TMS320C6000 DSP Inter-Integrated Circuit (I2C) Module Reference Guide

Literature Number: SPRU175D
March 2007

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
Low Power Wireless	www.ti.com/lpw	Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

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

Preface

Read This First

About This Manual

This document describes the inter−integrated circuit (I2C) module in the TMS320C6000™ DSP family. This document assumes the reader is familiar with the I2C-bus specification.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.
  - Registers in this document are shown in figures and described in tables.
  - Each register figure shows a rectangle divided into fields that represent the fields of the register.
  - Each field is labeled with its bit name, its beginning and ending bit numbers above, and its read/write properties below. A legend explains the notation used for the properties.
  - Reserved bits in a register figure designate a bit that is used for future device expansion.

Related Documentation From Texas Instruments

The following documents describe the C6000™ 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.

TMS320C6000 CPU and Instruction Set Reference Guide (literature number SPRU189) describes the TMS320C6000™ CPU architecture, instruction set, pipeline, and interrupts for these digital signal processors.

TMS320C6000 DSP Peripherals Overview Reference Guide (literature number SPRU190) describes the peripherals available on the TMS320C6000™ DSPs.

TMS320C6000 Technical Brief (literature number SPRU197) gives an introduction to the TMS320C62x™ and TMS320C67x™ DSPs, development tools, and third-party support.

TMS320C6000 Programmer's Guide (literature number SPRU198) describes ways to optimize C and assembly code for the TMS320C6000™ DSPs and includes application program examples.

TMS320C6000 Code Composer Studio Tutorial (literature number SPRU301) introduces the Code Composer Studio™ integrated development environment and software tools.

Code Composer Studio Application Programming Interface Reference Guide (literature number SPRU321) describes the Code Composer Studio™ application programming interface (API), which allows you to program custom plug-ins for Code Composer.

TMS320C6x Peripheral Support Library Programmer's Reference (literature number SPRU273) describes the contents of the TMS320C6000™ peripheral support library of functions and macros. It lists functions and macros both by header file and alphabetically, provides a complete description of each, and gives code examples to show how they are used.

TMS320C6000 Chip Support Library API Reference Guide (literature number SPRU401) describes a set of application programming interfaces (APIs) used to configure and control the on-chip peripherals.

Trademarks

Code Composer Studio, C6000, C62x, C64x, C67x, TMS320C6000, TMS320C62x, TMS320C64x, TMS320C67x, and VelociTI are trademarks of Texas Instruments.

Contents

Figures

Tables

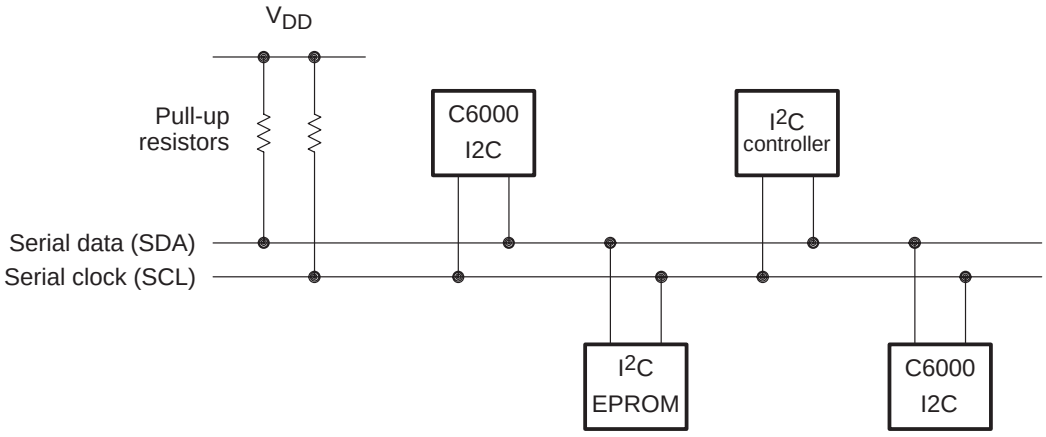
Inter-Integrated Circuit (I2C) Module

The inter-integrated circuit (I2C) module provides an interface between a TMS320C6000™ DSP and other 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 up to 8-bit data to/from the C6000™ DSP through the I2C module. To determine whether a particular C6000 DSP has an I2C module, see the data sheet for that DSP. This chapter assumes the reader is familiar with the I^2C-bus specification.

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



C6000 and TMS320C6000 are trademarks of Texas Instruments.

1.1 Features

The I2C module has the following features:

- Compliance with the Philips Semiconductors I^2C-bus specification (version 2.1):
  - Support for byte format transfer
  - 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 EDMA event and one write EDMA event, which can be used by the EDMA 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

2 Functional Overview

Each device connected to an I^2C-bus, including any C6000 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. In addition, a device connected to the I^2C-bus can also be considered as the master or the slave when performing data transfers. Note that 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.

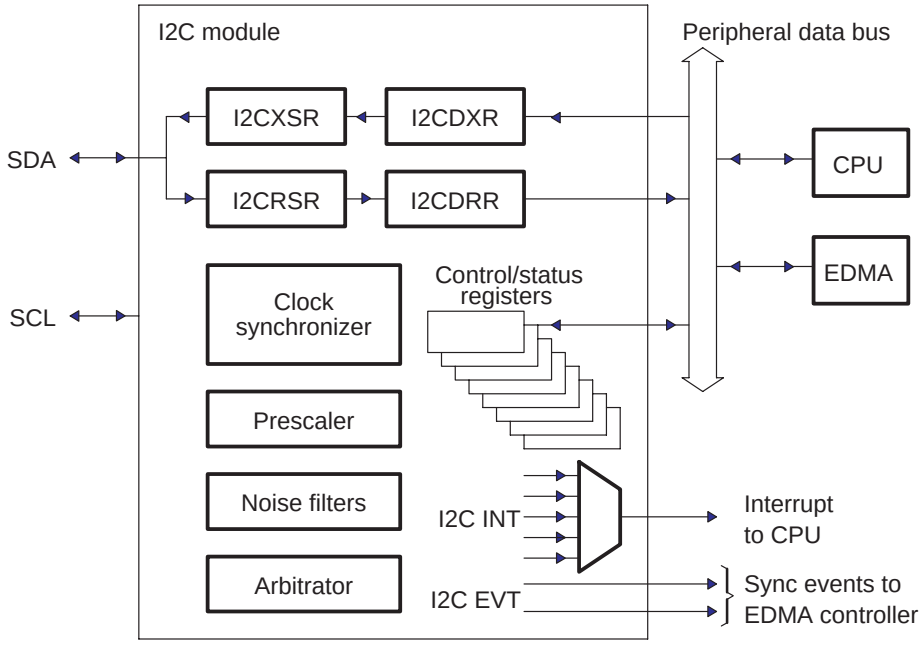
For data communication, the I2C module has a serial data pin (SDA) and a serial clock pin (SCL), as shown in Figure 2. These two pins carry information between the C6000 device and other devices connected to the I^2C-bus. The SDA and SCL pins both are bidirectional. They each must be connected to a positive 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.

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 EDMA controller
- Control and status registers
- A peripheral data bus interface to enable the CPU and the EDMA 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
- EDMA event generation logic, so that activity in the EDMA 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 EDMA 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



3 Operational Details

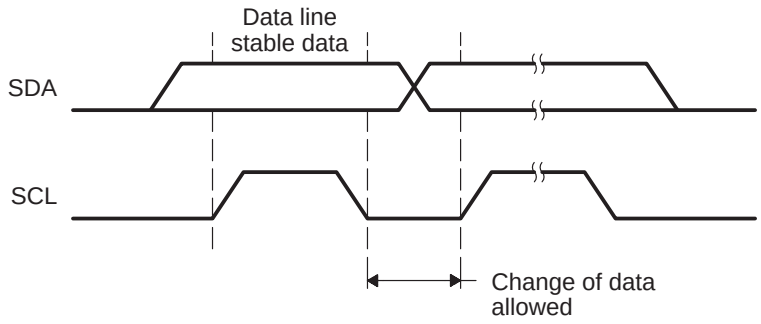
3.1 Input and Output Voltage Levels

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 device-specific datasheet.

3.2 Data Validity

The data on SDA must be stable during the high period of the clock (see Figure 3). The high or low state of the data line, SDA, can change only when the clock signal on SCL is low.

Figure 3. Bit Transfer on the I^2C-Bus



3.3 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. In order 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

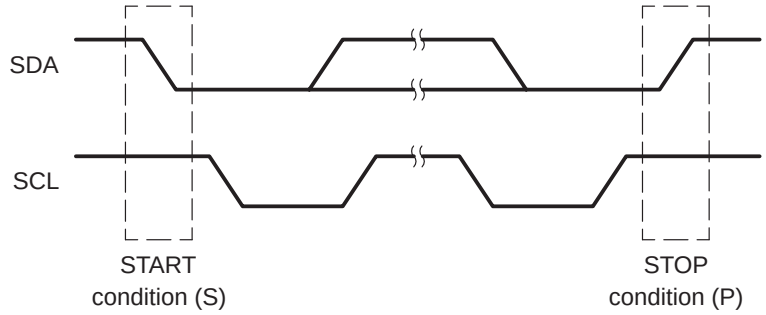
Operating Mode	Description
Slave-receiver mode	The I2C module is a slave and receives data from a master. All slave modules begin in this mode. In this mode, serial data bits received on SDA are shifted in with the clock pulses that are generated by the master. As a slave, the I2C module does not generate the clock signal, but it can hold SCL low while the intervention of the DSP is required (RSFULL = 1 in I2CSTR) after data has been received.
Slave-transmitter mode	The I2C module is a slave and transmits data to a master. This mode can only be entered 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 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 data has been transmitted.
Master-receiver mode	The I2C module is a master and receives data from a slave. This mode can only be entered 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 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 data has been received.
Master-transmitter mode	The I2C module is a master and transmits control information and data to a slave. All master modules 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 data has been transmitted.

3.4 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^{2}C-bus. As shown in Figure 4:

- 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 4. I2C Module START and STOP Conditions



After a START condition and before a subsequent STOP condition, the I^{2}C-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.

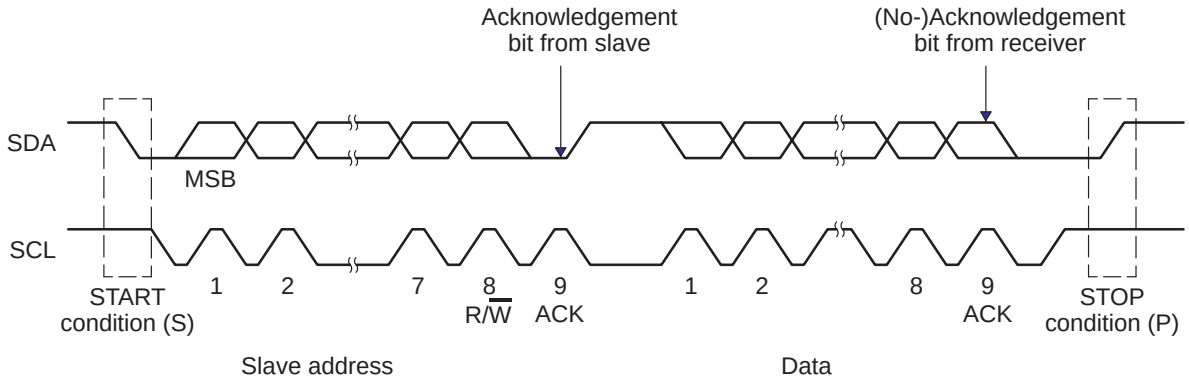
For the I2C module to start a data transfer with a START condition, the master mode (MST) bit and the START condition (STT) bit in I2CMDR must both be 1. For the I2C module to end a data transfer with a STOP condition, the STOP condition (STP) bit must be set to 1. When BB is set to 1 and STT is set to 1, a repeated START condition is generated. For a description of I2CMDR (including the MST, STT, and STP bits), see section 8.9.

3.5 Serial Data Formats

Figure 5 shows an example of a data transfer on the I2C-bus. The I2C module supports 1-bit to 8-bit data values. Figure 5 is shown in an 8-bit data format (BC = 000 in I2CMDR). Each bit put on the SDA line equates to 1 pulse on the SCL line and the data is always transferred with the most-significant bit (MSB) first. The number of data values that can be transmitted or received is unrestricted; however, the transmitters and receivers must agree on the number of data values being transferred. The I2C module supports the following data formats.

- 7-bit addressing mode
- 10-bit addressing mode
- Free data format mode

Figure 5. I2C Module Data Transfer



3.5.1 7-Bit Addressing Format

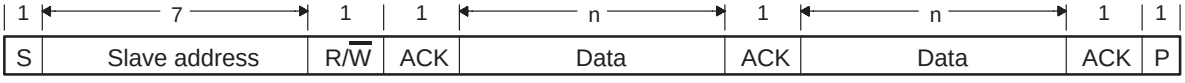
In the 7-bit addressing format (Figure 6), the first byte after a START condition (S) consists of a 7-bit slave address followed by a R/$\overline{W}$ bit. The R/$\overline{W}$ bit determines the direction of the data:

- 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 the R/$\overline{W}$ bit. If the ACK bit is inserted by the slave, 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) bits 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.

Figure 6. I2C Module 7-Bit Addressing Format (FDF = 0, XA = 0 in I2CMDR)



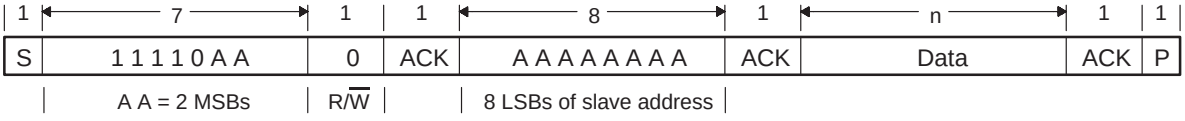
Note: n = The number of data bits (from 1 to 8) specified by the bit count (BC) field of I2CMDR.

3.5.2 10-Bit Addressing Format

The 10-bit addressing format (Figure 7) 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 (ACK) 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.

Figure 7. I2C Module 10-Bit Addressing Format With Master-Transmitter Writing to Slave-Receiver (FDF = 0, XA = 1 in I2CMDR)



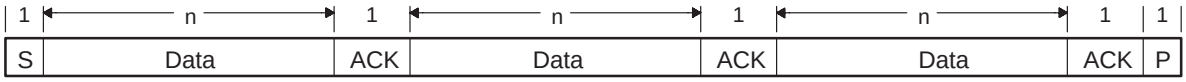
Note: n = The number of data bits (from 1 to 8) specified by the bit count (BC) field of I2CMDR.

3.5.3 Free Data Format

In the free data format (Figure 8), the first bits after a START condition (S) are a data word. An ACK bit is inserted after each data word, which can be from 1 to 8 bits, depending on BC 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.

Figure 8. I2C Module Free Data Format (FDF = 1 in I2CMDR)

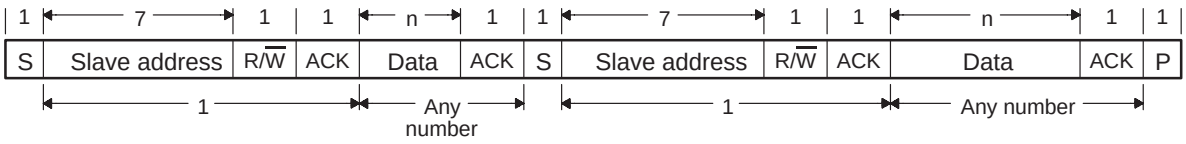


Note: n = The number of data bits (from 1 to 8) specified by the bit count (BC) field of I2CMDR.

3.5.4 Using a Repeated START Condition

The repeated START condition can be used with the 7-bit addressing, 10-bit addressing, and free data formats. The 7-bit addressing format using a repeated START condition (S) is shown in Figure 9. At the end of each data word, the master can drive another START condition. Using this capability, a master can transmit/receive any number of data words before driving a STOP condition. The length of a data word can be from 1 to 8 bits and is selected with BC of I2CMDR.

Figure 9. I2C Module 7-Bit Addressing Format With Repeated START Condition (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.

3.6 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 the I2C module sends a NACK bit.

Table 2. Ways to Generate a NACK Bit

	NACK Bit Generation	
I2C Module Condition	**Basic**	**Optional**
Slave-receiver mode	❑ Disable data transfers (STT = 0 in I2CMDR). ❑ 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 I2C MDR)	❑ 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.
Master-receiver mode AND Nonrepeat mode (RM = 0 in I2C MDR)	❑ If STP = 1 in I2CMDR, allow the internal data counter to count down to 0 and 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.

3.7 Arbitration

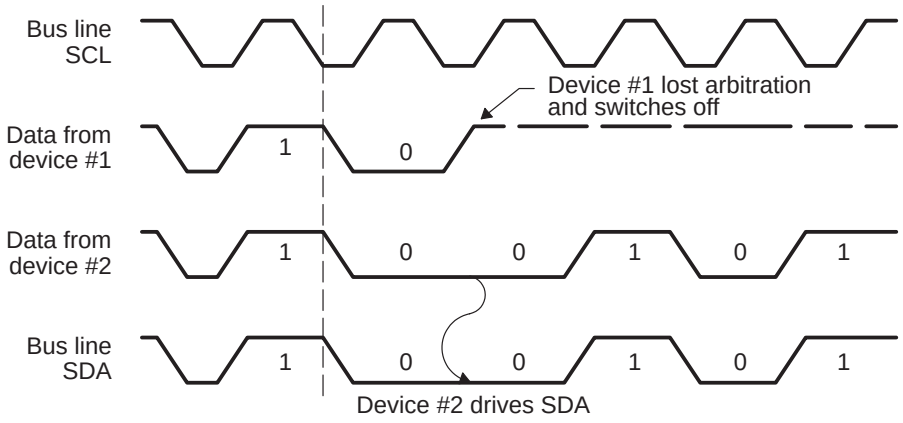
If two or more master-transmitters simultaneously start a transmission on the same bus, an arbitration procedure is invoked. The arbitration procedure uses the data presented on the serial data bus (SDA) by the competing transmitters. Figure 10 illustrates the arbitration procedure between two devices. 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.

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

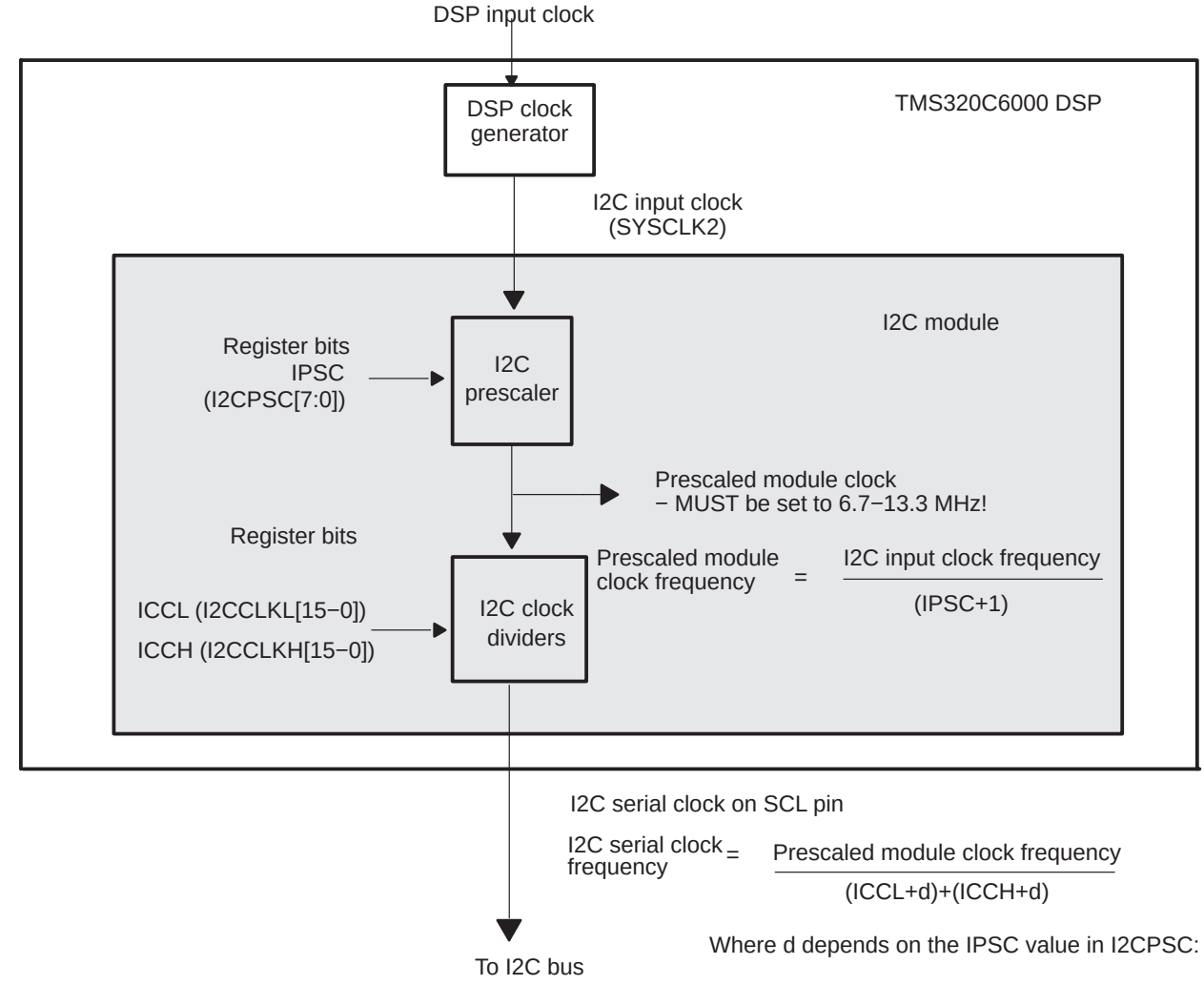
Figure 10. Arbitration Procedure Between Two Master-Transmitters



3.8 Clock Generation

The DSP clock generator receives a signal from an external clock source and produces an **I2C input clock** as shown in Figure 11 below. A programmable prescaler in the I2C module divides the I2C input clock down to produce a **prescaled module clock**. You **must** operate the prescaled module clock within the 6.7−13.3 MHz range. The I2C clock dividers divide the high and low portions of the prescaled module clock signal down to produce the **I2C serial clock**, which appears on the SCL pin when you configure the I2C module to be a master on the I2C bus.

Figure 11. I2C Input Clock



IPSC value	d
0	7
1	6
2h−FFh	5

CAUTION

Prescaled Module Clock Frequency Range

The I2C module must be operated with a prescaled module clock frequency of 6.7 to 13.3 MHz. The I2C prescaler register (I2CPSC) must be configured to this frequency range.

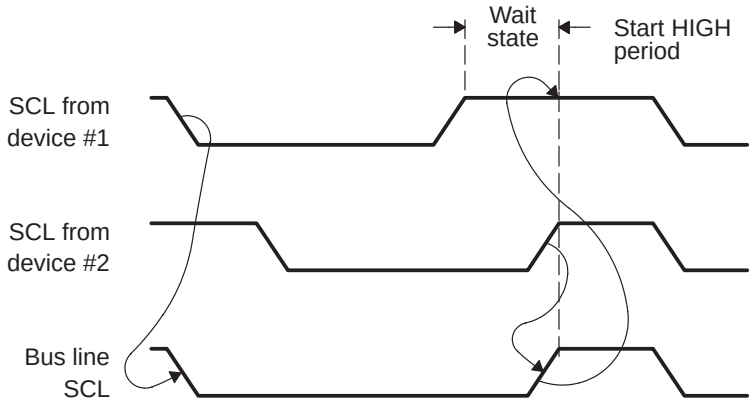
You must only initialize the prescaler (I2CPSC) while the I2C module is in the reset state (the IRS bit = 0 in the I2CMDR register). The prescaled frequency only takes effect when you change the IRS bit to 1. Changing the IPSC value while IRS = 1 has no effect. Likewise, you must configure the I2C clock dividers (I2CCLKL and I2CCLKH) while the I2C module is still in reset (the IRS bit = 0 in the I2CMDR register).

3.9 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 (device #1) 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 data word or to prepare a data word to be transmitted.

Figure 12. Synchronization of Two I2C Clock Generators During Arbitration



4 Interrupt Requests Generated by the I2C Module

The I2C module can generate the interrupt requests described in Table 3. Two of these requests indicate to the CPU when to write transmit data and when to read receive data. 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, I2CISR (described in section 8.10). Then the ISR can branch to the appropriate subroutine.

After the CPU reads I2CISR, the following events occur:

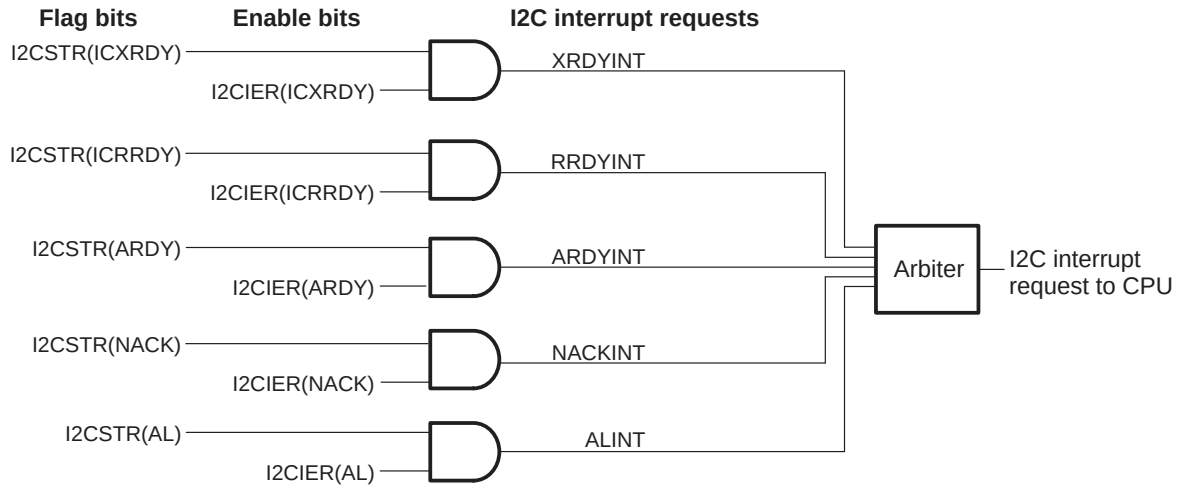
1) The flag for the source interrupt is cleared in I2CSTR. *Exception:* The ARDY, ICRRDY, and ICXRDY bits in I2CSTR are not cleared when I2CISR is read. To clear one of these bits, write a 1 to the corresponding bit.

2) The arbiter determines which of the remaining interrupt requests has the highest priority, writes the code for that interrupt to I2CISR, 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 ICXRDY bit of I2CSTR (described in section 8.3).
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 ICRRDY bit of I2CSTR (described in section 8.3).
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 (described in section 8.3). 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 (described in section 8.3).
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 (described in section 8.3).

Figure 13. Enable Paths of I2C Interrupt Requests



5 EDMA Events Generated by the I2C Module

For the EDMA controller to handle transmit and receive data, the I2C module generates the following two EDMA events. Activity in EDMA 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 EDMA controller. In response, the EDMA 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 EDMA controller. In response, the EDMA controller can write the next transmit data value to I2CDXR.

6 Resetting/Disabling the I2C Module

The I2C module can be reset/disabled 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.

7 Programming Guide

7.1 Main Program

7.1.1 State After Reset

1) Program the I2C prescaler register (I2CPSC) to obtain the desired prescaled module clock for the operation of the I2C module (see section 8.11). This value is to be calculated and is dependent on the CPU frequency.

2) Take the I2C module out of reset (IRS = 1).

   a) If using interrupt for transmit/receive data, enable the appropriate interrupt in I2CIER.

   b) If using the EDMA controller for transmit/receive data, enable the EDMA and program the EDMA controller.

7.1.2 Initialization Procedure

Configure the I2C mode register (I2CMDR) (see section 8.9).

7.1.3 Program Clock Control Registers (I2CCLKL and I2CCLKH)

Program the I2C prescaled module clock (I2CCLKL and I2CCLKH) to obtain the desired SCL clock timing (see section 8.4). These values are to be calculated and are dependent on the CPU frequency.

7.1.4 Configure Address Registers

1) Configure its own address register (I2COAR) (see section 8.1)

2) Configure the slave address register (I2CSAR) (see section 8.7)

7.1.5 Program Transmit Data Register (I2CDXR)

If in master-transmitter mode, program the data transmit register (I2CDXR) (see section 8.8).

7.1.6 Configure Status and Mode Register (I2CSTR)

Poll the bus busy (BB) bit in the I2C status register (I2CSTR), if it is cleared to 0 (bus not busy), configure START/STOP condition to initiate a transfer (see section 8.3).

7.1.7 Poll Receive Data

Poll the receive-data-ready interrupt flag bit (ICRRDY) in the I2C status register (I2CSTR), use the RRDY interrupt, or use the EDMA controller to read the receive data in the data receive register (I2CDRR).

7.1.8 Poll Transmit Data

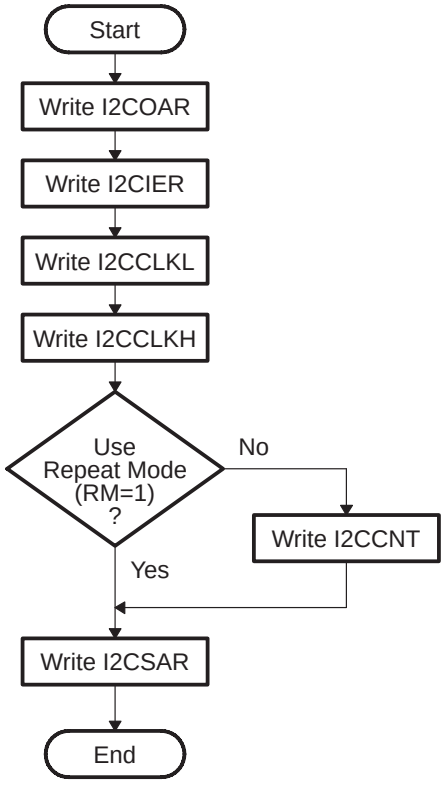
Poll the transmit data ready interrupt flag bit (ICXRDY) in the I2C status register (I2CSTR), use the XRDY interrupt, or use the EDMA controller to write data into the data transmit register (I2CDXR).

7.2 Interrupt Subroutines

1) Test for arbitration lost and resolve accordingly.
2) Test for no-acknowledge and resolve accordingly.
3) Test for register access ready and resolve accordingly.
4) Test for receive data and resolve accordingly.
5) Test for transmit data and resolve accordingly.

7.3 Flow Diagrams

Figure 14. Setup Procedure



7.3.1 Bit Polling Methods to Control Data Flow

Figure 15. Using Bit Polling for Master-Transmitter Operation, Repeat Mode (RM = 1)

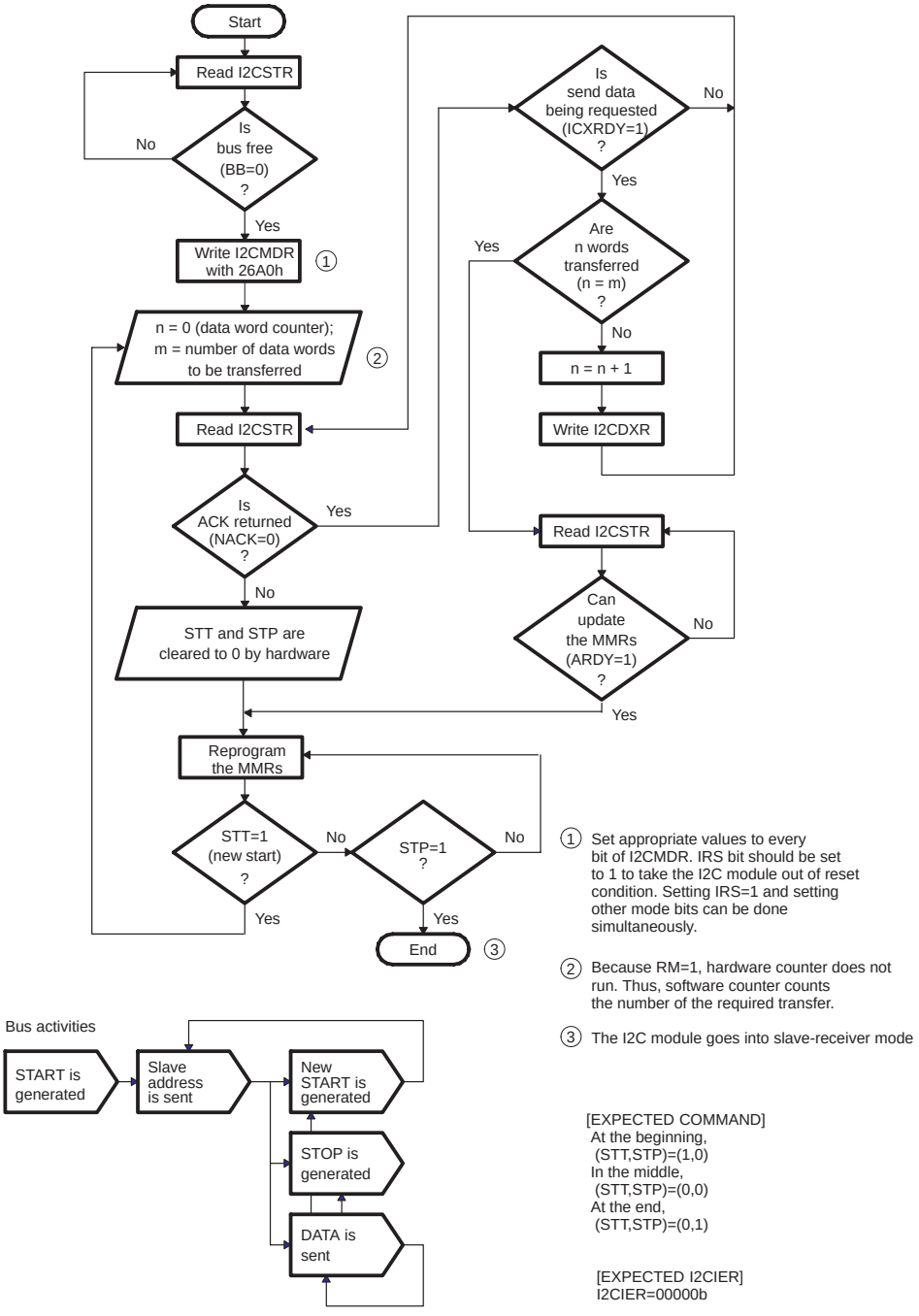


Figure 16. Using Bit Polling for Master-Transmitter Operation, Nonrepeat Mode (RM = 0)

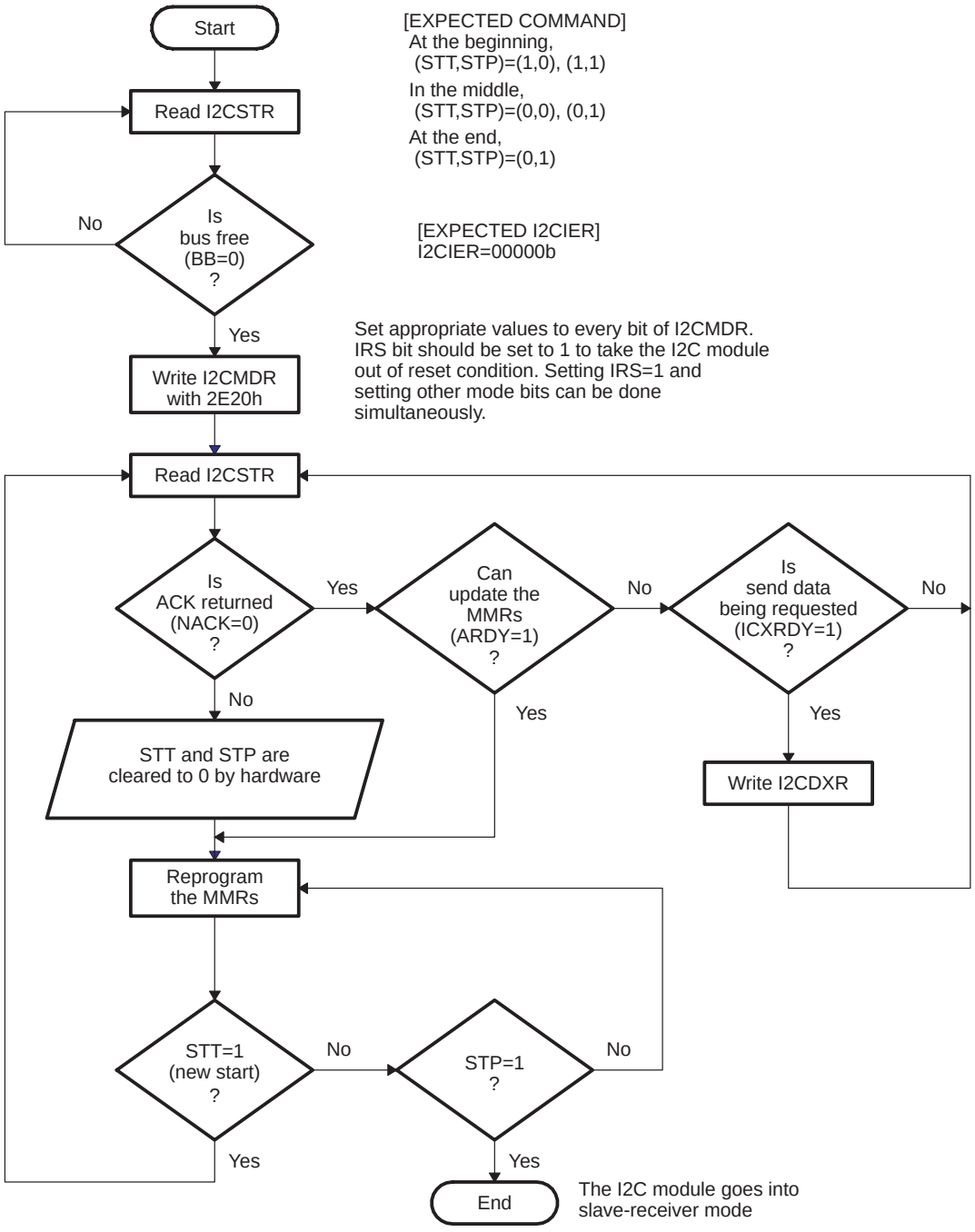
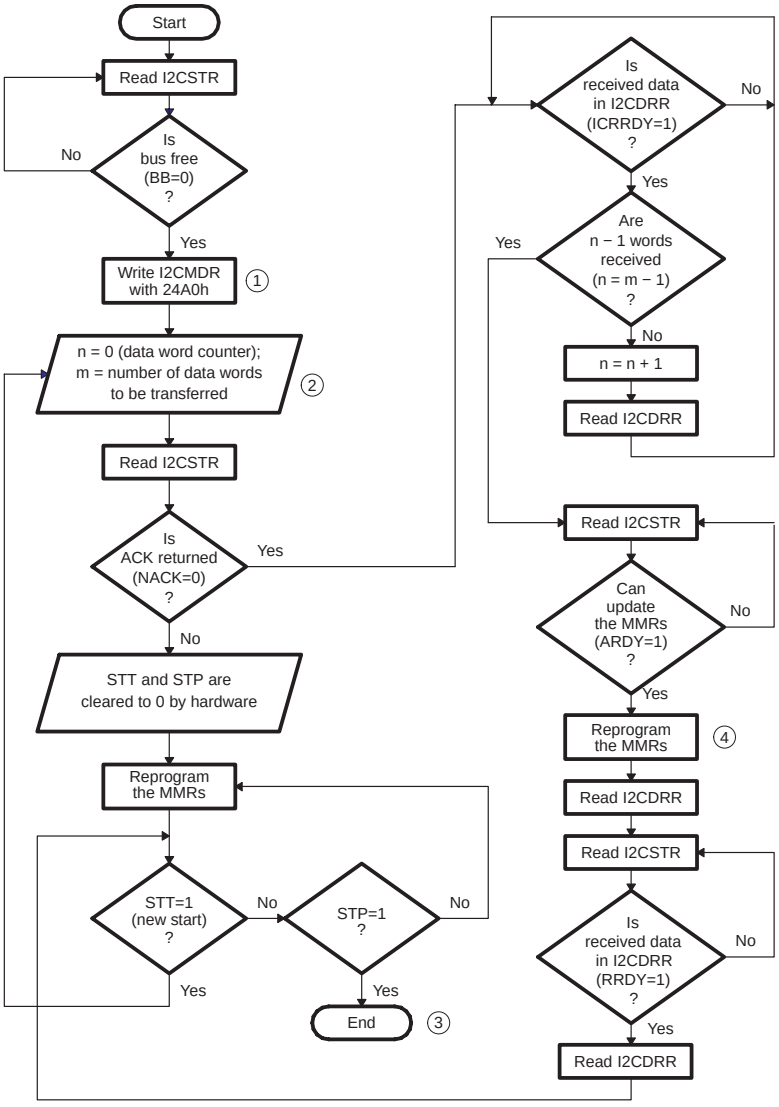


Figure 17. Using Bit Polling for Master-Receiver Operation, Repeat Mode (RM = 1), Fixed Number of Data Words

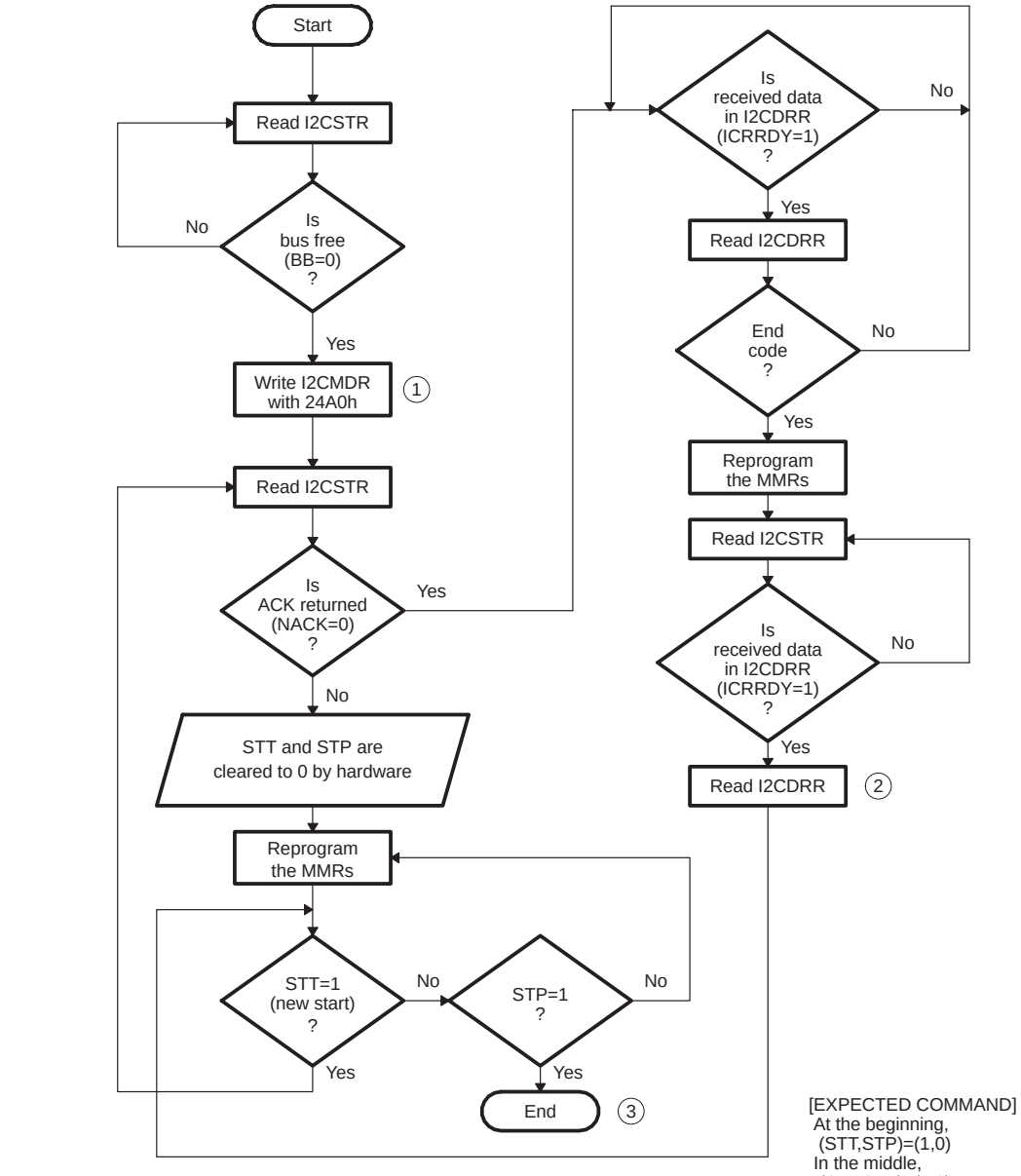


1. Set appropriate values to every bit of I2CMDR. IRS bit should be set to 1 to take the I2C module out of reset condition. Setting IRS=1 and setting other mode bits can be done simultaneously.
2. Because RM=1, hardware counter does not run. Thus, software counter counts the number of the required transfer.
3. The I2C module goes into slave-receiver mode
4. Set STP=1

[EXPECTED COMMAND]
At the beginning,
(STT,STP)=(1,0)
In the middle,
(STT,STP)=(0,0)
At the end,
(STT,STP)=(0,1)

[EXPECTED I2CIER]
I2CIER=00000b

Figure 18. Using Bit Polling for Master-Receiver Operation, Repeat Mode (RM = 1), Variable (Data-Dependent) Number of Data Words



1. Set appropriate values to every bit of I2CMDR. IRS bit should be set to 1 to take the I2C module out of reset condition. Setting IRS=1 and setting other mode bits can be done simultaneously.
2. Dummy read. The contents of this read data has no meaning.
3. The I2C module goes into slave-receiver mode

Figure 19. Using Bit Polling for Master-Receiver Operation, Nonrepeat Mode (RM = 0)

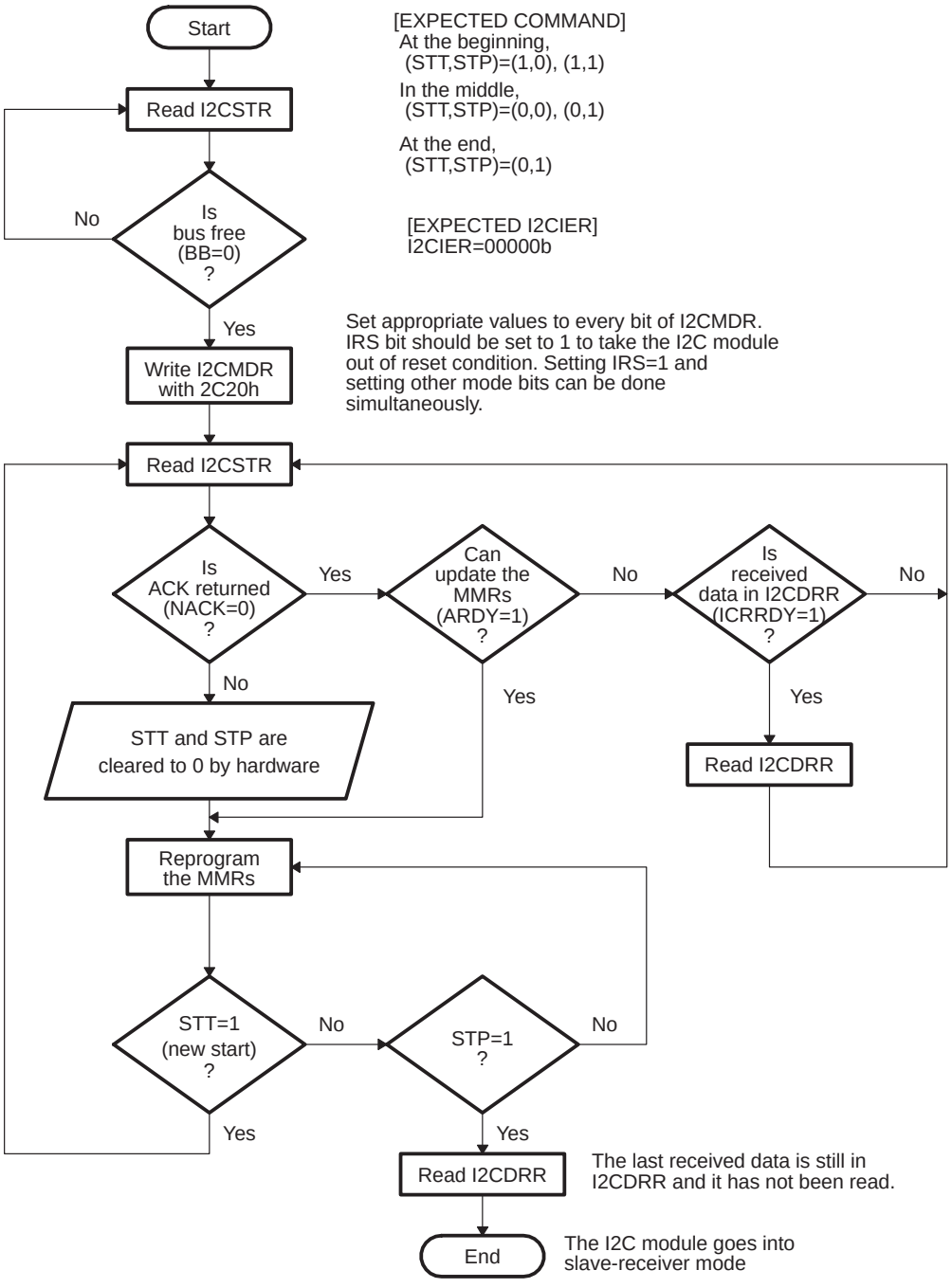
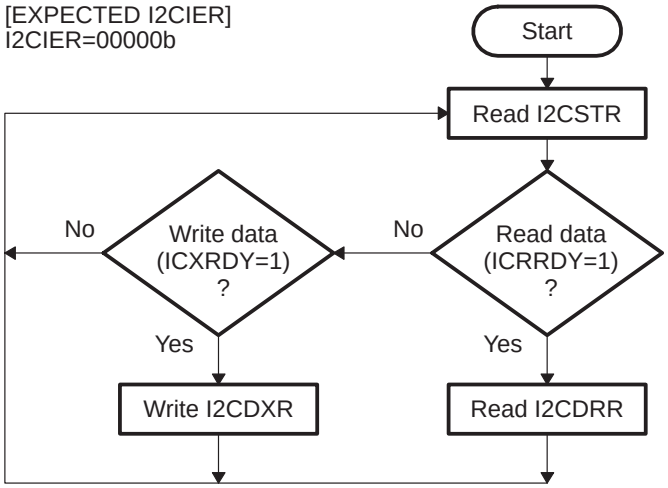


Figure 20. Using Bit Polling for Slave-Transmitter/Receiver Operation, Repeat Mode (RM = 1)



7.3.2 Interrupts Methods to Control Data Flow

Figure 21. Using Interrupts for Master-Transmitter Operation, Nonrepeat Mode (RM = 0)

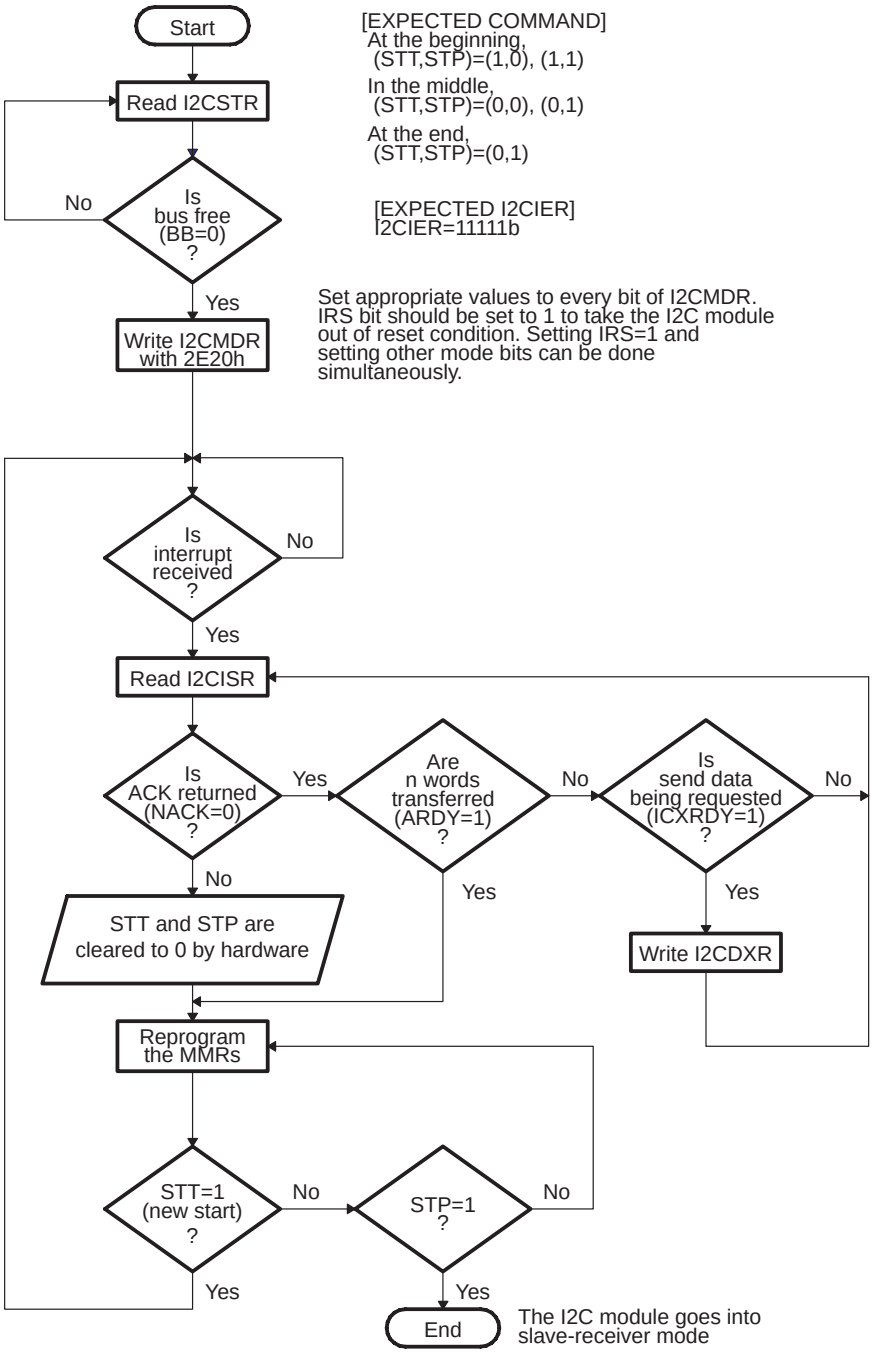


Figure 22. Using Interrupts for Master-Receiver Operation, Nonrepeat Mode (RM = 0)

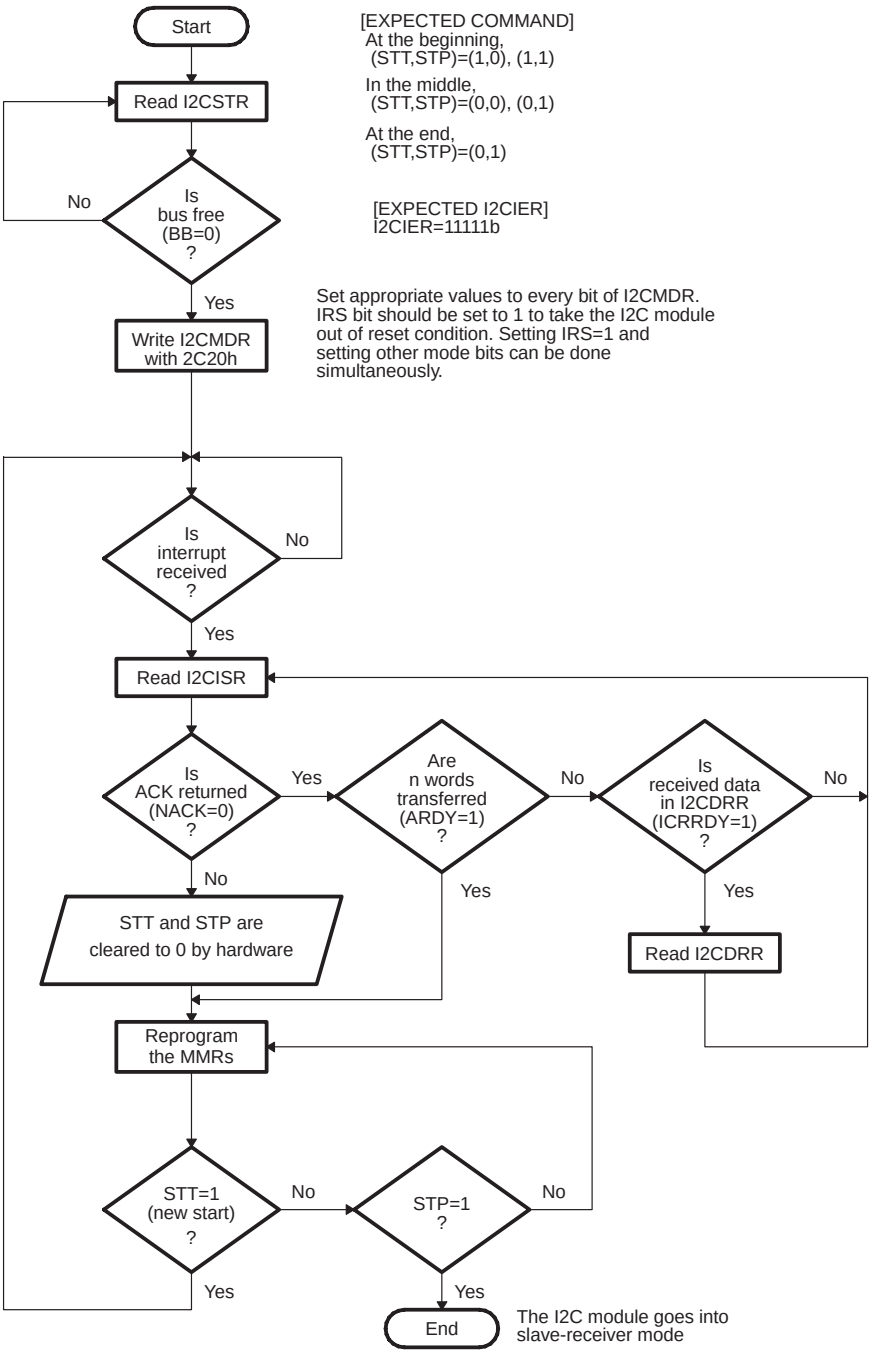


Figure 23. Using Interrupts for Master-Transmitter/Receiver Operation, Repeat Mode (RM = 1)

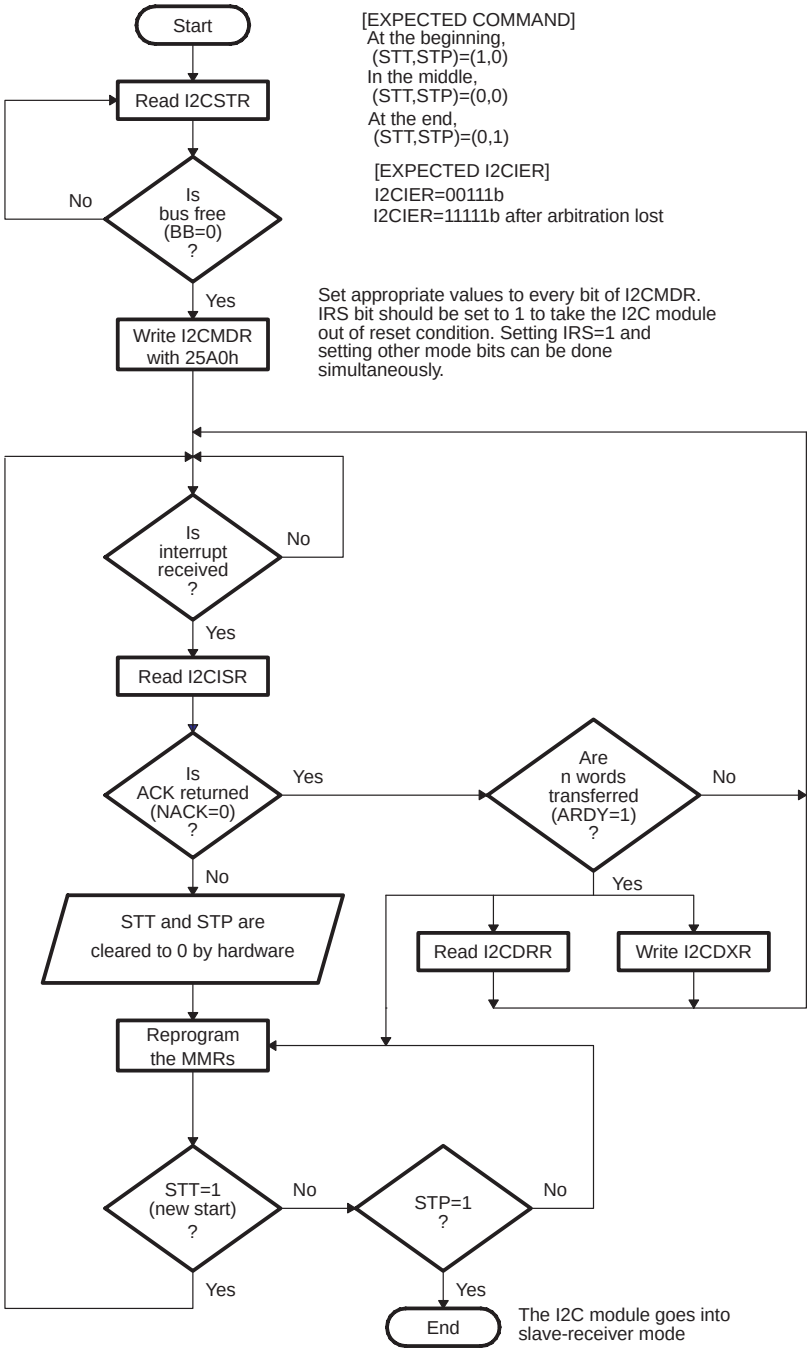
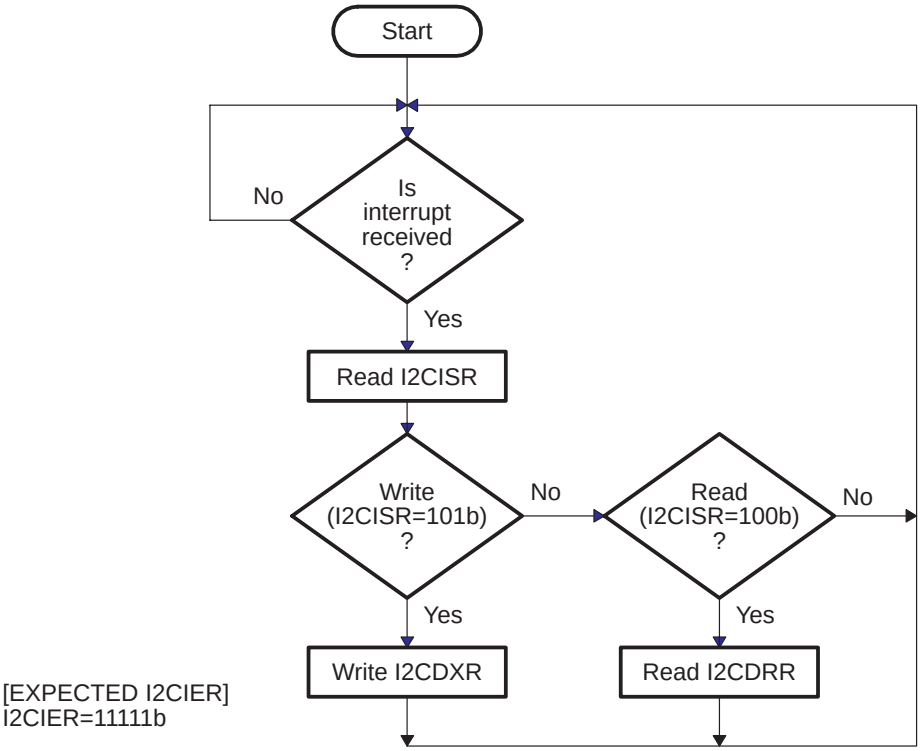


Figure 24. Using Interrupts for Slave-Transmitter/Receiver Operation, Repeat Mode (RM = 1)



7.3.3 EDMA Event Methods to Control Data Flow

Figure 25. Using EDMA Events for Master-Transmitter Operation, Nonrepeat Mode (RM = 0)

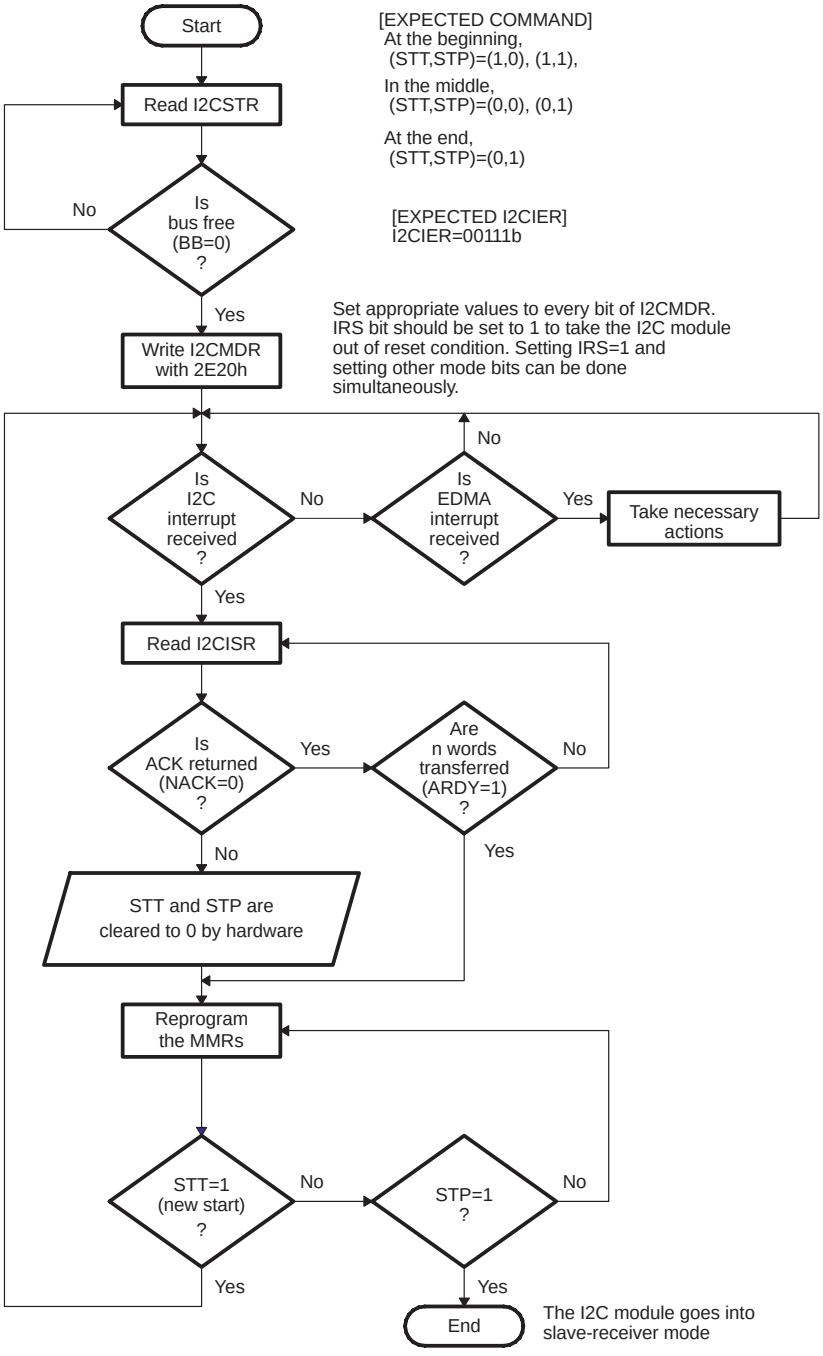
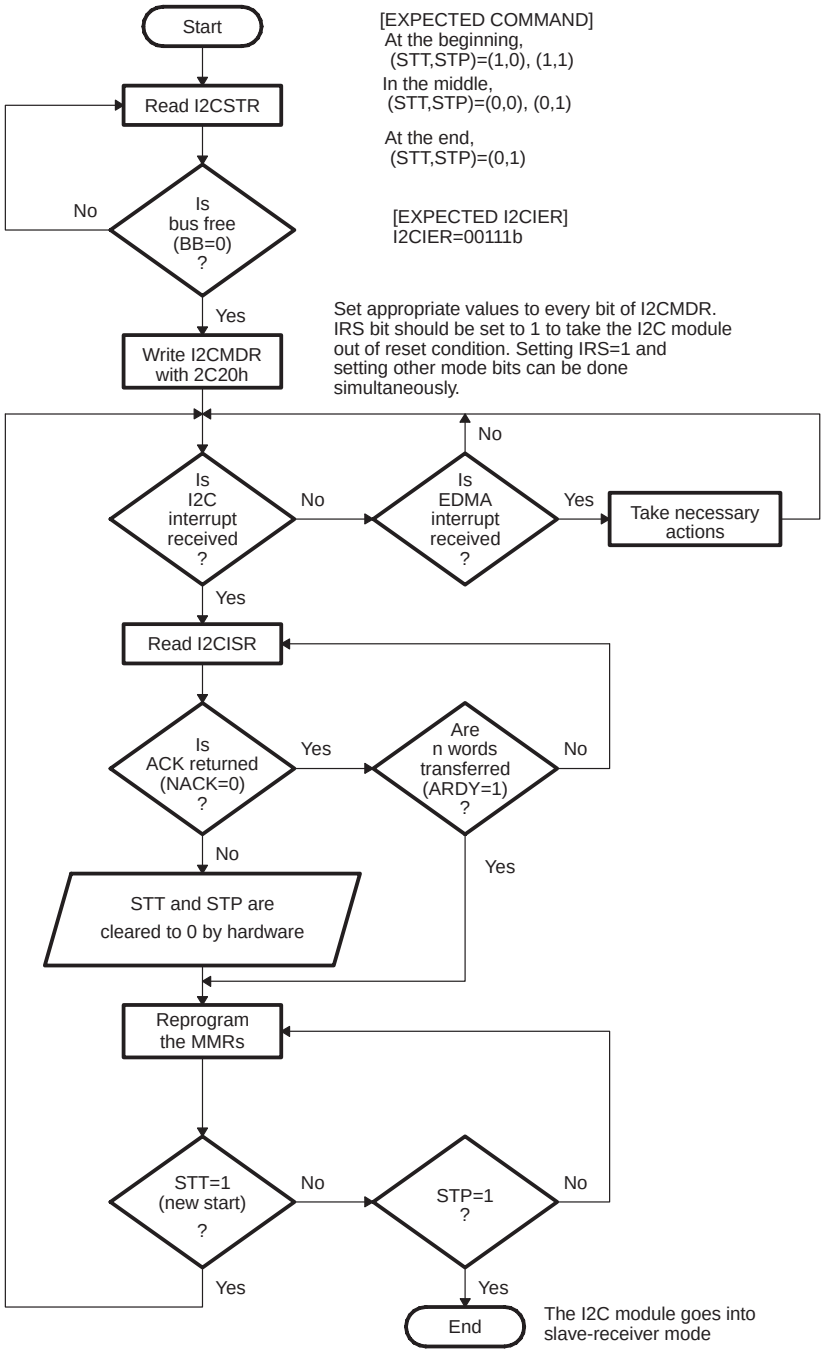


Figure 26. Using EDMA Events for Master-Receiver Operation, Nonrepeat Mode (RM = 0)



8 Registers

Table 4 lists the I2C module registers. All but the data registers (I2CRSR and I2CXSR) are accessible to the CPU. See the device-specific datasheet for the memory address of these registers.

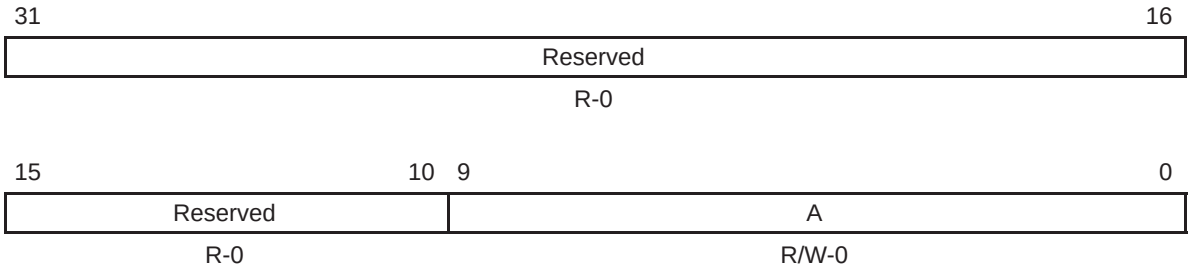
Table 4. I2C Module Registers

Acronym	Register Name	Address Offset (hex)	Section
I2COAR	I2C own address register	00	8.1
I2CIER	I2C interrupt enable register	04	8.2
I2CSTR	I2C status register	08	8.3
I2CCLKL	I2C clock low-time divider register	0C	8.4
I2CCLKH	I2C clock high-time divider register	10	8.4
I2CCNT	I2C data count register	14	8.5
I2CDRR	I2C data receive register	18	8.6
I2CSAR	I2C slave address register	1C	8.7
I2CDXR	I2C data transmit register	20	8.8
I2CMDR	I2C mode register	24	8.9
I2CISR	I2C interrupt source register	28	8.10
I2CPSC	I2C prescaler register	30	8.11
I2CPID1	I2C peripheral identification register 1	—	8.12
I2CPID2	I2C peripheral identification register 2	—	8.12
I2CRSR	I2C receive shift register (not accessible to the CPU or EDMA)	—	—
I2CXSR	I2C transmit shift register (not accessible to the CPU or EDMA)	—	—

8.1 I2C Own Address Register (I2COAR)

The I2C own address register (I2COAR) is a 32-bit register mapped used 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; bits 9−7 are ignored. The I2COAR is shown in Figure 27 and described in Table 5.

Figure 27. I2C Own Address Register (I2COAR)



Legend: R = Read only; R/W = Read/write; -*n* = value after reset

Table 5. I2C Own Address Register (I2COAR) Field Descriptions

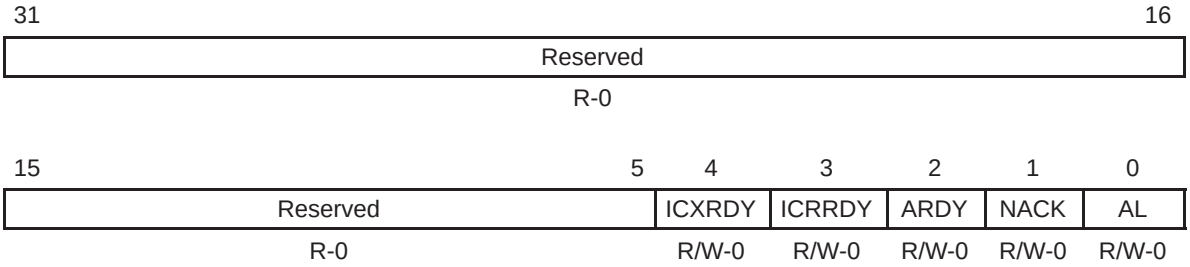
Bit	Field	*symval*†	Value	Description
31−10	Reserved	−	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
9−0	A	OF(*value*)		**In 7-bit addressing mode (XA = 0 in I2CMDR):**
			00h−7Fh	Bits 6−0 provide the 7-bit slave address of the I2C module. Bits 9−7 are ignored.
				In 10-bit addressing mode (XA = 1 in I2CMDR):
			000h−3FFh	Bits 9−0 provide the 10-bit slave address of the I2C module.

† For CSL C macro implementation, use the notation I2C_I2COAR_A_*symval*

8.2 I2C Interrupt Enable Register (I2CIER)

The I2C interrupt enable register (I2CIER) is used by the CPU to individually enable or disable I2C interrupt requests. The I2CIER is shown in Figure 28 and described Table 6.

Figure 28. I2C Interrupt Enable Register (I2CIER)



Legend: R = Read only; R/W = Read/write; -*n* = value after reset

Table 6. I2C Interrupt Enable Register (I2CIER) Field Descriptions

Bit	*field*†	*symval*†	Value	Description
31−5	Reserved	−	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
4	ICXRDY			Transmit-data-ready interrupt enable bit.
		MSK	0	Interrupt request is disabled.
		UNMSK	1	Interrupt request is enabled.
3	ICRRDY			Receive-data-ready interrupt enable bit.
		MSK	0	Interrupt request is disabled.
		UNMSK	1	Interrupt request is enabled.
2	ARDY			Register-access-ready interrupt enable bit.
		MSK	0	Interrupt request is disabled.
		UNMSK	1	Interrupt request is enabled.
1	NACK			No-acknowledgement interrupt enable bit.
		MSK	0	Interrupt request is disabled.
		UNMSK	1	Interrupt request is enabled.

† For CSL C macro implementation, use the notation I2C_I2CIER_*field_symval*

Table 6. I2C Interrupt Enable Register (I2CIER) Field Descriptions (Continued)

Bit	*field*†	*symval*†	Value	Description
0	AL			Arbitration-lost interrupt enable bit
		MSK	0	Interrupt request is disabled.
		UNMSK	1	Interrupt request is enabled.

† For CSL C macro implementation, use the notation I2C_I2CIER_*field_symval*

8.3 I2C Status Register (I2CSTR)

The I2C status register (I2CSTR) is used by the CPU to determine which interrupt has occurred and to read status information. The I2CSTR is shown in Figure 29 and described in Table 7.

Figure 29. I2C Status Register (I2CSTR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

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	6	5	4	3	2	1	0
Reserved			ICXRDY	ICRRDY	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

Table 7. I2C Status Register (I2CSTR) Field Descriptions

Bit	*field*†	*symval*†	Value	Description
31−14	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 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 8.9).
		NONE	0	NACK is 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).
		INT CLR	1	NACK is sent: A no-acknowledge bit was sent during the acknowledge cycle on the I^2C-bus.
12	BB			Bus busy bit. BB indicates whether the I^2C-bus is busy or is free for another data transfer.
		NONE	0	Bus is 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.
		INT CLR	1	Bus is 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 receive shift register (I2CRSR) is full with new data but the previous data has not been read from the data receive register (I2CDRR). The new data will not be copied to I2CDRR until the previous data is read. As new bits arrive from the SDA pin, they overwrite the bits in I2CRSR.
		NONE	0	No overrun is detected. RSFULL is cleared by any one of the following events: ❑ I2CDRR is read. ❑ The I2C module is reset.
		INT	1	Overrun is detected.

† For CSL C macro implementation, use the notation I2C_I2CSTR_*field_symval*

Table 7. I2C Status Register (I2CSTR) Field Descriptions (Continued)

Bit	*field*†	*symval*†	Value	Description
10	XSMT			Transmit shift register empty bit. XSMT 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.
		NONE	0	Underflow is detected.
		INT	1	No underflow is detected. XSMT is set by one of the following events: ❑ Data is written to I2CDXR. ❑ The I2C module is reset.
9	AAS			Addressed-as-slave bit.
		NONE	0	The AAS bit has been cleared by a repeated START condition or by a STOP condition.
		INT	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 data word has been received in the free data format (FDF = 1 in I2CMDR).
8	AD0			Address 0 bit.
		NONE	0	AD0 has been cleared by a START or STOP condition.
		INT	1	An address of all zeros (general call) is detected.
7−5	Reserved	−	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.

† For CSL C macro implementation, use the notation I2C_I2CSTR_*field_symval*

Table 7. I2C Status Register (I2CSTR) Field Descriptions (Continued)

Bit	*field*†	*symval*†	Value	Description
4	ICXRDY			Transmit-data-ready interrupt flag bit. ICXRDY 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 ICXRDY or use the XRDY interrupt request (see section 4).
		NONE	0	I2CDXR is not ready. ICXRDY is cleared by one of the following events: ❑ Data is written to I2CDXR. ❑ ICXRDY is manually cleared. To clear this bit, write a 1 to it.
		INT CLR	1	I2CDXR is ready: Data has been copied from I2CDXR to I2CXSR. ICXRDY is forced to 1 when the I2C module is reset.
3	ICRRDY			Receive-data-ready interrupt flag bit. ICRRDY 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 ICRRDY or use the RRDY interrupt request (see section 4).
		NONE	0	I2CDRR is not ready. ICRRDY is cleared by any one of the following events: ❑ I2CDRR is read. ❑ ICRRDY is manually cleared. To clear this bit, write a 1 to it. ❑ The I2C module is reset.
		INT CLR	1	I2CDRR is ready: Data has been copied from I2CRSR to I2CDRR.

† For CSL C macro implementation, use the notation I2C_I2CSTR_*field_symval*

Table 7. I2C Status Register (I2CSTR) Field Descriptions (Continued)

Bit	*field*†	*symval*†	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 4).
		NONE	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.
		INT CLR	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 data word 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 4).
		NONE	0	ACK received/NACK is 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 (I2CISR) when the register contains the code for a NACK interrupt. ❑ The I2C module is reset.
		INT CLR	1	NACK bit is 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.

† For CSL C macro implementation, use the notation I2C_I2CSTR_*field_symval*

Table 7. I2C Status Register (I2CSTR) Field Descriptions (Continued)

Bit	*field*†	*symval*†	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 4).
		NONE	0	Arbitration is 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 (I2CISR) when the register contains the code for an AL interrupt. ❑ The I2C module is reset.
		INT CLR	1	Arbitration is 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.

† For CSL C macro implementation, use the notation I2C_I2CSTR_*field_symval*

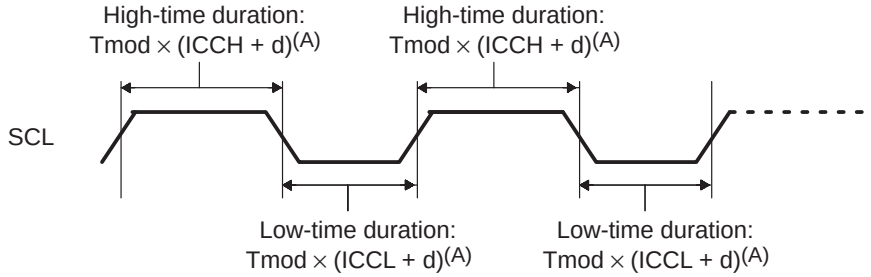
8.4 I2C Clock Divider Registers (I2CCLKL and I2CCLKH)

When the I2C is a master, the prescaled module clock is divided down for use as the master clock on the SCL pin. As shown in Figure 30, the shape of the master clock depends on two divide-down values:, ICCL and ICCH.

The frequency of the master clock is calculated as:

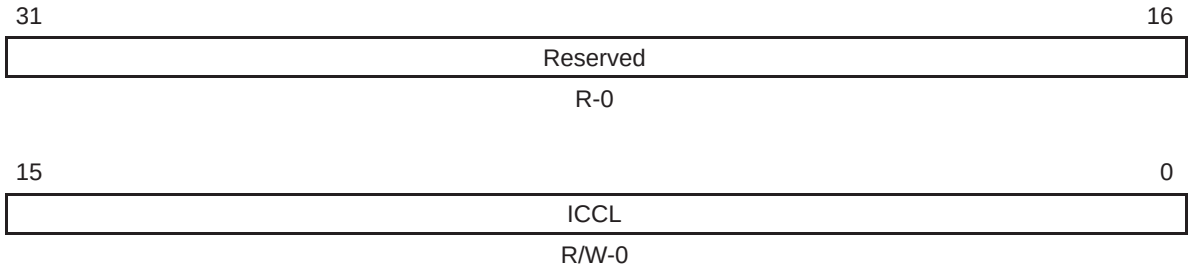
$$\text{master clock frequency} = \frac{\text{prescaled module clock frequency}}{(\text{ICCL} + \text{d}) + (\text{ICCH} + \text{d})}$$

Figure 30. Roles of the Clock Divide-Down Values (ICCL and ICCH)



(A)Tmod = prescaled module clock period = 1 /prescaled module clock frequency; d = 5, 6, or 7.

Figure 31. I2C Clock Low-Time Divider Register (I2CCLKL)



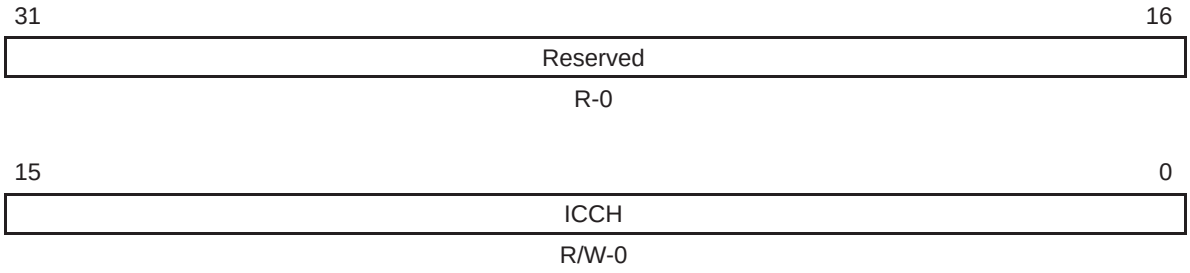
Legend: R = Read only; R/W = Read/write; -*n* = value after reset

Table 8. I2C Clock Low-Time Divider Register (I2CCLKL) Field Descriptions

Bit	Field	*symval*†	Value	Description
31−16	Reserved	−	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
15−0	ICCL	OF(*value*)	0	Not supported. Note that 0 is the default value; therefore, this field must be set to a value of 1 or greater.
			1−FFFFh	Clock low-time divide-down value of 2−65536. The period of the prescaled module clock is multiplied by (ICCL + d) to produce the low-time duration of the master clock on the SCL pin.

† For CSL C macro implementation, use the notation I2C_I2CCLKL_ICCL_*symval*

Figure 32. I2C Clock High-Time Divider Register (I2CCLKH)



Legend: R = Read only; R/W = Read/write; -*n* = value after reset

Table 9. I2C Clock High-Time Divider Register (I2CCLKH) Field Descriptions

Bit	Field	*symval*†	Value	Description
31−16	Reserved	−	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
15−0	ICCH	OF(value)	0	Not supported. Note that 0 is the default value; therefore, this field must be set to a value of 1 or greater.
			1−FFFFh	Clock low-time divide-down value of 2−65536. The period of the prescaled module clock is multiplied by (ICCL + d) to produce the low-time duration of the master clock on the SCL pin.

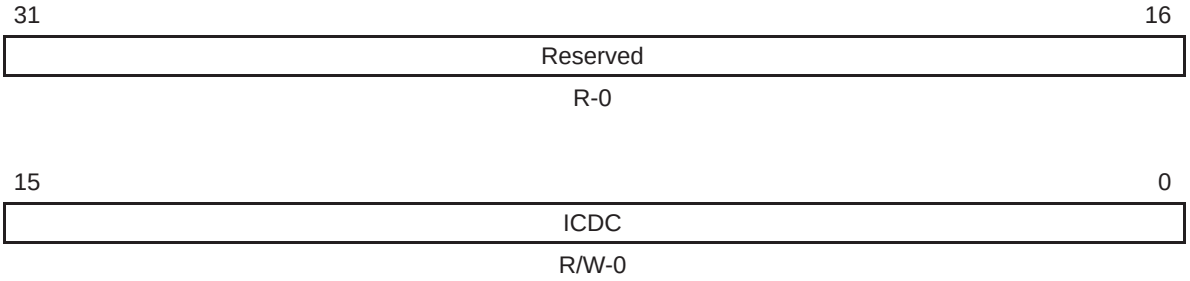
† For CSL C macro implementation, use the notation I2C_I2CCLKH_ICCH_*symval*

8.5 I2C Data Count Register (I2CCNT)

The I2C data count register (I2CCNT) is used to indicate how many data words 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 I2CCNT is shown in Figure 33 and described in Table 10.

The value written to I2CCNT is copied to an internal data counter. The internal data counter is decremented by 1 for each data word 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 count-down is complete (that is, when the last data word has been transferred).

Figure 33. I2C Data Count Register (I2CCNT)



Legend: R = Read only; R/W = Read/write; -*n* = value after reset

Table 10. I2C Data Count Register (I2CCNT) Field Descriptions

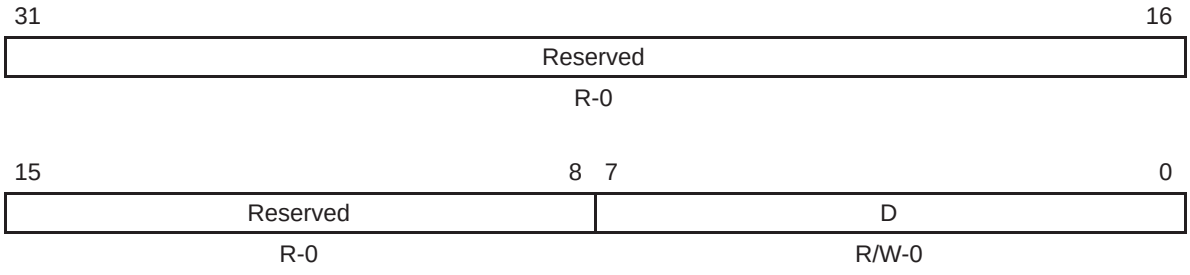
Bit	Field	*symval*†	Value	Description
31−16	Reserved	−	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
15−0	ICDC	OF(*value*)		Data count value. ICDC indicates the number of data words to transfer in the nonrepeat mode (RM = 0 in I2CMDR). The value 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.

† For CSL C macro implementation, use the notation I2C_I2CCNT_ICDC_*symval*

8.6 I2C Data Receive Register (I2CDRR)

The I2C data receive register (I2CDRR) is used by the DSP to read the receive data. The I2CDRR can receive a data value of up to 8 bits; data values with fewer than 8 bits are right-aligned in the D bits and the remaining D bits are undefined. The number of data bits is selected by the bit count bits (BC) of I2CMDR. The I2C receive shift register (I2CRSR) shifts in the received data from the SDA pin. Once data is complete, the I2C module copies the contents of I2CRSR into I2CDRR. The CPU and the EDMA controller cannot access I2CRSR. The I2CDRR is shown in Figure 34 and described in Table 11.

Figure 34. I2C Data Receive Register (I2CDRR)



Legend: R = Read only; R/W = Read/write; -*n* = value after reset

Table 11. I2C Data Receive Register (I2CDRR) Field Descriptions

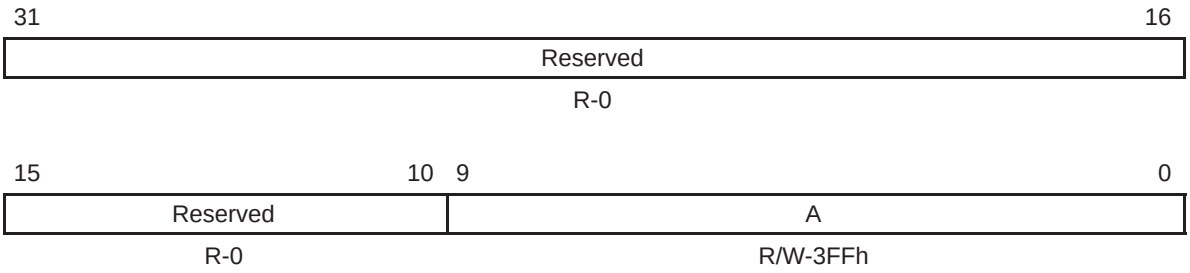
Bit	Field	*symval*†	Value	Description
31−8	Reserved	–	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
7−0	D	OF(value)	00h−FFh	Receive data.

† For CSL C macro implementation, use the notation I2C_I2CDRR_D_*symval*

8.7 I2C Slave Address Register (I2CSAR)

The I2C slave address register (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; bits 9−7 are ignored. The I2CSAR is shown in Figure 35 and described in Table 12.

Figure 35. I2C Slave Address Register (I2CSAR)



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

Table 12. I2C Slave Address Register (I2CSAR) Field Descriptions

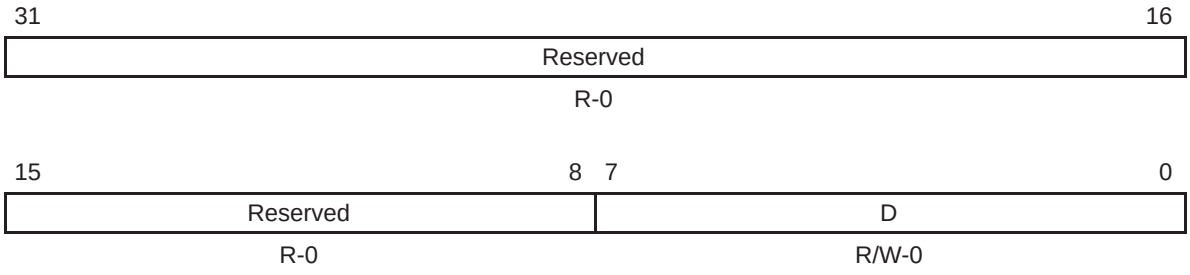
Bit	Field	*symval*†	Value	Description
31−10	Reserved	−	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
9−0	A	OF(*value*)		**In 7-bit addressing mode (XA = 0 in I2CMDR):**
			00h−7Fh	Bits 6−0 provide the 7-bit slave address that the I2C module transmits when it is in the master-transmitter mode. Bits 9−7 are ignored.
				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.

† For CSL C macro implementation, use the notation I2C_I2CSAR_A_*symval*

8.8 I2C Data Transmit Register (I2CDXR)

The DSP writes transmit data to the I2C data transmit register (I2CDXR). The I2CDXR can accept a data value of up to 8 bits. When writing a data value with fewer than 8 bits, the DSP must make sure that the value is right-aligned in the D bits. The number of data bits is selected by the bit count bits (BC) of I2CMDR. Once data is written to I2CDXR, the I2C module copies the contents of I2CDXR into the I2C transmit shift register (I2CXSR). The I2CXSR shifts out the transmit data from the SDA pin. The CPU and the EDMA controller cannot access I2CXSR. The I2CDXR is shown in Figure 36 and described in Table 13.

Figure 36. I2C Data Transmit Register (I2CDXR)



Legend: R = Read only; R/W = Read/write; -*n* = value after reset

Table 13. I2C Data Transmit Register (I2CDXR) Field Descriptions

Bit	Field	*symval*†	Value	Description
31−8	Reserved	–	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
7−0	D	OF(value)	00h−FFh	Transmit data

† For CSL C macro implementation, use the notation I2C_I2CDXR_D_*symval*

8.9 I2C Mode Register (I2CMDR)

The I2C mode register (I2CMDR) contains the control bits of the I2C module. The I2CMDR is shown in Figure 37 and described in Table 14.

Figure 37. I2C Mode Register (I2CMDR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R/W-0															

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	1	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/W = Read/write; -*n* = value after reset

Table 14. I2C Mode Register (I2CMDR) Field Descriptions

Bit	***field***†	***symval***†	Value	Description
31−16	Reserved	−	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
15	NACKMOD			NACK mode bit is only applicable when the I2C module is acting as a receiver.
		ACK	0	**In 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 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.
		NACK	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. To send a NACK bit in the next acknowledge cycle, you must set NACKMOD before the rising edge of the last data bit.

† For CSL C macro implementation, use the notation I2C_I2CMDR_*field_symval*

Table 14. I2C Mode Register (I2CMDR) Field Descriptions (Continued)

Bit	*field*†	*symval*†	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.
		BSTOP	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.
		RFREE	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 15). Note that the STT and STP bits can be used to terminate the repeat mode.
		NONE	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.
		START	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.
12	Reserved	–	0	This reserved bit location is always read as zero. A value written to this field has no effect.

† For CSL C macro implementation, use the notation I2C_I2CMDR_*field_symval*

Table 14. I2C Mode Register (I2CMDR) Field Descriptions (Continued)

Bit	*field*†	*symval*†	Value	Description
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 15). Note that the STT and STP bits can be used to terminate the repeat mode.
		NONE	0	STP is automatically cleared after the STOP condition has been generated.
		STOP	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.
		SLAVE	0	Slave mode. The I2C module is a slave and receives the serial clock from the master.
		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 16 summarizes when TRX is used and when it is a don't care.
		RCV	0	Receiver mode. The I2C module is a receiver and receives data on the SDA pin.
		XMT	1	Transmitter mode. The I2C module is a transmitter and transmits data on the SDA pin.
8	XA			Expanded address enable bit.
		7BIT	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).
		10BIT	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).

† For CSL C macro implementation, use the notation I2C_I2CMDR_*field_symval*

Table 14. I2C Mode Register (I2CMDR) Field Descriptions (Continued)

Bit	*field*†	*symval*†	Value	Description
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 15).
		NONE	0	Nonrepeat mode. The value in the data count register (I2CCNT) determines how many data words are received/ transmitted by the I2C module.
		REPEAD	1	Repeat mode. Data words 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. This bit disables or enables the digital loopback mode of the I2C module. The effects of this bit are shown in Figure 38. Note that DLB in the free data format mode (DLB = 1 and FDF = 1) is not supported.
		NONE	0	Digital loopback mode is disabled.
		LOOPBACK	1	Digital loopback mode is enabled. In this mode, the MST bit must be set to 1 and data transmitted out of I2CDXR is received in I2CDRR after *n* DSP cycles by an internal path, where: *n* = ((I2C input clock frequency/prescaled 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.
5	IRS			I2C module reset bit.
		RST	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.
		NRST	1	The I2C module is enabled.

† For CSL C macro implementation, use the notation I2C_I2CMDR_*field_symval*

Table 14. I2C Mode Register (I2CMDR) Field Descriptions (Continued)

Bit	*field*†	*symval*†	Value	Description
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 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, the I2C module ignores a START byte from a master, regardless of the value of the STB bit.
		NONE	0	The I2C module is not in the START byte mode.
		SET	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 The I2C module sends the slave address that is in I2CSAR.
3	FDF			Free data format mode bit. Note that DLB in the free data format mode (DLB = 1 and FDF = 1) is not supported.
		NONE	0	Free data format mode is disabled. Transfers use the 7-/10-bit addressing format selected by the XA bit.
		SET	1	Free data format mode is enabled. Transfers have the free data (no address) format described in section 3.5.

† For CSL C macro implementation, use the notation I2C_I2CMDR_*field_symval*

Table 14. I2C Mode Register (I2CMDR) Field Descriptions (Continued)

Bit	*field*†	*symval*†	Value	Description
2−0	BC	OF(*value*)		Bit count bits. BC defines the number of bits (1 to 8) in the next data word 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 word has 8 bits. If the bit count is less than 8, receive data is right aligned in the D bits of I2CDRR and the remaining D bits are undefined. Also, transmit data written to I2CDXR must be right aligned.
		BIT8FDF	0	8 bits per data word
		BIT1FDF	1h	1 bit per data word
		BIT2FDF	2h	2 bits per data word
		BIT3FDF	3h	3 bits per data word
		BIT4FDF	4h	4 bits per data word
		BIT5FDF	5h	5 bits per data word
		BIT6FDF	6h	6 bits per data word
		BIT7FDF	7h	7 bits per data word

† For CSL C macro implementation, use the notation I2C_I2CMDR_*field_symval*

Table 15. Master-Transmitter/Receiver Bus Activity Defined by RM, STT, and STP Bits

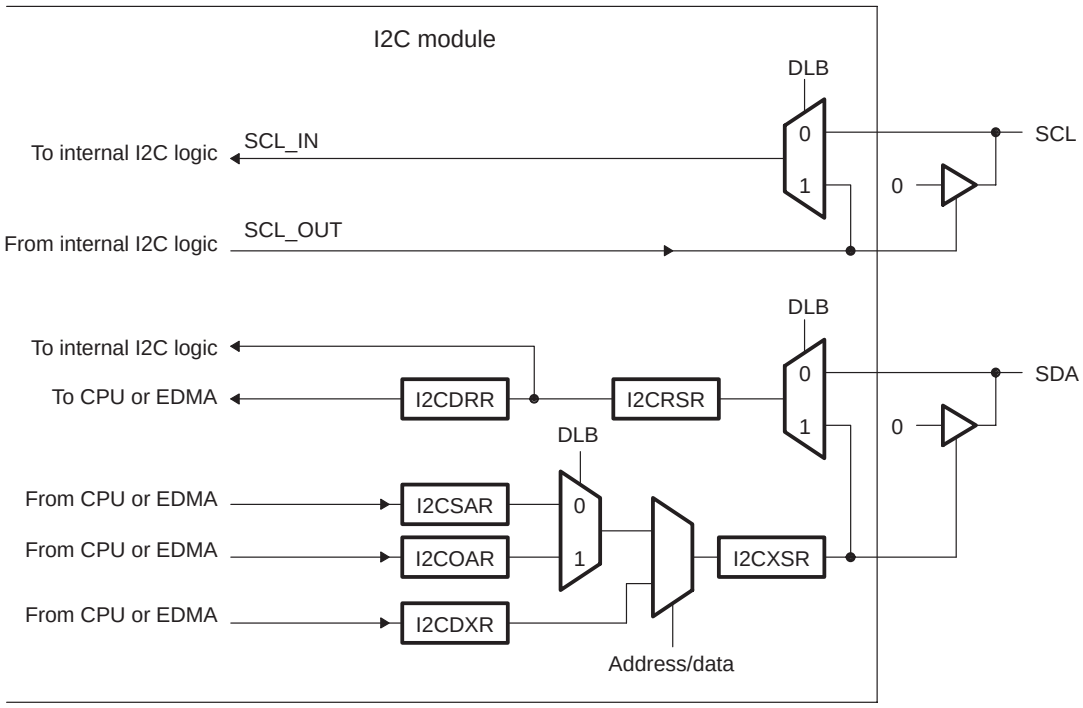
I2CMDR Bit				
RM	**STT**	**STP**	**Bus Activity†**	**Description**
0	0	0	None	No activity
0	0	1	P	STOP condition
0	1	0	S-A-D..(*n*)..D	START condition, slave address, *n* data words (*n* = value in I2CCNT)
0	1	1	S-A-D..(*n*)..D-P	START condition, slave address, *n* data words, STOP condition (*n* = 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)

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

Table 16. How the MST and FDF Bits Affect the Role of TRX Bit

I2CMDR Bit			
MST	**FDF**	**I2C Module State**	**Function of TRX Bit**
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 transmitter and receiver be fixed. 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.

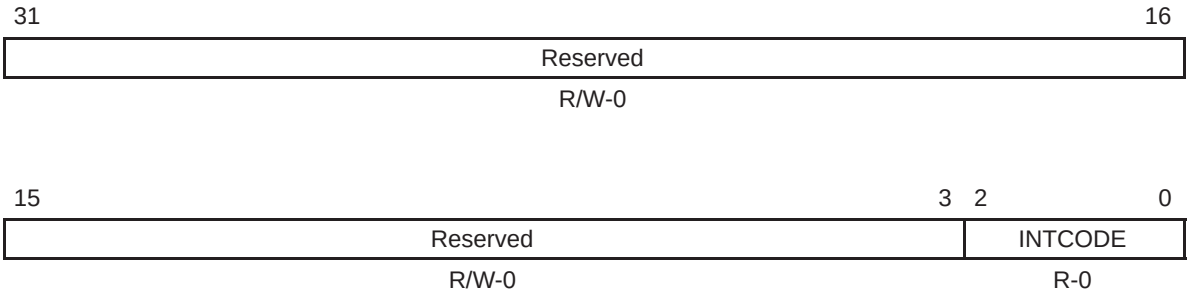
Figure 38. Block Diagram Showing the Effects of the Digital Loopback Mode (DLB) Bit



8.10 I2C Interrupt Source Register (I2CISR)

The I2C interrupt source register (I2CISR) is 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. The I2CISR is shown in Figure 39 and described in Table 17.

Figure 39. I2C Interrupt Source Register (I2CISR)



Legend: R = Read only; R/W = Read/write; -*n* = value after reset

Table 17. I2C Interrupt Source Register (I2CISR) Field Descriptions

Bit	Field	*symval*†	Value	Description
31−3	Reserved	−	0	These reserved bit locations are always read as zeros. Always write 0 to this field.
2−0	INTCODE			Interrupt code bits. The binary code in INTCODE indicates which event generated an I2C interrupt.
		NONE	000	None
		AL	001	Arbitration is lost.
		NACK	010	No-acknowledgement condition is detected.
		RAR	011	Registers are ready to be accessed.
		RDR	100	Receive data is ready.
		XDR	101	Transmit data is ready.
		−	110−111	Reserved

† For CSL C macro implementation, use the notation I2C_I2CISR_INTCODE_*symval*

8.11 I2C Prescaler Register (I2CPSC)

The I2C prescaler register (I2CPSC) is used for dividing down SYSCLK2 to obtain the desired prescaled module clock for the operation of I2C.

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

CAUTION

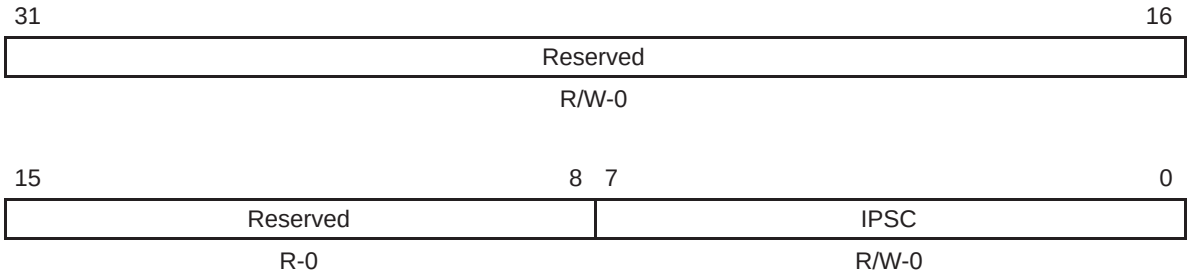
Prescaled Module Clock Frequency Range

The I2C module must be operated with a prescaled module clock frequency of 6.7 to 13.3 MHz.

The I2C prescaler register (I2CPSC) must be configured to this frequency range.

The I2C prescaler register (I2CPSC) is shown in Figure 40 and described in Table 18.

Figure 40. I2C Prescaler Register (I2CPSC)



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

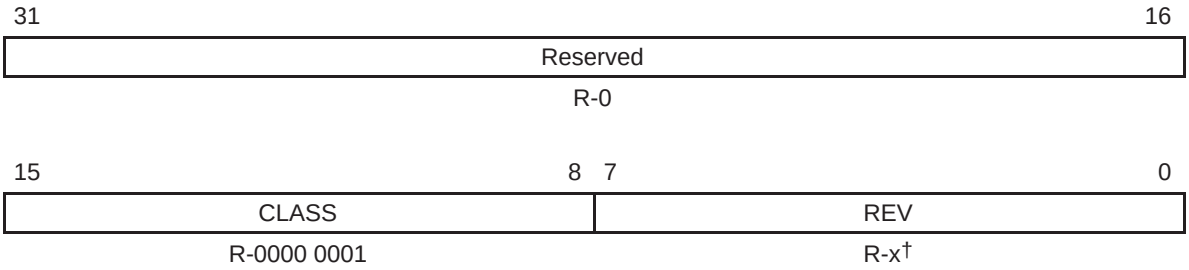
Table 18. I2C Prescaler Register (I2CPSC) Field Descriptions

Bit	Field	Value	Description
31−8	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
7−0	IPSC	0−FFh	I2C prescaler divide-down value. The I2C module must be operated with a prescaled module clock frequency of 6.7 to 13.3 MHz. IPSC must be set to a value for this frequency range. IPSC determines how much SYSCLK2 is divided to create the I2C clock: prescaled module clock frequency = SYSCLK2 frequency/(IPSC + 1) **Note:** IPSC must be initialized while I2C is in reset (IRS = 0 in I2CMDR).

8.12 I2C Peripheral Identification Registers (I2CPID1 and I2CPID2)

The peripheral identification registers (PID) contain identification data for the I2C module. I2CPID1 is shown in Figure 41 and described in Table 19. I2CPID2 is shown in Figure 42 and described in Table 20.

Figure 41. I2C Peripheral Identification Register 1 (I2CPID1)

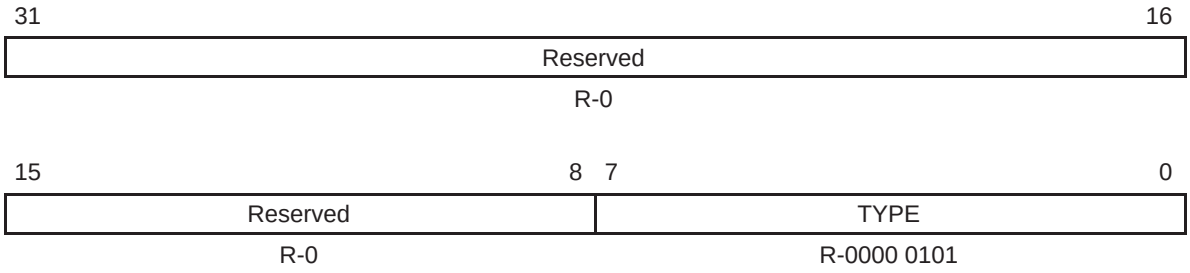


Legend: R = Read only; -x = value after reset
† See the device-specific datasheet for the default value of this field.

Table 19. I2C Peripheral Identification Register 1 (I2CPID1) Field Descriptions

Bit	Field	Value	Description
31−16	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
15−8	CLASS		Identifies class of peripheral.
		1	Serial port
7−0	REV		Identifies revision of peripheral.
		x	See the device-specific datasheet for the value.

Figure 42. I2C Peripheral Identification Register 2 (I2CPID2)



Legend: R = Read only; -x = value after reset

Table 20. I2C Peripheral Identification Register 2 (I2CPID2) Field Descriptions

Bit	Field	Value	Description
31−8	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
7−0	TYPE		Identifies type of peripheral.
		05h	I2C

Revision History

Table 21 lists the changes made since the previous version of this document.

Table 21. Document Revision History

Page	Additions/Modifications/Deletions
Table 2	Updated Basic column of Slave-Receiver Mode cell.

This page is intentionally left blank.

Index

M

N

O

P

R

S

T

V

X