-Wire and the HDQ protocols both use HDQ timing.

7.1 Functional Description

The module works with both HDQ and the 1-Wire protocols. The protocols use a single wire to communicate between the master and the slave. The protocols employ a return-to-1 mechanism, where after any command, the line is pulled up to a high. This requires an external pullup.

An open-drain configuration is used on the wire. The DX is connected to the GZ pin of an external 3-state output driver, and the input pin is connected to a zero level.

A control bit determines whether the HDQ or the 1-Wire protocol is to be used. Although this bit can be modified at any point, it is recommended that it be modified only as part of boot-up configuration. For the design, the bit is assumed static. By default, the configuration complies with the HDQ spec.

7.1.1 Receive and Transmit Operation

The receive and transmit operations are performed according to the timing specified in the HDQ protocol. This is done to keep the hardware interface section compatible between the two devices. In essence, the 1-Wire mode runs at slower speeds than the capabilities of the mode. The differences between the protocol at the hardware layer are described in the following subsections.

7.1.2 HDQ Mode (Default)

In HDQ mode, the firmware does not require the host to create an initialization pulse to the slave. However, the slave can be reset using an initialization pulse (also referred to as a break pulse). The pulse is created by setting the appropriate bit in the control and status register. The slave does not respond with a presence pulse, as it does in the 1-Wire protocol.

In a typical write to the slave, 2 bytes of data are sent to the slave. This is the command/address byte followed by the data that must be written. In a typical read, 1 command/address byte is sent to the slave, and the slave returns 1 byte of data.

The master implementation is a byte engine. The firmware is responsible for sending the ID, command/address, and data. The master engine provides only one data TX register.

HDQ is a return-to-1 protocol. This means that after a data byte (either command/address + write data for writes, or just command/address for reads) is sent to the slave, the host pulls the line high. This is accomplished in the device by setting the line to high (with an external pullup). The slave pulls the line low to initiate a transaction. This is the case when a read occurs, and the slave must send the read data back to the host.

If the host initiates a read and data is not received in a specified interval (the slave does not pull the line low within this time), a time-out status bit is set. This indicates that a read was not successfully completed. On successful completion, the time-out bit is cleared. The bit remains set or cleared until the next transaction by the host.

An interrupt condition indicates either a TX complete, RX complete, or time-out condition. The read of the interrupt status register clears all of the interrupt conditions. Only one interrupt signal is sent to the microcontroller, and only an overall mask bit exists for the enabling and disabling of the interrupt. The interrupt conditions cannot be individually masked.

The programmer must perform the following sequence for the reads and writes to the slave:

Write operation:

1) Write the command or data value to the TX write register.
2) Write 0 to the R/W bit of the control and status register to indicate a write.
3) Write 1 to the go bit of the control and status register to start the actual transmit. This step and the above step can be done at the same time.

- The hardware sends the byte from the TX data register.
- The time-out bit is always cleared in a write, because the hardware has no acknowledge mechanism from the slave.
- The completion of the operation sets the TX complete flag in the interrupt status register. If interrupts are masked, no interrupt is generated. The interrupt status register is always cleared at the beginning of any read or write operation.
- At the end of the write, the go bit is cleared.

4) Software must read the interrupt status register to clear the interrupt.

5) Repeat for each successive byte.

Read operation:

1) Write the command value to the TX write register.

2) Write 0 to R/W bit, 1 to the go bit, and wait for TX complete interrupt.

3) Write 1 to the R/W bit of the control and status register to indicate a read.

4) Write 1 to the go bit of the control and status register to start the actual read. This step and the above step can be done at the same time.

   - The hardware detects a low-going edge of the line (created by the slave) and receives 8 bits of data in the RX receive buffer register. The first bit that is received from the slave is the LSB and the last bit is the MSB of the byte. The master performs this step as soon as the slave sends the data, irrespective of the state of the go bit. However, an RX complete interrupt is generated only when the software writes the go bit.
   - If a time-out occurs, a time-out bit is set in the control and status register.
   - The completion of the operation sets the RX complete flag in the interrupt status register. If interrupts are masked, no interrupt is generated. The interrupt status register is always cleared at the beginning of either a read or a write operation.
   - At the end of the read, the go bit is cleared. It is also cleared if a time-out is detected.

5) Software must read the interrupt status register to determine whether RX was complete or whether a time-out occurred.

6) Software does a read of the RX buffer register to retrieve the read data from slave.

7) Repeat for each successive byte.

In HDQ16 mode, the address/command is only written once to the slave. However, after the first byte is received, if an RX complete interrupt is received, the software must initiate the read of the second byte by writing the go bit of the control and status register. The first byte that was received is shadowed and provided to the software while the hardware is fetching the second byte of data.

7.2 1-Wire Mode (SDQ)

This section highlights the primary differences between the HDQ and the 1-Wire protocols.

In the 1-Wire mode, the firmware must send an initialization pulse to the multiple slaves that can be connected on the interface. If any slave is present, the slave responds with a presence pulse.

The initialization pulse is sent by setting the appropriate bit in the control and status register. A presence detect is indicated in the appropriate bit of the register. If no presence is received, a time-out bit is set in the status register. The initialization bit is cleared at the end of the initialization pulse. Also, the presence detect and the time-out bits are cleared at the end of the initialization pulse, if a presence detect is received. The time-out bit has no other significance in this mode; that is, unlike in HDQ mode, it is always cleared during a read operation.

1-Wire mode is a bit-by-bit protocol for a read. Unlike HDQ, which sends eight bits of data on a read, the slave must be clocked by the host in 1-Wire protocol for each bit. At the end of the command/address byte, the line is pulled high and the host creates a low-going edge to initiate a bit read from the slave. The host then pulls the line high, and the slave either pulls the line low to indicate a 0 or does not drive the line to indicate a 1. The host repeats the operation for the next bit that need to be read.

The first bit that is received is the LSB and the last bit is the MSB in the RX data register.

An interrupt condition indicates either a TX complete, RX complete, or time-out condition. The read of the interrupt status register clears all of the interrupt conditions. Only one interrupt signal is sent to the microcontroller, and only an overall mask bit exists for the enabling and disabling of the interrupt. The interrupt conditions cannot be individually masked.

The programmer must perform the following sequence for the reads and writes to the slave:

Write operation:

1) Write the ID, command, or data value to the TX write register.
2) Write 0 to the R/W bit of the control and status register to indicate a write.
3) Write 1 to the go bit of the control and status register to start the actual transmit. This step and the above step can be done at the same time.
   - The hardware sends the one byte of the TX write data register.
   - The time-out bit is always cleared in a write.
   - The completion of the operation sets the TX complete flag in the interrupt status register. If interrupts are masked, no interrupt is generated. The interrupt status register is always cleared at the beginning of any read or write operation.
   - At the end of the write, the go bit is cleared.
4) If interrupt is enabled, software must read the interrupt status register to clear the interrupt.
5) Repeat for each successive byte.

Read operation:

1) Write the ID value to the TX write register.
2) Write 0 to R/W bit and 1 to the go bit and wait for TX complete interrupt.
3) Write the command value to the TX write register.
4) Write 0 to R/W bit and 1 to the GO bit and wait for TX complete interrupt.
5) Write 1 to the R/W bit of the control and status register to indicate a read.
6) Write 1 to the go bit of the control and status register to start the actual read. This step and the above step can be done at the same time.
   - The hardware creates a low-going edge of the line (created by the slave), and clocks 8 bits of data into the RX receive buffer register. The first bit that is received from the slave is the LSB and the last bit is the MSB of the byte.
   - The time-out bit is always cleared in a read.
   - The completion of the operation sets the RX complete flag in the interrupt status register. If interrupts are masked, no interrupt is generated. The interrupt status register is always cleared at the beginning of any read or write operation.

- At the end of the read, the go bit is cleared. It is also cleared if a time-out is detected.

7) If interrupt is enabled, software must read the interrupt status register to determine if RX was completed or whether there was a time-out.

8) Software does a read of the RX buffer register to retrieve the read data from the slave.

9) Repeat for each successive byte.

7.3 1-Wire Bit Mode Operation

A single-bit mode can be entered by writing to the appropriate bit in the control and status register. In this mode, only one bit of data is received each time from the slave. After the bit is received, an RX complete interrupt is generated. Bit 0 of the receive buffer is updated each time a bit is received.

The mode has no effect in HDQ mode, as HDQ does not support single-bit protocol.

7.3.1 Timing Diagrams

Figure 78 shows the timing diagram for the read, reset, and write. In the HDQ, the reset pulse contains only the initialization and not the presence pulse. The timing required for the various signals are specified in *Single-Wire Advanced Battery Monitor IC for Cellular and PDA Applications* (SLUS480).

The master works at the timing of the HDQ interface, which encompasses the HDQ and the 1-Wire timing. Therefore, in 1-Wire mode, the master runs slower than the full performance capability of the protocol.

Figure 78. Read Timing Diagram

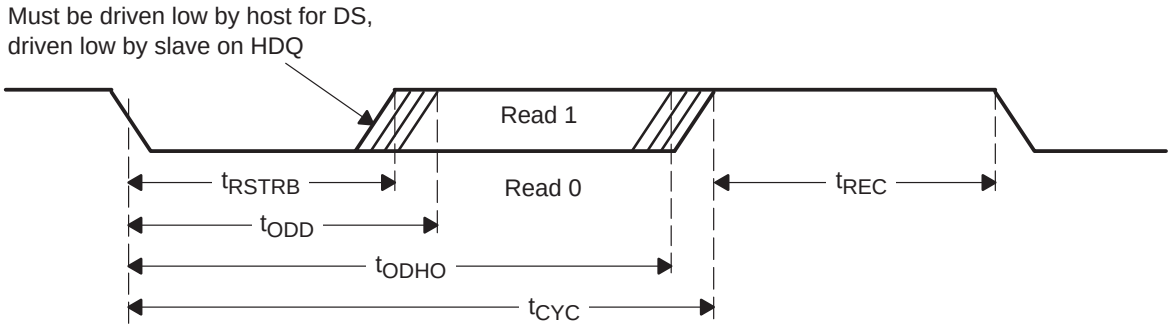


Figure 79. Reset Timing Diagram

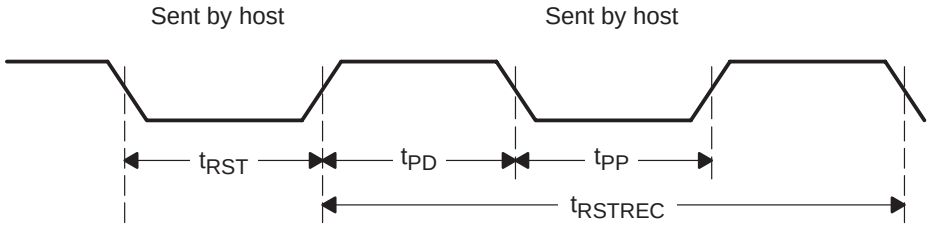
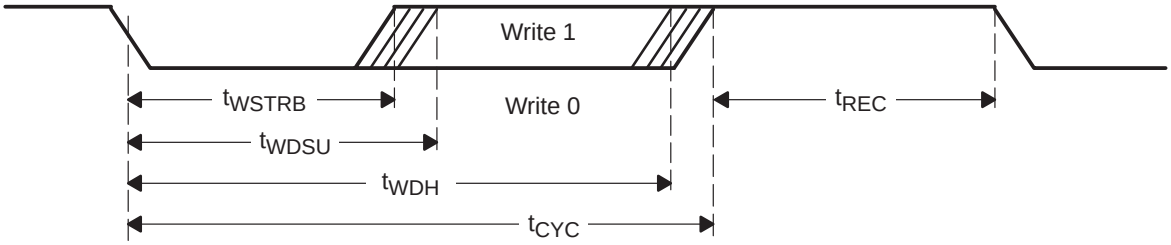
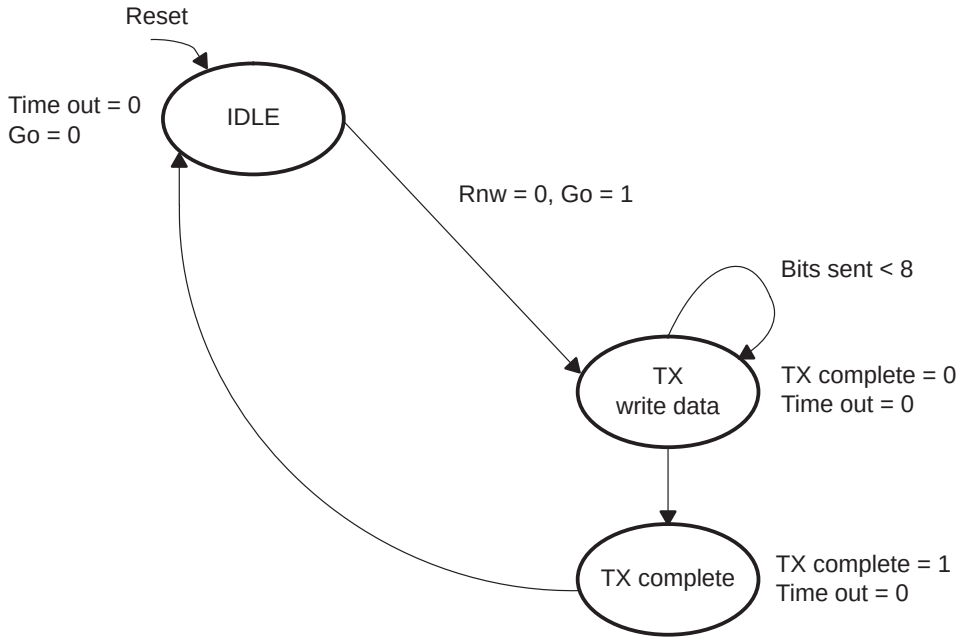


Figure 80. Write Timing Diagram



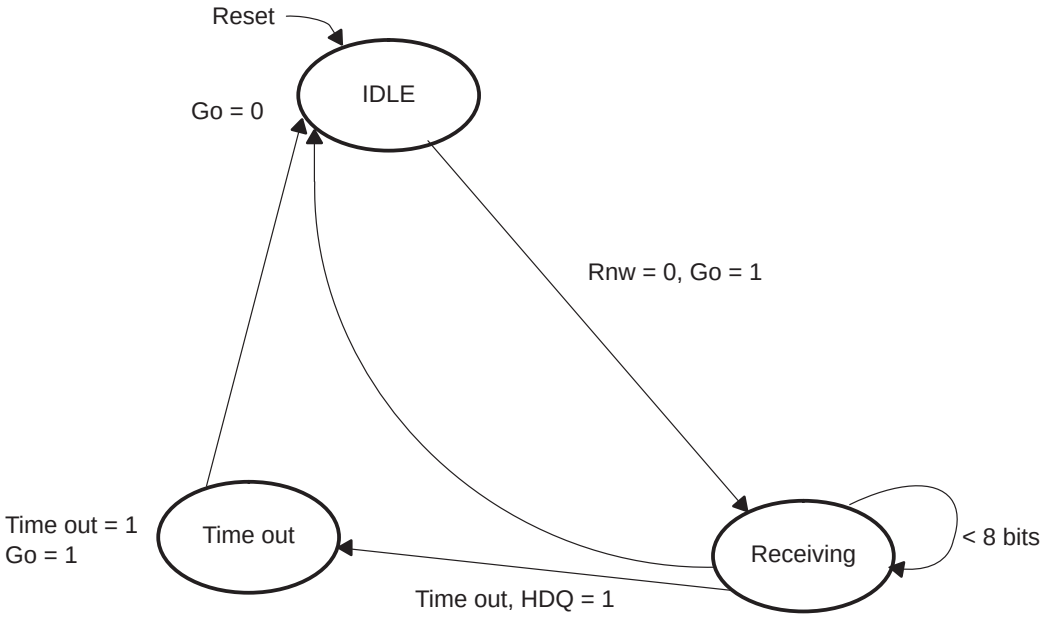
7.3.2 Write State Diagram

Figure 81. Write State Machine #1



7.3.3 Read State Diagram

Figure 82. Read State Machine #1



7.3.4 Status Flags

The status register contains status flags from the transmitter, the receiver, and the presence detect logic.

The presence-condition-detected status flag is contained in the status register. This is valid only in 1-Wire mode. It is cleared when the host sends an initialization pulse and then is set to 1 if a pulse is received. Otherwise, it stays cleared at 0.

7.3.5 Interrupts

The module provides the following interrupt status:

- Transmitter complete

  A write of one byte was completed. Successful or unsuccessful completion is not indicated, because there is no acknowledge from the slave in either HDQ or 1-Wire mode. Cleared at beginning of write command.

- Read complete

  Indicates successful completion of a byte read in both modes. Cleared at beginning of read command.

- Presence detect/time-out

  - In 1-Wire mode, it indicates that it is now valid to check the presence detect received bit. Cleared at beginning of initialization sequence.
  - In HDQ mode, it indicates that after a read command was issued by the host, the slave did not pull the line low within the specified time. In HDQ mode, the bit is cleared at beginning of the read command.

Only one interrupt is generated to the microcontroller, based on any of the above interrupt status conditions. A read to the interrupt status register clears all of the status bits that have been set.

The interrupt can be masked by setting the appropriate bit in the control and status register.

A read of the interrupt status register clears the interrupt. If there is a pending interrupt, the interrupt line stays low and no low-high-low transition is created. The interrupt therefore must be handled as a level interrupt (where a low-going edge is not needed) in an upstream interrupt handler (or processor).

7.4 Power-Down Mode

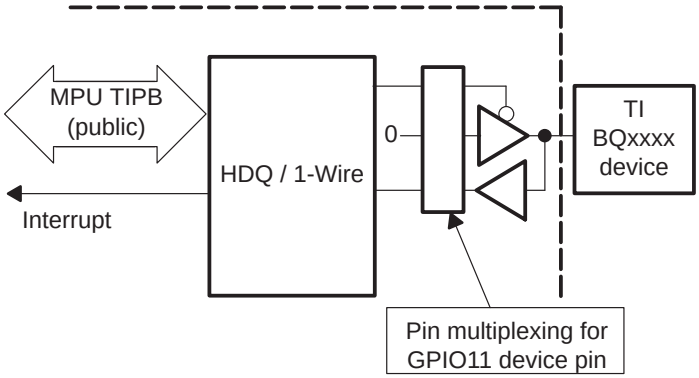
Writing to the appropriate bit in the control and status register shuts the clock to the state machine. The state machines are reset when the clock is disabled, and if any transaction is being performed, it is aborted into the reset state. The register values are not affected by disabling the clock. No register access must be performed to the module registers after the software puts the module in power-down mode by setting bit 5 of the control and status register to 0, other than a write to the power-down bit to take it out of power-down mode.

7.5 HDQ and 1-Wire Battery Monitoring Serial Interface

The HDQ and 1-Wire battery monitoring serial interface module implements the hardware protocol of the master function of the Benchmarq HDQ and the Dallas Semiconductor 1-Wire™ protocol. The module works off a command structure that is programmed into transmit command registers. The received data is in the received data register. The firmware is responsible for correct sequencing in the command registers. The module implements only the hardware interface layer of the protocols.

The HDQ and the 1-Wire mode are selectable in software, which must be done before any transmit and receive from the module is performed. The mode is assumed static during operation of the device.

Figure 83. HDQ and 1-Wire Overview



7.6 Software Interface

Mapping the registers to the TI peripheral bus (TIPB) address signals is shown in Table 113. The memory map identifies the 2K space associated with the peripheral.

The hardware provides no synchronization between the register clock domain and the state machine domain. This means that during a read, the hardware has the capability to modify the receive buffer. It is also possible that any access to the transmit write data register corrupts the data that is being sent if a TX is being performed.

However, these hazards can be avoided in software by observing the following limitations:

- A read is not performed from the interrupt status register or receive buffer register unless the processor has been interrupted by the peripheral.
- After the release of the go bit in the control and status register, no access to the TX write data buffer or the control and status registers is performed until the processor has been interrupted by the peripheral.
- Software is not allowed to poll the interrupt status register to determine whether an interrupt was generated.
- No register access can be done to the module registers after the software puts the module in power-down mode (by setting bit 5 of the control and status register to 0), except to reenable the clock.

Table 113. HDQ and 1-Wire Registers

Base Address = 0xFFFB C000			
Register	**Description**	**Type**	**Address**
HDQ1W_TX	TX write data	R/W	FFFB:C000
HDQ1W_RX	RX receive buffer	R	FFFB:C004
HDQ1W_CTRL	Control and status	R/W	FFFB:C008
HDQ1W_INTS	Interrupt status, read to clear	R	FFFB:C00C

Table 114. HDQ/1−Wire TX Write Data Register (HDQ1W_TX)

Base Address = 0xFFFB C000, Offset Address = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:24	Reserved	Reserved − read aliased to bit 7:0, writes ignored	R/W	0x00
23:16	Reserved	Reserved − read aliased to bit 7:0, writes ignored	R/W	0x00
15:8	Reserved	Reserved − read aliased to bit 7:0, writes ignored	R/W	0x00
7:0	WD	Write data (used in both HDQ and 1−Wire modes)	R/W	0x00

Table 115. HDQ/1−Wire RX Receive Buffer Register (HDQ1W_RX)

Base Address = 0xFFFB C000, Offset Address = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:24	Reserved	Reserved − read aliased to bit 7:0, writes ignored	R	U
23:16	Reserved	Reserved − read aliased to bit 7:0, writes ignored	R	U
15:8	Reserved	Reserved − read aliased to bit 7:0, writes ignored	R	U
7:0	RD	Next received character	R	U

Table 116. HDQ/1−Wire Control Register (HDQ1W_CTRL)

Base Address = 0xFFFB C000, Offset Address = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:24	Reserved	Reserved − read aliased to bit 7:0, writes ignored	R/W	0x00
23:16	Reserved	Reserved − read aliased to bit 7:0, writes ignored	R/W	0x00
15:8	Reserved	Reserved − read aliased to bit 7:0, writes ignored	R/W	0x00
7	SBM	Single-bit mode for 1-Wire	R/W	0
6	IM	Interrupt mask (1: Enable, 0: Disable interrupts)	R/W	0
5	PDM	Power-down mode (1: Enable clocks, 0: Disable clocks)	R/W	0
4	GB	Go bit Write 1 to send the appropriate commands. Bit returns to 0 after the command is complete.	R/W	0

Table 116. HDQ/1−Wire Control Register (HDQ1W_CTRL) (Continued)

Base Address = 0xFFFB C000, Offset Address = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
3	PD	Presence detect received, 1-Wire mode only. 0: Not detected. 1: Detected.	R	0
2	IP	Write 1 to send Initialization pulse. Bit returns to 0 after pulse is sent.	R/W	0
1	RWB	R/W bit (determines if next command is read or write) 0: Write. 1: Read.	R/W	0
0	MODE	Set mode 0: HDQ. 1: 1-Wire.	R/W	0

Table 117. HDQ/1−Wire Interrupt Status Register (HDQ1W_INTS)

Base Address = 0xFFFB C000, Offset Address = 0x0C				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:24	Reserved	Reserved − read aliased to bit 7:0, writes ignored	R	0x00
23:16	Reserved	Reserved − read aliased to bit 7:0, writes ignored	R	0x00
15:8	Reserved	Reserved − read aliased to bit 7:0, writes ignored	R	0x00
7:3	Reserved	Reserved − reads 0, writes ignored	R	0x0
2	TC	TX completed	R/C	0
1	RC	Read complete	R/C	0
0	DTO	Presence detect/time-out: In 1-Wire mode this is due to presence detect, and in HDQ mode this is due to time-out on read	R/C	0

Note: R/C − read clears bit

8 Frame Adjustment Counter

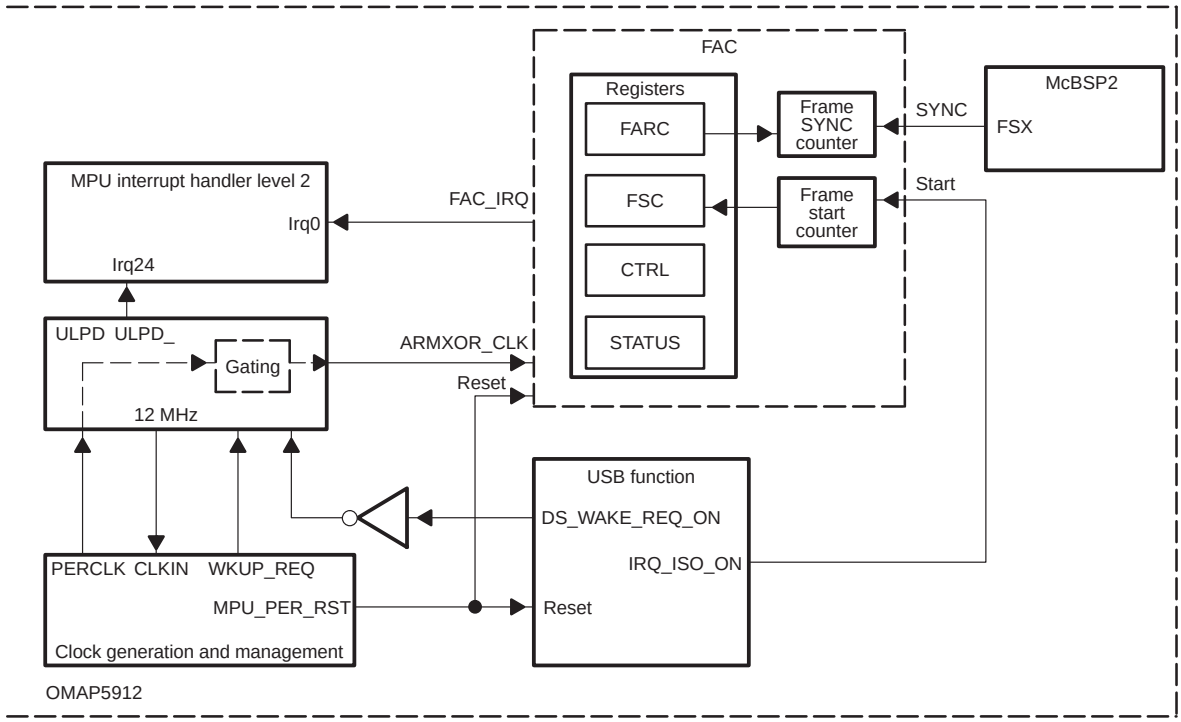
The frame adjustment counter counts the number of rising edges of one signal (start of frame interrupt of the USB function) during a programmable number of rising edges of a second signal (transit frame synchronization of McBSP2). System-level software uses this count value to adjust the duration of the two time domains with respect to each other to reduce overflow and underflow. If the data being transferred is audio data, this module can be part of a solution that reduces pops and clicks.

8.1 Features

The frame adjustment counter (FAC) is a module that consists of a frame-synchronization capture pin and a frame-start capture pin. System-level software uses the respective count values to adjust the duration of the two time domains with respect to each other to reduce the overflow and underflow.

A frame-adjustment reference count (FARC) register is programmed with the number of frame-synchronization pulses over which the frame-start pulses are to be counted. A frame-start count (FSC) register is updated with the number of frame-start rising edges that occur during the programmable FARC period. A control and configuration (CTRL) register allows for the module to be put into either continuous or halt mode. In continuous mode, the FSC register is periodically updated with a new value each time the FARC register value is met, and a new count is automatically initiated. In halt mode, the FSC register is updated with a new value when the FARC register value is met, counting halts, and an interrupt is generated. In halt mode, a new count is initiated upon software servicing the interrupt by reading the FSC register. The RUN bit in the control and configuration register can enable and disable the counters. If the RUN bit is set to 0, the counters are reset immediately, even though the count is not finished. The software can use this bit as a software reset. Additionally, a status register (STATUS) containing a FSC_FULL bit indicates to the system software whether FSC has been read subsequent to the last FSC update.

Figure 84. FAC Top-Level Diagram



The main FAC features are:

- Frame-synchronization capture pin (SYNC)
- Frame-start capture pin (START)
- Programmable frame-adjustment reference count register (FARC)
- Read-only frame-start count register (FSC)
- Interrupt generation logic
- Configuration and control register (CTRL)
- Status register (STATUS)

8.2 Synchronization and Counter Control

Because the frame-start and frame-synchronization signals are from different time domains, the FAC module synchronizes them to the system clock domain and uses the synchronized signals as the count enables. The actual counters for frame synchronization and frame start are clocked by the system clock.

8.2.1 Synchronization

The synchronization mechanism is based on the assumption that the system clock is running at least eight times faster than frame synchronization and frame start. Figure 85 and Figure 86 show the synchronization logic and the counter hookup.

Figure 85. FAC Module Counters and Clock Synchronization

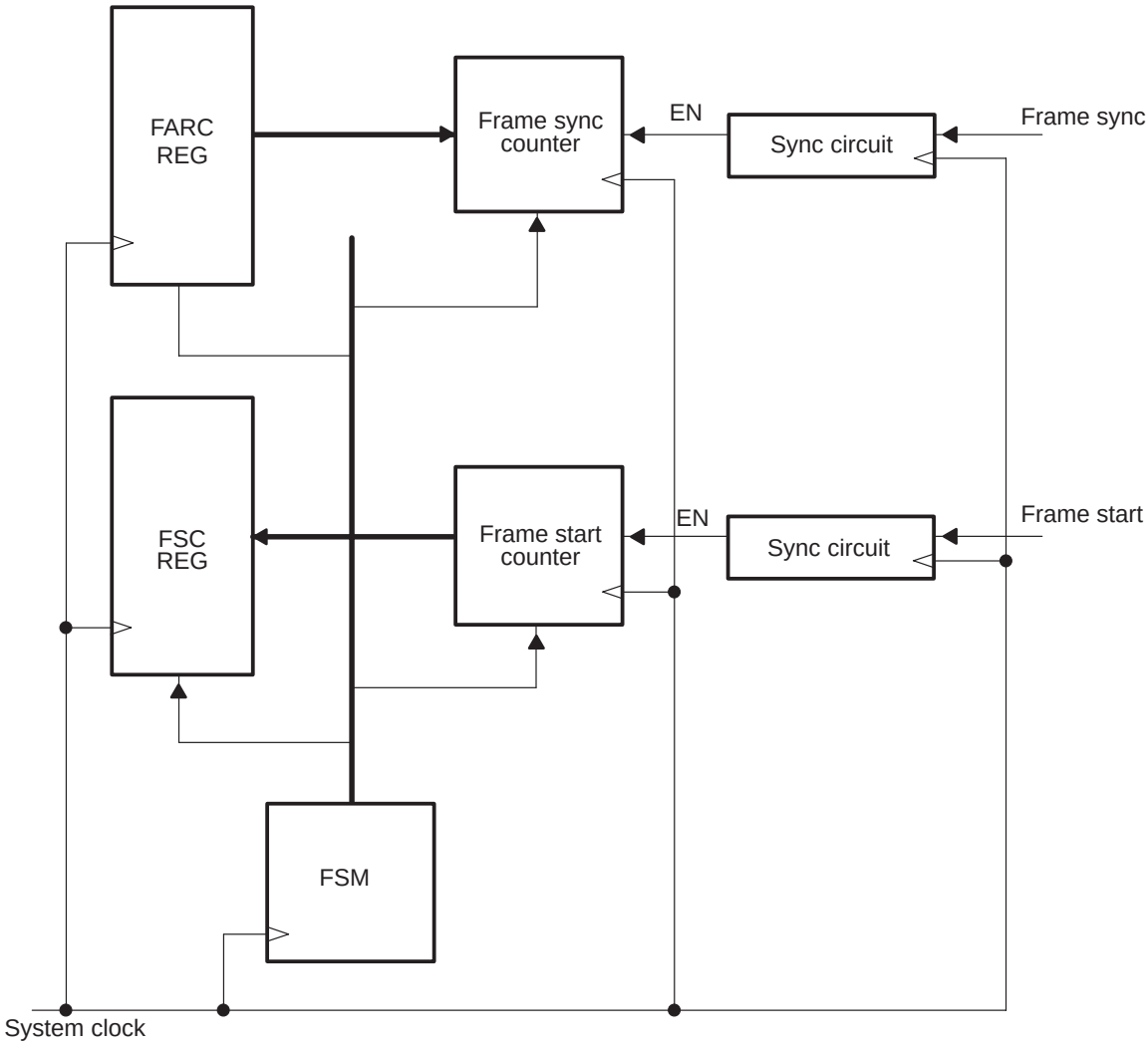


Figure 86. Synchronization Circuit for Frame Synchronization and Frame Start Signals

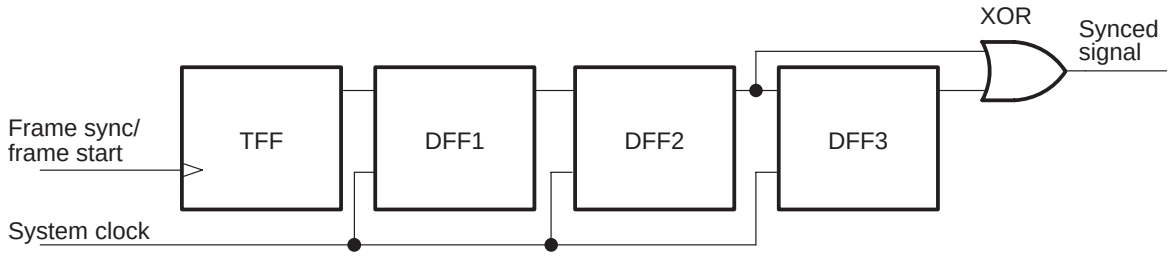
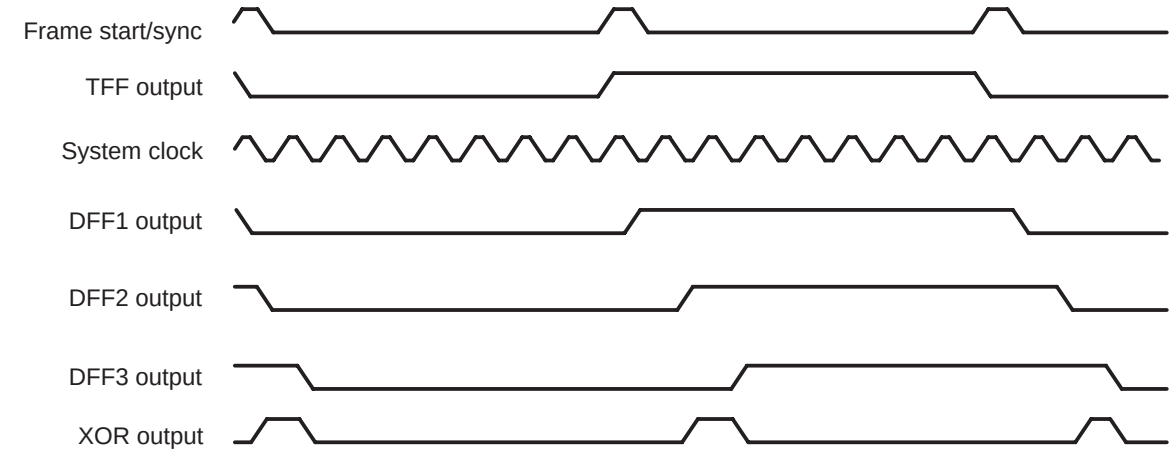


Figure 87 shows the actual waveforms at the output of each flip-flop and the XOR output.

Figure 87. Synchronization Circuit Waveforms



8.3 FAC Interrupt

The FAC generates one interrupt, FAC_IRQ (in halt mode when the FARC value is met), connected to the MPU level 2 interrupt handler, line 0 (level-sensitive).

8.4 FAC Clocks and Reset

The FAC works with a clock (PCLK) provided by the ULPD from a request generated by the USB function (DS_WAKE_REQ_ON).

The DS_WAKE_REQ_ON request does not wake up the system itself.

The ULPD module uses this request to generate an interrupt (ULPD_nIrq) to the MPU, which wakes up the system via its wake-up request (WKUP_REQ).

Once the system is awakened (12 MHz provided to the MPU), the MPU programmable peripheral clock (PERCLK) is used as the source clock for the FAC clock.

The MPU TIPB reset (MPU_PER_RST) resets the FAC.

8.5 Software Interface

Table 118 lists the FAC registers. Table 119 through Table 122 describe the register bits.

Table 118. FAC Registers

Register	Description	Type	Address
FARC	Frame adjustment reference count	R/W	FFFB:A800
FSC	Frame start count	R	FFFB:A804
CTRL	Control and configuration	R/W	FFFB:A808
STATUS	Status	R	FFFB:A80C
SYNC CNT	Frame synchronization counter	R	FFFB:A810
START CNT	Frame start counter	R	FFFB:A814

The FAC module is a word16 module with 32-bit aligned addresses.

The frame-adjustment counter register (FARC) is programmed with the number of frame synchronization counts over which the frame start pulses are counted. This is a 16-bit programmable fixed reference in the range of 0-65536. A value of 0 disables the count operation.

Table 119. Frame Adjustment Reference Count Register (FARC)

Base Address = 0xFFFB A800, Offset Address = 0x00				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	FARC	16-bit value in the range 0-65536: 0—disable counting	R/W	0x0000

The frame-start count (FSC) register is a 16-bit read-only register that contains the number of frame-start rising edges that occur during the programmable FARC period. The frame-start counting can be in one of two modes. When the CNT bit in the control and configuration (CTRL) register is set to 1, the counting is in continuous mode and this register is updated periodically (every time the frame adjustment reference count is met) with the new count value. If CNT is 0, the counting is in halt mode. The frame-start count register is updated when the frame adjustment reference count is met, and the counting halts until the software reads the FSC register.

A level-sensitive interrupt can be generated to indicate that the frame-start counting is finished, and the FSC register is loaded with a new count value. The interrupt is controlled by the INT_ENABLE bit in the control and configuration (CTRL) register. If this bit is set to 1, an interrupt is generated when the FSC register is updated. Because the interrupt is level-sensitive, the interrupt signal is kept low until the software reads the FSC register, or the RUN bit in the control register is set to 0. When the FSC is read or RUN bit in control register is set to 0, the interrupt signal is reset to 1. When the IN_ENABLE bit is set to 0, no interrupt is generated. The interrupt can be enabled or disabled for both continuous mode and halt mode.

Table 120. Frame Start Count Register (FSC)

Base Address = 0xFFFB A800, Offset Address = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	FS	16-bit value	R	0x0000

The control and configuration (CTRL) register is a R/W register used to configure the module. The RUN bit enables the frame-start counter. If this bit is set to 0, the frame-start counting is disabled immediately. The software can use this bit as a software reset for the FAC module by setting the RUN bit to 0. When the RUN bit is set to 0, the frame-start counter, the frame-synchronization counter, and the FSC register are reset to 0. The software reset also clears the status register FSC_FULL bit to 0. If an interrupt has been generated and the FAC module is waiting for an FSC register read, a software reset puts the counter control back to idle state. This means that after the software has been reset the counter starts counting again, regardless of whether the FSC register has been read or not.

Table 121. FAC Control and Configuration Register (CTRL)

Base Address = 0xFFFB A800, Offset Address = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:3	Reserved	Reserved	R	0x0000
2	INT_ENABLE	The INT_ENABLE bit is independent of the CNT bit. The interrupt can be enabled or disabled in either continuous mode or halt mode. 0: No interrupt is generated. 1: An interrupt is generated when FSC is updated.	R/W	0
1	RUN	Enables operation of the counter. 0: The frame start counter, the frame-synchronization counter, and the FSC are reset to 0. Any pending interrupt also is cleared when RUN is set to 0. 1: Enables the frame start counter.	R/W	0
0	CNT	0: Halt mode− updates FSC value when the frame-adjustment reference count is met and halts operation until FSC is read. 1: Continuous mode− periodically updates FSC value each time the frame-adjustment reference count is met.	R/W	0

Table 122. FAC Status Register (STATUS)

Base Address = 0xFFFB A800, Offset Address = 0x0C				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:1	Reserved	Reserved	R	0x0000
0	FSC_FULL	This bit is set to a 1 when FSC is updated. This bit is set back to a 0 when the FSC has been read or RUN bit in control is 0.	R/C	0

The status register (STATUS) contains an interrupt status bit.

Table 123. SYNC Counter Register (SYNC_CNT)

Base Address = 0xFFFB A800, Offset Address = 0x10				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	SC	Sync count value	R	0x0000

Table 124. Start Counter Register (START_CNT)

Base Address = 0xFFFB A800, Offset Address = 0x14				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	SC	Start count value	R	0x0000

Revision History

Table 125 lists the changes made since the previous version of this document.

Table 125. Document Revision History

Page	Additions/Modifications/Deletions
114	Modified introduction to section 4, *Multichannel Serial Interfaces.*

This page is intentionally left blank.

Index

M

N

P

R

S

T

W