

TMS320C672x DSP Inter-Integrated Circuit (I2C) Module

Reference Guide

Literature Number: SPRU877B
August 2005

Contents

Preface	8
1 Introduction to the I2C Module	9
1.1 Features	9
1.2 Features Not Supported	10
2 Functional Overview	10
3 Operational Details	11
3.1 Electrical Considerations	11
3.2 Data Validity	12
3.3 Operating Modes	12
3.4 START and STOP Conditions	13
3.5 Serial Data Formats	14
3.6 NACK Bit Generation	16
3.7 Arbitration	17
3.8 Clock Generation	17
3.9 Clock Synchronization	19
4 Events Generated by the I2C Module	20
4.1 Interrupt Requests	20
4.2 dMAX Events	21
4.3 Transmit Data Ready Interrupt/DMA Event	21
5 GPIO Pin Control	23
6 Resetting/Disabling the I2C Module	23
7 Programming Guide	23
7.1 Main Program	23
7.2 Interrupt Subroutines	24
7.3 Flow Diagrams	24
8 Emulation Considerations	37
9 Registers	37
9.1 I2C Own Address Register (I2COAR)	38
9.2 I2C Interrupt Enable Register (I2CIER)	39
9.3 I2C Interrupt Status Register (I2CSTR)	40
9.4 I2C Clock Divider Registers (I2CCLKL and I2CCLKH)	43
9.5 I2C Data Count Register (I2CCNT)	45
9.6 I2C Data Receive Register (I2CDRR)	46
9.7 I2C Slave Address Register (I2CSAR)	47
9.8 I2C Data Transmit Register (I2CDXR)	48
9.9 I2C Mode Register (I2CMDR)	49
9.10 I2C Interrupt Source Register (I2CISR)	53
9.11 I2C Extended Mode Register (I2CEMDR)	54
9.12 I2C Prescaler Register (I2CPSC)	55
9.13 I2C Peripheral Identification Registers (I2CPID1 and I2CPID2)	56
9.14 I2C Pin Function Register (I2CPFUNC)	57
9.15 I2C Pin Direction Register (I2CPDIR)	58
9.16 I2C Pin Data Input Register (I2CPDIN)	59

9.17	I2C Pin Data Output Register (I2CPDOUT)	60
9.18	I2C Pin Data Set Register (I2CPDSET)	61
9.19	I2C Pin Data Clear Register (I2CPDCLR)	62
Appendix A	Revision History	63

List of Figures

1	Multiple I2C Modules Connected	9
2	I2C Module Conceptual Block Diagram	11
3	Bit Transfer on the I ² C-Bus	12
4	I2C Module START and STOP Conditions	13
5	I2C Module Data Transfer.....	14
6	I2C Module 7-Bit Addressing Format (FDF = 0, XA = 0 in I2CMMDR)	14
7	I2C Module 10-Bit Addressing Format With Master-Transmitter Writing to Slave-Receiver (FDF = 0, XA = 1 in I2CMMDR).....	15
8	I2C Module Free Data Format (FDF = 1 in I2CMMDR)	15
9	I2C Module 7-Bit Addressing Format With Repeated START Condition (FDF = 0, XA = 0 in I2CMMDR)	15
10	Arbitration Procedure Between Two Master-Transmitters	17
11	Clocking Diagram for the I2C Module	18
12	Synchronization of Two I ² C Clock Generators During Arbitration	19
13	Enable Paths of I2C Interrupt Requests	21
14	Interrupt Generation When XRDYM = 1	22
15	Interrupt Generation When XRDYM = 0	22
16	Setup Procedure	24
17	Using Bit Polling for Master-Transmitter Operation, Repeat Mode (RM = 1)	25
18	Using Bit Polling for Master-Transmitter Operation, Nonrepeat Mode (RM = 0).....	26
19	Using Bit Polling for Master-Receiver Operation, Repeat Mode (RM = 1), Fixed Number of Data Words	27
20	Using Bit Polling for Master-Receiver Operation, Repeat Mode (RM = 1), Variable (Data-Dependent) Number of Data Words	28
21	Using Bit Polling for Master-Receiver Operation, Nonrepeat Mode (RM = 0).....	29
22	Using Bit Polling for Slave-Transmitter/Receiver Operation, Repeat Mode (RM = 1).....	30
23	Using Interrupts for Master-Transmitter Operation, Nonrepeat Mode (RM = 0)	31
24	Using Interrupts for Master-Receiver Operation, Nonrepeat Mode (RM = 0)	32
25	Using Interrupts for Master-Transmitter/Receiver Operation, Repeat Mode (RM = 1).....	33
26	Using Interrupts for Slave-Transmitter/Receiver Operation, Repeat Mode (RM = 1)	34
27	Using dMAX Events for Master-Transmitter Operation, Nonrepeat Mode (RM = 0)	35
28	Using dMAX Events for Master-Receiver Operation, Nonrepeat Mode (RM = 0)	36
29	I2C Own Address Register (I2COAR) [Offset = 00h].....	38
30	I2C Interrupt Enable Register (I2CIER) [Offset = 04h]	39
31	I2C Status Register (I2CSTR) [Offset = 08h].....	40
32	Roles of the Clock Divide-Down Values (ICCL and ICCH)	43
33	I2C Clock Low-Time Divider Register (I2CCLKL) [Offset = 0Ch]	43
34	I2C Clock High-Time Divider Register (I2CCLKH) [Offset = 10h].....	44
35	I2C Data Count Register (I2CCNT) [Offset = 14h].....	45
36	I2C Data Receive Register (I2CDRR) [Offset = 18h].....	46
37	I2C Slave Address Register (I2CSAR) [Offset = 1Ch]	47
38	I2C Data Transmit Register (I2CDXR) [Offset = 20h]	48
39	I2C Mode Register (I2CMMDR) [Offset = 24h].....	49
40	Block Diagram Showing the Effects of the Digital Loopback Mode (DLB) Bit.....	52
41	I2C Interrupt Source Register (I2CISR) [Offset = 28h]	53
42	I2C Extended Mode Register (I2CEMDR) [Offset = 2Ch].....	54
43	I2C Prescaler Register (I2CPSC) [Offset = 30h]	55
44	I2C Peripheral Identification Register 1 (I2CPID1) [Offset = 34h]	56
45	I2C Peripheral Identification Register 2 (I2CPID2) [Offset = 38h]	56
46	I2C Pin Function Register (I2CPFUNC) [Offset = 48h].....	57
47	I2C Pin Direction Register (I2CPDIR) [Offset = 4Ch].....	58
48	I2C Pin Data Input Register (I2CPDIN) [Offset = 50h]	59
49	I2C Pin Data Output Register (I2CPDOUT) [Offset = 54h]	60

50	I2C Pin Data Set Register (I2CPDSET) [Offset = 58h].....	61
51	I2C Pin Data Clear Register (I2CPDCLR) [Offset = 5Ch].....	62

List of Tables

1	Operating Modes of the I2C Module	12
2	Ways to Generate a NACK Bit	16
3	Descriptions of the I2C Interrupt Requests	20
4	Inter-Integrated Circuit (I2C) Registers	37
5	I2C Own Address Register (I2COAR) Field Descriptions.....	38
6	I2C Interrupt Enable Register (I2CIER) Field Descriptions	39
7	I2C Status Register (I2CSTR) Field Descriptions.....	40
8	I2C Clock Low-Time Divider Register (I2CCLKL) Field Descriptions	43
9	I2C Clock High-Time Divider Register (I2CCLKH) Field Descriptions	44
10	I2C Data Count Register (I2CCNT) Field Descriptions.....	45
11	I2C Data Receive Register (I2CDRR) Field Descriptions.....	46
12	I2C Slave Address Register (I2CSAR) Field Descriptions	47
13	I2C Data Transmit Register (I2CDXR) Field Descriptions	48
14	I2C Mode Register (I2CMDR) Field Descriptions	49
15	Master-Transmitter/Receiver Bus Activity Defined by RM, STT, and STP Bits	51
16	How the MST and FDF Bits Affect the Role of TRX Bit.....	51
17	I2C Interrupt Source Register (I2CISR) Field Descriptions	53
18	I2C Extended Mode Register (I2CEMDR) Field Descriptions	54
19	I2C Prescaler Register (I2CPSC) Field Descriptions	55
20	I2C Peripheral Identification Register 1 (I2CPID1) Field Descriptions	56
21	I2C Peripheral Identification Register 2 (I2CPID2) Field Descriptions	56
22	I2C Pin Function Register (I2CPFUNC) Field Descriptions	57
23	I2C Pin Direction Register (I2CPDIR) Field Descriptions	58
24	I2C Pin Data Input Register (I2CPDIN) Field Descriptions	59
25	I2C Pin Data Output Register (I2CPDOUT) Field Descriptions	60
26	I2C Pin Data Set Register (I2CPDSET) Field Descriptions.....	61
27	I2C Pin Data Clear Register (I2CPDCLR) Field Descriptions	62
A-1	Document Revision History	63

Read This First

About This Manual

This document describes the inter-integrated circuit (I²C) module in the TMS320C672x™ digital signal processors (DSPs) of the TMS320C6000™ DSP family. The I²C provides an interface between the C672x™ DSP and other devices compliant with Philips Semiconductors Inter-IC bus (I²C-bus) specification version 2.1 and connected by way of an I²C-bus. 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. 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.

The current documentation that describes the C6000 devices, related peripherals, and other technical collateral, is available in the C6000 DSP product folder at: www.ti.com/c6000.

SPRU733 — TMS320C67x/C67x+ DSP CPU and Instruction Set Reference Guide. Describes the CPU architecture, pipeline, instruction set, and interrupts for the TMS320C67x and TMS320C67x+ digital signal processors (DSPs) of the TMS320C6000 DSP family. The C67x/C67x+ DSP generation comprises floating-point devices in the C6000 DSP platform. The C67x+ DSP is an enhancement of the C67x DSP with added functionality and an expanded instruction set.

SPRU723 — TMS320C672x DSP Peripherals Overview Reference Guide. This document provides an overview and briefly describes the peripherals available on the TMS320C672x digital signal processors (DSPs) of the TMS320C6000 family.

SPRU197 — TMS320C6000 Technical Brief. Provides an introduction to the TMS320C62x and TMS320C67x digital signal processors (DSPs) of the TMS320C6000 DSP family. Describes the CPU architecture, peripherals, development tools and third-party support for the C62x and C67x DSPs.

Trademarks

TMS320C672x, TMS320C6000, C672x, C6000, Code Composer Studio are trademarks of Texas Instruments.

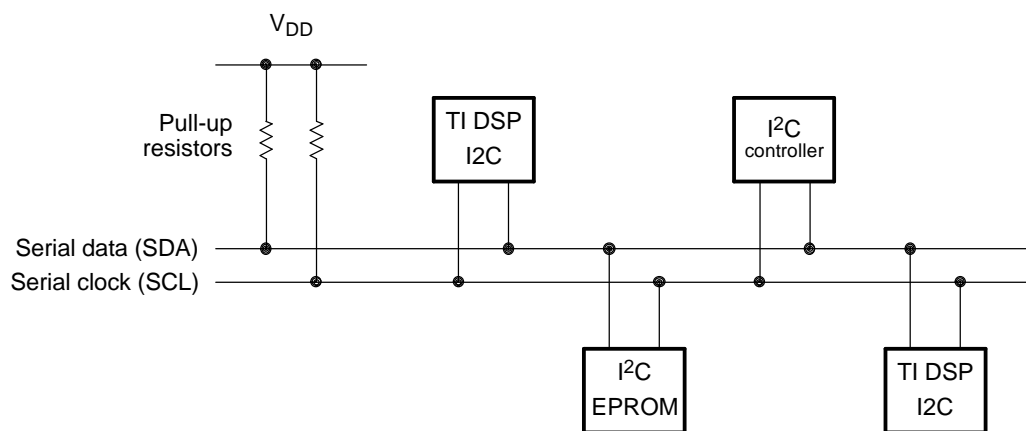
Inter-Integrated Circuit (I2C) Module

The inter-integrated circuit (I2C) module provides an interface between a TMS320C672x™ DSP and other devices compliant with Philips Semiconductors Inter-IC bus (I2C-bus) specification version 2.1 and connected by way of an I2C-bus. External components attached to this 2-wire serial bus can transmit/receive up to 8-bit data to/from the C672x™ DSP through the I2C module. To determine whether a particular C672x DSP has an I2C module, see the device-specific data manual for that DSP. This chapter assumes the reader is familiar with the I2C-bus specification.

1 Introduction to the I2C Module

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

Figure 1. Multiple I2C Modules Connected



1.1 Features

The I2C module has the following features:

- Compliance with the Philips Semiconductors I2C-bus specification (version 2.1):
 - Support for 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 (in 7-bit addressing mode only)
 - Data transfer rate of from 10 kbps up to 400 kbps (Philips Fast-mode rate)
- One dMAX receive event and one dMAX transmit event, which can be used by the dMAX 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
 - stop-condition detected
 - addressed-as-slave detected
- Module enable/disable capability
- Free data format mode
- GPIO capability

1.2 Features Not Supported

The I2C module does not support:

- High-speed mode (Hs-mode)
- CBUS compatibility mode
- 7-bit and 10-bit addressing within one serial transfer
- Combined formats in 10-bit addressing mode. These include:
 - master transmits data to multiple slaves within one serial transfer
 - master transmits data to slave, then reads data from same slave within one serial transfer

2 Functional Overview

Each device connected to an I²C-bus 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 I²C-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 I²C-bus can be connected to the same I²C-bus.

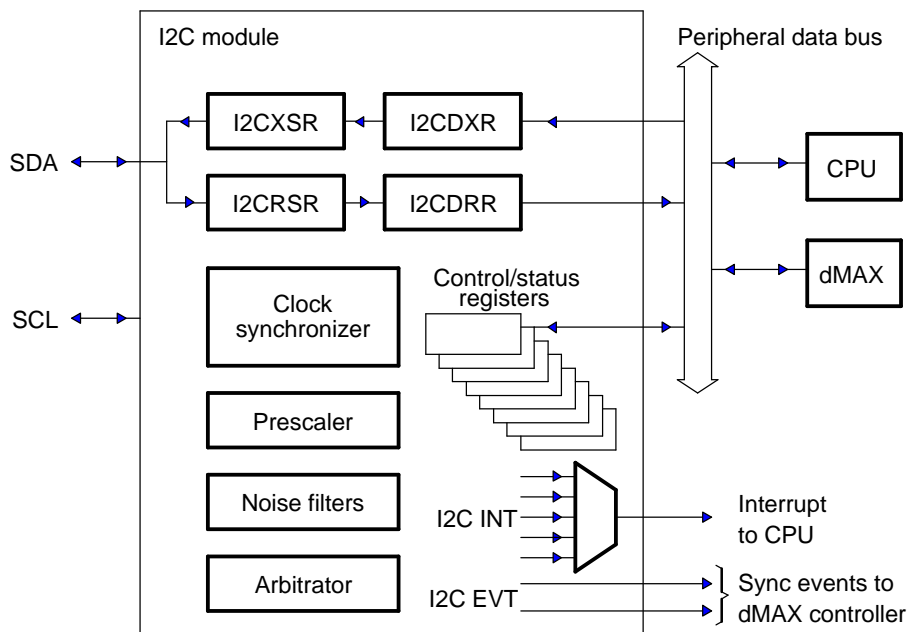
For data communication, the I2C module has a serial data pin (SDA) and a serial clock pin (SCL), as shown in [Figure 2](#). These two pins carry information between the C672x device and other devices connected to the I²C-bus. The SDA and SCL pins are both bidirectional. Each must be connected to a positive 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 dMAX controller
- Control and status registers
- A peripheral data bus interface to enable the CPU and the dMAX controller to access the I2C module registers
- A clock synchronizer to synchronize the I2C input clock (SYSCLK2 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 (SYSCLK2) 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
- dMAX event generation logic, so that activity in the dMAX 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 dMAX 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 I2CXHR and shifted out on the SDA pin one bit at a time. When the I2C module is configured as a receiver, received data is shifted into I2CHSR and then copied to I2CDRR.

Figure 2. I2C Module Conceptual Block Diagram



3 Operational Details

3.1 Electrical Considerations

3.1.1 Input and Output Voltage Levels

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

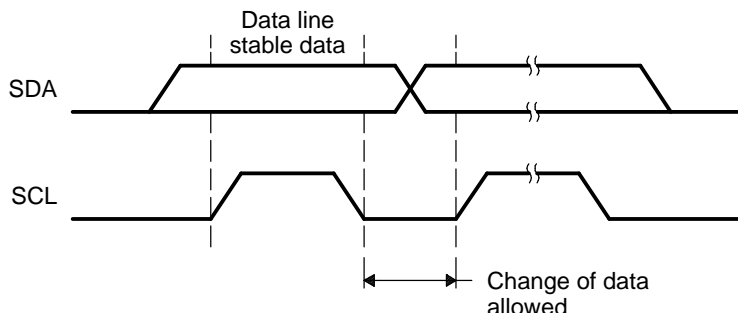
3.1.2 Buffers

Note that the C672x DSP uses strong (± 8 mA) buffers. For this reason, series resistors may be needed on the SDA and SCL pins to slow down the clock/data edges.

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 I²C-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

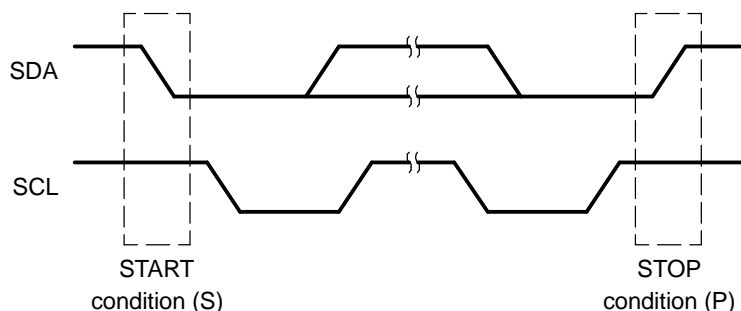
Operating Mode	Description
Slave-receiver mode	The I2C module is a slave and receives data from a master. All slave modules begin in this mode. In this mode, serial data bits received on SDA are shifted in with the clock pulses that are generated by the master. As a slave, the I2C module does not generate 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. The I2C module enters its slave-transmitter mode, if the slave address (7-bit or 10-bit) 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. The I2C module enters its master-receiver mode after transmitting the slave address (7-bit or 10-bit) 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 I²C-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 I²C-bus is considered busy, and the bus busy (BB) bit of I2CSTR is 1. Between a STOP condition and the next START condition, the bus is considered free, and BB is 0.

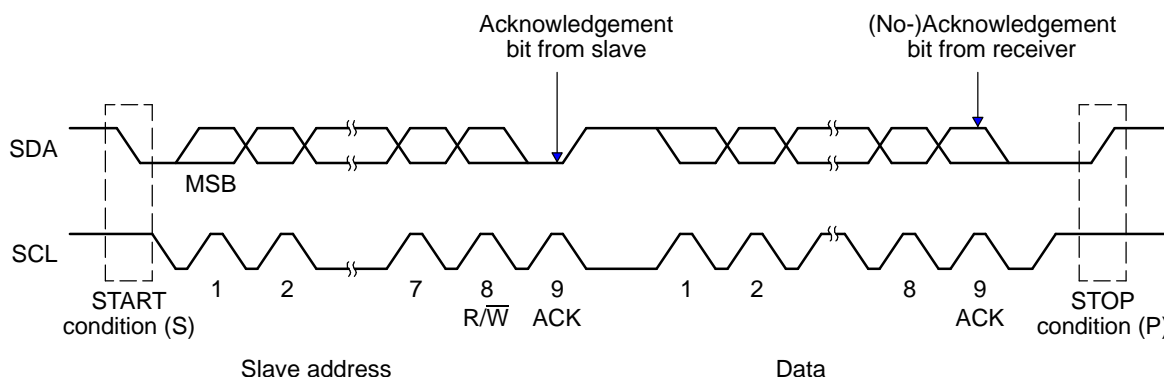
For the I²C module to start a data transfer with a START condition, the master mode (MST) bit and the START condition (STT) bit in I2CMDR must both be 1. For the I²C module to end a data transfer with a STOP condition, the STOP condition (STP) bit must be set to 1. When BB is set to 1 and STT is set to 1, a repeated START condition is generated. For a description of I2CMDR (including the MST, STT, and STP bits), see [Section 9.9](#).

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

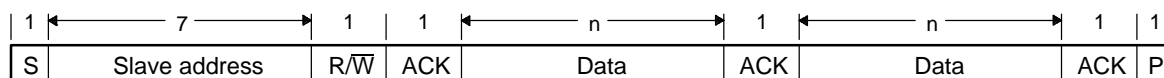
In the 7-bit addressing format (Figure 6), the first byte after a START condition (S) consists of a 7-bit slave address followed by a R/W bit. The R/W bit determines the direction of the data:

- $R/\overline{W} = 0$: 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/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/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)



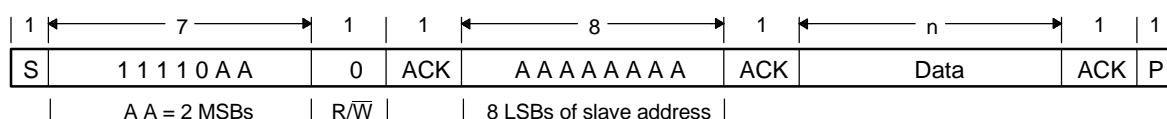
(1) n = The number of data bits (from 1 to 8) specified by the bit count (BC) field of I2CMDR.

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/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 I²C-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)



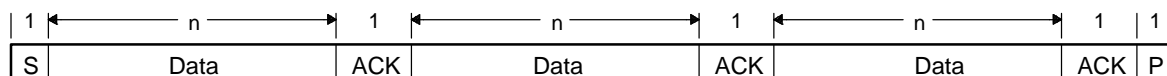
(1) n = The number of data bits (from 1 to 8) specified by the bit count (BC) field of I2CMDR.

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)

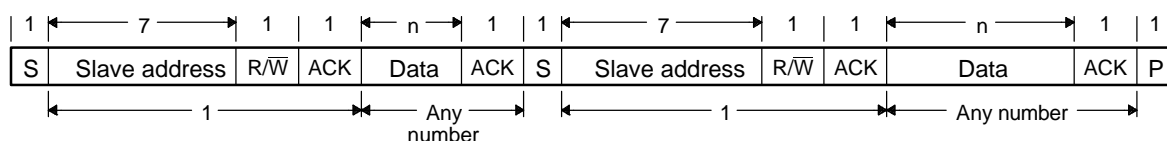


(1) n = The number of data bits (from 1 to 8) specified by the bit count (BC) field of I2CMDR.

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)



(1) 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

I2C Module Condition	NACK Bit Generation	
	Basic	Optional
Slave-receiver mode	<ul style="list-style-type: none"> 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)	<ul style="list-style-type: none"> 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 I2CMDR)	<ul style="list-style-type: none"> 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

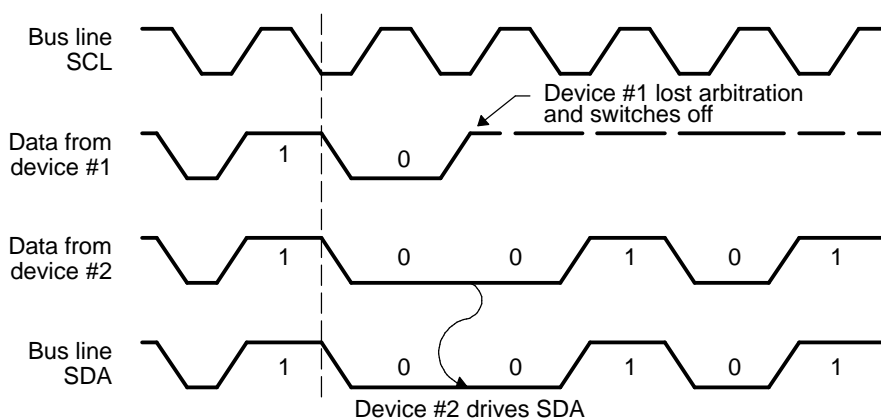
If two or more master-transmitters simultaneously start a transmission on the same bus, an arbitration procedure is invoked. The arbitration procedure uses the data presented on the serial data bus (SDA) by the competing transmitters. Figure 10 illustrates the arbitration procedure between two devices.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



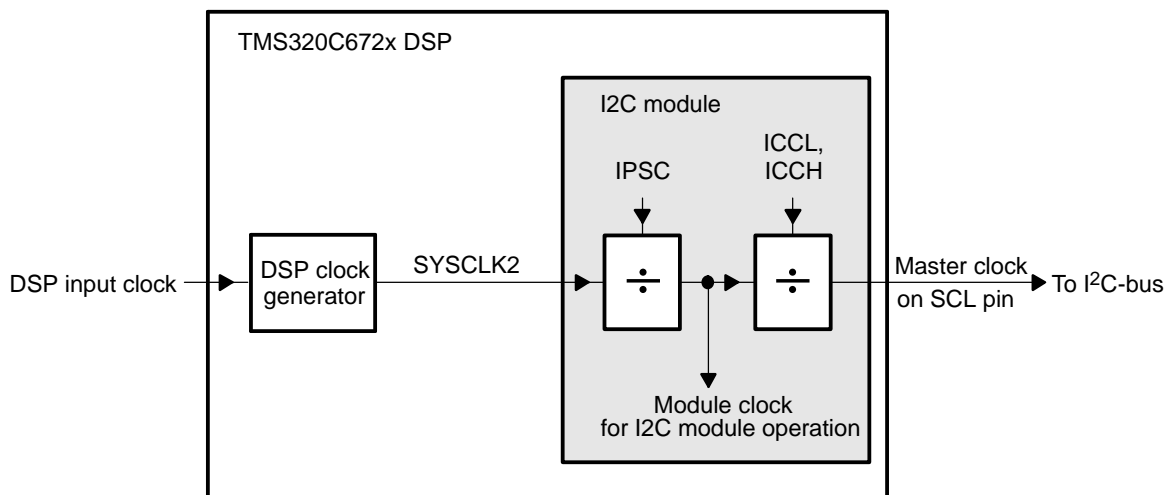
3.8 Clock Generation

As shown in Figure 11, the DSP clock generator receives a signal from an external clock source and produces an **I2C input clock** (SYSCLK2) with a programmed frequency. SYSCLK2 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 C672x DSP. (See the device-specific data manual for generation of SYSCLK2). SYSCLK2 is then divided twice more inside the I2C module to produce the module clock and the master clock.

The **module clock** determines the frequency at which the I2C module operates. The I2C module must be operated with a module clock frequency of 7 to 12 MHz. Figure 11 shows how this clock is generated. A programmable prescaler in the I2C module divides down SYSCLK2 to produce the module clock. To specify the divide-down value, initialize the IPSC field of the I2C prescaler register, I2CPSC (described in Section 9.12). The resulting frequency is:

$$\text{module clock frequency} = \frac{\text{SYSCLK2 frequency}}{(\text{IPSC} + 1)}$$

Figure 11. Clocking Diagram for the I2C Module



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.

CAUTION

Module Clock Frequency Range

The I2C module must be operated with a module clock frequency of 7 to 12 MHz. The I2C prescaler register (I2CPSC) must be configured to this frequency range.

The **master clock** appears on the SCL pin when the I2C module is configured to be a master on the I²C-bus. This clock controls the timing of communication between the I2C module and a slave. As shown in [Figure 11](#), a second clock divider in the I2C module divides down the module clock to produce the master clock. The clock divider uses 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, as described in [Section 9.4](#). The resulting frequency is:

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

Where d depends on the IPSC value in I2CPSC:

IPSC value	d
0	7
1	6
2h-FFh	5

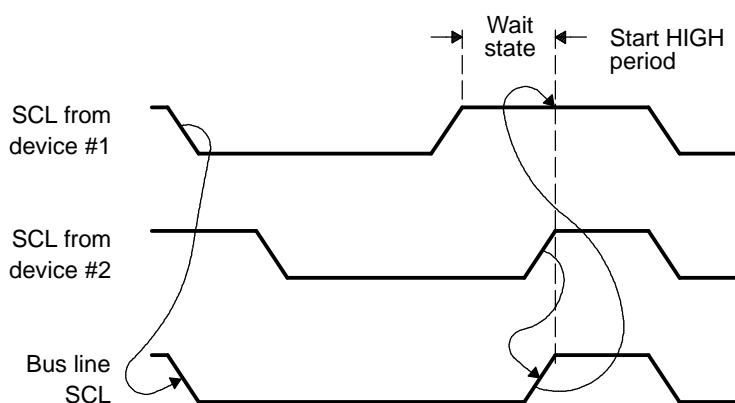
Note that the equation **does not include rise and fall time effects** and may need to be modified depending on bus capacitance.

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 a 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 for transmission.

Figure 12. Synchronization of Two I²C Clock Generators During Arbitration



4 Events Generated by the I2C Module

4.1 Interrupt Requests

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. Then the ISR can branch to the appropriate subroutine.

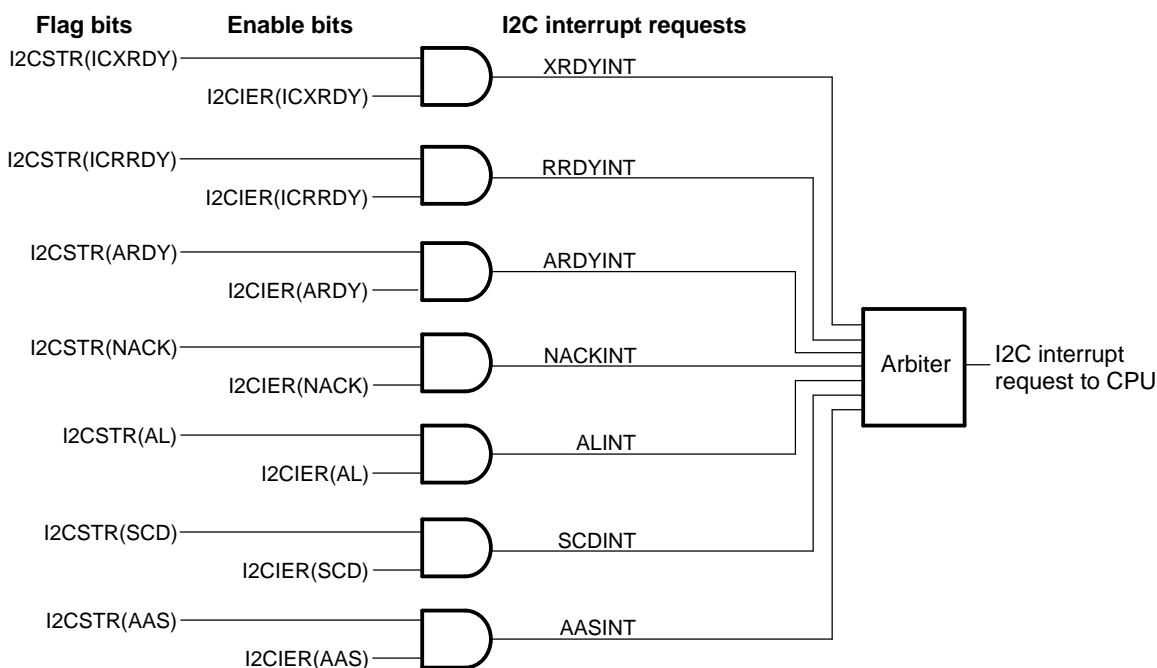
After the CPU reads I2CISR, the following events occur:

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 9.3).
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 9.3).
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 9.3). As an alternative to using ARDYINT, the CPU can poll the ARDY bit of I2CSTR (described in Section 9.3).
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 9.3).
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 9.3).
SCDINT	Stop-condition detected: The I2C module has detected a STOP condition. As an alternative to using SCDINT, the CPU can poll the SCD bit of I2CSTR (described in Section 9.3).
AASINT	Address-as-slave condition: The I2C module has recognized its own slave address or an address of all zeroes (general call). As an alternative to using AASINT, the CPU can poll the AAS bit of I2CSTR (described in Section 9.3).

Figure 13. Enable Paths of I2C Interrupt Requests



4.2 dMAX Events

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

4.3 Transmit Data Ready Interrupt/DMA Event

The C672x DSP generates the transmit data ready interrupt in two different ways when operating in slave-transmitter mode through the use of the transmit data ready interrupt mode (XRDYM) bit in the I2C extended mode register (I2CEMDR). The I2C module enters slave-transmitter mode, if the slave address transmitted on the I2C bus is the same as its own address (in I2COAR) and the master has transmitted R/W = 1. After the I2C has entered slave-transmitter mode, it generates transmit data ready interrupts (if enabled in I2CIER) to the CPU and DMA events based on the setting specified through the XRDYM bit.

When XRDYM = 1, the first transmit data ready interrupt and event are generated after the I2C module has recognized its slave address and R/W was transmitted as 1 by the master device. All subsequent transmit data ready interrupts and events are generated when the data in the data transmit register (I2CDXR) is copied to the transmit shift register (I2CXSR). The I2C module keeps copying data from I2CDXR to I2CXSR as long as the master keeps generating acknowledge signals to request more data. [Figure 14](#) shows the operation of the transmit data ready interrupt when XRDYM = 1.

When the XRDYM = 0, the first transmit data ready interrupt and event are generated after the I2C module has recognized its slave address and R/W was transmitted as 1 by the master device. All subsequent transmit data ready interrupts and events are generated when the master generates an acknowledge signal to request more data. [Figure 15](#) shows the operation of the transmit data ready interrupt when XRDYM = 0.

The XRDYM bit only has an effect when the I2C module is operating in slave-transmitter mode.

CAUTION

Code Compatibility

Devices other than the C672x DSP operate as if the XRDYM bit is set to 1. Clearing the XRDYM bit to 0 may have an undesired effect when porting I2C module code from other devices. The XRDYM bit should remain as 1 when code compatibility is desired.

Figure 14. Interrupt Generation When XRDYM = 1

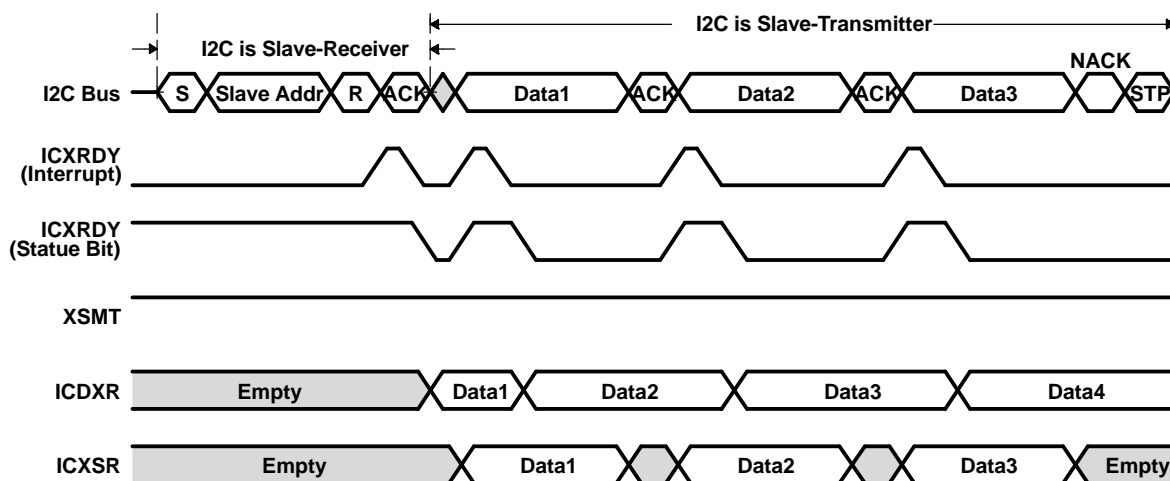
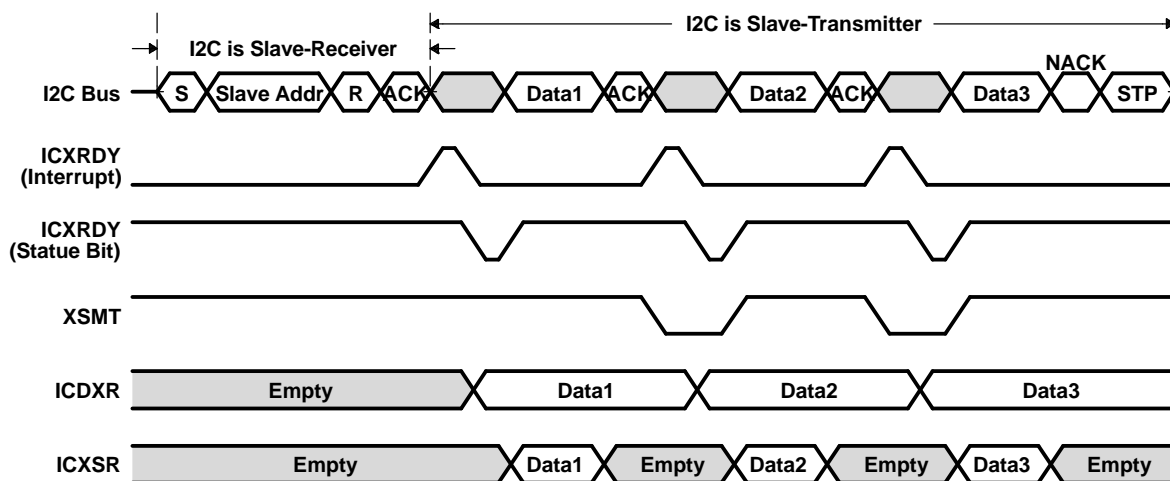


Figure 15. Interrupt Generation When XRDYM = 0



5 GPIO Pin Control

The SDA and SCL pins of the I2C module can be used for general-purpose input/output (GPIO). To use the GPIO mode of the I2C pins:

1. Place the I2C module in reset by clearing the IRS bit to 0 in the I2C mode register (I2CMDR).
2. Enable the GPIO mode by setting the GPMODE bit to 1 in the I2C pin function register (I2CPFUNC).

Some DSPs may require pull-up resistors on the SDA and SCL pins in order to use the GPIO mode. See your device-specific data manual to determine if pull-up resistors are necessary for your DSP.

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 module clock for the operation of the I2C module (see [Section 9.12](#)). 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 dMAX controller for transmit/receive data, enable the dMAX and program the dMAX controller.

7.1.2 Initialization Procedure

Configure the I2C mode register (I2CMDR) (see [Section 9.9](#)).

7.1.3 Program Clock Control Registers (I2CCLKL and I2CCLKH)

Program the I2C module clock (I2CCLKL and I2CCLKH) to obtain the desired SCL clock timing (see [Section 9.4](#)). 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 9.1](#)))
2. Configure the slave address register (I2CSAR (see [Section 9.7](#)))

7.1.5 Program Transmit Data Register (I2CDXR)

If in master-transmitter mode, program the data transmit register (I2CDXR) (see [Section 9.8](#)).

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 9.3](#)).

7.1.7 Poll Receive Data

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

7.1.8 Poll Transmit Data

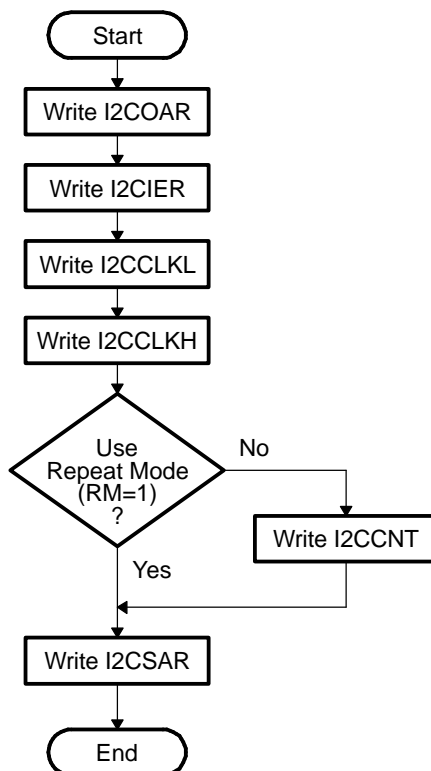
Poll the transmit-data-ready interrupt (ICXRDY) bit in the I2C status register (I2CSTR), use the XRDY interrupt, or use the dMAX controller to write data into the data transmit register (I2CDXR).

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 ready and resolve accordingly.
5. Test for transmit-data ready and resolve accordingly.
6. Test for stop-condition and resolve accordingly.
7. Test for address-as-slave condition and resolve accordingly.

7.3 Flow Diagrams

Figure 16. Setup Procedure



7.3.1 Bit Polling Methods to Control Data Flow

Figure 17. Using Bit Polling for Master-Transmitter Operation, Repeat Mode (RM = 1)

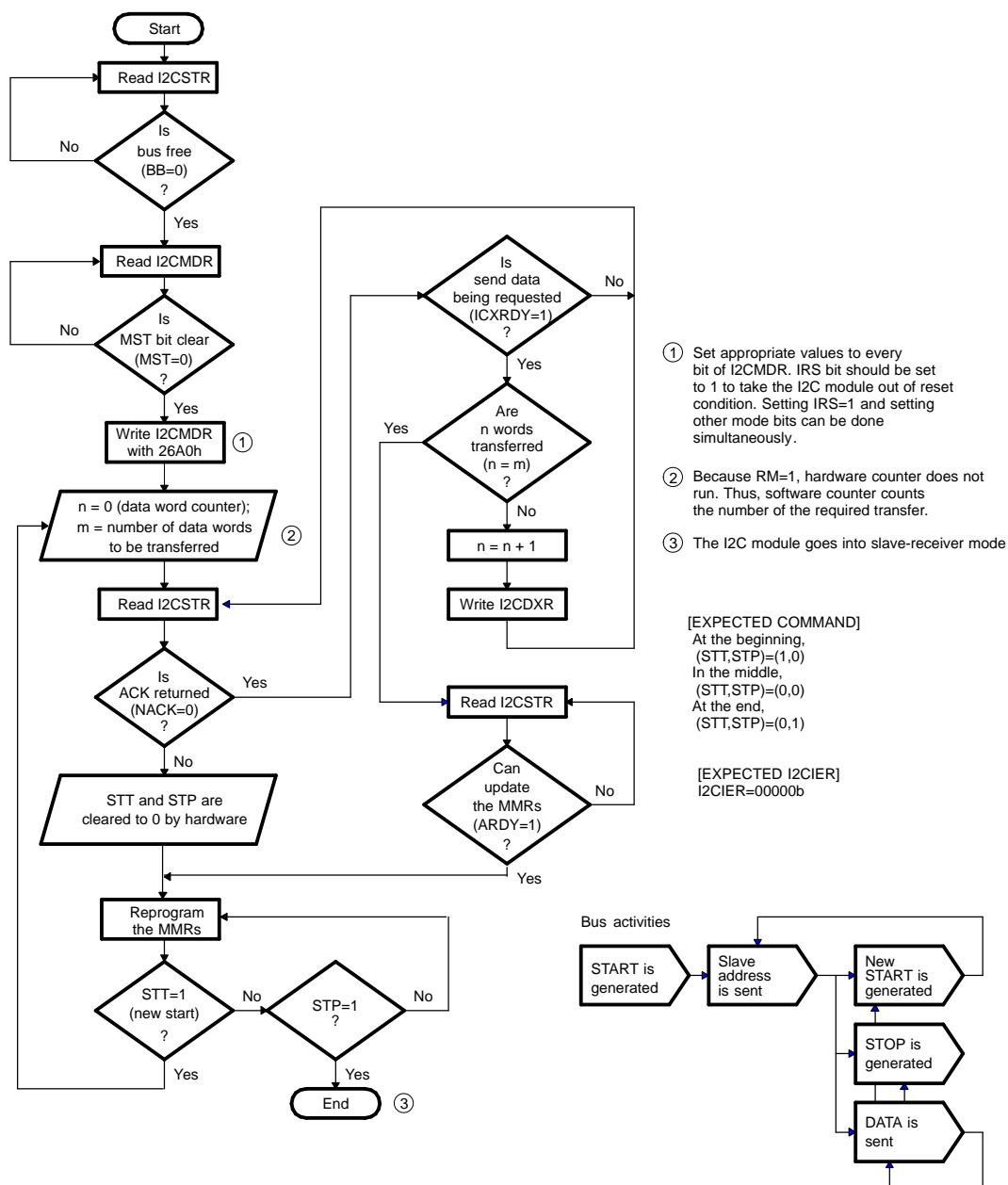


Figure 18. Using Bit Polling for Master-Transmitter Operation, Nonrepeat Mode (RM = 0)

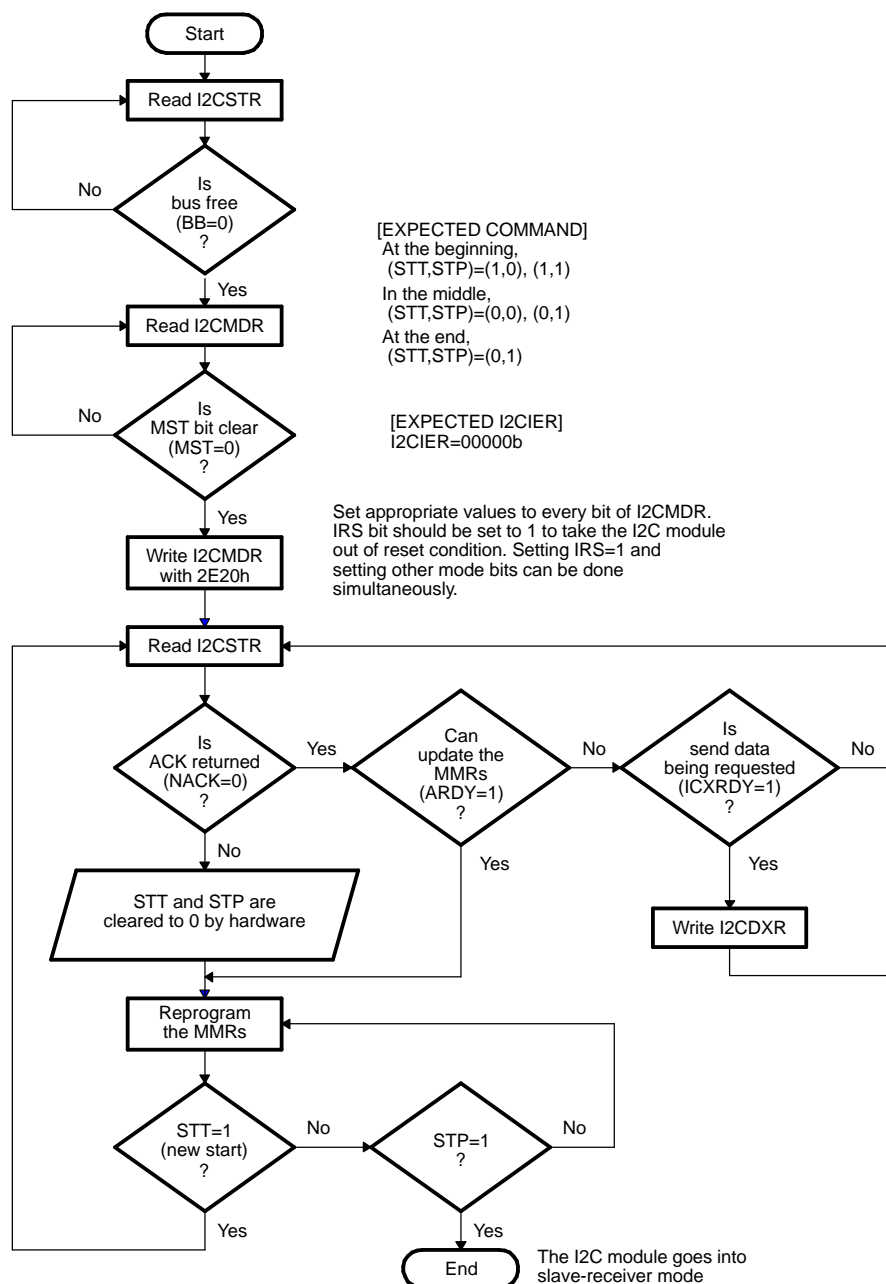


Figure 19. Using Bit Polling for Master-Receiver Operation, Repeat Mode (RM = 1), Fixed Number of Data Words

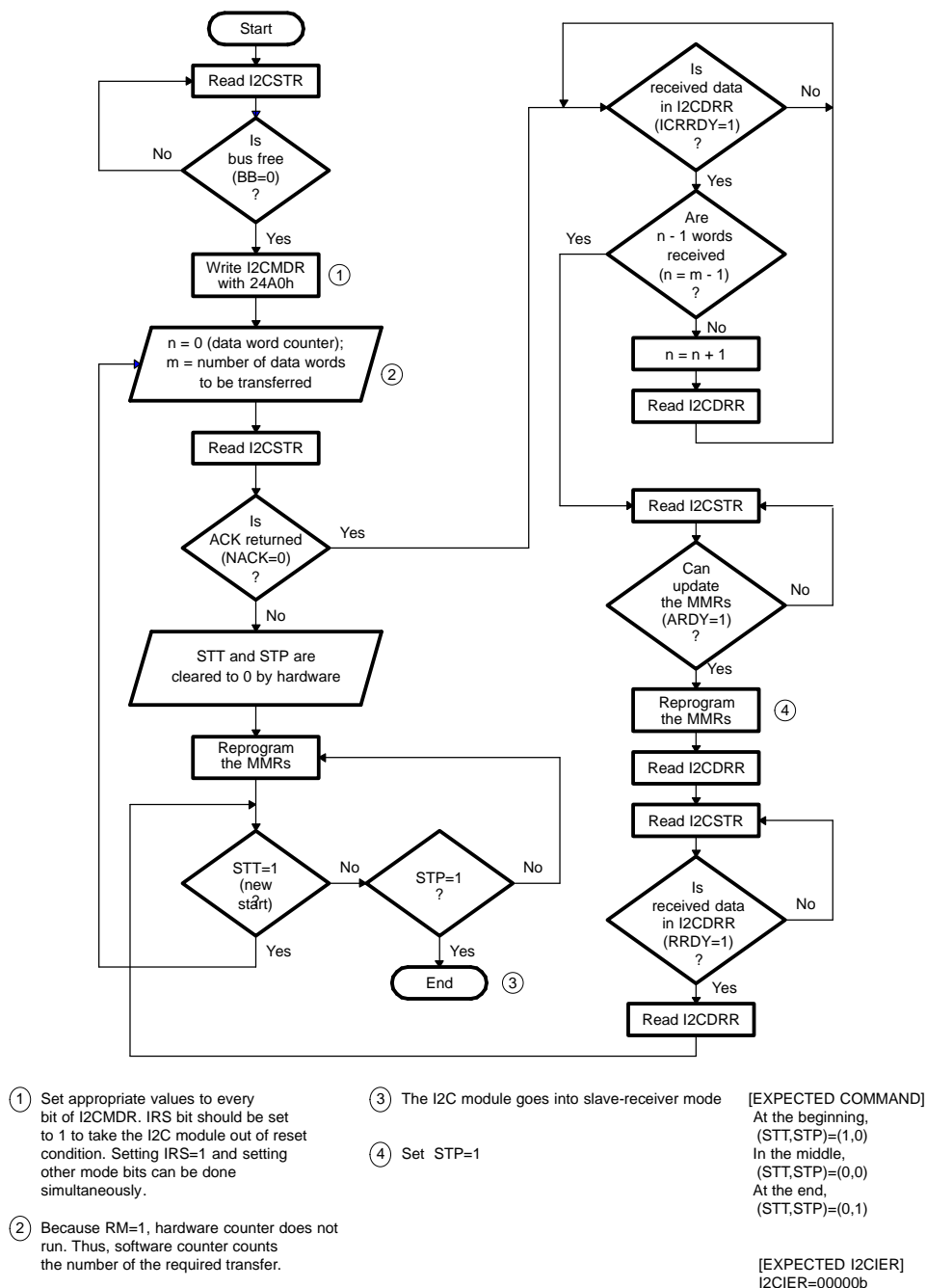


Figure 20. Using Bit Polling for Master-Receiver Operation, Repeat Mode (RM = 1), Variable (Data-Dependent) Number of Data Words

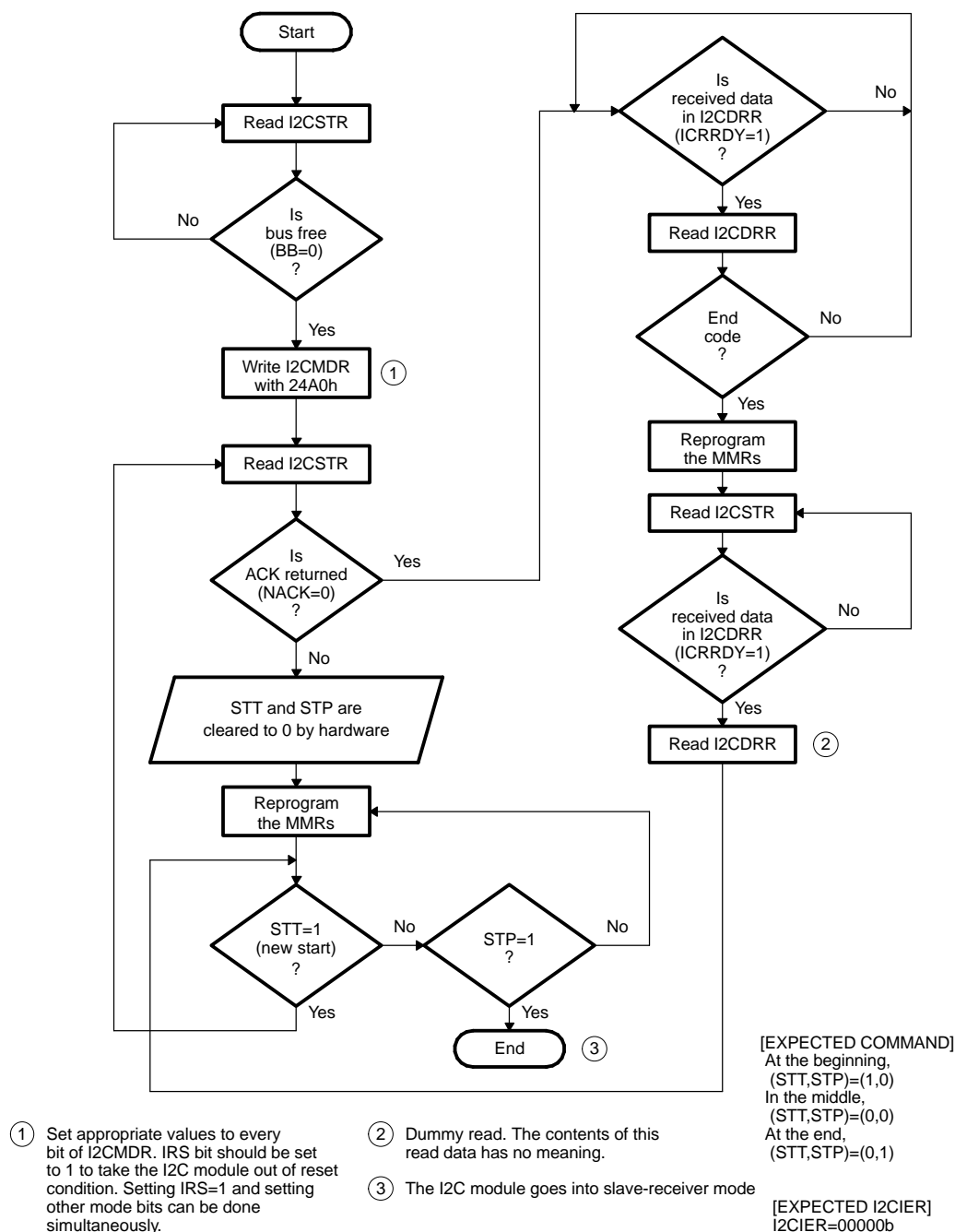


Figure 21. Using Bit Polling for Master-Receiver Operation, Nonrepeat Mode (RM = 0)

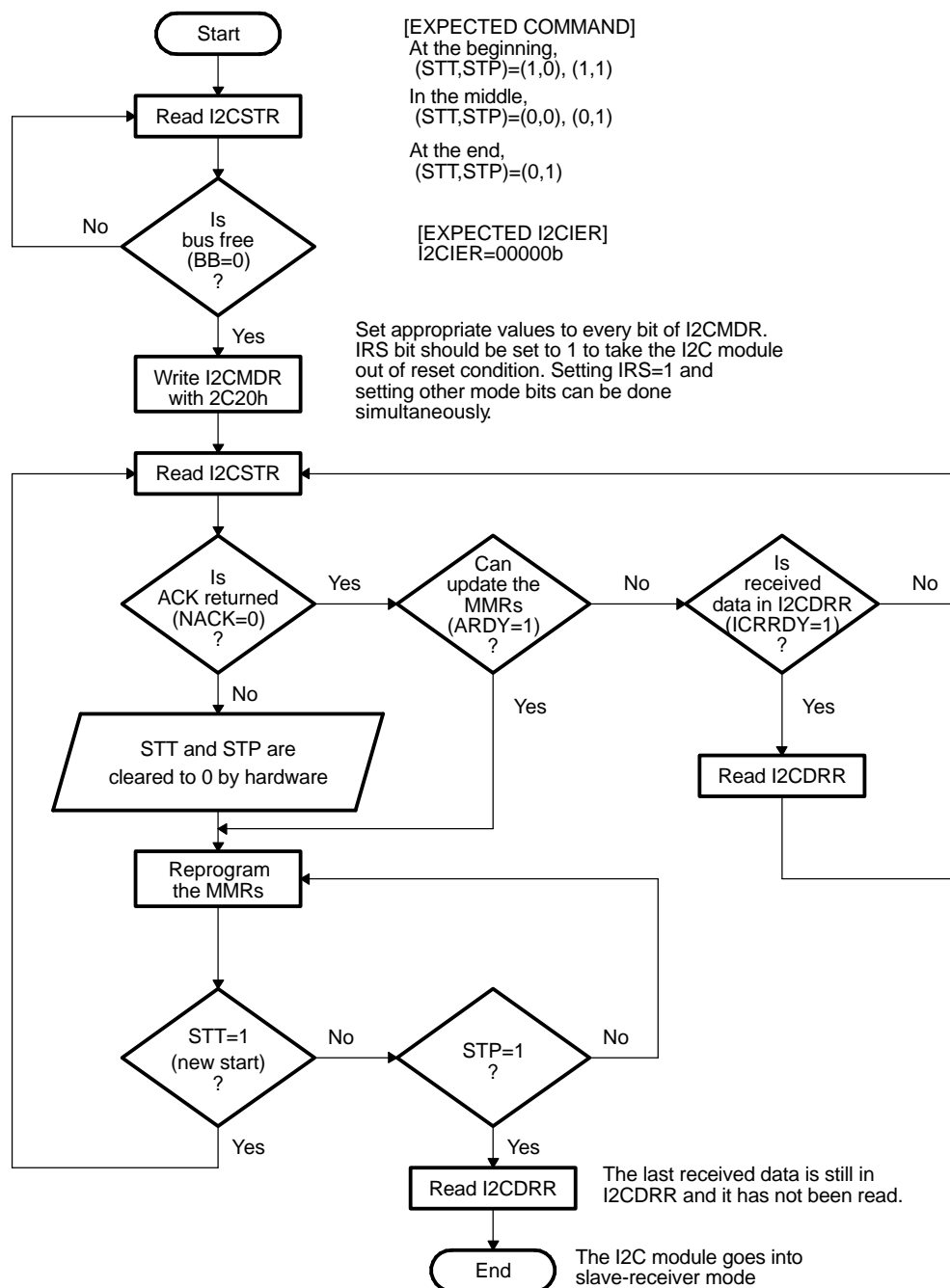
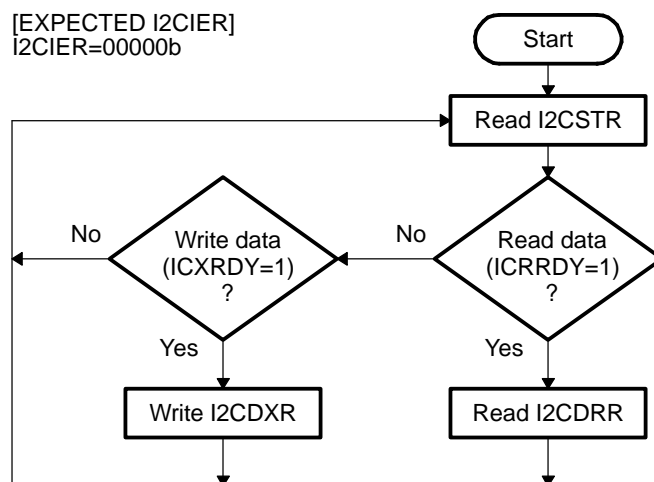


Figure 22. Using Bit Polling for Slave-Transmitter/Receiver Operation, Repeat Mode (RM = 1)



7.3.2 Interrupt Methods to Control Data Flow

Figure 23. Using Interrupts for Master-Transmitter Operation, Nonrepeat Mode (RM = 0)

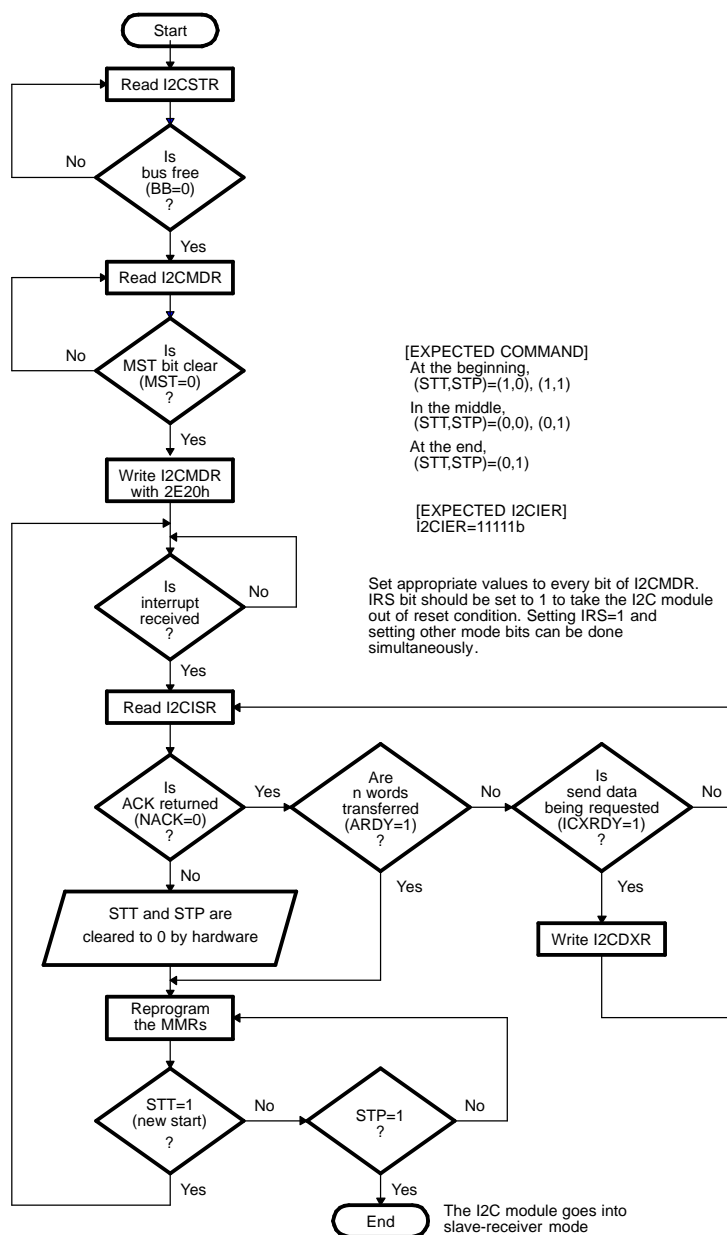


Figure 24. Using Interrupts for Master-Receiver Operation, Nonrepeat Mode (RM = 0)

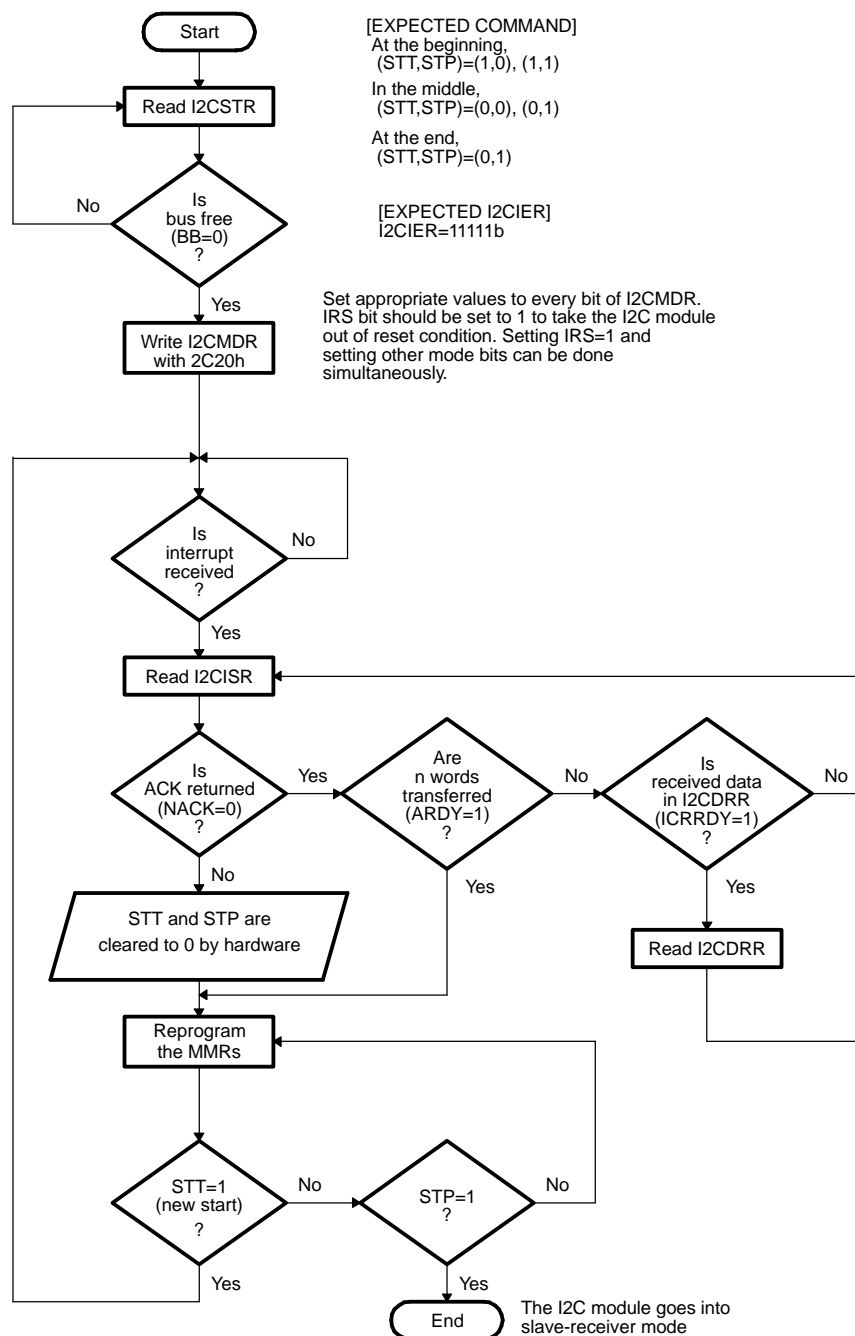


Figure 25. Using Interrupts for Master-Transmitter/Receiver Operation, Repeat Mode (RM = 1)

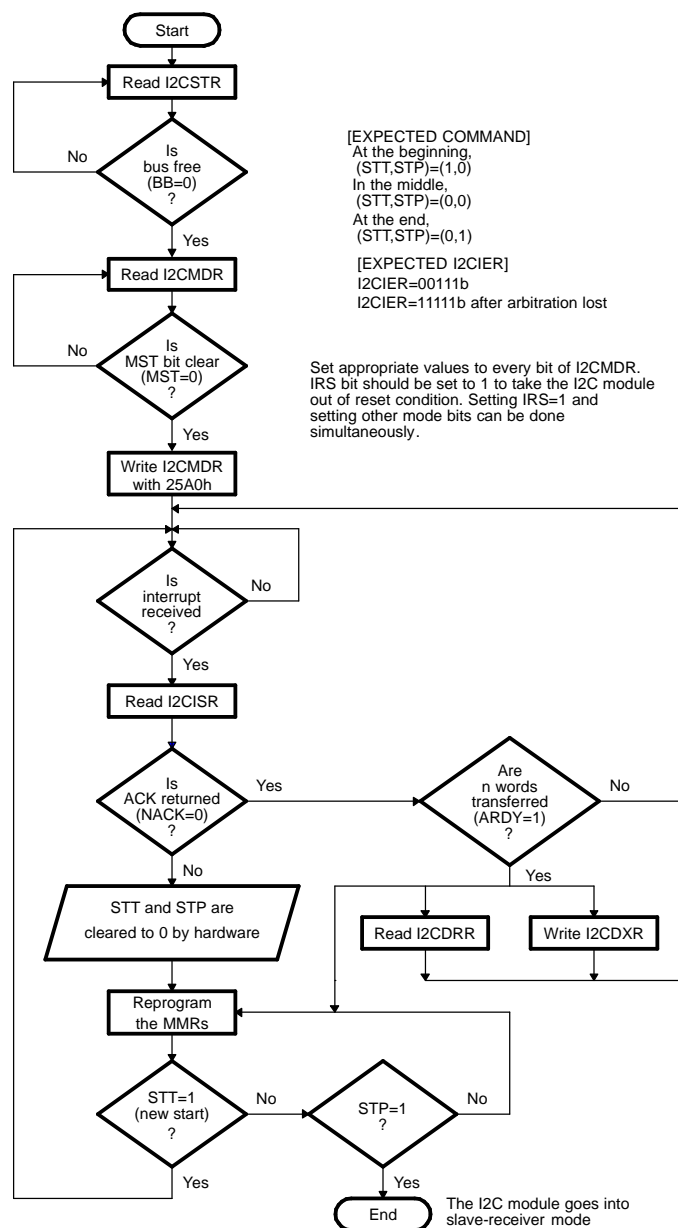
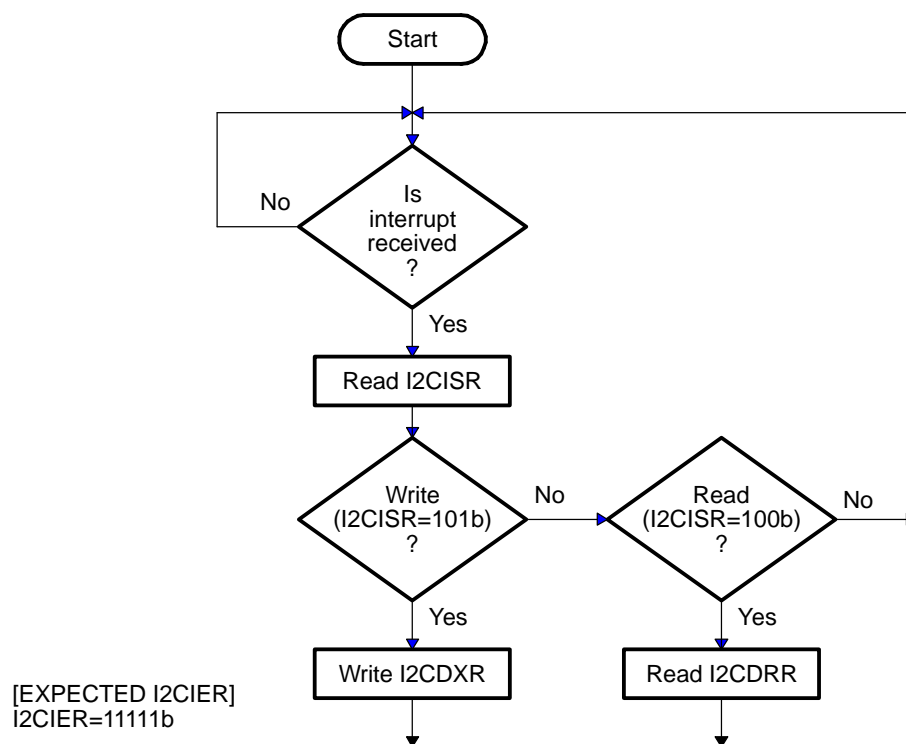


Figure 26. Using Interrupts for Slave-Transmitter/Receiver Operation, Repeat Mode (RM = 1)



7.3.3 dMAX Event Methods to Control Data Flow

Figure 27. Using dMAX Events for Master-Transmitter Operation, Nonrepeat Mode (RM = 0)

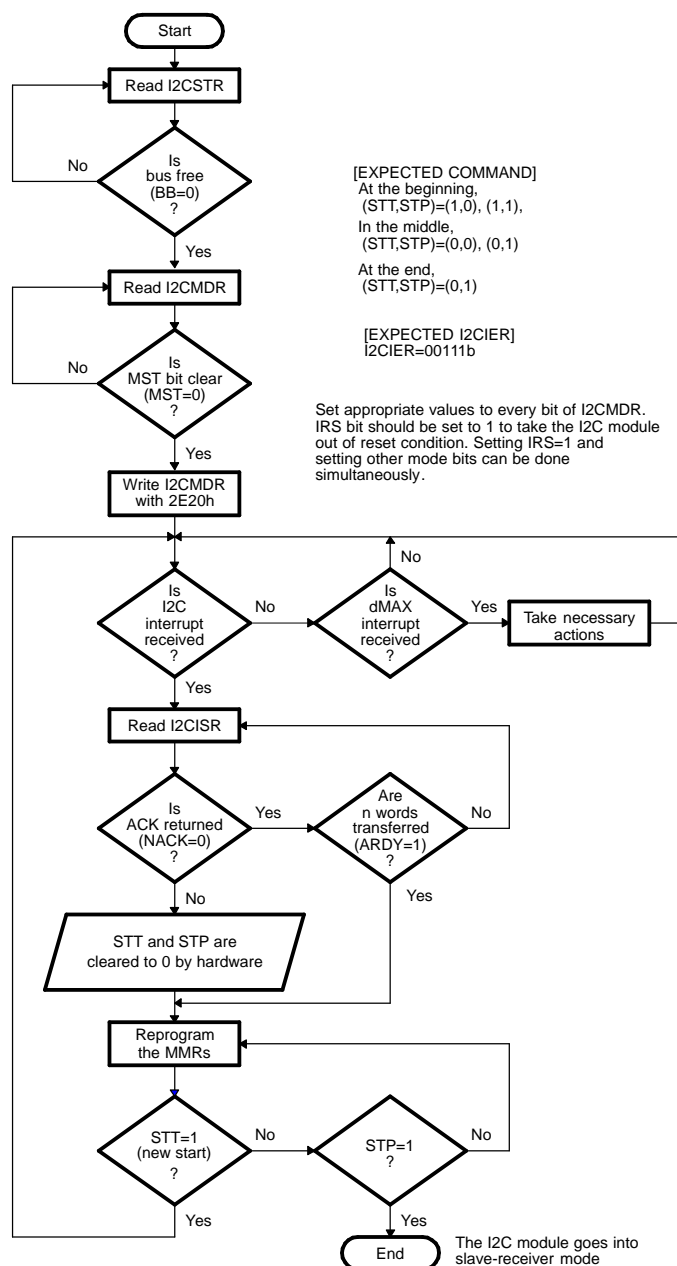
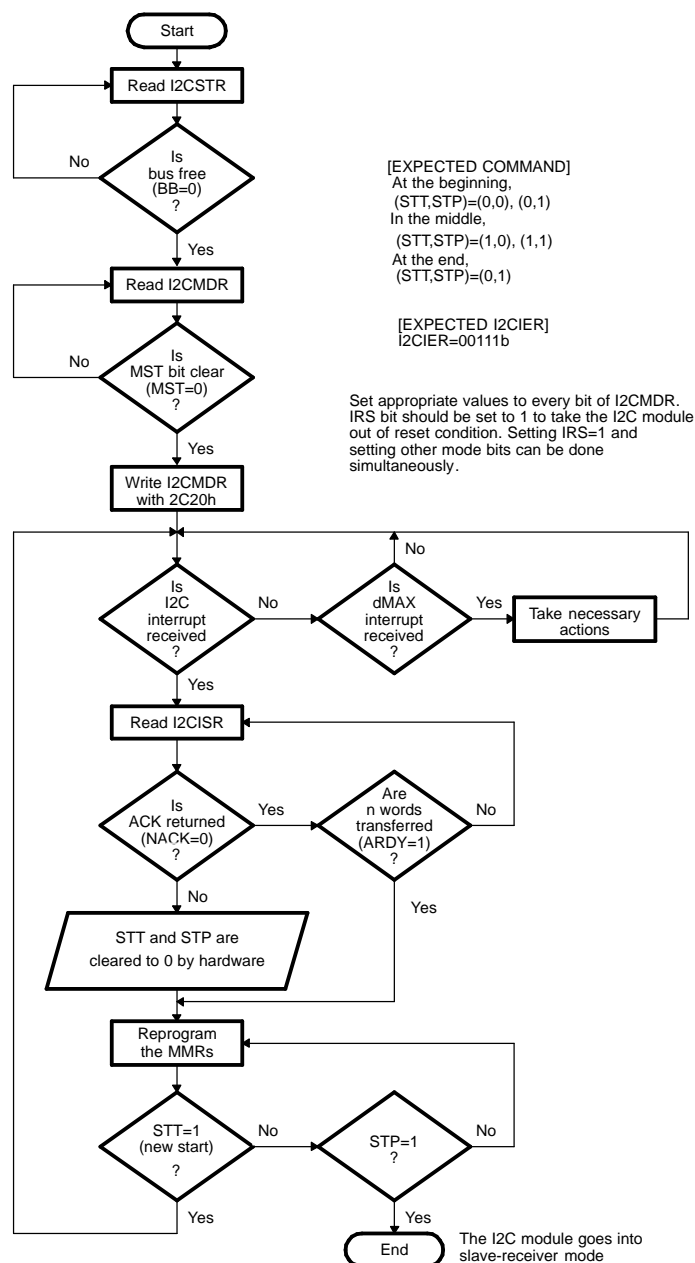


Figure 28. Using dMAX Events for Master-Receiver Operation, Nonrepeat Mode (RM = 0)



8 Emulation Considerations

The emulation suspend feature is not supported on the C672x devices. Since emulation suspend is not supported, the FREE bit in the I2C mode register (I2CMDR) has no effect on the I2C module. The I2C module will always run free, regardless of the value in the FREE bit.

Having the I2C data receive register (I2CDRR) open in a memory window in the Code Composer Studio™ integrated development environment acts as a read of I2CDRR. The receive-data-ready interrupt (ICRRDY) bit in the I2C status register (I2CSTR) is set to 1 when I2CDRR is ready to be read, but the ICRRDY bit is cleared to 0 when I2CDRR has been read. Since having I2CDRR open in a memory window acts as a read of I2CDRR, this action clears the ICRRDY bit and typically results in a missed byte of data. Any operation after this action is not assured to be reliable.

For debugging purposes, you are advised not to view the I2C registers in a Code Composer Studio memory window. The I2C registers may be read using C6000™ DSP instructions and using the watch windows or log windows to check the register values.

9 Registers

Table 4 lists the memory-mapped registers for the inter-integrated circuit (I2C). See the device-specific data manual for the memory address of these registers. All other register offset addresses not listed in Table 4 should be considered as reserved locations and the register contents should not be modified.

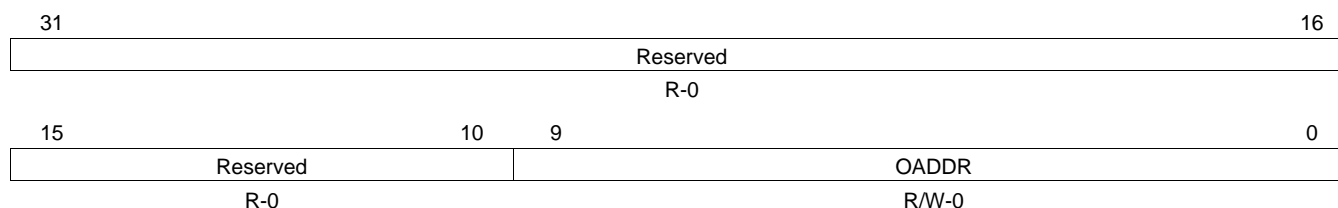
Table 4. Inter-Integrated Circuit (I2C) Registers

Offset	Acronym	Register Description	Section
00h	I2COAR	I2C own address register	Section 9.1
04h	I2CIER	I2C interrupt enable register	Section 9.2
08h	I2CSTR	I2C interrupt status register	Section 9.3
0Ch	I2CCLKL	I2C clock low-time divider register	Section 9.4
10h	I2CCLKH	I2C clock high-time divider register	Section 9.4
14h	I2CCNT	I2C data count register	Section 9.5
18h	I2CDRR	I2C data receive register	Section 9.6
1Ch	I2CSAR	I2C slave address register	Section 9.7
20h	I2CDXR	I2C data transmit register	Section 9.8
24h	I2CMDR	I2C mode register	Section 9.9
28h	I2CISR	I2C interrupt source register	Section 9.10
2Ch	I2CEMDR	I2C extended mode register	Section 9.11
30h	I2CPSC	I2C prescaler register	Section 9.12
34h	I2CPID1	I2C peripheral identification register 1	Section 9.13
38h	I2CPID2	I2C peripheral identification register 2	Section 9.13
48h	I2CPFUNC	I2C pin function register	Section 9.14
4Ch	I2CPDIR	I2C pin direction register	Section 9.15
50h	I2CPDIN	I2C pin data input register	Section 9.16
54h	I2CPDOUT	I2C pin data output register	Section 9.17
58h	I2CPDSET	I2C pin data set register	Section 9.18
5Ch	I2CPDCLR	I2C pin data clear register	Section 9.19

9.1 I2C Own Address Register (I2COAR)

The I2C own address register (I2COAR) is used to specify its own slave address, which distinguishes it from other slaves connected to the I2C-bus. If the 7-bit addressing mode is selected (XA = 0 in I2CMDR), only bits 6-0 are used; bits 9-7 are ignored. The I2COAR is shown in [Figure 29](#) and described in [Table 5](#).

Figure 29. I2C Own Address Register (I2COAR) [Offset = 00h]



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

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

Bit	Field	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	OADDR	0-3FFh	Own slave address. Provides the slave address of the I2C. In 7-bit addressing mode (XA = 0 in I2CMDR): Bits 6-0 provide the 7-bit slave address of the I2C. Bits 9-7 are ignored. In 10-bit addressing mode (XA = 1 in I2CMDR): Bits 9-0 provide the 10-bit slave address of the I2C.

9.2 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 30](#) and described [Table 6](#).

Figure 30. I2C Interrupt Enable Register (I2CIER) [Offset = 04h]

31																8																																															
Reserved																																																															
R-0																																																															
7								6								5								4								3								2								1								0							
Reserved								AAS								SCD								ICXRDY								ICRRDY								ARDY								NACK								AL							
R-0								R/W-0								R/W-0								R/W-0								R/W-0								R/W-0								R/W-0								R/W-0							

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

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

Bit	Field	Value	Description
31-7	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
6	AAS	0 1	Address-as-slave interrupt enable bit. Interrupt request is disabled. Interrupt request is enabled.
5	SCD	0 1	Stop-condition detected interrupt enable bit. Interrupt request is disabled. Interrupt request is enabled.
4	ICXRDY	0 1	Transmit-data-ready interrupt enable bit. Interrupt request is disabled. Interrupt request is enabled.
3	ICRRDY	0 1	Receive-data-ready interrupt enable bit. Interrupt request is disabled. Interrupt request is enabled.
2	ARDY	0 1	Register-access-ready interrupt enable bit. Interrupt request is disabled. Interrupt request is enabled.
1	NACK	0 1	No-acknowledgement interrupt enable bit. Interrupt request is disabled. Interrupt request is enabled.
0	AL	0 1	Arbitration-lost interrupt enable bit Interrupt request is disabled. Interrupt request is enabled.

9.3 I2C Interrupt 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 31](#) and described in [Table 7](#).

Figure 31. I2C Status Register (I2CSTR) [Offset = 08h]

31																16															
Reserved																															
R-0																															
15				14				13				12				11				10				9				8			
Reserved				SDIR				NACKSNT				BB				RSFULL				XSMT				AAS				AD0			
R-0				R/W1C-0				R/W1C-0				R/W1C-0				R-0				R-1				R-0				R-0			
7				6				5				4				3				2				1				0			
Reserved								SCD				ICXRDY				ICRRDY				ARDY				NACK				AL			
R-0								R/W1C-0				R/W1C-1				R/W1C-0				R/W1C-0				R/W1C-0				R/W1C-0			

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing 0 has no effect); -n = value after reset

Table 7. I2C Status Register (I2CSTR) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
14	SDIR	0 1	Slave direction bit. In digital-loopback mode (DLB), the SDIR bit is cleared to 0. I2C is acting as a master-transmitter/receiver or a slave-receiver. SDIR is cleared by one of the following events: <ul style="list-style-type: none"> A STOP or a START condition. SDIR is manually cleared. To clear this bit, write a 1 to it. I2C is acting as a slave-transmitter.
13	NACKSNT	0 1	No-acknowledgement sent bit. NACKSNT bit is used when the I2C 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 9.9). NACK is not sent. NACKSNT is cleared by one of the following events: <ul style="list-style-type: none"> It is manually cleared. To clear this bit, write a 1 to it. The I2C is reset (either when 0 is written to the IRS bit of I2CMDR or when the DSP is reset). NACK is sent. A no-acknowledge bit was sent during the acknowledge cycle on the I2C-bus.
12	BB	0 1	Bus busy bit. BB bit indicates whether the I2C-bus is busy or is free for another data transfer. In the master mode, BB is controlled by the software. Bus is free. BB is cleared by one of the following events: <ul style="list-style-type: none"> The I2C receives or transmits a STOP bit (bus free). BB is manually cleared. To clear this bit, write a 1 to it. The I2C is reset (either when 0 is written to the IRS bit of I2CMDR or when the DSP is reset). Bus is busy. When the STT bit in I2CMDR is set to 1, a restart condition is generated. BB is set by one of the following events: <ul style="list-style-type: none"> The I2C has received or transmitted a START bit on the bus. SCL is in a low state and the IRS bit in I2CMDR is 0.
11	RSFULL	0 1	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. No overrun is detected. RSFULL is cleared by one of the following events: <ul style="list-style-type: none"> I2CDRR is read. The I2C is reset (either when 0 is written to the IRS bit of I2CMDR or when the DSP is reset). Overrun is detected.

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

Bit	Field	Value	Description
10	XSMT	0 1	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. Underflow is detected. No underflow is detected. XSMT is set by one of the following events: <ul style="list-style-type: none">• Data is written to I2CDXR.• The I2C is reset (either when 0 is written to the IRS bit of I2CMDR or when the DSP is reset).
9	AAS	0 1	Addressed-as-slave bit. The AAS bit has been cleared by a repeated START condition or by a STOP condition. AAS is set by one of the following events: <ul style="list-style-type: none">• I2C has recognized its own slave address or an address of all zeros (general call).• The first data word has been received in the free data format (FDF = 1 in I2CMDR).
8	AD0	0 1	Address 0 bit. AD0 has been cleared by a START or STOP condition. An address of all zeros (general call) is detected.
7-6	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
5	SCD	0 1	Stop-condition detected bit. SCD indicates when a STOP condition has been detected on the I2C bus. The STOP condition could be generated by the I2C or by another I2C device connected to the bus. No STOP condition has been detected. SCD is cleared by one of the following events: <ul style="list-style-type: none">• CPU reads the INTCODE bits in I2CISR as 110b.• SCD is manually cleared. To clear this bit, write a 1 to it. A STOP condition has been detected.
4	ICXRDY	0 1	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. I2CDXR is not ready. ICXRDY is cleared by one of the following events: <ul style="list-style-type: none">• Data is written to I2CDXR.• ICXRDY is manually cleared. To clear this bit, write a 1 to it. I2CDXR is ready. Data has been copied from I2CDXR to I2CXSR. I2CXRDY is forced to 1 when the I2C is reset.
3	ICRRDY	0 1	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. I2CDRR is not ready. ICRRDY is cleared by one of the following events: <ul style="list-style-type: none">• I2CDRR is read (see Section 8).• ICRRDY is manually cleared. To clear this bit, write a 1 to it.• The I2C is reset (either when 0 is written to the IRS bit of I2CMDR or when the DSP is reset). I2CDRR is ready. Data has been copied from I2CRSR to I2CDRR.
2	ARDY	0 1	Register-access-ready interrupt flag bit (only applicable when the I2C is in the master mode). ARDY indicates that the I2C 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. The registers are not ready to be accessed. ARDY is cleared by one of the following events: <ul style="list-style-type: none">• The I2C starts using the current register contents.• ARDY is manually cleared. To clear this bit, write a 1 to it.• The I2C is reset (either when 0 is written to the IRS bit of I2CMDR or when the DSP is reset). The registers are ready to be accessed. <ul style="list-style-type: none">• In the nonrepeat mode (RM = 0 in I2CMDR): If STP = 0 in I2CMDR, ARDY is set when the internal data counter counts down to 0. If STP = 1, ARDY is not affected (instead, the I2C 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.

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

Bit	Field	Value	Description
1	NACK	0	No-acknowledgement interrupt flag bit. NACK applies when the I2C is a transmitter (master or slave). NACK indicates whether the I2C 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.
		1	ACK received/NACK is not received. NACK is cleared by one of the following events: <ul style="list-style-type: none">• 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 is reset (either when 0 is written to the IRS bit of I2CMDR or when the DSP is reset).
0	AL	0	NACK bit is received. The hardware detects that a no-acknowledge (NACK) bit has been received. Note: While the I2C performs a general call transfer, NACK is 1, even if one or more slaves send acknowledgement.
		1	Arbitration is lost. The hardware detects that a no-acknowledge (NACK) bit has been received. Note: While the I2C performs a general call transfer, NACK is 1, even if one or more slaves send acknowledgement.
0	AL	0	Arbitration-lost interrupt flag bit (only applicable when the I2C is a master-transmitter). AL primarily indicates when the I2C has lost an arbitration contest with another master-transmitter. The CPU can poll AL or use the AL interrupt request.
		1	Arbitration is not lost. AL is cleared by one of the following events: <ul style="list-style-type: none">• 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 is reset (either when 0 is written to the IRS bit of I2CMDR or when the DSP is reset).
		1	Arbitration is lost. AL is set by one of the following events: <ul style="list-style-type: none">• The I2C senses that it has lost an arbitration with two or more competing transmitters that started a transmission almost simultaneously.• The I2C attempts to start a transfer while the BB (bus busy) bit is set to 1.
			When AL is set to 1, the MST and STP bits of I2CMDR are cleared, and the I2C becomes a slave-receiver.

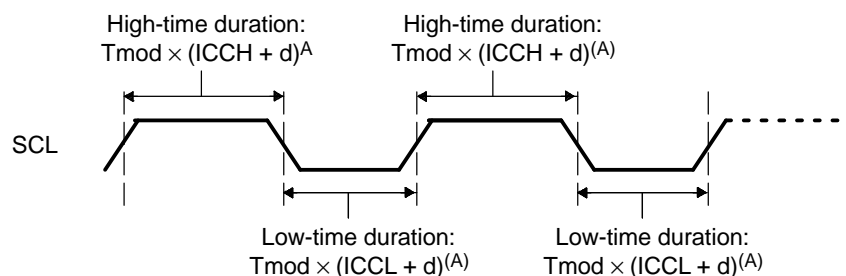
9.4 I2C Clock Divider Registers (I2CCLKL and I2CCLKH)

When the I2C is a master, the module clock is divided down for use as the master clock on the SCL pin. As shown in Figure 32, the shape of the master clock depends on two divide-down values, ICCL and ICCH.

The frequency of the master clock is calculated as:

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

Figure 32. Roles of the Clock Divide-Down Values (ICCL and ICCH)



A As described in Section 3.8, T_{mod} = module clock period = $1/\text{module clock frequency}$; $d = 5, 6, \text{ or } 7$.

9.4.1 I2C Clock Low-Time Divider Register (I2CCLKL)

The I2C clock low-time divider register (I2CCLKL) is shown in Figure 33 and described in Table 8. For each master clock cycle, I2CCL determines the amount of time the signal is low. I2CCLKL must be configured while the I2C is still in reset ($\text{IRS} = 0$ in I2CMDR).

Figure 33. I2C Clock Low-Time Divider Register (I2CCLKL) [Offset = 0Ch]

31	Reserved	16
R-0		
15	ICCL	0
R/W-0		

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

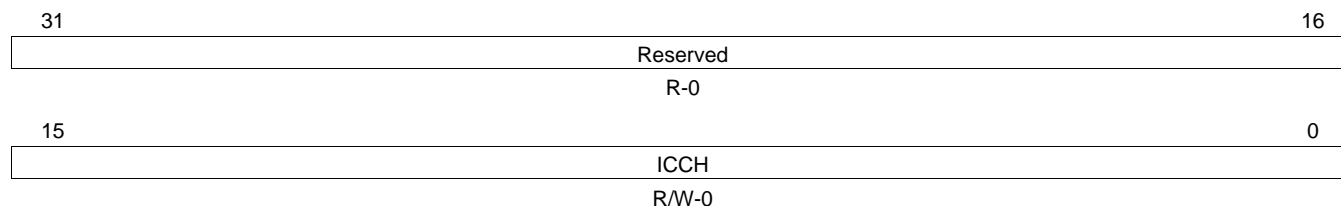
Table 8. I2C Clock Low-Time Divider Register (I2CCLKL) Field Descriptions

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.
15-0	ICCL	0-FFFFh	Clock low-time divide-down value of 1-65536. The period of the module clock is multiplied by $(\text{ICCL} + d)$ to produce the low-time duration of the master clock on the SCL pin.

9.4.2 I2C Clock High-Time Divider Register (I2CCLKH)

The I2C clock high-time divider register (I2CCLKH) is shown in [Figure 34](#) and described in [Table 9](#). For each master clock cycle, ICCH determines the amount of time the signal is high. I2CCLKH must be configured while the I2C is still in reset (IRS = 0 in I2CMMDR).

Figure 34. I2C Clock High-Time Divider Register (I2CCLKH) [Offset = 10h]



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

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

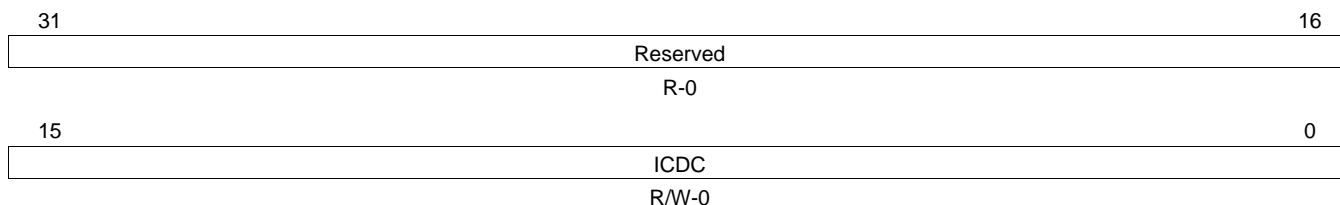
Bit	Field	Value	Description
31-16	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
15-0	ICCH	0-FFFFh	Clock high-time divide-down value of 1-65536. The period of the module clock is multiplied by (ICCH + d) to produce the high-time duration of the master clock on the SCL pin.

9.5 I2C Data Count Register (I2CCNT)

The I2C data count register (I2CCNT) is used to indicate how many data words to transfer when the I2C 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 35 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 terminates the transfer with a STOP condition when the countdown is complete (that is, when the last data word has been transferred).

Figure 35. I2C Data Count Register (I2CCNT) [Offset = 14h]



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

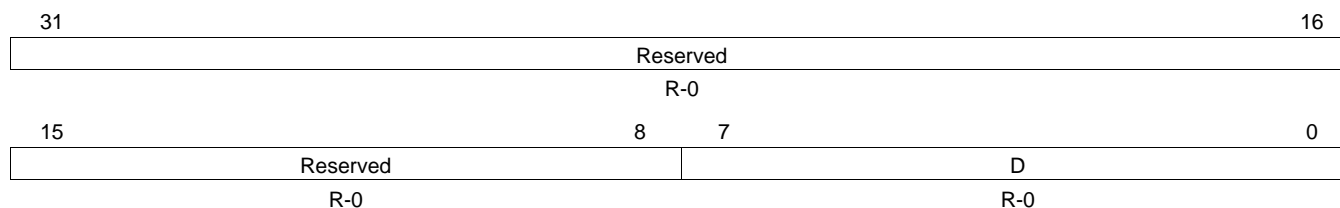
Table 10. I2C Data Count Register (I2CCNT) Field Descriptions

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.
15-0	ICDC	0-FFFFh	Data count value. When RM = 0 in I2CMDR, ICDC indicates the number of data words to transfer in the nonrepeat mode. When RM = 1 in I2CMDR, the value in I2CCNT is a don't care. If STP = 1 in I2CMDR, a STOP condition is generated when the internal data counter counts down to 0.
		0	The start value loaded to the internal data counter is 65536.
		1h- FFFFh	The start value loaded to internal data counter is 1-65535.

9.6 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 copies the contents of I2CRSR into I2CDRR. The CPU and the dMAX controller cannot access I2CRSR. The I2CDRR is shown in [Figure 36](#) and described in [Table 11](#).

Figure 36. I2C Data Receive Register (I2CDRR) [Offset = 18h]



LEGEND: R = Read only; -n = value after reset

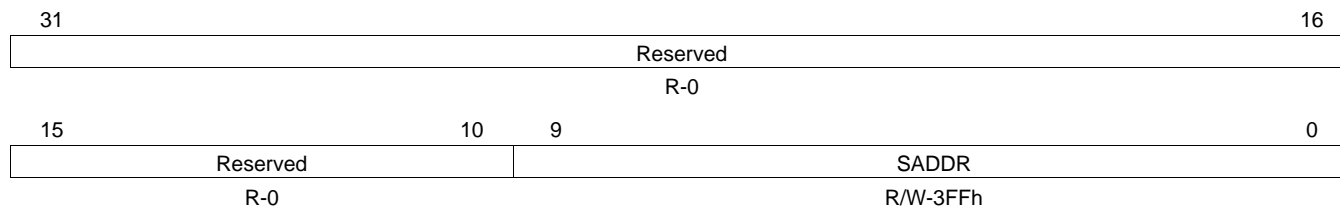
Table 11. I2C Data Receive Register (I2CDRR) Field Descriptions

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	D	0-FFh	Receive data.

9.7 I2C Slave Address Register (I2CSAR)

The I2C slave address register (I2CSAR) contains a 7-bit or 10-bit slave address. When the I2C 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 37](#) and described in [Table 12](#).

Figure 37. I2C Slave Address Register (I2CSAR) [Offset = 1Ch]



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

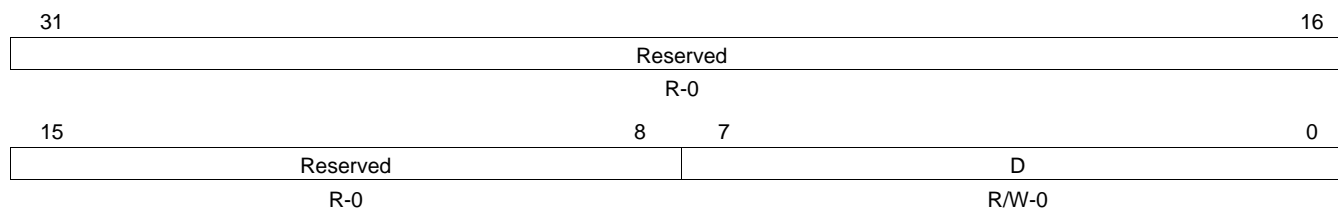
Table 12. I2C Slave Address Register (I2CSAR) Field Descriptions

Bit	Field	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	SADDR	0-3FFh	Slave address. Provides the slave address of the I2C. In 7-bit addressing mode (XA = 0 in I2CMDR): Bits 6-0 provide the 7-bit slave address that the I2C transmits when it is in the master-transmitter mode. Bits 9-7 are ignored. In 10-bit addressing mode (XA = 1 in I2CMDR): Bits 9-0 provide the 10-bit slave address that the I2C transmits when it is in the master-transmitter mode.

9.8 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 copies the contents of I2CDXR into the I2C transmit shift register (I2CXS). The I2CXS shifts out the transmit data from the SDA pin. The CPU and the dMAX controller cannot access I2CXS. The I2CDXR is shown in [Figure 38](#) and described in [Table 13](#).

Figure 38. I2C Data Transmit Register (I2CDXR) [Offset = 20h]



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 13. I2C Data Transmit Register (I2CDXR) 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	D	0-FFh	Transmit data.

9.9 I2C Mode Register (I2CMDR)

The I2C mode register (I2CMDR) contains the control bits of the I2C. The I2CMDR is shown in shown in Figure 39 and described in Table 14.

Figure 39. I2C Mode Register (I2CMDR) [Offset = 24h]

31																16															
Reserved																															
R-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-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/W = Read/Write; R = Read only; -n = value after reset

Table 14. I2C Mode Register (I2CMDR) 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	NACKMOD	0	No-acknowledge (NACK) mode bit (only applicable when the I2C is a receiver). In slave-receiver mode: The I2C sends an acknowledge (ACK) bit to the transmitter during each acknowledge cycle on the bus. The I2C only sends a no-acknowledge (NACK) bit if you set the NACKMOD bit.
		1	In master-receiver mode: The I2C sends an ACK bit during each acknowledge cycle until the internal data counter counts down to 0. When the counter reaches 0, the I2C sends a NACK bit to the transmitter. To have a NACK bit sent earlier, you must set the NACKMOD bit. In either slave-receiver or master-receiver mode: The I2C 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.
14	FREE	0	This emulation mode bit is not supported on C672x devices . The I2C module is always in run-free mode; therefore, FREE has no effect on the I2C module. See Section 8 . Stop-on-suspend mode. Not supported on C672x devices. I2C module is always in run-free mode.
		1	Run-free mode. The I2C runs free; that is, it continues to operate when a breakpoint occurs.
13	STT	0	START condition bit (only applicable when the I2C is a master). The RM, STT, and STP bits determine when the I2C starts and stops data transmissions (see Table 15). Note that the STT and STP bits can be used to terminate the repeat mode. In master mode: STT is automatically cleared after the START condition has been generated.
		1	In slave mode: if STT is 0, the I2C does not monitor the bus for commands from a master. As a result, the I2C performs no data transfers. In master mode: setting STT to 1 causes the I2C to generate a START condition on the I2C-bus. In slave mode: if STT is 1, the I2C monitors the bus and transmits/receives data in response to commands from a master.
12	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
11	STP	0	STOP condition bit (only applicable when the I2C is a master). The RM, STT, and STP bits determine when the I2C starts and stops data transmissions (see Table 15). Note that the STT and STP bits can be used to terminate the repeat mode. 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 counts down to 0.

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

Bit	Field	Value	Description
10	MST	0 1	Master mode bit. MST determines whether the I2C 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. See Table 16 . Slave mode. The I2C is a slave and receives the serial clock from the master. Master mode. The I2C is a master and generates the serial clock on the SCL pin.
9	TRX	0 1	Transmitter mode bit. When relevant, TRX selects whether the I2C is in the transmitter mode or the receiver mode. Table 16 summarizes when TRX is used and when it is a don't care. Receiver mode. The I2C is a receiver and receives data on the SDA pin. Transmitter mode. The I2C is a transmitter and transmits data on the SDA pin.
8	XA	0 1	Expanded address enable bit. 7-bit addressing mode (normal address mode). The I2C transmits 7-bit slave addresses (from bits 6-0 of I2CSAR), and its own slave address has 7 bits (bits 6-0 of I2COAR). 10-bit addressing mode (expanded address mode). The I2C 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	0 1	Repeat mode bit (only applicable when the I2C is a master-transmitter). The RM, STT, and STP bits determine when the I2C starts and stops data transmissions (see Table 15). If the I2C is configured in slave mode, the RM bit is don't care. Nonrepeat mode. The value in the data count register (I2CCNT) determines how many data words are received/transmitted by the I2C. Repeat mode. Data words are continuously received/transmitted by the I2C until the STP bit is manually set to 1, regardless of the value in I2CCNT.
6	DLB	0 1	Digital loopback mode bit (only applicable when the I2C is a master-transmitter). This bit disables or enables the digital loopback mode of the I2C. The effects of this bit are shown in Figure 40 . Note that DLB in the free data format mode (DLB = 1 and FDF = 1) is not supported. Digital loopback mode is disabled. 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: $n = ((\text{SYSCLK2 frequency} / \text{module clock frequency}) \times 8)$ The transmit clock is also the receive clock. The address transmitted on the SDA pin is the address in I2COAR.
5	IRS	0 1	I2C reset bit. Note that if IRS is reset during a transfer, it can cause the I2C bus to hang (SDA and SCL are in a high-impedance state). The I2C is in reset/disabled. When this bit is cleared to 0, all status bits (in I2CSTR) are set to their default values. The I2C is enabled.
4	STB	0 1	START byte mode bit (only applicable when the I2C 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 is a slave, the I2C ignores a START byte from a master, regardless of the value of the STB bit. The I2C is not in the START byte mode. The I2C is in the START byte mode. When the START condition bit (STT) is set, the I2C begins the transfer by generating: 1. A START condition 2. A START byte (0000 0001b) 3. A dummy acknowledge clock pulse 4. A repeated START condition The I2C sends the slave address that is in I2CSAR.
3	FDF	0 1	Free data format mode bit. Note that DLB in the free data format mode (DLB = 1 and FDF = 1) is not supported. See Table 16 . Free data format mode is disabled. Transfers use the 7-/10-bit addressing format selected by the XA bit. Free data format mode is enabled.

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

Bit	Field	Value	Description
2-0	BC	0-7h	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. The number of bits selected with BC must match the data size of the other device. Note that when BC = 0, 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.
		0	8 bits per data word
		1h	1 bit per data word
		2h	2 bits per data word
		3h	3 bits per data word
		4h	4 bits per data word
		5h	5 bits per data word
		6h	6 bits per data word
		7h	7 bits per data word

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

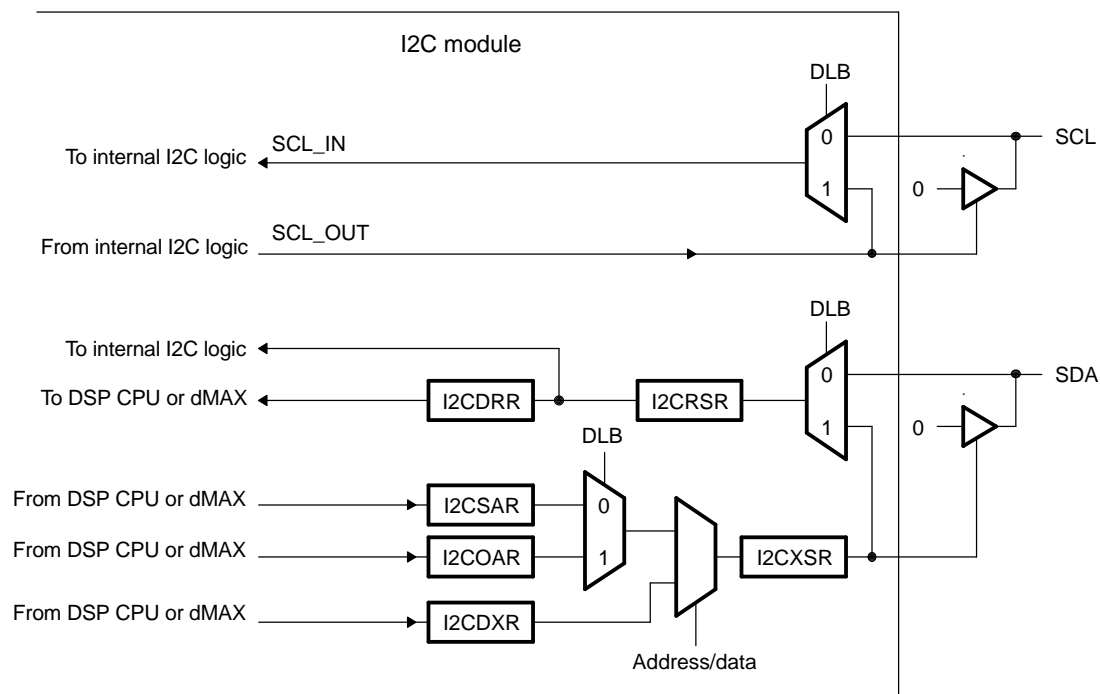
I2CMDR Bit			Bus Activity ⁽¹⁾	Description
RM	STT	STP		
0	0	0	None	No activity
0	0	1	P	STOP condition
0	1	0	S-A-D.. <i>(n)</i> ..D	START condition, slave address, <i>n</i> data words (<i>n</i> = value in I2CCNT)
0	1	1	S-A-D.. <i>(n)</i> ..D-P	START condition, slave address, <i>n</i> data words, STOP condition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

⁽¹⁾ 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		I2C State	Function of TRX Bit
MST	FDF		
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 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: TRX = 0: The I2C is a receiver. TRX = 1: The I2C is a transmitter.
1	0	In master mode but not free data format mode	TRX identifies the role of the I2C: TRX = 0: The I2C is a receiver. TRX = 1: The I2C is a transmitter.
1	1	In master 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: TRX = 0: The I2C is a receiver. TRX = 1: The I2C is a transmitter.

Figure 40. Block Diagram Showing the Effects of the Digital Loopback Mode (DLB) Bit

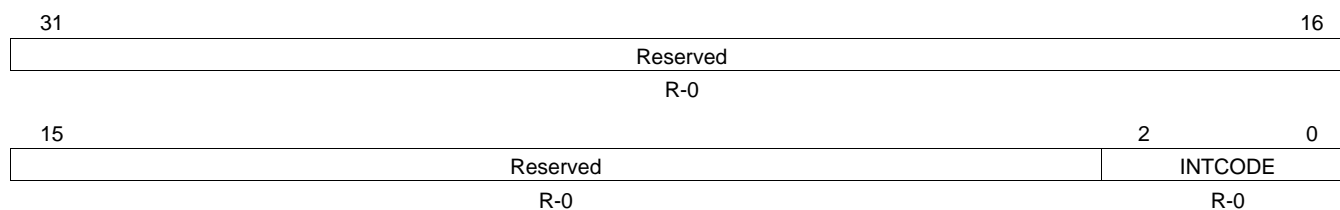


9.10 I2C Interrupt Source Register (I2CISR)

The I2C interrupt source register (I2CISR) is used by the CPU to determine which event generated an I2C interrupt (to read more about these events, see [Table 3](#)). The I2CISR is shown in [Figure 41](#) and described in [Table 17](#).

Reading I2CISR clears the current interrupt flag bit; if other interrupts are pending, a new interrupt is generated. If more than one interrupt flag bit is set, reading I2CISR clears the highest priority interrupt flag. Note that it is necessary to read (and therefore, clear) I2CISR before doing another start; otherwise, I2CISR could contain an incorrect value (old interrupt flags).

Figure 41. I2C Interrupt Source Register (I2CISR) [Offset = 28h]



LEGEND: R= Read only; -n = value after reset

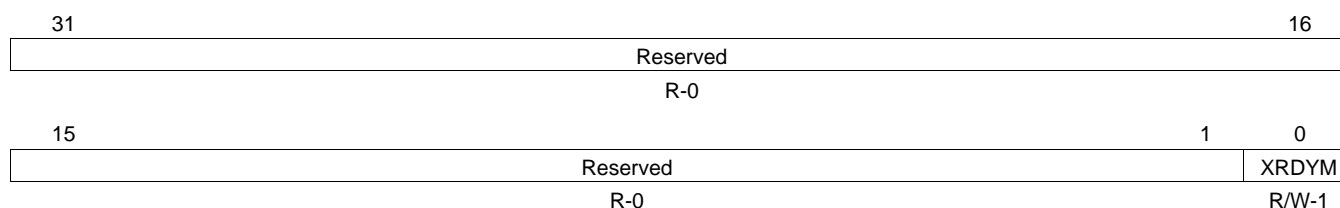
Table 17. I2C Interrupt Source Register (I2CISR) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
2-0	INTCODE	0-7h	Interrupt code bits. The binary code in INTCODE indicates which event generated an I2C interrupt.
		0	None
		1h	Arbitration-lost interrupt (AL)
		2h	No-acknowledgement interrupt (NACK)
		3h	Register-access-ready interrupt (ARDY)
		4h	Receive-data-ready interrupt (ICRRDY)
		5h	Transmit-data-ready interrupt (ICXRDY)
		6h	Stop-condition detected interrupt (SCD)
		7h	Address-as-slave interrupt (AAS)

9.11 I2C Extended Mode Register (I2CEMDR)

The I2C extended mode register (I2CEMDR) is used to indicate which condition generates a transmit data ready interrupt. The I2CEMDR is shown in [Figure 42](#) and described in [Table 18](#).

Figure 42. I2C Extended Mode Register (I2CEMDR) [Offset = 2Ch]



LEGEND: R/W = Read/Write; R= Read only; -n = value after reset

Table 18. I2C Extended Mode Register (I2CEMDR) Field Descriptions

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	XRDYM	0	Transmit-data-ready interrupt bit. Determines which condition generates a transmit-data-ready interrupt. XRDYM only has an effect when the I2C is operating as a slave-transmitter. The transmit-data-ready interrupt is generated when the master requests more data by sending an acknowledge signal after the transmission of the last data.
		1	The transmit-data-ready interrupt is generated when the data in I2CDXR is copied to I2CXSR.

9.12 I2C Prescaler Register (I2CPSC)

The I2C prescaler register (I2CPSC) is used for dividing down SYSCLK2 to obtain the desired module clock for the operation of the I2C. The I2CPSC is shown in [Figure 43](#) and described in [Table 19](#).

The IPSC bits must be initialized while the 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

Module Clock Frequency Range

The I2C module must be operated with a module clock frequency of 7 to 12 MHz. The I2C prescaler register (I2CPSC) must be configured to this frequency range.

Figure 43. I2C Prescaler Register (I2CPSC) [Offset = 30h]

31	Reserved	16
	R-0	
15	8 7	0
	Reserved	IPSC
	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

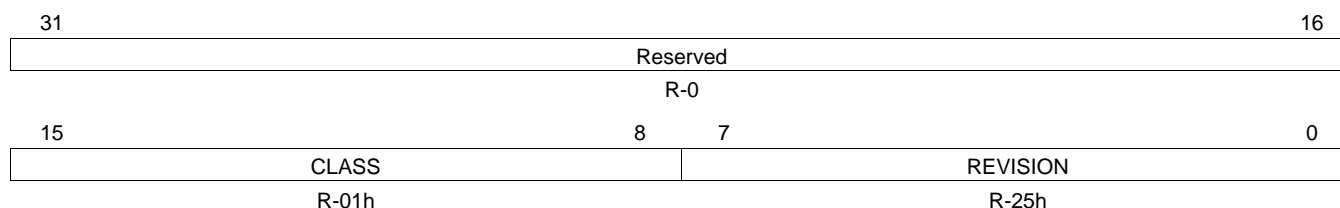
Table 19. I2C Prescaler Register (I2CPSC) Field Descriptions

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	<p>I2C prescaler divide-down value. The I2C module must be operated with a module clock frequency of 7 to 12 MHz. IPSC must be set to a value for this frequency range. IPSC determines how much SYSCLK2 is divided to create the I2C clock:</p> <p>module clock frequency = SYSCLK2 frequency/(IPSC + 1)</p> <p>Note: IPSC must be initialized while the I2C is in reset (IRS = 0 in I2CMDR).</p>

9.13 I2C Peripheral Identification Registers (I2CPID1 and I2CPID2)

The I2C peripheral identification registers (I2CPID) contain identification data (class, revision, and type) for the peripheral. I2CPID1 is shown in [Figure 44](#) and described in [Table 20](#). I2CPID2 is shown in [Figure 45](#) and described in [Table 21](#).

Figure 44. I2C Peripheral Identification Register 1 (I2CPID1) [Offset = 34h]

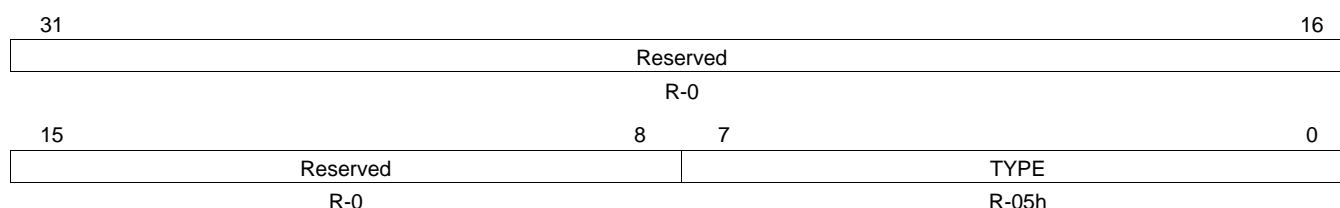


LEGEND: R = Read only; -n = value after reset

Table 20. 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	1h	Identifies class of peripheral. Serial port
7-0	REVISION	25h	Identifies revision of peripheral. Current revision of peripheral.

Figure 45. I2C Peripheral Identification Register 2 (I2CPID2) [Offset = 38h]



LEGEND: R = Read only; -n = value after reset

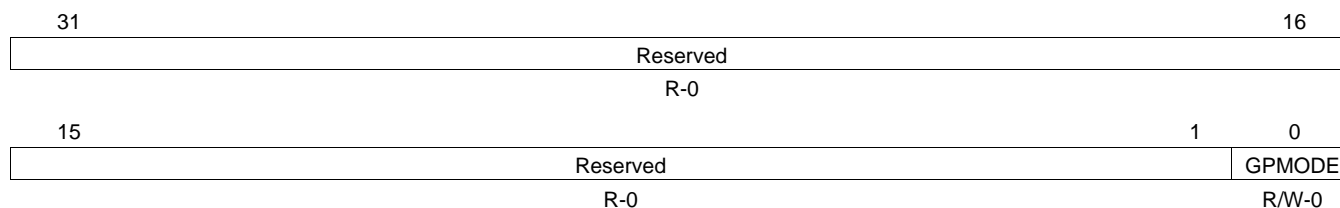
Table 21. I2C Peripheral Identification Register 2 (I2CPID2) Field Descriptions

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	05h	Identifies type of peripheral. I2C

9.14 I2C Pin Function Register (I2CPFUNC)

The I2C pin function register (I2CPFUNC) selects the SDA and SCL pins as GPIO. The I2CPFUNC is shown in [Figure 46](#) and described in [Table 22](#). No hardware protection is required to disable I2C functionality when GPMODE = 1 and the IRS bit in I2CMDR is set to 1. When GPMODE = 1 (GPIO mode), the module that controls the I2C function receives a logic-high value for SCL and SDA. The IRS bit in I2CMDR can be set to 1, regardless of the GPMODE bit value. I2C functionality works when IRS = 1. The I2C must be held in reset (IRS = 0) when changing to/from GPIO mode using the GPMODE bit.

Figure 46. I2C Pin Function Register (I2CPFUNC) [Offset = 48h]



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

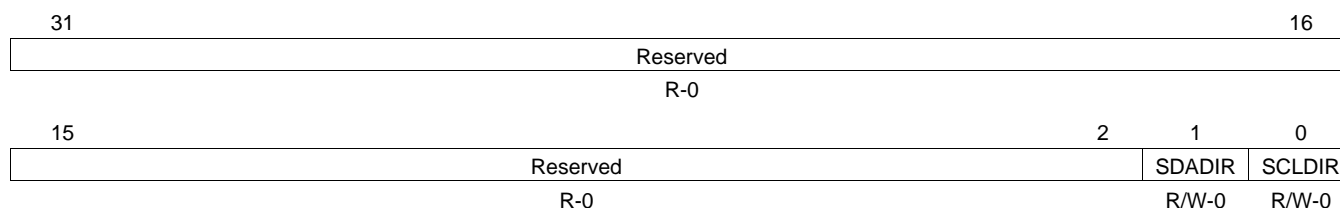
Table 22. I2C Pin Function Register (I2CPFUNC) Field Descriptions

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	GPMODE	0	GPIO mode enable bit for SCL and SDA pins. The I2C must be placed in reset (IRS = 0 in I2CMDR) before enabling the GPIO function of the SCL and SDA pins.
		0	GPIO mode is disabled; SCL and SDA pins have I2C functionality.
		1	GPIO mode is enabled; SCL and SDA pins have GPIO functionality.

9.15 I2C Pin Direction Register (I2CPDIR)

The I2C pin direction register (I2CPDIR) controls the direction of the SDA and SCL pins when configured as GPIO. The I2CPDIR is shown in [Figure 47](#) and described in [Table 23](#). If a bit is set to 1, the pin functions as an output; if a bit is cleared to 0, the pin functions as an input.

Figure 47. I2C Pin Direction Register (I2CPDIR) [Offset = 4Ch]



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

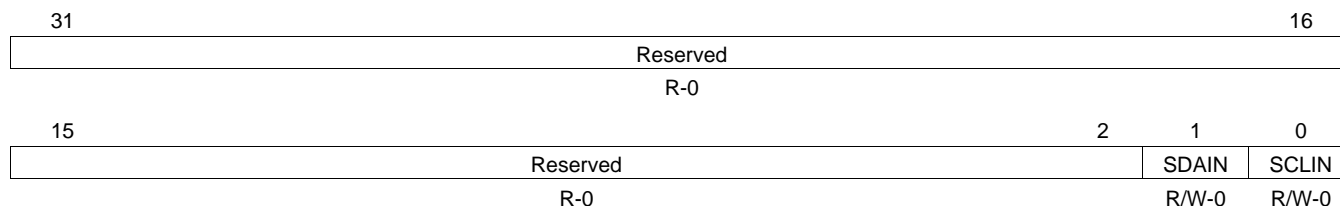
Table 23. I2C Pin Direction Register (I2CPDIR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
1	SDADIR	0 1	SDA direction bit. Controls the direction of the SDA pin when configured as GPIO. SDA pin functions as input. SDA pin functions as output.
0	SCLDIR	0 1	SCL direction bit. Controls the direction of the SCL pin when configured as GPIO. SCL pin functions as input. SCL pin functions as output.

9.16 I2C Pin Data Input Register (I2CPDIN)

The I2C pin data input register (I2CPDIN) reflects the state of the SDA and SCL pins. The I2CPDIN is shown in Figure 48 and described in Table 24. When read, I2CPDIN returns the value from the pin's input buffer regardless of the state of the corresponding I2CPFUNC or I2CPDIR bits.

Figure 48. I2C Pin Data Input Register (I2CPDIN) [Offset = 50h]



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 24. I2C Pin Data Input Register (I2CPDIN) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
1	SDAIN	0 1	Indicates the logic level present on the SDA pin. SDAIN is set regardless of the GPMODE bit value in I2CPFUNC. A value written to this bit has no effect. A logic low is present at the SDA pin. A logic high is present at the SDA pin.
0	SCLIN	0 1	Indicates the logic level present on the SCL pin. SCLIN is set regardless of the GPMODE bit value in I2CPFUNC. A value written to this bit has no effect. A logic low is present at the SCL pin. A logic high is present at the SCL pin.

9.17 I2C Pin Data Output Register (I2CPDOUT)

The I2C pin data output register (I2CPDOUT) determines the value driven on the SDA and SCL pins, if the pin is configured as an output. The I2CPDOUT is shown in [Figure 49](#) and described in [Table 25](#). Writes do not affect pins not configured as GPIO outputs.

The I2CPDOUT bits are set or cleared by writing to this register directly. A read of I2CPDOUT returns the value of the register not the value at the pin (that might be configured as an input). An alternative way to set bits in I2CPDOUT is to write a 1 to the corresponding bit of I2CPDSET. An alternative way to clear bits in I2CPDOUT is to write a 1 to the corresponding bit of I2CPDCLR.

I2CPDOUT has these aliases:

- I2CPDSET - writing a 1 to a bit in I2CPDSET sets the corresponding bit in I2CPDOUT to 1; writing a 0 has no effect and keeps the bits in I2CPDOUT unchanged.
- I2CPDCLR - writing a 1 to a bit in I2CPDCLR clears the corresponding bit in I2CPDOUT to 0; writing a 0 has no effect and keeps the bits in I2CPDOUT unchanged.

Figure 49. I2C Pin Data Output Register (I2CPDOUT) [Offset = 54h]

31	Reserved															16	
R-0																	
15	Reserved														2	1	0
R-0														SDAOUT		SCLOUT	
R-0														R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

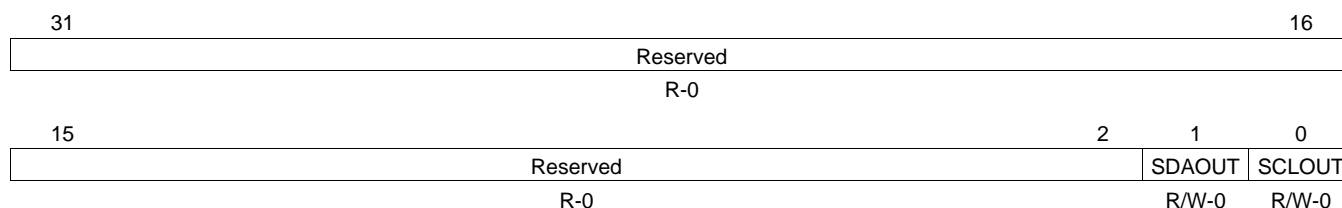
Table 25. I2C Pin Data Output Register (I2CPDOUT) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
1	SDAOUT	0 1	Controls the value driven on the SDA pin when the pin is configured as an output (GPIO mode must be enabled by setting GPMODE = 1 in I2CPFUNC). When reading data, returns the value in the SDAOUT bit, does not return the level on the pin. When writing data, writes to the SDAOUT bit. SDA pin is driven to a logic low. SDA pin is driven to a logic high.
0	SCLOUT	0 1	Controls the value driven on the SCL pin when the pin is configured as an output (GPIO mode must be enabled by setting GPMODE = 1 in I2CPFUNC). When reading data, returns the value in the SCLOUT bit, does not return the level on the pin. When writing data, writes to the SCLOUT bit. SCL pin is driven to a logic low. SCL pin is driven to a logic high.

9.18 I2C Pin Data Set Register (I2CPDSET)

The I2C pin data set register (I2CPDSET) is shown in Figure 50 and described in Table 26. I2CPDSET is an alias of the I2C pin data output register (I2CPDOUT) for writes only and provides an alternate means of driving GPIO outputs high. Writing a 1 to a bit of I2CPDSET sets the corresponding bit in I2CPDOUT. Writing a 0 has no effect. Register reads are indeterminate.

Figure 50. I2C Pin Data Set Register (I2CPDSET) [Offset = 58h]



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

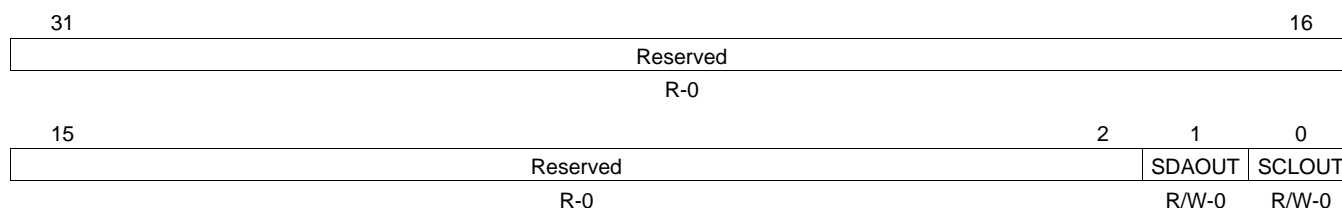
Table 26. I2C Pin Data Set Register (I2CPDSET) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
1	SDAOUT		Sets the value of the SDAOUT bit in I2CPDOUT (SDA pin). A write of 0 to this bit has no effect. This bit location has an indeterminate value when read.
		0	No effect.
		1	Sets the SDAOUT bit in I2CPDOUT to 1.
0	SCLOUT		Sets the value of the SCLOUT bit in I2CPDOUT (SCL pin). A write of 0 to this bit has no effect. This bit location has an indeterminate value when read.
		0	No effect.
		1	Sets the SCLOUT bit in I2CPDOUT to 1.

9.19 I2C Pin Data Clear Register (I2CPDCLR)

The I2C pin data clear register (I2CPDCLR) is shown in [Figure 51](#) and described in [Table 27](#). I2CPDCLR is an alias of the I2C pin data output register (I2CPDOUT) for writes only and provides an alternate means of driving GPIO outputs low. Writing a 1 to a bit of I2CPDCLR clears the corresponding bit in I2CPDOUT. Writing a 0 has no effect. Register reads are indeterminate.

Figure 51. I2C Pin Data Clear Register (I2CPDCLR) [Offset = 5Ch]



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 27. I2C Pin Data Clear Register (I2CPDCLR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
1	SDAOUT		Clears the value of the SDAOUT bit in I2CPDOUT (SDA pin). A write of 0 to this bit has no effect. This bit location has an indeterminate value when read.
		0	No effect.
		1	Clears the SDAOUT bit in I2CPDOUT to 0.
0	SCLOUT		Clears the value of the SCLOUT bit in I2CPDOUT (SCL pin). A write of 0 to this bit has no effect. This bit location has an indeterminate value when read.
		0	No effect.
		1	Clears the SCLOUT bit in I2CPDOUT to 0.

Appendix A Revision History

Table A-1 lists the changes made since the previous version of this document.

Table A-1. Document Revision History

Reference	Additions/Modifications/Deletions
Section 3.8	Modified master clock frequency equation. Added IPSC value table. Added sentence after IPSC value table: Note that the equation does not include rise and fall time effects and may need to be modified depending on bus capacitance.
Section 9.4	Modified master clock frequency equation. Modified Figure 32 and figure footnote.
Section 9.4.1	Modified ICCL bit description in Table 8: Clock low-time divide-down value of 1-65536. The period of the module clock is multiplied by (ICCL + d) to produce the low-time duration of the master clock on the SCL pin.
Section 9.4.2	Modified ICCH bit description in Table 9: Clock high-time divide-down value of 1-65536. The period of the module clock is multiplied by (ICCH + d) to produce the high-time duration of the master clock on the SCL pin.
Section 9.12	Modified IPSC bit description in Table 19: The I2C module must be operated with a module clock frequency of 7 to 12 MHz.

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

Copyright © 2005, Texas Instruments Incorporated