TMS320VC5501/5502/5503/5507/5509 DSP Inter-Integrated Circuit (I2C) Module Reference Guide

SPRU146D
October 2005

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 (I2C) module that is available on the TMS320VC5501, TMS320VC5502, TMS320VC5503, TMS320VC5507, TMS320VC5509, and TMS320VC5509A digital signal processors (DSPs) in the TMS320C55x™ (C55x™) DSP generation. The I2C module provides an interface between one of these C55x 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 TMS320C55x is often abbreviated as C55x.
- 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 C55x devices and related support tools. Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

TMS320VC5501 Fixed-Point Digital Signal Processor Data Manual (literature number SPRS206) describes the features of the TMS320VC5501 fixed-point DSP and provides signal descriptions, pinouts, electrical specifications, and timings for the device.

TMS320VC5502 Fixed-Point Digital Signal Processor Data Manual (literature number SPRS166) describes the features of the TMS320VC5502 fixed-point DSP and provides signal descriptions, pinouts, electrical specifications, and timings for the device.

TMS320VC5503 Fixed-Point Digital Signal Processor Data Manual (literature number SPRS245) describes the features of the TMS320VC5503 fixed-point DSP and provides signal descriptions, pinouts, electrical specifications, and timings for the device.

TMS320VC5507 Fixed-Point Digital Signal Processor Data Manual (literature number SPRS244) describes the features of the TMS320VC5507 fixed-point DSP and provides signal descriptions, pinouts, electrical specifications, and timings for the device.

TMS320VC5509 Fixed-Point Digital Signal Processor Data Manual (literature number SPRS163) describes the features of the TMS320VC5509 fixed-point DSP and provides signal descriptions, pinouts, electrical specifications, and timings for the device.

TMS320VC5509A Fixed-Point Digital Signal Processor Data Manual (literature number SPRS205) describes the features of the TMS320VC5509A fixed-point DSP and provides signal descriptions, pinouts, electrical specifications, and timings for the device.

TMS320C55x Technical Overview (literature number SPRU393) introduces the TMS320C55x DSPs, the latest generation of fixed-point DSPs in the TMS320C5000™ DSP platform. Like the previous generations, this processor is optimized for high performance and low-power operation. This book describes the CPU architecture, low-power enhancements, and embedded emulation features.

TMS320C55x DSP CPU Reference Guide (literature number SPRU371) describes the architecture, registers, and operation of the CPU for the TMS320C55x DSPs.

TMS320C55x DSP Peripherals Reference Guide (literature number SPRU317) introduces the peripherals, interfaces, and related hardware that are available on TMS320C55x DSPs.

TMS320C55x DSP Algebraic Instruction Set Reference Guide (literature number SPRU375) describes the TMS320C55x DSP algebraic instructions individually. Also includes a summary of the instruction set, a list of the instruction opcodes, and a cross-reference to the mnemonic instruction set.

TMS320C55x DSP Mnemonic Instruction Set Reference Guide (literature number SPRU374) describes the TMS320C55x DSP mnemonic instructions individually. Also includes a summary of the instruction set, a list of the instruction opcodes, and a cross-reference to the algebraic instruction set.

TMS320C55x Optimizing C/C++ Compiler User's Guide (literature number SPRU281) describes the TMS320C55x C/C++ Compiler. This C/C++ compiler accepts ISO standard C and C++ source code and produces assembly language source code for TMS320C55x devices.

TMS320C55x Assembly Language Tools User's Guide (literature number SPRU280) describes the assembly language tools (assembler, linker, and other tools used to develop assembly language code), assembler directives, macros, common object file format, and symbolic debugging directives for TMS320C55x devices.

TMS320C55x DSP Programmer's Guide (literature number SPRU376) describes ways to optimize C and assembly code for the TMS320C55x DSPs and explains how to write code that uses special features and instructions of the DSPs.

Trademarks

Contents

Figures

Tables

I2C Module

This chapter describes the features and operation of the inter-integrated circuit (I2C) module that is available on the TMS320VC5501, TMS320VC5502, TMS320VC5503, TMS320VC5507, TMS320VC5509, and TMS320VC5509A digital signal processors (DSPs) in the TMS320C55x™ (C55x™) DSP generation. The I2C module provides an interface between one of these C55x 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. External components attached to this 2-wire serial bus can transmit/receive 1- to 8-bit data to/from the C55x DSP through the I2C module. This chapter assumes the reader is familiar with the I^2C-bus specification.

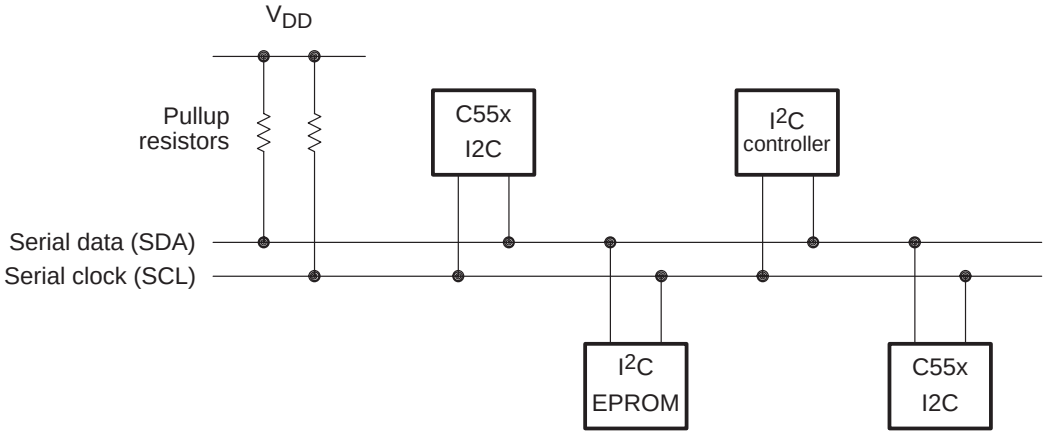
Note:

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.)

1 Introduction to the I2C Module

The I2C module supports any slave or master I^2C-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 read DMA event and one write DMA event, which can be used by the DMA controller
- 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.
- 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 C55x 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 I^2C-bus can be connected to the same I^2C-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 C55x device and other devices connected to the I^2C-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:

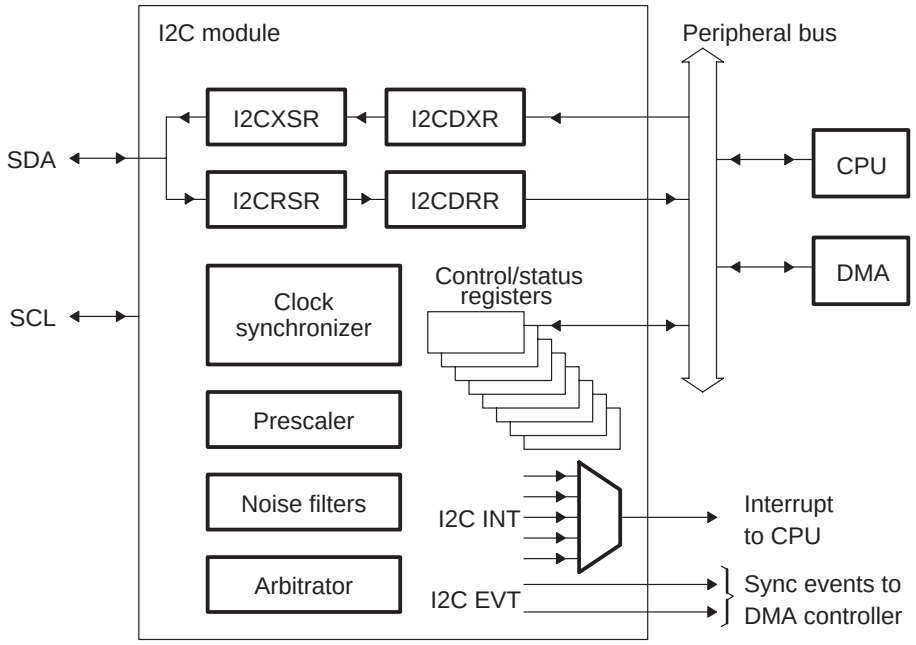
- Standard Mode: Send exactly n data values, where n is a value you program in an I2C module register. See Table 5 for register information.
- Repeat Mode: Keep sending data values until you use software to initiate a STOP condition or a new START condition. See Table 5 for RM bit information.

The I2C module consists of the following primary blocks:

- A serial interface: one data pin (SDA) and one clock pin (SCL)
- Data registers to temporarily hold receive data and transmit data traveling between the SDA pin and the CPU or the DMA controller
- Control and status registers
- A peripheral bus interface to enable the CPU and the DMA controller to access the I2C module registers
- 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
- DMA event generation logic, so that activity in the DMA controller can be synchronized to data reception and data transmission in the I2C module

Figure 2 shows the four registers used for transmission and reception. The CPU or the DMA controller 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.

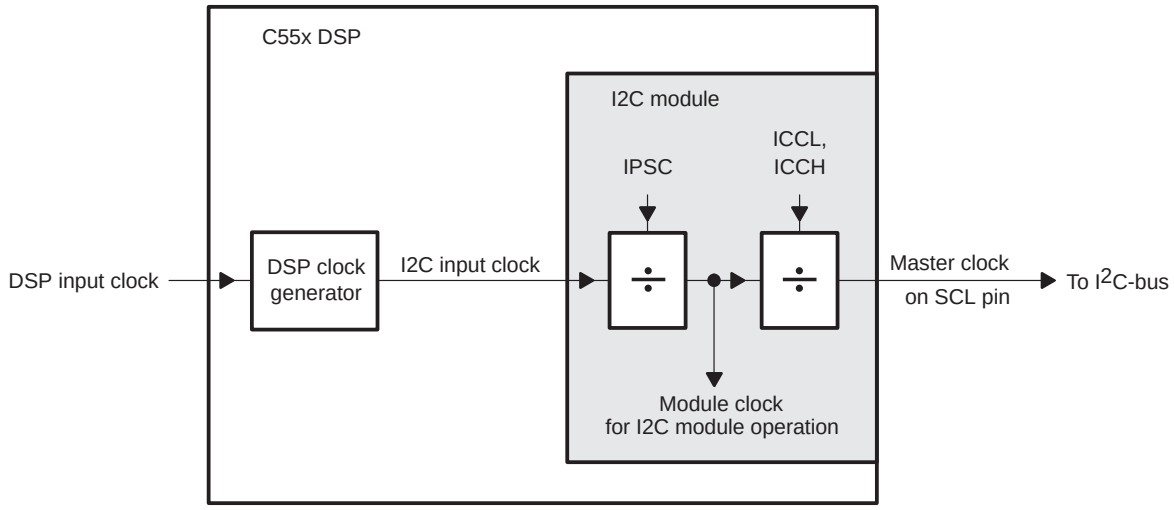
Figure 2. I2C Module Conceptual Block Diagram



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 can be equivalent to the CPU clock or it can be equivalent to the CPU clock divided by an integer, depending on the capabilities of the particular C55x DSP. The clock 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. The module clock must be in the range from 7 MHz to 12 MHz. This is necessary for proper operation. With the I2C module clock in this range, the noise filters on the SDA and SCL pins suppress noise that has a duration of 50 ns or shorter. 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 44). The resulting frequency is:

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

See the device-specific data manual for the supported range of values for the module clock frequency. 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 I^2C-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 5.7 (page 44).

2 I2C Module Operational Details

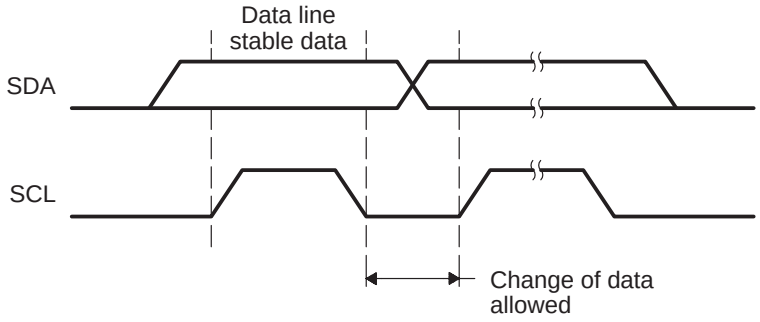
2.1 Input and Output Voltage Levels

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 I^2C-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 C55x 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 I^2C-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

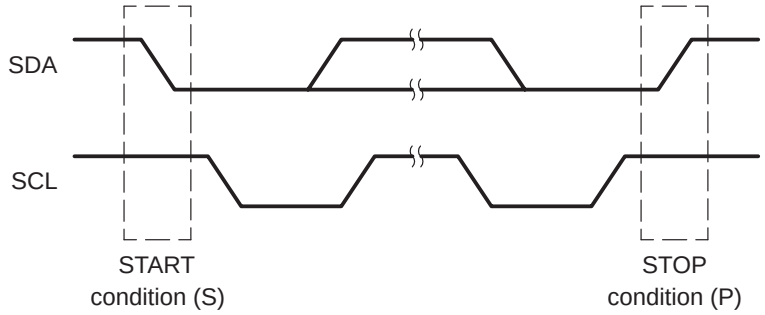
Operating Mode	Description
Slave-receiver mode	The I2C module is a slave and receives data from a master. All slaves begin in this mode. In this mode, serial data bits received on SDA are shifted in with the clock pulses that are generated by the master. As a slave, the I2C module does not generate the 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 I^2C-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 I^2C-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 28.

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)

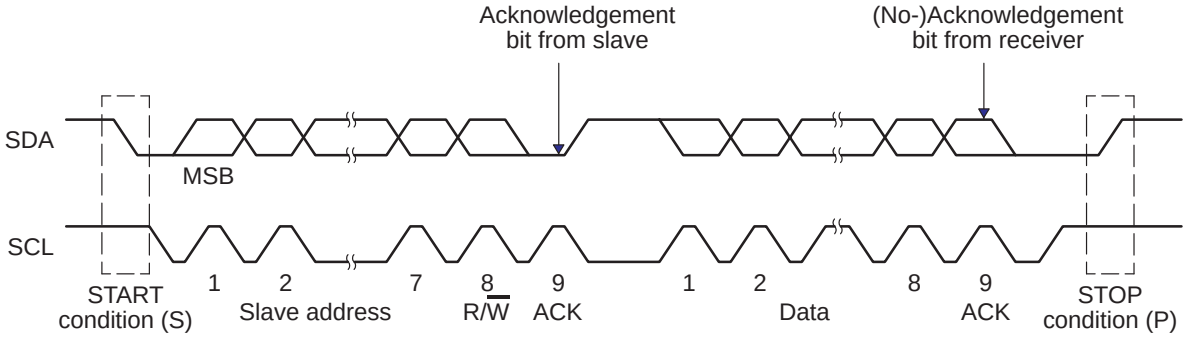
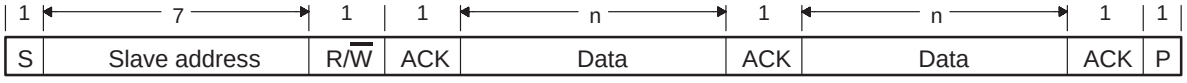
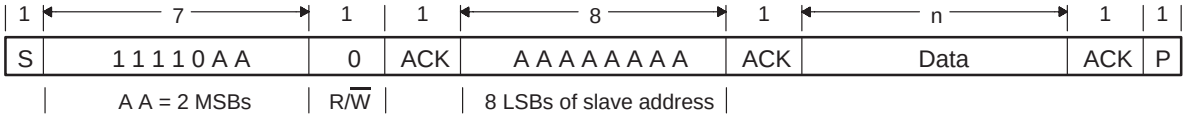


Figure 7. I2C Module 7-Bit Addressing Format (FDF = 0, XA = 0 in I2CMDR)



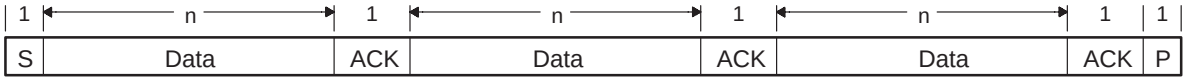
Note: n = The number of data bits (from 1 to 8) specified by the bit count (BC) field of I2CMDR.

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 18), 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 18) 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 I^2C-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

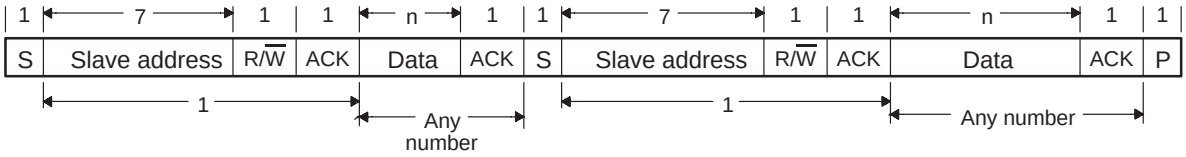
In this format (see Figure 9 on page 18), the first byte after a START condition (S) is a data byte. An ACK bit is inserted after each data byte, which can be from 1 to 8 bits, depending on the BC field of I2CMDR. No address or data-direction bit is sent. Therefore, the transmitter and the receiver must 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	Basic NACK Bit Generation Options	Additional Option (Not Supported on TMS320VC5503/5507/5509 DSPs)
Slave-receiver mode	❑ Disable data transfers (STT = 0 in I2CSTR). ❑ 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 (STOP = 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.

Table 2. Ways to Generate a NACK Bit (Continued)

I2C Module Condition	Basic NACK Bit Generation Options	Additional Option (Not Supported on TMS320VC5503/5507/5509 DSPs)
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

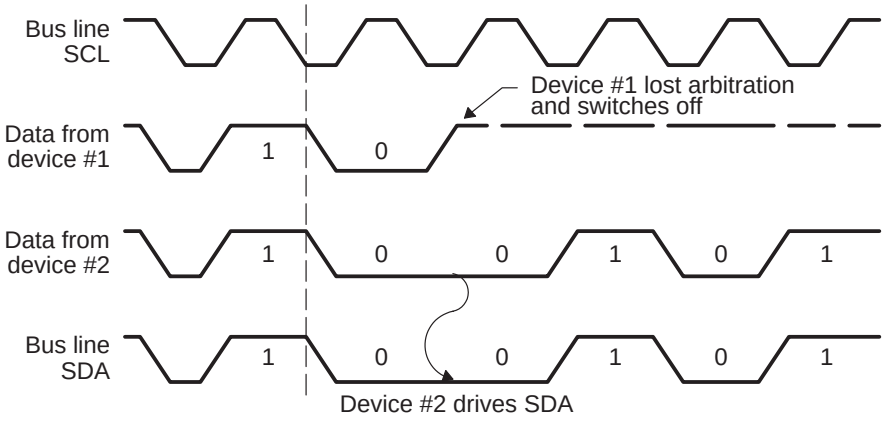
If two or more master-transmitters simultaneously start a transmission on the same bus, an arbitration procedure is invoked. The arbitration procedure uses the data presented on the serial data bus (SDA) by the competing transmitters. Figure 11 illustrates the arbitration procedure between two devices.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

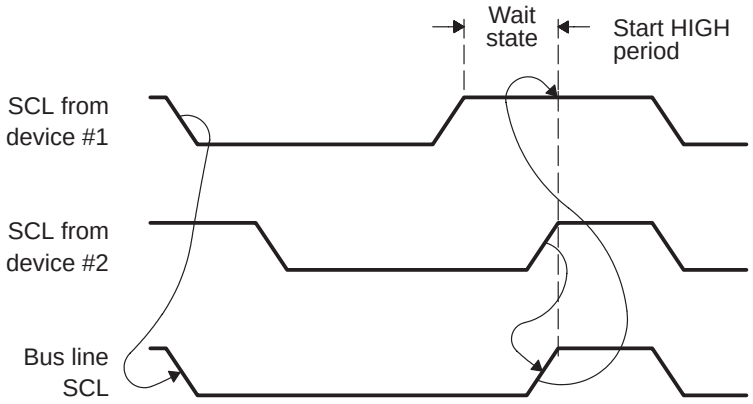


2.8 Clock Synchronization

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 and DMA Events Generated by the I2C Module

The I2C module can generate five types of 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 DMA controller to handle transmit and receive data, you can use the two DMA events described in section 3.2.

3.1 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 (ISR). The ISR for the I2C interrupt can determine the interrupt source by reading the interrupt source register, I2CISRC (see page 43). Then the ISR can branch to the appropriate subroutine.

After the CPU reads I2CISRC, the following events occur:

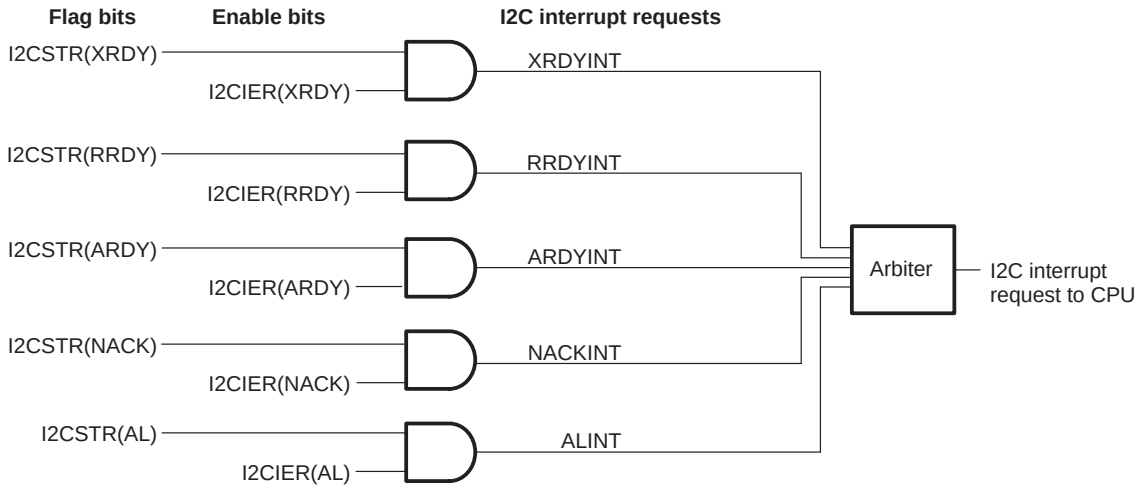
1) The flag for the source interrupt is cleared in I2CSTR. *Exception:* The ARDY, RRDY, and XRDY bits in I2CSTR are not cleared when I2CISRC is read. To clear one of these bits, write a 1 to it.

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 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 (see page 38).
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, which is described in section 5.4 (page 38). 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.

Figure 13. Enable Paths of the I2C Interrupt Requests



3.2 I2C Module DMA Events

The I2C module generates the following two DMA events. Activity in DMA channels can be synchronized to these events.

- Receive event (REVT): When receive data has been copied from the receive shift register (I2CRSR) to the data receive register (I2CDRR), the I2C module sends an REVT signal to the DMA controller. In response, the DMA controller can read the data from I2CDRR.
- Transmit event (XEVT): When transmit data has been copied from the data transmit register (I2CDXR) to the transmit shift register (I2CXSR), the I2C module sends an XEVT signal to the DMA controller. In response, the DMA controller can write the next transmit data value to I2CDXR.

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 $\overline{\text{RESET}}$ pin low. The entire DSP is reset and is held in the reset state until you drive the pin high. When $\overline{\text{RESET}}$ 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 at 16-bit addresses in I/O space. For the addresses, see the data manual for your particular TMS320C55x DSP.

Table 4. I2C Module Registers

Name	Description	See...
I2CMDR	I2C mode register	Page 28
I2CMDR2	I2C mode register 2	Page 35
I2CIER	I2C interrupt enable register	Page 36
I2CSTR	I2C status register	Page 38
I2CISRC	I2C interrupt source register	Page 43
I2CPSC	I2C prescaler register	Page 44
I2CCLKL	I2C clock low-time divider register	Page 44
I2CCLKH	I2C clock high-time divider register	Page 44
I2CSAR	I2C slave address register	Page 46
I2COAR	I2C own address register	Page 47
I2CCNT	I2C data count register	Page 48
I2CDRR	I2C data receive register	Page 49
I2CRSR	I2C receive shift register (not accessible to the CPU)	Page 49 (see I2CDRR description)
I2CDXR	I2C data transmit register	Page 49
I2CXSR	I2C transmit shift register (not accessible to the CPU)	Page 49 (see I2CDXR description)

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	IDLEEN‡	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

† NACKMOD is not available in TMS320VC5503/5507/5509 DSPs. In these devices, writing to bit 15 has no effect and reading returns 0.

‡ IDLEEN is in I2CMDR *only* in TMS320VC5509 DSPs, except for TMS320VC5503/5507/5509A. In other TMS320C55x DSPs, writing to bit 12 has no effect and reading returns 0. Also, in other TMS320C55x DSPs, the idle control is provided through a peripheral idle control register (see the data manual). To idle the I2C peripheral in the 5503/5507/5509A device, please see section 5.2 for the I2C idle enable bit in the I2C Mode Register 2 (I2CMDR2).

Table 5. I2C Mode Register (I2CMDR) Field Descriptions

Bit	Field	Value	Description
15	NACKMOD†		NACK mode bit (not available in TMS320VC5503/5507/5509 DSPs). 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 the 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.

† NACKMOD is not available in TMS320VC5503/5507/5509 DSPs. In these devices, bit 15 is reserved.

‡ IDLEEN is in I2CMDR *only* in TMS320VC5509 DSPs, except TMS320VC5503/5507/5509A. In other TMS320C55x DSPs, writing to bit 12 has no effect and reading returns 0. Also, in other TMS320C55x DSPs, the idle control is provided through a peripheral idle control register (see the data manual). To idle the I2C peripheral in the 5503/5507/5509A device, please see section 5.2 for the I2C idle enable bit in the I2C Mode Register 2 (I2CMDR2).

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 34). 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. In the slave mode, if STT is 0, the I2C module does not monitor the bus for commands from a master. As a result, the I2C module performs no data transfers.
		1	In the master mode, setting STT to 1 causes the I2C module to generate a START condition on the I^2C-bus. In the slave mode, if STT is 1, the I2C module monitors the bus and transmits/receives data in response to commands from a master.

† NACKMOD is not available in TMS320VC5503/5507/5509 DSPs. In these devices, bit 15 is reserved.
‡ IDLEEN is in I2CMDR *only* in TMS320VC5509 DSPs, except TMS320VC5503/5507/5509A. In other TMS320C55x DSPs, writing to bit 12 has no effect and reading returns 0. Also, in other TMS320C55x DSPs, the idle control is provided through a peripheral idle control register (see the data manual). To idle the I2C peripheral in the 5503/5507/5509A device, please see section 5.2 for the I2C idle enable bit in the I2C Mode Register 2 (I2CMDR2).

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

Bit	Field	Value	Description
12	IDLEEN‡		Idle enable bit
		0	The I2C module cannot be affected when the CPU executes an IDLE instruction.
		1	Setting this bit allows the I2C module to be deactivated by an IDLE instruction. If the PERIPH domain has been deactivated by an IDLE instruction (PERIS = 1 in the idle status register), the I2C module is inactive.
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 34). 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.
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 34 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.

† NACKMOD is not available in TMS320VC5503/5507/5509 DSPs. In these devices, bit 15 is reserved.
‡ IDLEEN is in I2CMDR *only* in TMS320VC5509 DSPs, except TMS320VC5503/5507/5509A. In other TMS320C55x DSPs, writing to bit 12 has no effect and reading returns 0. Also, in other TMS320C55x DSPs, the idle control is provided through a peripheral idle control register (see the data manual). To idle the I2C peripheral in the 5503/5507/5509A device, please see section 5.2 for the I2C idle enable bit in the I2C Mode Register 2 (I2CMDR2).

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

Bit	Field	Value	Description
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 34).
		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.
6	DLB		Digital loopback mode bit. The effects of this bit are shown in Figure 15 (page 35).
		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) $\times$ 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.

† NACKMOD is not available in TMS320VC5503/5507/5509 DSPs. In these devices, bit 15 is reserved.

‡ IDLEEN is in I2CMDR *only* in TMS320VC5509 DSPs, except TMS320VC5503/5507/5509A. In other TMS320C55x DSPs, writing to bit 12 has no effect and reading returns 0. Also, in other TMS320C55x DSPs, the idle control is provided through a peripheral idle control register (see the data manual). To idle the I2C peripheral in the 5503/5507/5509A device, please see section 5.2 for the I2C idle enable bit in the I2C Mode Register 2 (I2CMDR2).

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

Bit	Field	Value	Description
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 I^2C-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.
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 18). **Note:** The free data format is not supported in the digital loopback mode (DLB = 1).

† NACKMOD is not available in TMS320VC5503/5507/5509 DSPs. In these devices, bit 15 is reserved.
‡ IDLEEN is in I2CMDR *only* in TMS320VC5509 DSPs, except TMS320VC5503/5507/5509A. In other TMS320C55x DSPs, writing to bit 12 has no effect and reading returns 0. Also, in other TMS320C55x DSPs, the idle control is provided through a peripheral idle control register (see the data manual). To idle the I2C peripheral in the 5503/5507/5509A device, please see section 5.2 for the I2C idle enable bit in the I2C Mode Register 2 (I2CMDR2).

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

Bit	Field	Value	Description
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

† NACKMOD is not available in TMS320VC5503/5507/5509 DSPs. In these devices, bit 15 is reserved.
‡ IDLEEN is in I2CMDR *only* in TMS320VC5509 DSPs, except TMS320VC5503/5507/5509A. In other TMS320C55x DSPs, writing to bit 12 has no effect and reading returns 0. Also, in other TMS320C55x DSPs, the idle control is provided through a peripheral idle control register (see the data manual). To idle the I2C peripheral in the 5503/5507/5509A device, please see section 5.2 for the I2C idle enable bit in the I2C Mode Register 2 (I2CMDR2).

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

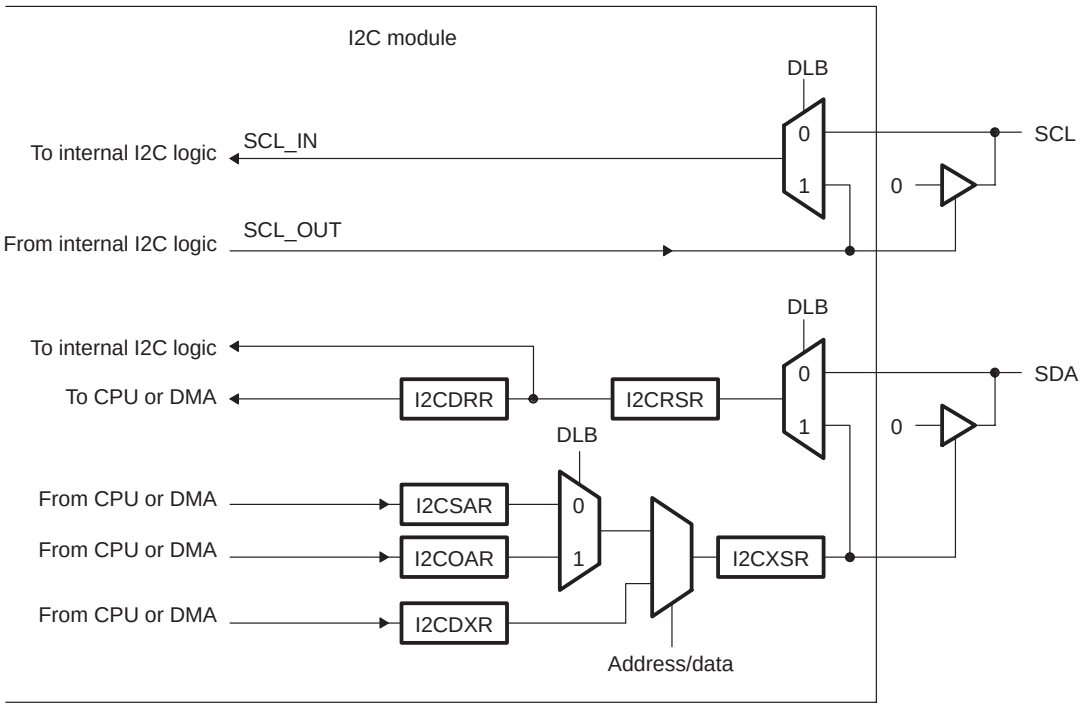
RM	STT	STP	Bus Activity†	Description
0	0	0	None	No activity
0	0	1	P	STOP condition
0	1	0	S-A-D..(n)..D	START condition, slave address, n data bytes (n = value in I2CCNT)
0	1	1	S-A-D..(n)..D-P	START condition, slave address, n data bytes, STOP condition (n = value in I2CCNT)
1	0	0	None	No activity
1	0	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 Mode Register 2 (I2CMDR2) (For TMS320VC5503/5507/5509A only)

The bits of I2CMDR2 are shown and described in Figure 16 and Table 8, respectively. The I2C idle enable bit is mapped at 0x3C0F in the reserved area of I2C, and disables access to all I2C module registers except I2CMDR2.

Figure 16. I2C Mode Register 2 (I2CMDR2) (TMS320VC5503/5507/5509A only)

31	1	0
Reserved		IDLEEN
R-0		R/W

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

Table 8. I2C Mode Register 2 (I2CMDR2) Field Descriptions (TMS320VC5503/5507/5509A only)

Bit	Field	Value	Description
31-1	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
0	IDLEEN		I2C idle enable bit
		0	The I2C does not go into an IDLE state when the DSP executes the IDLE instruction.
		1	The I2C does go into an IDLE state when the DSP executes the IDLE instruction. This action disables access to all I2C module registers except I2CMDR2.

5.3 I2C Interrupt Enable Register (I2CIER)

I2CIER, a 16-bit register mapped to the I/O space of the DSP, is used by the CPU to individually enable or disable I2C interrupt requests. The bits of I2CIER are shown and described in Figure 17 and Table 9, respectively.

Figure 17. I2C Interrupt Enable Register (I2CIER)

15	8
Reserved	
R-0	

7	5	4	3	2	1	0
Reserved		XRDY	RRDY	ARDY	NACK	AL
R-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 9. I2C Interrupt Enable Register (I2CIER) Field Descriptions

Bit	Field	Value	Description
15-5	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
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

5.4 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 18 and Table 10, respectively.

Figure 18. I2C Status Register (I2CSTR)

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

7	5	4	3	2	1	0
Reserved		XRDY	RRDY	ARDY	NACK	AL
R-0		R/W1C-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); -*n* = Value after reset

† NACKSNT is not available in TMS320VC5503/5507/5509 DSPs. In these devices, writing to bit 13 has no effect and reading returns 0.

Table 10. I2C Status Register (I2CSTR) Field Descriptions

Bit	Field	Value	Description
15-14	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
13	NACKSNT†		NACK sent bit (not available in TMS320VC5503/5507/5509 DSPs). 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 DSP is reset).
		1	NACK sent: A no-acknowledge bit was sent during the acknowledge cycle on the I^2C-bus.

† NACKSNT is not available in TMS320VC5503/5507/5509 DSPs. In these devices, bit 13 is reserved.

Table 10. I2C Status Register (I2CSTR) Field Descriptions (Continued)

Bit	Field	Value	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.

† NACKSNT is not available in TMS320VC5503/5507/5509 DSPs. In these devices, bit 13 is reserved.

Table 10. I2C Status Register (I2CSTR) Field Descriptions (Continued)

Bit	Field	Value	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.
7-5	Reserved		These reserved bit locations are always read as zeros. A value written to this field has no effect.
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 24).
		0	I2CDXR not ready. XRDY is cleared by one of the following events: ❑ Data is written to I2CDXR. ❑ XRDY is manually cleared. To clear this bit, write a 1 to it.
		1	I2CDXR ready: Data has been copied from I2CDXR to I2CXSR. XRDY is also forced to 1 when the I2C module is reset.
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 24).
		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.

† NACKSNT is not available in TMS320VC5503/5507/5509 DSPs. In these devices, bit 13 is reserved.

Table 10. I2C Status Register (I2CSTR) Field Descriptions (Continued)

Bit	Field	Value	Description
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 24).
		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.
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 24).
		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.

† NACKSNT is not available in TMS320VC5503/5507/5509 DSPs. In these devices, bit 13 is reserved.

Table 10. I2C Status Register (I2CSTR) Field Descriptions (Continued)

Bit	Field	Value	Description
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 24).
		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.

† NACKSNT is not available in TMS320VC5503/5507/5509 DSPs. In these devices, bit 13 is reserved.

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.

5.5 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 25).

Figure 19. I2C Interrupt Source Register (I2CISRC)

15 12	11 8	7 3	2 0
Reserved	TESTMD†	Reserved	INTCODE
R-0	R/W-0	R-0	R-0

Legend: R = Read; W = Write; -*n* = Value after reset
† The TESTMD bits must be kept 0 for proper operation of the I2C module.

Table 11. I2C Interrupt Source Register (I2CISRC) Field Descriptions

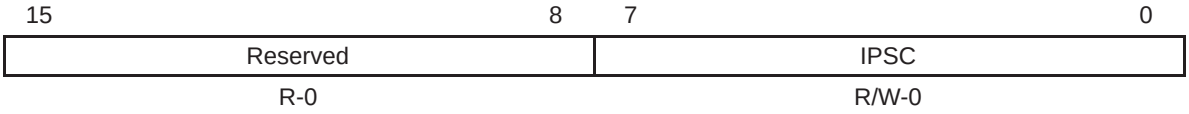
Bit	Field	Value	Description
15-12	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
11-8	TESTMD	0	Although there is no reason for you to write to I2CISRC, remember that bits 11-8 must remain 0 for proper operation of the I2C module. These bits are reserved for test purposes.
7-3	Reserved	0	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 which event 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
		110-111	Reserved

5.6 I2C Prescaler Register (I2CPSC)

The I2C prescaler register (I2CPSC) is a 16-bit I/O-mapped 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 13).

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 20. I2C Prescaler Register (I2CPSC)



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

Table 12. I2C Prescaler Register (I2CPSC) Field Descriptions

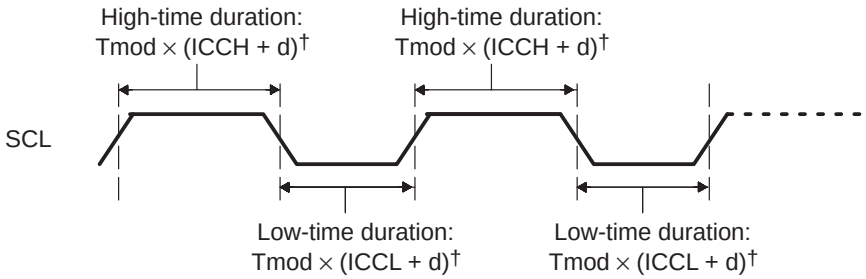
Bit	Field	Value	Description
15-8	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
7-0	IPSC	00h-FFh	I2C prescaler divide-down value. IPSC determines how much the CPU clock is divided to create the module clock of the I2C module: module clock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

5.7 I2C Clock Divider Registers (I2CCLKL and I2CCLKH)

As explained in section 1.4 (page 13), 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 21, the shape of the master clock depends on two divide-down values:

- ICCL in I2CCLKL (summarized by Figure 22 and Table 13). For each master clock cycle, ICCL determines the amount of time the signal is low.
- ICCH in I2CCLKH (summarized by Figure 23 and Table 14). For each master clock cycle, ICCH determines the amount of time the signal is high.

Figure 21. The Roles of the Clock Divide-Down Values (ICCL and ICCH)



† As described in section 5.7.1, Tmod is the module clock period, and d is 5, 6, or 7.

Figure 22. I2C Clock Low-Time Divider Register (I2CCLKL)

15	0
ICCL	
R/W-0	

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

Table 13. 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 5.7.1.

Figure 23. I2C Clock High-Time Divider Register (I2CCLKH)

15	0
ICCH	
R/W-0	

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

Table 14. 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 5.7.1.

5.7.1 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 15. IPSC is described in section 5.6.

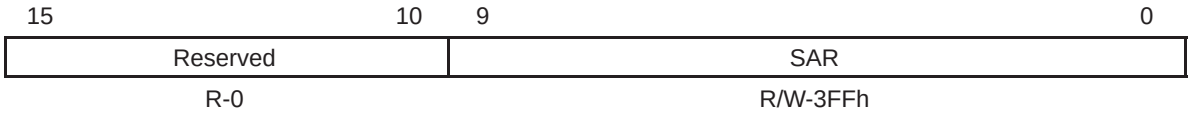
Table 15. Dependency of Delay d on the Divide-Down Value IPSC

IPSC	d
0	7
1	6
Greater than 1	5

5.8 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 24. As described in Table 16, 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 24. I2C Slave Address Register (I2CSAR)



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

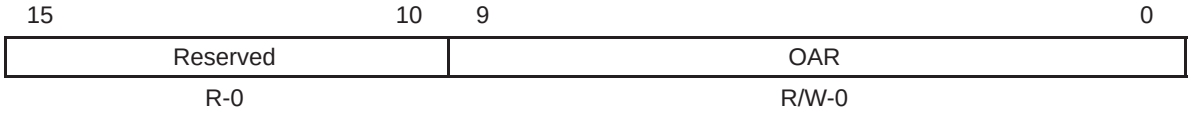
Table 16. I2C Slave Address Register (I2CSAR) Field Descriptions

Bit	Field	Value	Description
15-10	Reserved	0	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.

5.9 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 25 shows the format of I2COAR, and Table 17 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 I^2C-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 25. I2C Own Address Register (I2COAR)



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

Table 17. I2C Own Address Register (I2COAR) Field Descriptions

Bit	Field	Value	Description
15-10	Reserved	0	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.

5.10 I2C Data Count Register (I2CCNT)

I2CCNT is a 16-bit I/O-mapped register used to indicate how many data bytes to transfer when the I2C module is configured as a master-transmitter (MST = 1 and TRX = 1 in I2CMDR) and the repeat mode is off (RM = 0 in I2CMDR). In the repeat mode (RM = 1), I2CCNT is not used. The bits of I2CCNT are shown and described in Figure 26 and Table 18, 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 (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 26. I2C Data Count Register (I2CCNT)

15	0
ICDC	
R/W-0	

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

Table 18. I2C Data Count Register (I2CCNT) Field Description

Bit	Field	Value	Description
15-0	ICDC		Data count value. ICDC indicates the number of data bytes to transfer in the nonrepeat mode (RM = 0 in I2CMDR). 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.

5.11 I2C Data Receive Register (I2CDRR)

I2CDRR (see Figure 27 and Table 19) is a 16-bit I/O-mapped register used by the CPU or the DMA controller 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 and the DMA controller cannot access I2CRSR.

Figure 27. I2C Data Receive Register (I2CDRR)

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

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

Table 19. I2C Data Receive Register (I2CDRR) Field Description

Bit	Field	Value	Description
15-8	Reserved	0	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

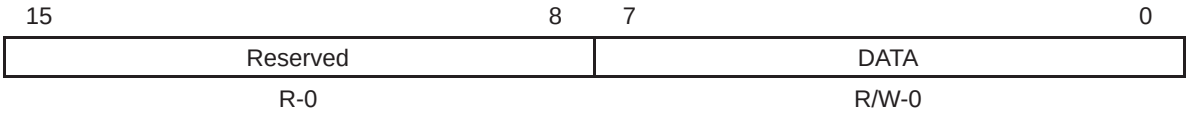
5.12 I2C Data Transmit Register (I2CDXR)

The CPU or the DMA controller writes transmit data to I2CDXR (see Figure 28 and Table 20). This 16-bit I/O-mapped 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 and the DMA controller cannot access I2CXSR.

Figure 28. I2C Data Transmit Register (I2CDXR)



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

Table 20. I2C Data Transmit Register (I2CDXR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	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

Revision History

This document was revised to SPRU146C from SPRU146B, which was released in December 2003.

The following changes were made in this revision:

Page	Additions/Modifications/Deletions
Global	Added the TMS320VC5503/5507 devices where applicable.
40	Changed description of the 0 Value for the AAS bit in Table 10 to: 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.

Index

I

M

N

O

P

R

S

T

V

X