TMS320x280x DSP Inter-Integrated Circuit (I^2C) Module Reference Guide

SPRU721
November 2004

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 manual describes the features and operation of the inter-integrated circuit (I^2C) module that is available on the TMS320x280x digital signal processors (DSPs). The I^2C module provides an interface between one of these 28x DSPs and devices compliant with Philips Semiconductors Inter-IC bus (I^2C-bus) specification version 2.1 and connected by way of an I^2C-bus. This manual assumes the reader has familiarity with the I^2C-bus specification.

Notational Conventions

This document uses the following conventions.

- The device number TMS320x280x is often abbreviated as 280x.
- In most cases, hexadecimal numbers are shown with the suffix h. For example, the following number is a hexadecimal 40 (decimal 64):

  40h

  Similarly, binary numbers often are shown with the suffix b. For example, the following number is the decimal number 4 shown in binary form:

  0100b

- If a signal or pin is active low, it has an overbar. For example, the $\overline{\text{RESET}}$ signal is active low.

Related Documentation From Texas Instruments

The following documents describe the 280x devices and related support tools. Copies of these documents are available on the Internet at www.ti.com.

TMS320C28x DSP CPU and Instruction Set Reference Guide (literature number SPRU430) describes the central processing unit (CPU) and the assembly language instructions of the TMS320C28x™ fixed-point digital signal processors (DSPs). It also describes emulation features available on these DSPs.

TMS320F2801, TMS320F2806, TMS320F2808 Digital Signal Processors (literature number SPRS230) data sheet contains the pinout, signal descriptions, as well as electrical and timing specifications for the F280x devices.

TMS320x280x Analog-to-Digital Converter (ADC) Reference Guide (literature number SPRU716) describes the ADC module. The module is a 12−bit pipelined ADC. The analog circuits of this converter, referred to as the core in this document, include the front-end analog multiplexers (MUXs), sample−and−hold (S/H) circuits, the conversion core, voltage regulators, and other analog supporting circuits. Digital circuits, referred to as the wrapper in this document, include programmable conversion sequencer, result registers, interface to analog circuits, interface to device peripheral bus, and interface to other on-chip modules.

TMS320x280x Boot ROM Reference Guide (literature number SPRU722) describes the purpose and features of the bootloader (factory-programmed boot-loading software). It also describes other contents of the device on-chip boot ROM and identifies where all of the information is located within that memory.

TMS320x281x, 280x Enhanced Controller Area Network (eCAN) Reference Guide (literature number SPRU074) describes the eCAN that uses established protocol to communicate serially with other controllers in electrically noisy environments. With 32 fully configurable mailboxes and time-stamping feature, the eCAN module provides a versatile and robust serial communication interface. The eCAN module implemented in the C28x DSP is compatible with the CAN 2.0B standard (active).

TMS320x281x, 280x Peripheral Reference Guide (literature number SPRU566) describes the peripheral reference guides of the 28x digital signal processors (DSPs).

TMS320x281x, 280x Serial Communication Interface (SCI) Reference Guide (literature number SPRU051) describes the SCI that is a two-wire asynchronous serial port, commonly known as a UART. The SCI modules support digital communications between the CPU and other asynchronous peripherals that use the standard non-return-to-zero (NRZ) format.

TMS320x281x, 280x Serial Peripheral Interface (SPI) Reference Guide (literature number SPRU059) describes the SPI − a high-speed synchronous serial input/output (I/O) port that allows a serial bit stream of programmed length (one to sixteen bits) to be shifted into and out of the device at a programmed bit−transfer rate. The SPI is used for communications between the DSP controller and external peripherals or another controller.

TMS320x280x System Control and Interrupts Reference Guide (literature number SPRU712) describes the various interrupts and system control features of the 280x digital signal processors (DSPs).

The TMS320C28x Instruction Set Simulator Technical Overview (literature number SPRU608) describes the simulator, available within the Code Composer Studio for TMS320C2000 IDE, that simulates the instruction set of the C28x core.

TMS320x280x Enhanced Quadrature Encoder Pulse (eQEP) Reference Guide (literature number SPRU790) describes the eQEP module, which is used for interfacing with a linear or rotary incremental encoder to get position, direction, and speed information from a rotating machine in high performance moton and position control systems. It includes the module description and registers.

TMS320x280x Enhanced Capture (eCAP) Module Reference Guide (literature number SPRU807) describes the enhanced Capture Module. It includes the module description and registers.

TMS320x280x Enhanced Pulse Width Modulator (ePWM) Module Reference Guide (literature number SPRU791). The PWM peripheral is an essential part of controlling many of the power related systems found in both commercial and industrial equipments. This guide describes the main areas that include digital motor control, switch mode power supply control, UPS (uninterruptible power supplies), and other forms of power conversion. The PWM peripheral can be considered as performing a DAC function, where the duty cycle is equivalent to a DAC analog value, it is sometimes referred to as a Power DAC.

TMS320C28x DSP/BIOS Application Programming Interface (API) Reference Guide (literature number SPRU625) describes development using DSP/BIOS.

3.3 V DSP for Digital Motor Control Application Report (literature number SPRA550). New generations of motor control digital signal processors (DSPs) lower their supply voltages from 5 V to 3.3 V to offer

higher performance at lower cost. Replacing traditional 5-V digital control circuitry by 3.3-V designs introduce no additional system cost and no significant complication in interfacing with TTL and CMOS compatible components, as well as with mixed voltage ICs such as power transistor gate drivers. Just like 5-V based designs, good engineering practice should be exercised to minimize noise and EMI effects by proper component layout and PCB design when 3.3-V DSP, ADC, and digital circuitry are used in a mixed signal environment, with high and low voltage analog and switching signals, such as a motor control system. In addition, software techniques such as Random PWM method can be used by special features of the Texas Instruments (TI) TMS320x24xx DSP controllers to significantly reduce noise effects caused by EMI radiation.

This application report reviews designs of 3.3-V DSP versus 5-V DSP for low HP motor control applications.The application report first describes a scenario of a 3.3-V-only motor controller indicating that for most applications, no significant issue of interfacing between 3.3 V and 5 V exists. Cost-effective 3.3-V − 5-V interfacing techniques are then discussed for the situations where such interfacing is needed. On-chip 3.3-V ADC versus 5-V ADC is also discussed. Sensitivity and noise effects in 3.3-V and 5-V ADC conversions are addressed. Guidelines for component layout and printed circuit board (PCB) design that can reduce system’s noise and EMI effects are summarized in the last section.

Trademarks

Contents

1 Introduction to the I2C Module 11
1.1 Features 12
1.2 Features Not Supported 13
1.3 Functional Overview 13
1.4 Clock Generation 16

2 I2C Module Operational Details 18
2.1 Input and Output Voltage Levels 18
2.2 Data Validity 18
2.3 Operating Modes 18
2.4 I2C Module START and STOP Conditions 20
2.5 Serial Data Formats 21
2.5.1 7-Bit Addressing Format 22
2.5.2 10-Bit Addressing Format 22
2.5.3 Free Data Format 23
2.5.4 Using a Repeated START Condition 23
2.6 NACK Bit Generation 24
2.7 Arbitration 25
2.8 Clock Synchronization 26

3 Interrupt Requests Generated by the I2C Module 27
3.1 Basic I2C Interrupt Requests 27
3.2 I2C FIFO Interrupts 29

4 Resetting/Disabling the I2C Module 30

5 I2C Module Registers 31
5.1 I2C Mode Register (I2CMDR) 32
5.2 I2C Interrupt Enable Register (I2CIER) 38
5.3 I2C Status Register (I2CSTR) 40
5.4 I2C Interrupt Source Register (I2CISRC) 46
5.5 I2C Prescaler Register (I2CPSC) 47
5.6 I2C Clock Divider Registers (I2CCLKL and I2CCLKH) 48
5.6.1 Formula for the Master Clock Period 49
5.7 I2C Slave Address Register (I2CSAR) 50
5.8 I2C Own Address Register (I2COAR) 51
5.9 I2C Data Count Register (I2CCNT) 52
5.10 I2C Data Receive Register (I2CDRR) 53
5.11 I2C Data Transmit Register (I2CDXR) 54
5.12 I2C Transmit FIFO Register (I2CFFTX) 55
5.13 I2C Receive FIFO Register (I2CFFRX) 56

Figures

1 Multiple I2C Modules Connected 12
2 I2C Module Conceptual Block Diagram 15
3 Clocking Diagram for the I2C Module 16
4 Bit Transfer on the I2C-Bus 18
5 I2C Module START and STOP Conditions 20
6 I2C Module Data Transfer (in This Case, 7-Bit Addressing and 8-Bit Data) 21
7 I2C Module 7-Bit Addressing Format (FDF = 0, XA = 0 in I2CMDR) 21
8 I2C Module 10-Bit Addressing Format (FDF = 0, XA = 1 in I2CMDR) (in This Case, Master-Transmitter Writing to Slave-Receiver) 21
9 I2C Module Free Data Format (FDF = 1 in I2CMDR) 22
10 Repeated START Condition (in This Case, 7-Bit Addressing Format) 23
11 Arbitration Procedure Between Two Master-Transmitters 25
12 Synchronization of Two I2C Clock Generators During Arbitration 26
13 Enable Paths of the I2C Interrupt Requests 29
14 I2C Mode Register (I2CMDR) 32
15 Pin Diagram Showing the Effects of the Digital Loopback Mode (DLB) Bit 38
16 I2C Interrupt Enable Register (I2CIER) 38
17 I2C Status Register (I2CSTR) 40
18 I2C Interrupt Source Register (I2CISRC) 46
19 I2C Prescaler Register (I2CPSC) 47
20 The Roles of the Clock Divide-Down Values (ICCL and ICCH) 48
21 I2C Clock Low-Time Divider Register (I2CCLKL) 48
22 I2C Clock High-Time Divider Register (I2CCLKH) 49
23 I2C Slave Address Register (I2CSAR) 50
24 I2C Own Address Register (I2COAR) 51
25 I2C Data Count Register (I2CCNT) 52
26 I2C Data Receive Register (I2CDRR) 53
27 I2C Data Transmit Register (I2CDXR) 54
28 I2C Transmit FIFO Register (I2CFFTX) 55
29 I2C Receive FIFO Register (I2CFFRX) 56

Tables

1 Operating Modes of the I2C Module 19
2 Ways to Generate a NACK Bit 24
3 Descriptions of the Basic I2C Interrupt Requests 28
4 I2C Module Registers 31
5 I2C Mode Register (I2CMDR) Field Descriptions 32
6 Master-Transmitter/Receiver Bus Activity Defined by the RM, STT, and STP Bits of I2CMDR 37
7 How the MST and FDF Bits of I2CMDR Affect the Role of the TRX Bit of I2CMDR 37
8 I2C Interrupt Enable Register (I2CIER) Field Descriptions 39
9 I2C Status Register (I2CSTR) Field Descriptions 40
10 I2C Interrupt Source Register (I2CISRC) Field Descriptions 46
11 I2C Prescaler Register (I2CPSC) Field Descriptions 47
12 I2C Clock Low-Time Divider Register (I2CCLKL) Field Description 48
13 I2C Clock High-Time Divider Register (I2CCLKH) Field Description 49
14 Dependency of Delay d on the Divide-Down Value IPSC 49
15 I2C Slave Address Register (I2CSAR) Field Descriptions 50
16 I2C Own Address Register (I2COAR) Field Descriptions 51
17 I2C Data Count Register (I2CCNT) Field Description 52
18 I2C Data Receive Register (I2CDRR) Field Description 53
19 I2C Data Transmit Register (I2CDXR) Field Descriptions 54
20 I2C Transmit FIFO Register (I2CFFTX) Field Descriptions 55
21 I2C Receive FIFO Register (I2CFFRX) Field Descriptions 57

This page is intentionally left blank.

I²C Module

This guide describes the features and operation of the inter-integrated circuit (I²C) module that is available on the TMS320x280x digital signal processor (DSP). The I²C module provides an interface between one of these DSPs and devices compliant with Philips Semiconductors Inter-IC bus (I²C-bus) specification version 2.1 and connected by way of an I²C-bus. External components attached to this 2-wire serial bus can transmit/receive 1- to 8-bit data to/from the 280x DSP through the I²C module. This guide assumes the reader is familiar with the I²C-bus specification.

Note:

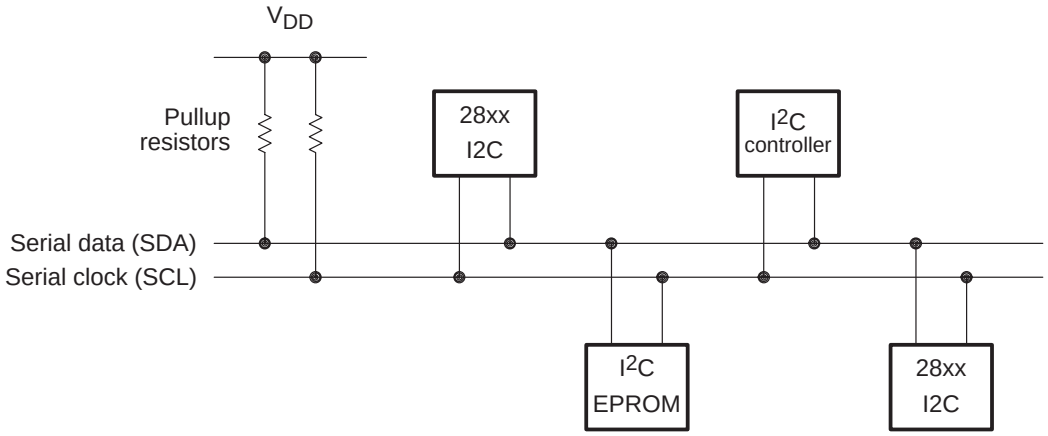
A unit of data transmitted or received by the I²C module can have fewer than 8 bits; however, for convenience, a unit of data is called a *data byte* throughout this document. (The number of bits in a data byte is selectable via the BC bits of the mode register, I2CMDR.)

This reference guide is applicable for the I²C found on the TMS320x280x family of processors. This includes all Flash-based, ROM-based, and RAM-based devices within the 280x family.

1 Introduction to the I²C Module

The I²C module supports any slave or master I²C-compatible device. Figure 1 shows an example of multiple I²C modules connected for a two-way transfer from one device to other devices.

Figure 1. Multiple I2C Modules Connected



1.1 Features

The I2C module has the following features:

- Compliance with the Philips Semiconductors I2C-bus specification (version 2.1):
  - Support for 8-bit format transfers
  - 7-bit and 10-bit addressing modes
  - General call
  - START byte mode
  - Support for multiple master-transmitters and slave-receivers
  - Support for multiple slave-transmitters and master-receivers
  - Combined master transmit/receive and receive/transmit mode
  - Data transfer rate of from 10 kbps up to 400 kbps (Philips Fast-mode rate)
- One 16-bit receive FIFO and one 16-bit transmit FIFO
- One interrupt that can be used by the CPU. This interrupt can be generated as a result of one of the following conditions: transmit-data ready, receive-data ready, register-access ready, no-acknowledgement received, arbitration lost, stop condition detected, addressed as slave.
- An additional interrupt that can be used by the CPU when in FIFO mode
- Module enable/disable capability
- Free data format mode

1.2 Features Not Supported

The I2C module does not support:

- High-speed mode (Hs-mode)
- CBUS compatibility mode

1.3 Functional Overview

Each device connected to an I2C-bus, including any 28xx DSP connected to the bus with an I2C module, is recognized by a unique address. Each device can operate as either a transmitter or a receiver, depending on the function of the device. A device connected to the I2C-bus can also be considered as the master or the slave when performing data transfers. A master device is the 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. The I2C module supports the multi-master mode, in which one or more devices capable of controlling an I2C-bus can be connected to the same I2C-bus.

For data communication, the I2C module has a serial data pin (SDA) and a serial clock pin (SCL), as shown in Figure 2. These two pins carry information between the 28xx device and other devices connected to the I2C-bus. The SDA and SCL pins both are bidirectional. They each must be connected to a positive supply voltage using a pull-up resistor. When the bus is free, both pins are high. The driver of these two pins has an open-drain configuration to perform the required wired-AND function.

There are two major transfer techniques:

- Standard Mode: Send exactly n data values, where n is a value you program in an I2C module register. See Table 5 for register information.
- Repeat Mode: Keep sending data values until you use software to initiate a STOP condition or a new START condition. See Table 5 for RM bit information.

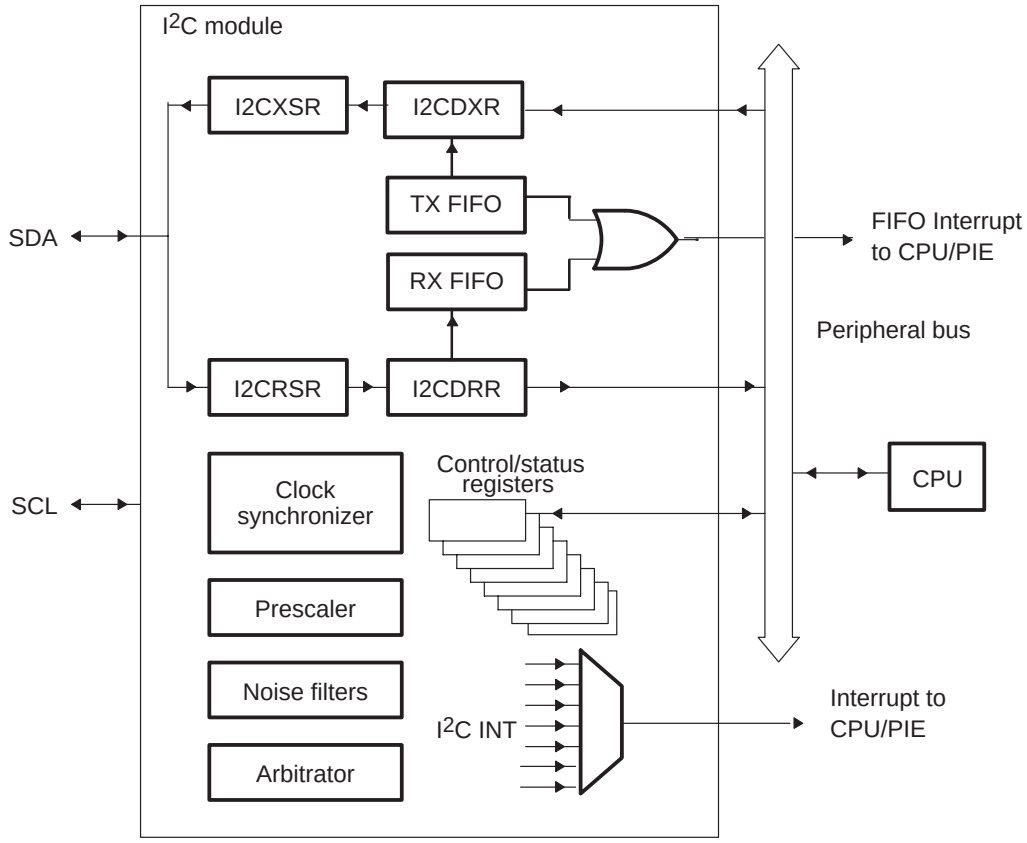
The I2C module consists of the following primary blocks:

- A serial interface: one data pin (SDA) and one clock pin (SCL)
- Data registers and FIFOs to temporarily hold receive data and transmit data traveling between the SDA pin and the CPU
- Control and status registers
- A peripheral bus interface to enable the CPU to access the I2C module registers and FIFOs.

- A clock synchronizer to synchronize the I2C input clock (from the DSP clock generator) and the clock on the SCL pin, and to synchronize data transfers with masters of different clock speeds
- A prescaler to divide down the input clock that is driven to the I2C module
- A noise filter on each of the two pins, SDA and SCL
- An arbitrator to handle arbitration between the I2C module (when it is a master) and another master
- Interrupt generation logic, so that an interrupt can be sent to the CPU
- FIFO interrupt generation logic, so that FIFO access can be synchronized to data reception and data transmission in the I2C module

Figure 2 shows the four registers used for transmission and reception in non-FIFO mode. The CPU writes data for transmission to I2CDXR and reads received data from I2CDRR. When the I2C module is configured as a transmitter, data written to I2CDXR is copied to I2CXSR and shifted out on the SDA pin one bit a time. When the I2C module is configured as a receiver, received data is shifted into I2CRSR and then copied to I2CDRR.

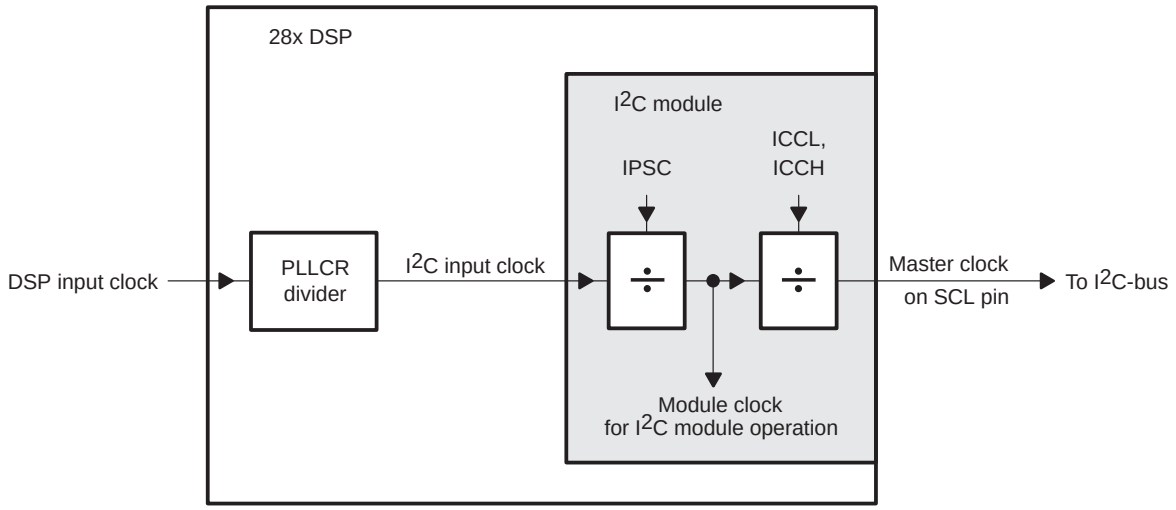
Figure 2. I2C Module Conceptual Block Diagram



1.4 Clock Generation

As shown in Figure 3, the DSP clock generator receives a signal from an external clock source and produces an I2C input clock with a programmed frequency. The I2C input clock is equivalent to the CPU clock and is then divided twice more inside the I2C module to produce the module clock and the master clock.

Figure 3. Clocking Diagram for the I2C Module



The module clock determines the frequency at which the I2C module operates. A programmable prescaler in the I2C module divides down the I2C input clock to produce the module clock. To specify the divide-down value, initialize the IPSC field of the prescaler register, I2CPSC (see page 47). The resulting frequency is:

$$\text{module clock frequency} = \frac{\text{I2C input clock frequency}}{(\text{IPSC} + 1)}$$

Note:

To meet all of the I2C protocol timing specifications, the module clock *must* be configured between 7 − 12 MHz.

The prescaler must be initialized only while the I2C module is in the reset state (IRS = 0 in I2CMDR). The prescaled frequency takes effect only when IRS is changed to 1. Changing the IPSC value while IRS = 1 has no effect.

The master clock appears on the SCL pin when the I2C module is configured to be a master on the I2C-bus. This clock controls the timing of communication between the I2C module and a slave. As shown in Figure 3, a second clock divider in the I2C module divides down the module clock to produce the master clock. The clock divider uses the ICCL value of I2CCLKL to divide down the low portion of the module clock signal and uses the ICCH value of I2CCLKH to divide down the high portion of the module clock signal. The resulting frequency is

$$\text{master clock frequency} = \frac{\text{module clock frequency}}{(\text{ICCL} + d) + (\text{ICCH} + d)}$$

$$\text{master clock frequency} = \frac{\text{I2C input clock frequency}}{(\text{IPSC} + 1)[(\text{ICCL} + d) + (\text{ICCH} + d)]}$$

where d depends on the value of IPSC:

IPSC	d
0	7
1	6
Greater than 1	5

The registers I2CCLKL and I2CCLKH are described in section 5.6 (page 48).

2 I2C Module Operational Details

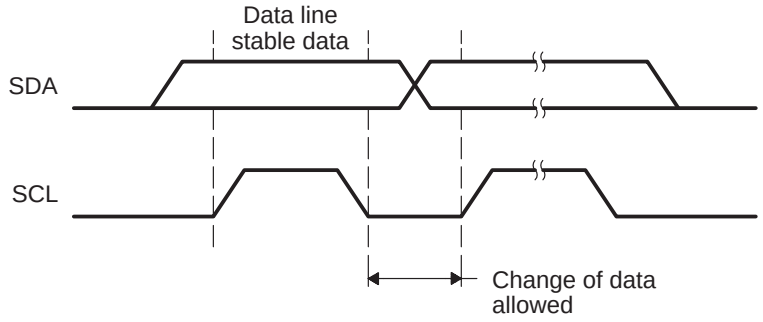
2.1 Input and Output Voltage Levels

One clock pulse is generated by the master device for each data bit transferred. Due to a variety of different technology devices that can be connected to the I2C-bus, the levels of logic 0 (low) and logic 1 (high) are not fixed and depend on the associated level of V_{DD}. For details, see the data manual for your particular 28x DSP.

2.2 Data Validity

The data on SDA must be stable during the high period of the clock (see Figure 4). The high or low state of the data line, SDA, can change only when the clock signal on SCL is low.

Figure 4. Bit Transfer on the I2C-Bus



2.3 Operating Modes

The I2C module has four basic operating modes to support data transfers as a master and as a slave. See Table 1 for the names and descriptions of the modes.

If the I2C module is a master, it begins as a master-transmitter and, typically, transmits an address for a particular slave. When giving data to the slave, the I2C module must remain a master-transmitter. To receive data from a slave, the I2C module must be changed to the master-receiver mode.

If the I2C module is a slave, it begins as a slave-receiver and, typically, sends acknowledgement when it recognizes its slave address from a master. If the master will be sending data to the I2C module, the module must remain a slave-receiver. If the master has requested data from the I2C module, the module must be changed to the slave-transmitter mode.

Table 1. Operating Modes of the I2C Module

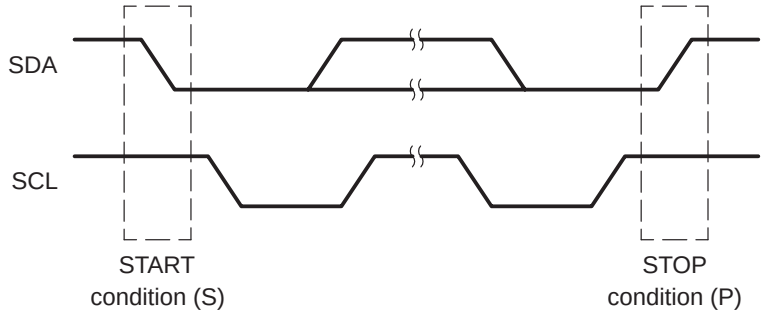
Operating Mode	Description
Slave-receiver mode	The I2C module is a slave and receives data from a master. All slaves begin in this mode. In this mode, serial data bits received on SDA are shifted in with the clock pulses that are generated by the master. As a slave, the I2C module does not generate the clock signal, but it can hold SCL low while the intervention of the DSP is required (RSFULL = 1 in I2CSTR) after a byte has been received.
Slave-transmitter mode	The I2C module is a slave and transmits data to a master. This mode can be entered only from the slave-receiver mode; the I2C module must first receive a command from the master. When you are using any of the 7-bit/10-bit addressing formats, the I2C module enters its slave-transmitter mode if the slave address byte is the same as its own address (in I2COAR) and the master has transmitted R/$\overline{W}$ = 1. As a slave-transmitter, the I2C module then shifts the serial data out on SDA with the clock pulses that are generated by the master. While a slave, the I2C module does not generate the clock signal, but it can hold SCL low while the intervention of the DSP is required (XSMT = 0 in I2CSTR) after a byte has been transmitted.
Master-receiver mode	The I2C module is a master and receives data from a slave. This mode can be entered only from the master-transmitter mode; the I2C module must first transmit a command to the slave. When you are using any of the 7-bit/10-bit addressing formats, the I2C module enters its master-receiver mode after transmitting the slave address byte and R/$\overline{W}$ = 1. Serial data bits on SDA are shifted into the I2C module with the clock pulses generated by the I2C module on SCL. The clock pulses are inhibited and SCL is held low when the intervention of the DSP is required (RSFULL = 1 in I2CSTR) after a byte has been received.
Master-transmitter mode	The I2C module is a master and transmits control information and data to a slave. All masters begin in this mode. In this mode, data assembled in any of the 7-bit/10-bit addressing formats is shifted out on SDA. The bit shifting is synchronized with the clock pulses generated by the I2C module on SCL. The clock pulses are inhibited and SCL is held low when the intervention of the DSP is required (XSMT = 0 in I2CSTR) after a byte has been transmitted.

2.4 I2C Module START and STOP Conditions

START and STOP conditions can be generated by the I2C module when the module is configured to be a master on the I2C-bus. As shown in Figure 5:

- The START condition is defined as a high-to-low transition on the SDA line while SCL is high. A master drives this condition to indicate the start of a data transfer.
- The STOP condition is defined as a low-to-high transition on the SDA line while SCL is high. A master drives this condition to indicate the end of a data transfer.

Figure 5. I2C Module START and STOP Conditions



After a START condition and before a subsequent STOP condition, the I2C-bus is considered busy, and the bus busy (BB) bit of I2CSTR is 1. Between a STOP condition and the next START condition, the bus is considered free, and BB is 0.

For the I2C module to start a data transfer with a START condition, the master mode bit (MST) and the START condition bit (STT) in I2CMDR must both be 1. For the I2C module to end a data transfer with a STOP condition, the STOP condition bit (STP) must be set to 1. When the BB bit is set to 1 and the STT bit is set to 1, a repeated START condition is generated. For a description of I2CMDR and its bits (including MST, STT, and STP), see section 5.1 on page 32.

2.5 Serial Data Formats

Figure 6 shows an example of a data transfer on the I2C-bus. The I2C module supports 1- to 8-bit data values. In Figure 6, 8-bit data is transferred. Each bit put on the SDA line equates to 1 pulse on the SCL line, and the values are always transferred with the most significant bit (MSB) first. The number of data values that can be transmitted or received is unrestricted. The serial data format used in Figure 6 is the 7-bit addressing format. The I2C module supports the formats shown in Figure 7 through Figure 9 and described in the paragraphs that follow the figures.

Figure 6. I2C Module Data Transfer (in This Case, 7-Bit Addressing and 8-Bit Data)

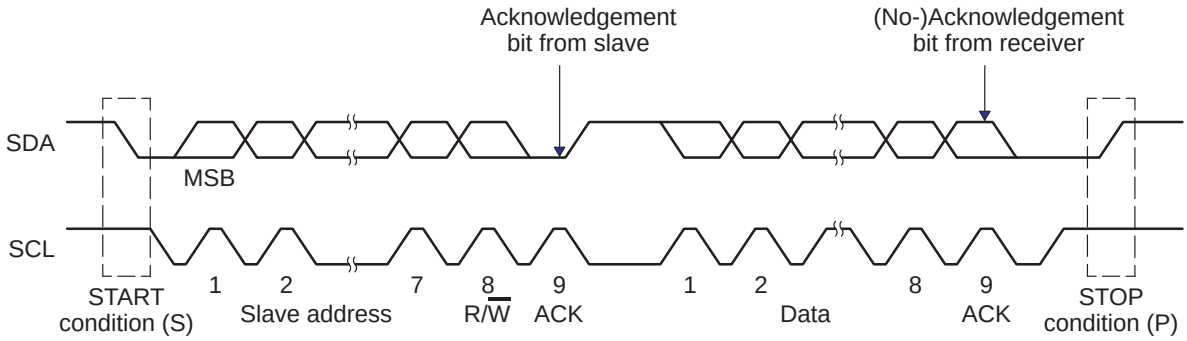
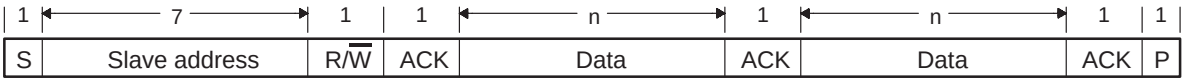
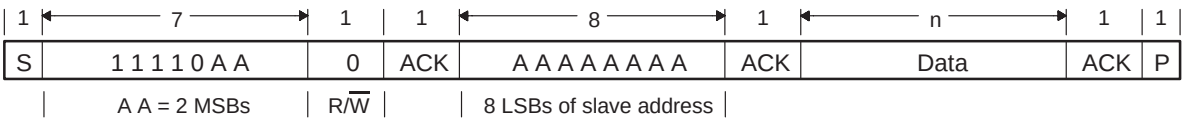


Figure 7. I2C Module 7-Bit Addressing Format (FDF = 0, XA = 0 in I2CMDR)



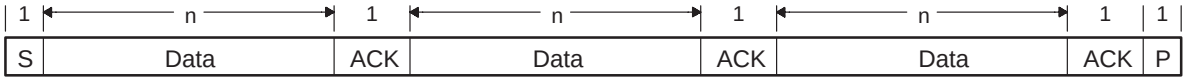
Note: n = The number of data bits (from 1 to 8) specified by the bit count (BC) field of I2CMDR.

Figure 8. I2C Module 10-Bit Addressing Format (FDF = 0, XA = 1 in I2CMDR) (in This Case, Master-Transmitter Writing to Slave-Receiver)



Note: n = The number of data bits (from 1 to 8) specified by the bit count (BC) field of I2CMDR.

Figure 9. I2C Module Free Data Format (FDF = 1 in I2CMDR)



Note: n = The number of data bits (from 1 to 8) specified by the bit count (BC) field of I2CMDR.

2.5.1 7-Bit Addressing Format

In the 7-bit addressing format (see Figure 7 on page 21), the first byte after a START condition (S) consists of a 7-bit slave address followed by a R/$\overline{W}$ bit. R/$\overline{W}$ determines the direction of the data:

- R/$\overline{W}$ = 0: The master writes (transmits) data to the addressed slave.
- R/$\overline{W}$ = 1: The master reads (receives) data from the slave.

An extra clock cycle dedicated for acknowledgement (ACK) is inserted after each byte. If the ACK bit is inserted by the slave after the first byte from the master, it is followed by *n* bits of data from the transmitter (master or slave, depending on the R/$\overline{W}$ bit). *n* is a number from 1 to 8 determined by the bit count (BC) field of I2CMDR. After the data bits have been transferred, the receiver inserts an ACK bit.

To select the 7-bit addressing format, write 0 to the expanded address enable (XA) bit of I2CMDR, and make sure the free data format mode is off (FDF = 0 in I2CMDR).

2.5.2 10-Bit Addressing Format

The 10-bit addressing format (see Figure 8 on page 21) is like the 7-bit addressing format, but the master sends the slave address in two separate byte transfers. The first byte consists of 11110b, the two MSBs of the 10-bit slave address, and R/$\overline{W}$ = 0 (write). The second byte is the remaining 8 bits of the 10-bit slave address. The slave must send acknowledgement after each of the two byte transfers. Once the master has written the second byte to the slave, the master can either write data or use a repeated START condition to change the data direction. (For more details about using 10-bit addressing, see the Philips Semiconductors I2C-bus specification.)

To select the 10-bit addressing format, write 1 to the XA bit of I2CMDR and make sure the free data format mode is off (FDF = 0 in I2CMDR).

2.5.3 Free Data Format

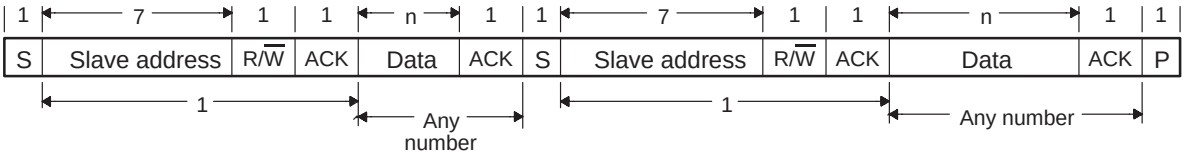
In this format (see Figure 9 on page 22), the first byte after a START condition (S) is a data byte. An ACK bit is inserted after each data byte, which can be from 1 to 8 bits, depending on the BC field of I2CMDR. No address or data-direction bit is sent. Therefore, the transmitter and the receiver must both support the free data format, and the direction of the data must be constant throughout the transfer.

To select the free data format, write 1 to the free data format (FDF) bit of I2CMDR. The free data format is not supported in the digital loopback mode (DLB = 1 in I2CMDR).

2.5.4 Using a Repeated START Condition

At the end of each data byte, the master can drive another START condition. Using this capability, a master can transmit/receive any number of data bytes before driving a STOP condition. The length of a data byte can be from 1 to 8 bits and is selected with the BC field of I2CMDR. The repeated START condition can be used with the 7-bit addressing, 10-bit addressing, and free data formats. Figure 10 shows a repeated START condition in the 7-bit addressing format.

Figure 10. Repeated START Condition (in This Case, 7-Bit Addressing Format)



Note: n = The number of data bits (from 1 to 8) specified by the bit count (BC) field of I2CMDR.

2.6 NACK Bit Generation

When the I2C module is a receiver (master or slave), it can acknowledge or ignore bits sent by the transmitter. To ignore any new bits, the I2C module must send a no-acknowledge (NACK) bit during the acknowledge cycle on the bus. Table 2 summarizes the various ways you can tell the I2C module to send a NACK bit.

Table 2. Ways to Generate a NACK Bit

I2C Module Condition	NACK Bit Generation Options
Slave-receiver mode	❑ Allow an overrun condition (RSFULL = 1 in I2CSTR). ❑ Reset the module (IRS = 0 in I2CMDR). ❑ Set the NACKMOD bit of I2CMDR before the rising edge of the last data bit you intend to receive.
Master-receiver mode AND Repeat mode (RM = 1 in I2CMDR)	❑ Generate a STOP condition (STP = 1 in I2CMDR). ❑ Reset the module (IRS = 0 in I2CMDR). ❑ Set the NACKMOD bit of I2CMDR before the rising edge of the last data bit you intend to receive.
Master-receiver mode AND Nonrepeat mode (RM = 0 in I2CMDR)	❑ If STP = 1 in I2CMDR, allow the internal data counter to count down to 0 and thus force a STOP condition. ❑ If STP = 0, make STP = 1 to generate a STOP condition. ❑ Reset the module (IRS = 0 in I2CMDR). ❑ Set the NACKMOD bit of I2CMDR before the rising edge of the last data bit you intend to receive.

2.7 Arbitration

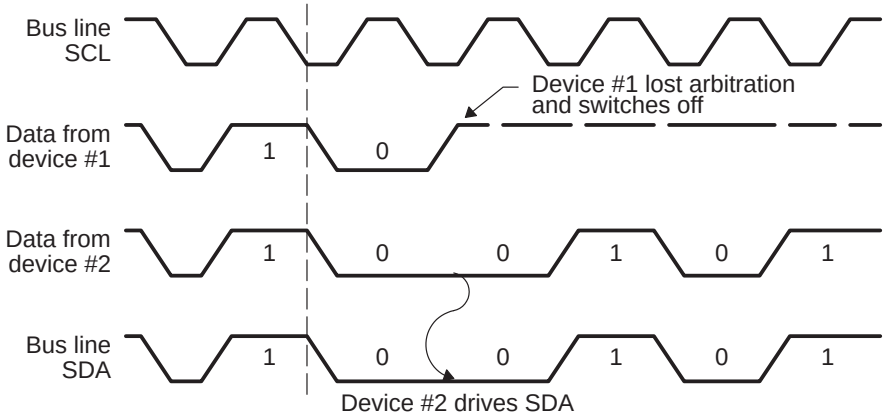
If two or more master-transmitters simultaneously start a transmission on the same bus, an arbitration procedure is invoked. The arbitration procedure uses the data presented on the serial data bus (SDA) by the competing transmitters. Figure 11 illustrates the arbitration procedure between two devices. The first master-transmitter, which drives SDA high, is overruled by another master-transmitter that drives SDA low. 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.

If the I2C module is the losing master, it switches to the slave-receiver mode, sets the arbitration lost (AL) flag, and generates the arbitration-lost interrupt request.

If during a serial transfer the arbitration procedure is still in progress when a repeated START condition or a STOP condition is transmitted to SDA, the master-transmitters involved must send the repeated START condition or the STOP condition at the same position in the format frame. Arbitration is not allowed between:

- A repeated START condition and a data bit
- A STOP condition and a data bit
- A repeated START condition and a STOP condition

Figure 11. Arbitration Procedure Between Two Master-Transmitters

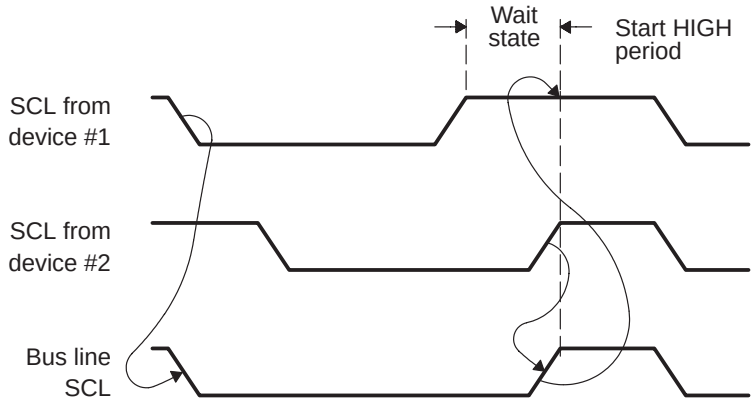


2.8 Clock Synchronization

Under normal conditions, only one master device generates the clock signal, SCL. During the arbitration procedure, however, there are two or more masters and the clock must be synchronized so that the data output can be compared. Figure 12 illustrates the clock synchronization. The wired-AND property of SCL means that a device that first generates a low period on SCL overrules the other devices. At this high-to-low transition, the clock generators of the other devices are forced to start their own low period. The SCL is held low by the device with the longest low period. The other devices that finish their low periods must wait for SCL to be released, before starting their high periods. A synchronized signal on SCL is obtained, where the slowest device determines the length of the low period and the fastest device determines 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 slows down a fast master and the slow device creates enough time to store a received byte or to prepare a byte to be transmitted.

Figure 12. Synchronization of Two I2C Clock Generators During Arbitration



3 Interrupt Requests Generated by the I2C Module

The I2C module can generate seven types of basic interrupt requests, which are described in section 3.1. Two of these can tell the CPU when to write transmit data and when to read receive data. If you want the FIFOs to handle transmit and receive data, you can also use the FIFO interrupts described in section 3.2. The basic I2C interrupts are combined to form PIE Group 8, Interrupt 1 (I2CINT1A_ISR), and the FIFO interrupts are combined to form PIE Group 8, Interrupt 2 (I2CINT2A_ISR).

3.1 Basic I2C Interrupt Requests

The I2C module generates the interrupt requests described in Table 3. As shown in Figure 13, all requests are multiplexed through an arbiter to a single I2C interrupt request to the CPU. Each interrupt request has a flag bit in the status register (I2CSTR) and an enable bit in the interrupt enable register (I2CIER). When one of the specified events occurs, its flag bit is set. If the corresponding enable bit is 0, the interrupt request is blocked. If the enable bit is 1, the request is forwarded to the CPU as an I2C interrupt.

The I2C interrupt is one of the maskable interrupts of the CPU. As with any maskable interrupt request, if it is properly enabled in the CPU, the CPU executes the corresponding interrupt service routine (I2CINT1A_ISR). The I2CINT1A_ISR for the I2C interrupt can determine the interrupt source by reading the interrupt source register, I2CISRC (see page 46). Then the I2CINT1A_ISR can branch to the appropriate subroutine.

After the CPU reads I2CISRC, the following events occur:

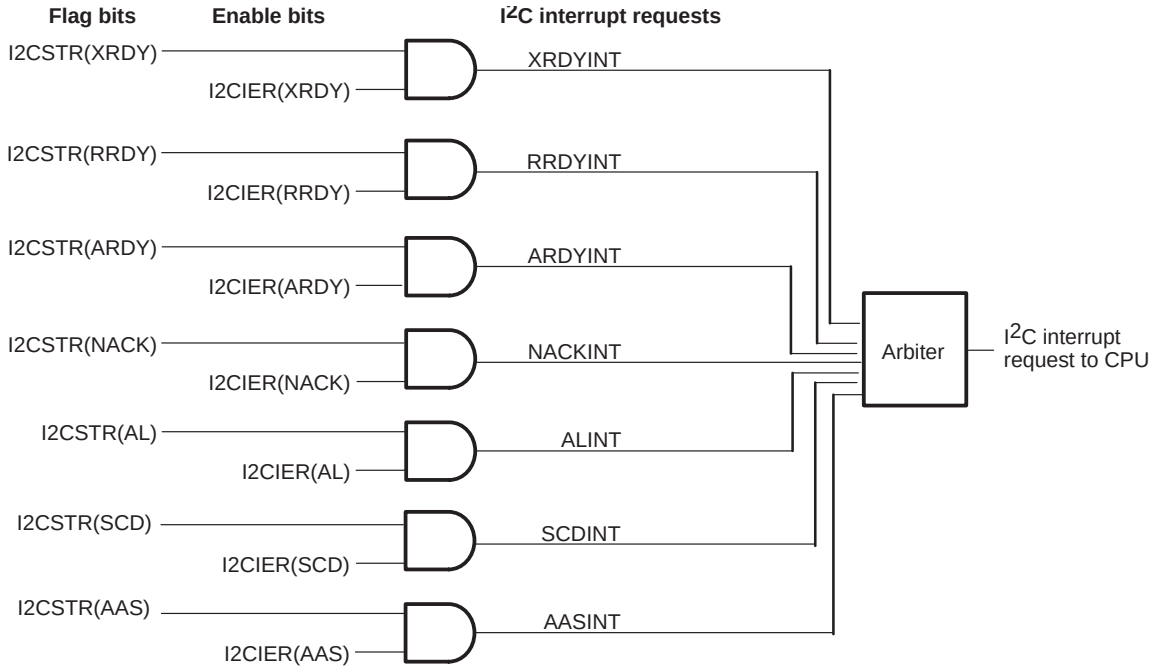
1) The flag for the source interrupt is cleared in I2CSTR. *Exception:* The ARDY, RRDY, and XRDY bits in I2CSTR are not cleared when I2CISRC is read. To clear one of these bits, write a 1 to it.

2) The arbiter determines which of the remaining interrupt requests has the highest priority, writes the code for that interrupt to I2CISRC, and forwards the interrupt request to the CPU.

Table 3. Descriptions of the Basic I2C Interrupt Requests

I2C Interrupt Request	Interrupt Source
XRDYINT	Transmit ready condition: The data transmit register (I2CDXR) is ready to accept new data because the previous data has been copied from I2CDXR to the transmit shift register (I2CXSR). As an alternative to using XRDYINT, the CPU can poll the XRDY bit of the status register, I2CSTR.
RRDYINT	Receive ready condition: The data receive register (I2CDRR) is ready to be read because data has been copied from the receive shift register (I2CRSR) to I2CDRR. As an alternative to using RRDYINT, the CPU can poll the RRDY bit of I2CSTR.
ARDYINT	Register-access ready condition: The I2C module registers are ready to be accessed because the previously programmed address, data, and command values have been used. The specific events that generate ARDYINT are the same events that set the ARDY bit of I2CSTR. As an alternative to using ARDYINT, the CPU can poll the ARDY bit.
NACKINT	No-acknowledgement condition: The I2C module is configured as a master-transmitter and did not received acknowledgement from the slave-receiver. As an alternative to using NACKINT, the CPU can poll the NACK bit of I2CSTR.
ALINT	Arbitration-lost condition: The I2C module has lost an arbitration contest with another master-transmitter. As an alternative to using ALINT, the CPU can poll the AL bit of I2CSTR.
SCDINT	Stop condition detected: A STOP condition was detected on the I2C bus. As an alternative to using SCDINT, the CPU can poll the SCD bit of the status register, I2CSTR.
AASINT	Addressed as slave condition: The I2C has been addressed as a slave device by another master on the I2C bus. As an alternative to using AASINT, the CPU can poll the AAS bit of the status register, I2CSTR.

Figure 13. Enable Paths of the I2C Interrupt Requests



3.2 I2C FIFO Interrupts

In addition to the seven basic I2C interrupts, the transmit and receive FIFOs each contain the ability to generate an interrupt (I2CINT2A). The transmit FIFO can be configured to generate an interrupt after transmitting a defined number of bytes, up to 16. The receive FIFO can be configured to generate an interrupt after receiving a defined number of bytes, up to 16. These two interrupt sources are ORed together into a single maskable CPU interrupt. The interrupt service routine can then read the FIFO interrupt status flags to determine from which source the interrupt came. See the I2C transmit FIFO register (I2CFFTX) and the I2C receive FIFO register (I2CFFRX) descriptions (page 55 and page 56) for more details.

4 Resetting/Disabling the I2C Module

You can reset/disable the I2C module in two ways:

- Write 0 to the I2C reset bit (IRS) in the I2C mode register (I2CMDR). All status bits (in I2CSTR) are forced to their default values, and the I2C module remains disabled until IRS is changed to 1. The SDA and SCL pins are in the high-impedance state.
- Initiate a DSP reset by driving the $\overline{\text{XRS}}$ pin low. The entire DSP is reset and is held in the reset state until you drive the pin high. When $\overline{\text{XRS}}$ is released, all I2C module registers are reset to their default values. The IRS bit is forced to 0, which resets the I2C module. The I2C module stays in the reset state until you write 1 to IRS.

IRS must be 0 while you configure/reconfigure the I2C module. Forcing IRS to 0 can be used to save power and to clear error conditions.

5 I2C Module Registers

Table 4 lists the I2C module registers. All but the receive and transmit shift registers (I2CRSR and I2CXSR) are accessible to the CPU.

Table 4. I2C Module Registers

Name	Address	Description
I2COAR	0x7900	I2C own address register
I2CIER	0x7901	I2C interrupt enable register
I2CSTR	0x7902	I2C status register
I2CCLKL	0x7903	I2C clock low-time divider register
I2CCLKH	0x7904	I2C clock high-time divider register
I2CCNT	0x7905	I2C data count register
I2CDRR	0x7906	I2C data receive register
I2CSAR	0x7907	I2C slave address register
I2CDXR	0x7908	I2C data transmit register
I2CMDR	0x7909	I2C mode register
I2CISRC	0x790A	I2C interrupt source register
I2CPSC	0x790C	I2C prescaler register
I2CFFTX	0x7920	I2C FIFO transmit register
I2CFFRX	0x7921	I2C FIFO receive register
I2CRSR	–	I2C receive shift register (not accessible to the CPU)
I2CXSR	–	I2C transmit shift register (not accessible to the CPU)

5.1 I2C Mode Register (I2CMDR)

The I2C mode register (I2CMDR) is a 16-bit I/O-mapped register that contains the control bits of the I2C module. The bit fields of I2CMDR are shown in Figure 14 and described in Table 5.

Figure 14. I2C Mode Register (I2CMDR)

15	14	13	12	11	10	9	8
NACKMOD	FREE	STT	Reserved	STP	MST	TRX	XA
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2		0
RM	DLB	IRS	STB	FDF	BC		
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0		

Legend: R = Read; W = Write; -*n* = Value after reset

Table 5. I2C Mode Register (I2CMDR) Field Descriptions

Bit	Field	Value	Description
15	NACKMOD		NACK mode bit. This bit is only applicable when the I2C module is acting as a receiver.
		0	In the slave-receiver mode: The I2C module sends an acknowledge (ACK) bit to the transmitter during each acknowledge cycle on the bus. The I2C module only sends a no-acknowledge (NACK) bit if you set the NACKMOD bit. In the master-receiver mode: The I2C module sends an ACK bit during each acknowledge cycle until the internal data counter counts down to 0. At that point, the I2C module sends a NACK bit to the transmitter. To have a NACK bit sent earlier, you must set the NACKMOD bit.
		1	In either slave-receiver or master-receiver mode: The I2C module sends a NACK bit to the transmitter during the next acknowledge cycle on the bus. Once the NACK bit has been sent, NACKMOD is cleared. *Important:* To send a NACK bit in the next acknowledge cycle, you must set NACKMOD before the rising edge of the last data bit.

Table 5. I2C Mode Register (I2CMDR) Field Descriptions (Continued)

Bit	Field	Value	Description
14	FREE		This emulation mode bit is used to determine what the state of the I2C module will be when a breakpoint is encountered in the high-level language debugger.
		0	When I2C module is master: If SCL is low when the breakpoint occurs, the I2C module stops immediately and keeps driving SCL low, whether the I2C module is the transmitter or the receiver. If SCL is high, the I2C module waits until SCL becomes low and then stops. When I2C module is slave: A breakpoint forces the I2C module to stop when the current transmission/reception is complete.
		1	The I2C module runs free; that is, it continues to operate when a breakpoint occurs.
13	STT		START condition bit (only applicable when the I2C module is a master). The RM, STT, and STP bits determine when the I2C module starts and stops data transmissions (see Table 6 on page 37). Note that the STT and STP bits can be used to terminate the repeat mode, and that this bit is not writable when IRS=0.
		0	In the master mode, STT is automatically cleared after the START condition has been generated.
		1	In the master mode, setting STT to 1 causes the I2C module to generate a START condition on the I2C-bus.
12	Reserved		This reserved bit location is always read as a 0. A value written to this bit has no effect.
11	STP		STOP condition bit (only applicable when the I2C module is a master). In the master mode, the RM, STT, and STP bits determine when the I2C module starts and stops data transmissions (see Table 6 on page 37). Note that the STT and STP bits can be used to terminate the repeat mode, and that this bit is not writable when IRS=0.
		0	STP is automatically cleared after the STOP condition has been generated.
		1	STP has been set by the DSP to generate a STOP condition when the internal data counter of the I2C module counts down to 0.

Table 5. I2C Mode Register (I2CMDR) Field Descriptions (Continued)

Bit	Field	Value	Description
10	MST		Master mode bit. MST determines whether the I2C module is in the slave mode or the master mode. MST is automatically changed from 1 to 0 when the I2C master generates a STOP condition.
		0	Slave mode. The I2C module is a slave and receives the serial clock from the master.
		1	Master mode. The I2C module is a master and generates the serial clock on the SCL pin.
9	TRX		Transmitter mode bit. When relevant, TRX selects whether the I2C module is in the transmitter mode or the receiver mode. Table 7 on page 37 summarizes when TRX is used and when it is a don't care.
		0	Receiver mode. The I2C module is a receiver and receives data on the SDA pin.
		1	Transmitter mode. The I2C module is a transmitter and transmits data on the SDA pin.
8	XA		Expanded address enable bit.
		0	7-bit addressing mode (normal address mode). The I2C module transmits 7-bit slave addresses (from bits 6−0 of I2CSAR), and its own slave address has 7 bits (bits 6−0 of I2COAR).
		1	10-bit addressing mode (expanded address mode). The I2C module transmits 10-bit slave addresses (from bits 9−0 of I2CSAR), and its own slave address has 10 bits (bits 9−0 of I2COAR).
7	RM		Repeat mode bit (only applicable when the I2C module is a master-transmitter). The RM, STT, and STP bits determine when the I2C module starts and stops data transmissions (see Table 6 on page 37).
		0	Nonrepeat mode. The value in the data count register (I2CCNT) determines how many bytes are received/transmitted by the I2C module.
		1	Repeat mode. Bytes are continuously received/transmitted by the I2C module until the STP bit is manually set to 1, regardless of the value in I2CCNT.

Table 5. I2C Mode Register (I2CMDR) Field Descriptions (Continued)

Bit	Field	Value	Description
6	DLB		Digital loopback mode bit. The effects of this bit are shown in Figure 15 (page 38).
		0	Digital loopback mode is disabled.
		1	Digital loopback mode is enabled. For proper operation in this mode, the MST bit must be 1. In the digital loopback mode, data transmitted out of I2CDXR is received in I2CDRR after n DSP cycles by an internal path, where: n = ((I2C input clock frequency/module clock frequency) × 8) The transmit clock is also the receive clock. The address transmitted on the SDA pin is the address in I2COAR. **Note:** The free data format (FDF = 1) is not supported in the digital loopback mode.
5	IRS		I2C module reset bit.
		0	The I2C module is in reset/disabled. When this bit is cleared to 0, all status bits (in I2CSTR) are set to their default values.
		1	The I2C module is enabled. This has the effect of releasing the I2C bus if the I2C peripheral is holding it.
4	STB		START byte mode bit. This bit is only applicable when the I2C module is a master. As described in version 2.1 of the Philips Semiconductors I2C-bus specification, the START byte can be used to help a slave that needs extra time to detect a START condition. When the I2C module is a slave, it ignores a START byte from a master, regardless of the value of the STB bit.
		0	The I2C module is not in the START byte mode.
		1	The I2C module is in the START byte mode. When you set the START condition bit (STT), the I2C module begins the transfer with more than just a START condition. Specifically, it generates: 1) A START condition 2) A START byte (0000 0001b) 3) A dummy acknowledge clock pulse 4) A repeated START condition Then, as normal, the I2C module sends the slave address that is in I2CSAR.

Table 5. I2C Mode Register (I2CMDR) Field Descriptions (Continued)

Bit	Field	Value	Description
3	FDF		Free data format mode bit.
		0	Free data format mode is disabled. Transfers use the 7-/10-bit addressing format selected by the XA bit.
		1	Free data format mode is enabled. Transfers have the free data (no address) format described in section 2.5 (page 21).
			Note: The free data format is not supported in the digital loopback mode (DLB = 1).
2−0	BC		Bit count bits. BC defines the number of bits (1 to 8) in the next data byte that is to be received or transmitted by the I2C module. The number of bits selected with BC must match the data size of the other device. Notice that when BC = 000b, a data byte has 8 bits. BC does not affect address bytes, which always have 8 bits.
			Note: If the bit count is less than 8, receive data is right-justified in I2CDRR(7−0), and the other bits of I2CDRR(7−0) are undefined. Also, transmit data written to I2CDXR must be right-justified.
		000	8 bits per data byte
		001	1 bit per data byte
		010	2 bits per data byte
		011	3 bits per data byte
		100	4 bits per data byte
		101	5 bits per data byte
		110	6 bits per data byte
		111	7 bits per data byte

Table 6. Master-Transmitter/Receiver Bus Activity Defined by the RM, STT, and STP Bits of I2CMDR

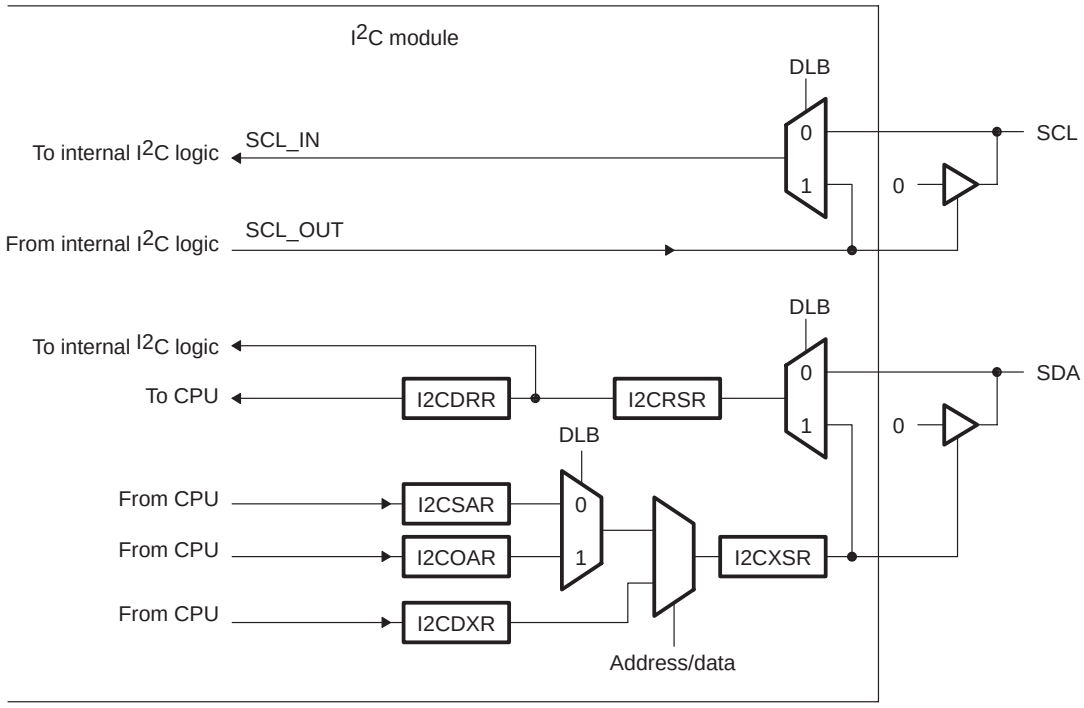
RM	STT	STP	Bus Activity†	Description
0	0	0	None	No activity
0	0	1	P	STOP condition
0	1	0	S-A-D..(n)..D	START condition, slave address, n data bytes (n = value in I2CCNT)
0	1	1	S-A-D..(n)..D-P	START condition, slave address, n data bytes, STOP condition (n = value in I2CCNT)
1	0	0	None	No activity
1	0	1	P	STOP condition
1	1	0	S-A-D-D-D…..	Repeat mode transfer: START condition, slave address, continuous data transfers until STOP condition or next START condition
1	1	1	None	Reserved bit combination (No activity)

† S = START condition; A = Address; D = Data byte; P = STOP condition

Table 7. How the MST and FDF Bits of I2CMDR Affect the Role of the TRX Bit of I2CMDR

MST	FDF	I2C Module State	Function of TRX
0	0	In slave mode but not free data format mode	TRX is a don't care. Depending on the command from the master, the I2C module responds as a receiver or a transmitter.
0	1	In slave mode and free data format mode	The free data format mode requires that the I2C module remains the transmitter or the receiver throughout the transfer. TRX identifies the role of the I2C module: TRX = 0: The I2C module is a receiver. TRX = 1: The I2C module is a transmitter.
1	X	In master mode; free data format mode on or off	TRX = 0: The I2C module is a receiver. TRX = 1: The I2C module is a transmitter.

Figure 15. Pin Diagram Showing the Effects of the Digital Loopback Mode (DLB) Bit



5.2 I2C Interrupt Enable Register (I2CIER)

I2CIER is used by the CPU to individually enable or disable I2C interrupt requests. The bits of I2CIER are shown and described in Figure 16 and Table 8, respectively.

Figure 16. I2C Interrupt Enable Register (I2CIER)

15	14	13	12	11	10	9	8
Reserved							
R-0							

7	6	5	4	3	2	1	0
Reserved	AAS	SCD	XRDY	RRDY	ARDY	NACK	AL
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = Read; W = Write; -*n* = Value after reset

Table 8. I2C Interrupt Enable Register (I2CIER) Field Descriptions

Bit	Field	Value	Description
15−7	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
6	AAS		Addressed as slave interrupt enable bit
		0	Interrupt request disabled
		1	Interrupt request enabled
5	SCD		Stop condition detected interrupt enable bit
		0	Interrupt request disabled
		1	Interrupt request enabled
4	XRDY		Transmit-data-ready interrupt enable bit
		0	Interrupt request disabled
		1	Interrupt request enabled
3	RRDY		Receive-data-ready interrupt enable bit
		0	Interrupt request disabled
		1	Interrupt request enabled
2	ARDY		Register-access-ready interrupt enable bit
		0	Interrupt request disabled
		1	Interrupt request enabled
1	NACK		No-acknowledgement interrupt enable bit
		0	Interrupt request disabled
		1	Interrupt request enabled
0	AL		Arbitration-lost interrupt enable bit
		0	Interrupt request disabled
		1	Interrupt request enabled

5.3 I2C Status Register (I2CSTR)

The I2C status register (I2CSTR) is a 16-bit I/O-mapped register used to determine which interrupt has occurred and to read status information. The bits of I2CSTR are shown and described in Figure 17 and Table 9, respectively.

Figure 17. I2C Status Register (I2CSTR)

15	14	13	12	11	10	9	8
Reserved	SDIR	NACKSNT	BB	RSFULL	XSMT	AAS	AD0
R-0	R/W1C-0	R/W1C-0	R/W1C-0	R-0	R-1	R-0	R-0

7	6	5	4	3	2	1	0
Reserved		SCD	XRDY	RRDY	ARDY	NACK	AL
R-0		R/W1C-0	R-1	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0

Legend: R = Read; W1C = Write 1 to clear (writing 0 has no effect); W1 = Write one to set; *-n* = Value after reset

Table 9. I2C Status Register (I2CSTR) Field Descriptions

Bit	Field	Description
15	Reserved	This reserved bit location is always read as zeros. A value written to this bit has no effect.
14	SDIR	Slave direction bit 0 I2C is not addressed as a slave transmitter. SDIR is cleared by one of the following events: ❑ It is manually cleared. To clear this bit, write a 1 to it. ❑ Digital loopback mode is enabled. ❑ A START or STOP condition occurs on the I2C bus. 1 I2C is addressed as a slave transmitter.
13	NACKSNT	NACK sent bit. This bit is used when the I2C module is in the receiver mode. One instance in which NACKSNT is affected is when the NACK mode is used (see the description for NACKMOD in section 5.1). 0 NACK not sent. NACKSNT bit is cleared by any one of the following events: ❑ It is manually cleared. To clear this bit, write a 1 to it. ❑ The I2C module is reset (either when 0 is written to the IRS bit of I2CMDR or when the whole device is reset). 1 NACK sent: A no-acknowledge bit was sent during the acknowledge cycle on the I2C-bus.

Table 9. I2C Status Register (I2CSTR) Field Descriptions (Continued)

Bit	Field	Description
12	BB	Bus busy bit. BB indicates whether the I2C-bus is busy or is free for another data transfer. Note that setting this bit to 1 will clear it. See paragraph following table for more information. 0 Bus free. BB is cleared by any one of the following events: ❑ The I2C module receives or transmits a STOP bit (bus free). ❑ BB is manually cleared. To clear this bit, write a 1 to it. ❑ The I2C module is reset. 1 Bus busy: The I2C module has received or transmitted a START bit on the bus.
11	RSFULL	Receive shift register full bit. RSFULL indicates an overrun condition during reception. Overrun occurs when the data has been received in the shift register (I2CRSR) and copied into the data receive register (I2CDRR). As new bits arrive from the SDA pin, they overwrite the bits in I2CRSR. The new data will not be copied to ICDRR until the previous data is read. 0 No overrun detected. RSFULL is cleared by any one of the following events: ❑ I2CDRR is read. ❑ The I2C module is reset. 1 Overrun detected
10	XSMT	Transmit shift register empty bit. XSMT = 0 indicates that the transmitter has experienced underflow. Underflow occurs when the transmit shift register (I2CXSR) is empty but the data transmit register (I2CDXR) has not been loaded since the last I2CDXR-to-I2CXSR transfer. The next I2CDXR-to-I2CXSR transfer will not occur until new data is in I2CDXR. If new data is not transferred in time, the previous data may be re-transmitted on the SDA pin. 0 Underflow detected (empty) 1 No underflow detected (not empty). XSMT is set by one of the following events: ❑ Data is written to I2CDXR. ❑ The I2C module is reset.

Table 9. I2C Status Register (I2CSTR) Field Descriptions (Continued)

Bit	Field	Description
9	AAS	Addressed-as-slave bit 0 In the 7-bit addressing mode, the AAS bit is cleared when receiving a NACK, a STOP condition, or a repeated START condition. In the 10-bit addressing mode, the AAS bit is cleared when receiving a NACK, a STOP condition, or by a slave address different from the I2C peripheral's own slave address. 1 The I2C module has recognized its own slave address or an address of all zeros (general call). The AAS bit is also set if the first byte has been received in the free data format (FDF = 1 in I2CMDR).
8	AD0	Address 0 bit 0 AD0 has been cleared by a START or STOP condition. 1 An address of all zeros (general call) is detected.
7−6	Reserved	These reserved bit locations are always read as zeros. A value written to this field has no effect.
5	SCD	Stop condition detected bit. SCD is set when the I2C sends or receives a STOP condition. 0 STOP condition not detected since SCD was last cleared. SCD is cleared by any one of the following events: ❑ I2CISRC is read when it contains the value 110b (stop condition detected). ❑ SCD is manually cleared. To clear this bit, write a 1 to it. ❑ The I2C module is reset. 1 A STOP condition has been detected on the I2C bus.
4	XRDY	Transmit-data-ready interrupt flag bit. XRDY indicates that the data transmit register (I2CDXR) is ready to accept new data because the previous data has been copied from I2CDXR to the transmit shift register (I2CXSR). The CPU can poll XRDY or use the XRDY interrupt request (see section 3.1, page 27). 0 I2CDXR not ready. XRDY is cleared when data is written to I2CDXR. 1 I2CDXR ready: Data has been copied from I2CDXR to I2CXSR. XRDY is also forced to 1 when the I2C module is reset.

Table 9. I2C Status Register (I2CSTR) Field Descriptions (Continued)

Bit	Field	Description
3	RRDY	Receive-data-ready interrupt flag bit. RRDY indicates that the data receive register (I2CDRR) is ready to be read because data has been copied from the receive shift register (I2CRSR) to I2CDRR. The CPU can poll RRDY or use the RRDY interrupt request (see section 3.1, page 27). 0 I2CDRR not ready. RRDY is cleared by any one of the following events: ❏ I2CDRR is read. ❏ RRDY is manually cleared. To clear this bit, write a 1 to it. ❏ The I2C module is reset. 1 I2CDRR ready: Data has been copied from I2CRSR to I2CDRR.
2	ARDY	Register-access-ready interrupt flag bit (only applicable when the I2C module is in the master mode). ARDY indicates that the I2C module registers are ready to be accessed because the previously programmed address, data, and command values have been used. The CPU can poll ARDY or use the ARDY interrupt request (see section 3.1, page 27). 0 The registers are not ready to be accessed. ARDY is cleared by any one of the following events: ❏ The I2C module starts using the current register contents. ❏ ARDY is manually cleared. To clear this bit, write a 1 to it. ❏ The I2C module is reset. 1 The registers are ready to be accessed. In the nonrepeat mode (RM = 0 in I2CMDR): If STP = 0 in I2CMDR, the ARDY bit is set when the internal data counter counts down to 0. If STP = 1, ARDY is not affected (instead, the I2C module generates a STOP condition when the counter reaches 0). In the repeat mode (RM = 1): ARDY is set at the end of each byte transmitted from I2CDXR.

Table 9. I2C Status Register (I2CSTR) Field Descriptions (Continued)

Bit	Field	Description
1	NACK	No-acknowledgement interrupt flag bit. NACK applies when the I2C module is a transmitter (master or slave). NACK indicates whether the I2C module has detected an acknowledge bit (ACK) or a no-acknowledge bit (NACK) from the receiver. The CPU can poll NACK or use the NACK interrupt request (see section 3.1, page 27). 0 ACK received/NACK not received. This bit is cleared by any one of the following events: ❏ An acknowledge bit (ACK) has been sent by the receiver. ❏ NACK is manually cleared. To clear this bit, write a 1 to it. ❏ The CPU reads the interrupt source register (I2CISRC) when the register contains the code for a NACK interrupt. ❏ The I2C module is reset. 1 NACK bit received. The hardware detects that a no-acknowledge (NACK) bit has been received. **Note:** While the I2C module performs a general call transfer, NACK is 1, even if one or more slaves send acknowledgement.
0	AL	Arbitration-lost interrupt flag bit (only applicable when the I2C module is a master-transmitter). AL primarily indicates when the I2C module has lost an arbitration contest with another master-transmitter. The CPU can poll AL or use the AL interrupt request (see section 3.1, page 27). 0 Arbitration not lost. AL is cleared by any one of the following events: ❏ AL is manually cleared. To clear this bit, write a 1 to it. ❏ The CPU reads the interrupt source register (I2CISRC) when the register contains the code for an AL interrupt. ❏ The I2C module is reset. 1 Arbitration lost. AL is set by any one of the following events: ❏ The I2C module senses that it has lost an arbitration with two or more competing transmitters that started a transmission almost simultaneously. ❏ The I2C module attempts to start a transfer while the BB (bus busy) bit is set to 1. When AL becomes 1, the MST and STP bits of I2CMDR are cleared, and the I2C module becomes a slave-receiver.

The I2C peripheral cannot detect a START or STOP condition when it is in reset, i.e. the IRS bit is set to 0. Therefore, the BB bit will remain in the state it was at when the peripheral was placed in reset. The BB bit will stay in that state until the I2C peripheral is taken out of reset, i.e. the IRS bit is set to 1, and a START or STOP condition is detected on the I2C bus.

Follow these steps before initiating the first data transfer with I2C :

1) After taking the I2C peripheral out of reset by setting the IRS bit to 1, wait a certain period to detect the actual bus status before starting the first data transfer. Set this period larger than the total time taken for the longest data transfer in the application. By waiting for a period of time after I2C comes out of reset, users can ensure that at least one START or STOP condition will have occurred on the I2C bus, and been captured by the BB bit. After this period, the BB bit will correctly reflect the state of the I2C bus.
2) Check the BB bit and verify that BB=0 (bus not busy) before proceeding.
3) Begin data transfers.

Not resetting the I2C peripheral in between transfers ensures that the BB bit reflects the actual bus status. If users must reset the I2C peripheral in between transfers, repeat steps 1 through 3 every time the I2C peripheral is taken out of reset.

5.4 I2C Interrupt Source Register (I2CISRC)

The I2C interrupt source register (I2CISRC) is a 16-bit I/O-mapped register used by the CPU to determine which event generated the I2C interrupt. For more information about these events, see the descriptions of the I2C interrupt requests in Table 3 (page 28).

Figure 18. I2C Interrupt Source Register (I2CISRC)

15	3	2	0
Reserved		INTCODE	
R-0		R-0	

Legend: R = Read; W = Write; -*n* = Value after reset

Table 10. I2C Interrupt Source Register (I2CISRC) Field Descriptions

Bit	Field	Description	
15−3	Reserved	These reserved bit locations are always read as zeros. A value written to this field has no effect.	
2−0	INTCODE	Interrupt code bits. The binary code in INTCODE indicates the event that generated an I2C interrupt.	
		000	None
		001	Arbitration lost
		010	No-acknowledgement condition detected
		011	Registers ready to be accessed
		100	Receive data ready
		101	Transmit data ready
		110	Stop condition detected
		111	Addressed as slave

5.5 I2C Prescaler Register (I2CPSC)

The I2C prescaler register (I2CPSC) is a 16-bit register used for dividing down the I2C input clock to obtain the desired module clock for the operation of the I2C module. See the device-specific data manual for the supported range of values for the module clock frequency. For more details about the module clock, see section 1.4 (page 16).

IPSC must be initialized while the I2C module is in reset (IRS = 0 in I2CMDR). The prescaled frequency takes effect only when IRS is changed to 1. Changing the IPSC value while IRS = 1 has no effect.

Figure 19. I2C Prescaler Register (I2CPSC)

15	8	7	0
Reserved		IPSC	
R-0		R/W-0	

Legend: R = Read; W = Write; -*n* = Value after reset

Table 11. I2C Prescaler Register (I2CPSC) Field Descriptions

Bit	Field	Description
15−8	Reserved	These reserved bit locations are always read as zeros. A value written to this field has no effect.
7−0	IPSC	I2C prescaler divide-down value.
	00h−FFh	IPSC determines how much the CPU clock is divided to create the module clock of the I2C module: module clock frequency = I2C input clock frequency/(IPSC + 1) **Note:** IPSC must be initialized while the I2C module is in reset (IRS = 0 in I2CMDR).

Note:

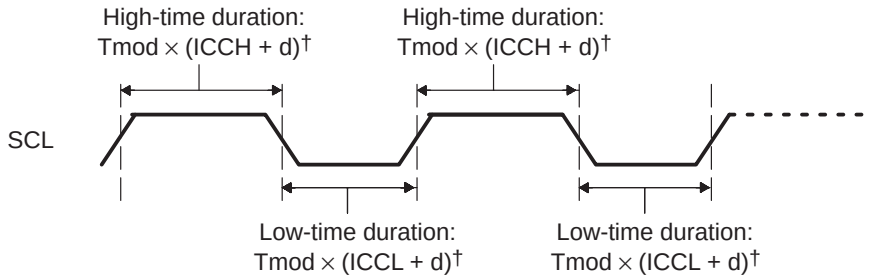
To meet all of the I2C protocol timing specifications, the module clock *must* be configured between 7−12 MHz.

5.6 I2C Clock Divider Registers (I2CCLKL and I2CCLKH)

As explained in section 1.4, when the I2C module is a master, the module clock is divided down for use as the master clock on the SCL pin. As shown in Figure 20, the shape of the master clock depends on two divide-down values:

- ICCL in I2CCLKL (summarized by Figure 21 and Table 12). For each master clock cycle, ICCL determines the amount of time the signal is low.
- ICCH in I2CCLKH (summarized by Figure 22 and Table 13). For each master clock cycle, ICCH determines the amount of time the signal is high.

Figure 20. The Roles of the Clock Divide-Down Values (ICCL and ICCH)



† As described in section 5.6.1, Tmod is the module clock period, and d is 5, 6, or 7.

Figure 21. I2C Clock Low-Time Divider Register (I2CCLKL)

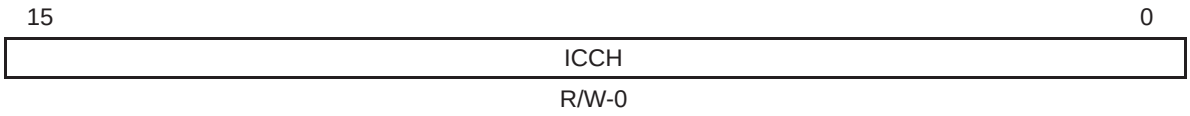
15	0
ICCL	
R/W-0	

Legend: R = Read; W = Write; -*n* = Value after reset

Table 12. I2C Clock Low-Time Divider Register (I2CCLKL) Field Description

Bit	Field	Value	Description
15−0	ICCL	0000h−FFFFh	Clock low-time divide-down value of 1−65536. To produce the low-time duration of the master clock, the period of the module clock is multiplied by (ICCL + d). d is 5, 6, or 7 as described in section 5.6.1. Note: These bits must be set to a non-zero value for proper I2C clock operation.

Figure 22. I2C Clock High-Time Divider Register (I2CCLKH)



Legend: R = Read; W = Write; -*n* = Value after reset

Table 13. I2C Clock High-Time Divider Register (I2CCLKH) Field Description

Bit	Field	Value	Description
15−0	ICCH	0000h−FFFFh	Clock high-time divide-down value of 1−65536. To produce the high-time duration of the master clock, the period of the module clock is multiplied by (ICCH + d). d is 5, 6, or 7 as described in section 5.6.1. Note: These bits must be set to a non-zero value for proper I2C clock operation.

5.6.1 Formula for the Master Clock Period

The period of the master clock (T_{mst}) is a multiple of the period of the module clock (T_{mod}):

$$T_{mst} = T_{mod} \times [(ICCL + d) + (ICCH + d)]$$

$$T_{mst} = \frac{(IPSC + 1)\ [(ICCL + d) + (ICCH + d)]}{\text{I2C input clock frequency}}$$

where d depends on the divide-down value IPSC, as shown in Table 14. IPSC is described in section 5.5.

Table 14. Dependency of Delay d on the Divide-Down Value IPSC

IPSC	d
0	7
1	6
Greater than 1	5

5.7 I2C Slave Address Register (I2CSAR)

The I2C slave address register (I2CSAR) is a register for storing the next slave address that will be transmitted by the I2C module when it is a master. It is a 16-bit I/O-mapped register with the format shown in Figure 23. As described in Table 15, the SAR field of I2CSAR contains a 7-bit or 10-bit slave address. When the I2C module is not using the free data format (FDF = 0 in I2CMDR), it uses this address to initiate data transfers with a slave or slaves. When the address is nonzero, the address is for a particular slave. When the address is 0, the address is a general call to all slaves. If the 7-bit addressing mode is selected (XA = 0 in I2CMDR), only bits 6−0 of I2CSAR are used; write 0s to bits 9−7.

Figure 23. I2C Slave Address Register (I2CSAR)

15	10	9	0
Reserved		SAR	
R-0		R/W-3FFh	

Legend: R = Read; W = Write; -*n* = Value after reset

Table 15. I2C Slave Address Register (I2CSAR) Field Descriptions

Bit	Field	Description	
15−10	Reserved	These reserved bit locations are always read as zeros. A value written to this field has no effect.	
9−0	SAR	In 7-bit addressing mode (XA = 0 in I2CMDR):	
		00h−7Fh	Bits 6−0 provide the 7-bit slave address that the I2C module transmits when it is in the master-transmitter mode. Write 0s to bits 9−7.
			In 10-bit addressing mode (XA = 1 in I2CMDR):
		000h−3FFh	Bits 9−0 provide the 10-bit slave address that the I2C module transmits when it is in the master-transmitter mode.

5.8 I2C Own Address Register (I2COAR)

The I2C own address register (I2COAR) is a 16-bit register mapped to the I/O space of the DSP. Figure 24 shows the format of I2COAR, and Table 16 describes its bit fields. The I2C module uses this register to specify its own slave address, which distinguishes it from other slaves connected to the I2C-bus. If the 7-bit addressing mode is selected (XA = 0 in I2CMDR), only bits 6−0 are used; write 0s to bits 9−7.

Figure 24. I2C Own Address Register (I2COAR)

Legend: R = Read; W = Write; -*n* = Value after reset

Table 16. I2C Own Address Register (I2COAR) Field Descriptions

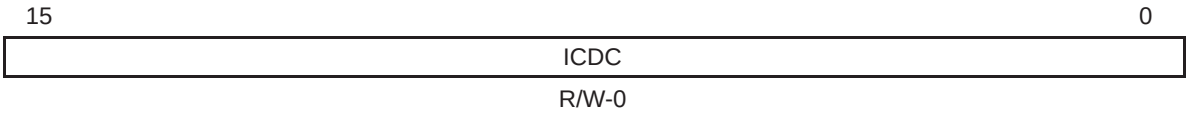
Bit	Field	Description	
15−10	Reserved	These reserved bit locations are always read as zeros. A value written to this field has no effect.	
9−0	OAR	In 7-bit addressing mode (XA = 0 in I2CMDR):	
		00h−7Fh	Bits 6−0 provide the 7-bit slave address of the I2C module. Write 0s to bits 9−7.
			In 10-bit addressing mode (XA = 1 in I2CMDR):
		000h−3FFh	Bits 9−0 provide the 10-bit slave address of the I2C module.

5.9 I2C Data Count Register (I2CCNT)

I2CCNT is a 16-bit register used to indicate how many data bytes to transfer when the I2C module is configured as a transmitter, or to receive when configured as a master receiver. In the repeat mode (RM = 1), I2CCNT is not used. The bits of I2CCNT are shown and described in Figure 25 and Table 17, respectively.

The value written to I2CCNT is copied to an internal data counter. The internal data counter is decremented by 1 for each byte transferred (I2CCNT remains unchanged). If a STOP condition is requested in the master mode (STP = 1 in I2CMDR), the I2C module terminates the transfer with a STOP condition when the countdown is complete (that is, when the last byte has been transferred).

Figure 25. I2C Data Count Register (I2CCNT)



Legend: R = Read; W = Write; -*n* = Value after reset

Table 17. I2C Data Count Register (I2CCNT) Field Description

Bit	Field	Description
15−0	ICDC	Data count value. ICDC indicates the number of data bytes to transfer or receive. The value in I2CCNT is a don't care when the RM bit in I2CMDR is set to 1.
		0000h The start value loaded to the internal data counter is 65536.
		0001h−FFFFh The start value loaded to internal data counter is 1−65535.

5.10 I2C Data Receive Register (I2CDRR)

I2CDRR (see Figure 26 and Table 18) is a 16-bit register used by the CPU to read received data. The I2C module can receive a data byte with 1 to 8 bits. The number of bits is selected with the bit count (BC) bits in I2CMDR. One bit at a time is shifted in from the the SDA pin to the receive shift register (I2CRSR). When a complete data byte has been received, the I2C module copies the data byte from I2CRSR to I2CDRR.

If a data byte with fewer than 8 bits is in I2CDRR, the data value is right-justified, and the other bits of I2CDRR(7−0) are undefined. For example, if BC = 011 (3-bit data size), the receive data is in I2CDRR(2−0), and the content of I2CDRR(7−3) is undefined.

The CPU cannot access I2CRSR.

In the receive FIFO mode, this register acts as the receive FIFO buffer.

Figure 26. I2C Data Receive Register (I2CDRR)

15	8	7	0
Reserved		DATA	
R-0		R-0	

Legend: R = Read; -*n* = Value after reset

Table 18. I2C Data Receive Register (I2CDRR) Field Description

Bit	Field	Description
15−8	Reserved	These reserved bit locations are always read as zeros. A value written to this field has no effect.
7−0	DATA	00h−FFh Receive data

5.11 I2C Data Transmit Register (I2CDXR)

The CPU writes transmit data to I2CDXR (see Figure 27 and Table 19). This 16-bit register accepts a data byte with 1 to 8 bits. Before writing to I2CDXR, specify how many bits are in a data byte by loading the appropriate value into the bit count (BC) bits of I2CMDR. When writing a data byte with fewer than 8 bits, make sure the value is right-aligned in I2CDXR.

After a data byte is written to I2CDXR, the I2C module copies the data byte to the transmit shift register (I2CXSR). From I2CXSR, the I2C module shifts the data byte out on the SDA pin, one bit at a time.

The CPU cannot access I2CXSR.

In the transmit FIFO mode, this register acts as the transmit FIFO buffer.

Figure 27. I2C Data Transmit Register (I2CDXR)

15	8	7	0
Reserved		DATA	
R-0		R/W-0	

Legend: R = Read; W = Write; -*n* = Value after reset

Table 19. I2C Data Transmit Register (I2CDXR) Field Descriptions

Bit	Field		Description
15−8	Reserved		These reserved bit locations are always read as zeros. A value written to this field has no effect.
7−0	DATA	00h−FFh	Transmit data

5.12 I2C Transmit FIFO Register (I2CFFTX)

The I2C transmit FIFO register (I2CFFTX) is a 16-bit register that contains the I2C FIFO mode enable bit as well as the control and status bits for the transmit FIFO mode of operation on the I2C peripheral. The bit fields are shown in Figure 28 and described in Table 20.

Figure 28. I2C Transmit FIFO Register (I2CFFTX)

15	14	13	12	11	10	9	8
Reserved	I2CFFEN	TXFFRST	TXFFST4	TXFFST3	TXFFST2	TXFFST1	TXFFST0
R-0	R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0

7		5	4	3	2	1	0
TXFFINT	TXFFINTCLR	TXFFIENA	TXFFIL4	TXFFIL3	TXFFIL2	TXFFIL1	TXFFIL0
R-0	R/W1C-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = Read; W = Write; W1C = Write one to clear; -*n* = Value after reset

Table 20. I2C Transmit FIFO Register (I2CFFTX) Field Descriptions

Bit	Field	Description
15	Reserved	Reserved. Reads will return a 0, writes have no effect
14	I2CFFEN	I2C FIFO mode enable bit. This bit must be enabled for either the transmit or the receive FIFO to operate correctly.
		0 Disable the I2C FIFO mode.
		1 Enable the I2C FIFO mode.
13	TXFFRST	I2C transmit FIFO reset bit.
		0 Reset the transmit FIFO pointer to 0000 and hold the transmit FIFO in the reset state.
		1 Enable the transmit FIFO operation.
12−8	TXFFST4−0	Contains the status of the transmit FIFO:
		10000 Transmit FIFO contains 16 bytes.
		0xxxx Transmit FIFO contains xxxx bytes.
		00000 Transmit FIFO is empty.

Table 20. I2C Transmit FIFO Register (I2CFFTX) Field Descriptions (Continued)

Bit	Field	Description
7	TXFFINT	Transmit FIFO interrupt flag. This bet cleared by a CPU write of a 1 to the TXFFINTCLR bit. If the TXFFIENA bit is set, this bit will generate an interrupt when it is set. 0 Transmit FIFO interrupt condition has not occurred. 1 Transmit FIFO interrupt condition has occurred.
6	TXFFINTCLR	Transmit FIFO interrupt flag clear bit. Writing a 1 to this bit clears the TXFFINT flag. Writes of zeros have no effect. Reads return a 0.
5	TXFFIENA	Transmit FIFO interrupt enable bit. 0 Disabled. TXFFINT flag does not generate an interrupt when set. 1 Enabled. TXFFINT flag does generate an interrupt when set.
4−0	TXFFIL4−0	Transmit FIFO interrupt level. These bits set the status level that will set the transmit interrupt flag. When the TXFFST4−0 bits reach a value equal to or less than these bits, the TXFFINT flag will be set. This will generate an interrupt if the TXFFIENA bit is set.

5.13 I2C Receive FIFO Register (I2CFFRX)

The I2C receive FIFO register (I2CFFRX) is a 16-bit register that contains the control and status bits for the receive FIFO mode of operation on the I2C peripheral. The bit fields are shown in Figure 29 and described in Table 21.

Figure 29. I2C Receive FIFO Register (I2CFFRX)

15	14	13	12	11	10	9	8
Reserved		RXFFRST	RXFFST4	RXFFST3	RXFFST2	RXFFST1	RXFFST0
R-0		R/W-0	R-0	R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
RXFFINT	RXFFINTCLR	RXFFIENA	RXFFIL4	RXFFIL3	RXFFIL2	RXFFIL1	RXFFIL0
R-0	R/W1C-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = Read; W = Write; W1C = Write 1 to clear; -*n* = Value after reset

Table 21. I2C Receive FIFO Register (I2CFFRX) Field Descriptions

Bit	Field	Description
15−14	Reserved	Reserved. Reads will return a 0, writes have no effect
13	RXFFRST	I2C receive FIFO reset bit. 0 Reset the receive FIFO pointer to 0000 and hold the receive FIFO in the reset state. 1 Enable the receive FIFO operation.
12−8	RXFFST4−0	Contains the status of the receive FIFO: 10000 Receive FIFO contains 16 bytes. 0xxxx Receive FIFO contains xxxx bytes. 00000 Receive FIFO is empty.
7	RXFFINT	Receive FIFO interrupt flag. This bet cleared by a CPU write of a 1 to the RXFFINTCLR bit. If the RXFFIENA bit is set, this bit will generate an interrupt when it is set. 0 Receive FIFO interrupt condition has not occurred. 1 Receive FIFO interrupt condition has occurred.
6	RXFFINTCLR	Receive FIFO interrupt flag clear bit. Writes of zeros have no effect. Reads return a zero. Writing a 1 to this bit clears the RXFFINT flag.
5	RXFFIENA	Receive FIFO interrupt enable bit. 0 Disabled. RXFFINT flag does not generate an interrupt when set. 1 Enabled. RXFFINT flag does generate an interrupt when set.
4−0	RXFFIL4−0	Receive FIFO interrupt level. These bits set the status level that will set the receive interrupt flag. When the RXFFST4−0 bits reach a value equal to or greater than these bits, the RXFFINT flag is set. This will generate an interrupt if the RXFFIENA bit is set.

This page is intentionally left blank.

Index

10-bit addressing format 22
7-bit addressing format 22

A

AAS bit of I2CSTR 40, 56
AD0 bit of I2CSTR 40
address 0 bit (AD0) 40
addressed as slave interrupt enable bit of I2CSTR 40
addressed−as−slave bit (AAS) 40
addressing formats (serial data formats) 21
AL bit of I2CIER 38
AL bit of I2CSTR 40
arbitration among master devices 25
arbitration−lost interrupt enable bit (AL) 38
arbitration−lost interrupt flag bit (AL) 40
ARDY bit of I2CIER 38
ARDY bit of I2CSTR 40

B

bit count (BC) bits of I2CMDR 32
bus busy (BB) bit of I2CSTR 40

C

clock generation 16
clock high−time divider register (I2CCLKH) 48
clock low−time divider register (I2CCLKL) 48
clock synchronization 26
count register (I2CCNT) 52

D

DATA bits of I2CDRR 53
DATA bits of I2CDXR 54
data count register (I2CCNT) 52
data formats 21
data receive register (I2CDRR) 53
data transmit register (I2CDXR) 54
data validity 18
digital loopback mode bit (DLB) bit 32
digital loopback shown in pin diagram 38
disabling the I2C module 30
DLB bit of I2CMDR 32

E

emulation mode bit (FREE) 32
expanded address enable bit (XA) 32

F

FDF bit of I2CMDR 32
features 12
features not supported 13
FIFO mode enable bit of I2CFFTX 55
formula for master clock period 49
FREE bit of I2CMDR 32
free data format 23
free data format mode bit (FDF) 32
functional overview 13

I

I2C FIFO Interrupts 29
I2C input clock 16
I2C module reset bit (IRS) 32
I2C modules connected (figure) 12
I2CCLKH 48
I2CCLKL 48
I2CCNT 52
I2CDRR 53
I2CDXR 54
I2CIER 38
I2CISRC 46
I2CMDR 32
I2COAR 51
I2CPSC 47
I2CRSR 53
I2CSAR 50
I2CSTR 40
I2CXSR 54
ICCH bits of I2CCLKH 49
ICCL bits of I2CCLKL 48
ICDC bits of I2CCNT 52
idle enable (IDLEEN) bit of I2CMDR 32
input and output voltage levels 18
input clock 16
interrupt code (INTCODE) bits of I2CISRC 46
interrupt enable register (I2CIER) 38
interrupt flags in I2CSTR 40
interrupt requests 27
interrupt source register (I2CISRC) 46
introduction 11
IPSC bits of I2CPSC 47
IRS bit of I2CMDR 32

M

master clock 16
master clock period formula 49
master mode bit (MST) 32
master−receiver mode 19
master−transmitter mode 19
mode register (I2CMDR) 32
module clock 16
MST bit of I2CMDR 32
multiple I2C modules connected (figure) 12

N

NACK bit generation 24
NACK bit of I2CIER 38
NACK bit of I2CSTR 40
NACK mode (NACKMOD) bit of I2CMDR 32
NACK sent (NACKSNT) bit of I2CSTR 40
no−acknowledgement interrupt enable bit (NACK) 38
no−acknowledgement interrupt flag bit (NACK) 40
nonrepeat mode selectable with RM bit 32

O

OAR bits of I2COAR 51
operating modes 18
operational details 18
output and input voltage levels 18
own address register (I2COAR) 51

P

pin diagram 38
prescaler register (I2CPSC) 47

R

receive data bits (DATA) 53
receive FIFO interrupt enable bit 56
receive FIFO interrupt flag bit 56
receive FIFO interrupt flag clear bit 56
receive FIFO interrupt level 56
receive FIFO reset bit of I2CFFRX 56
receive FIFO status bit 56
receive shift register (I2CRSR) 53
receive shift register full bit (RSFULL) 40
receive−data−ready interrupt enable bit (RRDY) 38
receive−data−ready interrupt flag bit (RRDY) 40
receiver mode selectable with TRX bit 32
register−access−ready interrupt enable bit (ARDY) 38
register−access−ready interrupt flag bit (ARDY) 40

registers 31
repeat mode bit (RM) 32
repeated START condition 23
resetting/disabling the I2C module 30
RM bit of I2CMDR 32
RRDY bit of I2CIER 38
RRDY bit of I2CSTR 40
RSFULL bit of I2CSTR 40

S

SAR bits of I2CSAR 50
serial data formats 21
slave address register (I2CSAR) 50
slave direction bit of I2CSTR 40
slave−receiver mode 19
slave−transmitter mode 19
START and STOP conditions 20
START byte mode bit (STB) 32
START condition bit (STT) 32
status register (I2CSTR) 40
STB bit of I2CMDR 32
STOP condition 20
STOP condition bit (STP) 32
STP bit of I2CMDR 32
STT bit of I2CMDR 32

T

transmit data bits (DATA) 54
transmit FIFO interrupt enable bit of I2CFFTX 55
transmit FIFO interrupt flag bit of I2CFFTX 55
transmit FIFO interrupt flag clear bit of I2CFFTX 55
transmit FIFO interrupt level bits of I2CFFTX 55
transmit FIFO reset bit of I2CFFTX 55
transmit shift register (I2CXSR) 54
transmit shift register empty bit (XSMT) 40, 56
transmit status bits of I2CFFTX 55
transmit−data−ready interrupt enable bit (XRDY) 38
transmit−data−ready interrupt flag bit (XRDY) 40
transmitter mode bit (TRX) 32
TRX bit of I2CMDR 32

V

voltage levels 18

X

XA bit of I2CMDR 32
XRDY bit of I2CIER 38
XRDY bit of I2CSTR 40
XSMT bit of I2CSTR 40