1. I2C - Inter-Integrated Circuit Module

The Inter-Integrated Circuit (I2C) bus is a multi-host serial data communication bus. Devices communicate in a host/client environment where the host devices initiate the communication. A client device is controlled through addressing.

The following figure shows a block diagram of the I2C interface module, and shows both Host and Client modes together.

**Figure 36-1.** I2C Block Diagram

**ABD**

(See I2CxCON2

**I2CxADR0/1/2/3**

Register)

**Address compare**

**TH**

(See RxyI2C Register)

**A**

I2CxSDAPPS



**SD**

**(in)**

**SDAHT**

(See I2CxCON2 Register)

**I2CxADB0/1**

**Transmit Buffer I2CxTXB**

**Register**

**TX Shift**

RxyPPS

**I2CxADB0/1**

**SDA**

**(out)**



See

I2CxCLK

Register

See I2CxBTO

Register

**SCL**

**CLK**

**BTO**



I2CxSCLPPS

**ABD**

|  |  |
| --- | --- |
|  |  |
| **RX Shift** | |
| **Register** | |
|  |  |

(See I2CxCON2

**Receive Buffer I2CxRXB**

Register)

**I2C Control Unit**

**Host**

**Module**

RxyPPS

**Client Module**

**SCL**

**(out)**



**(in)**

**TH**

**Interrupt Controller**

I2CxPIR

(See RxyI2C Register)

# I2C Features

The I2C supports the following modes and features:

* + - Modes
      * Host mode
      * Client mode
      * Multi-Host mode
    - Features
      * Supports Standard mode (100 kHz), Fast mode (400 kHz) and Fast mode Plus (1 MHz) modes of operation
      * Dedicated Address, Receive, and Transmit buffers
      * Up to four unique Client addresses
      * General Call addressing
      * 7-bit and 10-bit addressing with optional masking
      * Interrupts for:
        + Start condition
        + Restart condition
        + Stop condition
        + Address match
        + Data Write
        + Acknowledge Status
        + NACK detection
        + Data Byte Count
        + Bus Collision
        + Bus Time-out
      * Clock Stretching for:
        + RX buffer full
        + TX buffer empty
        + Incoming address match
        + Data Write
        + Acknowledge Status
      * Bus Collision Detection with Arbitration
      * Bus Time-out Detection
        + Selectable clock sources
        + Clock prescaler
      * Selectable Serial Data (SDA) Hold Time
      * Dedicated I2C Pad (I/O) Control
        + Standard GPIO or I2C-specific slew rate control
        + Selectable I2C pull-up levels
        + I2C-specific, SMBus 2.0/3.0, or standard GPIO input threshold level selections
      * Integrated Direct Memory Access (DMA) support
      * Remappable pin locations using Peripheral Pin Select (PPS)

# I2C Terminology

The I2C communication protocol terminology used throughout this document have been adapted from the Phillips I2C Specification and can be found in the table below.

**I2C Bus Terminology and Definitions**

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Host | The device that initiates a transfer, generates the clock signal and terminates a transfer |
| Client | The device addressed by the host |
| Multi-Host | A bus containing more than one host device that can initiate communication |
| Transmitter | The device that shifts data out onto the bus |
| Receiver | The device that shifts data in from the bus |
| Arbitration | Procedure that ensures only one host at a time controls the bus |
| Synchronization | Procedure that synchronizes the clock signal between two or more devices on the bus |
| Idle | The state in which no activity occurs on the bus and both bus lines are at a high logic level |

|  |  |
| --- | --- |
| Active | The state in which one or more devices are communicating on the bus |
| Matching Address | The address byte received by a client that matches the value that is stored in the I2CxADR0/1/2/3 registers |
| Addressed Client | Client device that has received a matching address and is actively being clocked by a host device |
| Write Request | Host transmits an address with the R/W bit clear indicating that it wishes to transmit data to a client device |
| Read Request | Host transmits an address with the R/W bit set indicating that it wishes to receive data from a client device |
| Clock Stretching | The action in which a device holds the SCL line low to stall communication |
| Bus Collision | Occurs when the module samples the SDA line and returns a low state while expecting a high state |
| Bus Time-out | Occurs whenever communication stalls for a period longer than acceptable |

# I2C Module Overview

The I2C module provides a synchronous serial interface between the microcontroller and other I2C- compatible devices using a bidirectional two-wire bus. Devices operate in a host/client environment that may contain one or more host devices and one or more client devices. The host device always initiates communication.

The I2C bus consists of two signal connections:

* + - Serial Clock (SCL)
    - Serial Data (SDA)

Both the SCL and SDA connections are open-drain lines, each line requiring pull-up resistors to the application’s supply voltage. Pulling the line to ground is considered a logic ‘0’, while allowing the line to float is considered a logic ‘1’. It is important to note that the voltage levels of the logic low and logic high are not fixed and are dependent on the bus supply voltage. According to the I2C Specification, a logic low input level is up to 30% of VDD (VIL ≤ 0.3 VDD), while the logic high input level is 70% to 100% of VDD (VIH ≥ 0.7 VDD). Both signal connections are considered bidirectional, although the SCL signal can only be an output in Host mode and an input in Client mode.

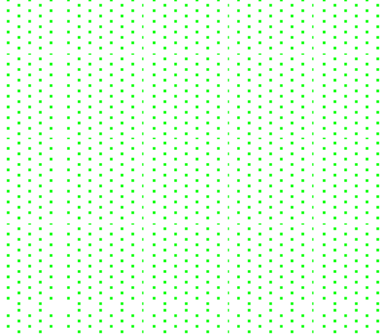
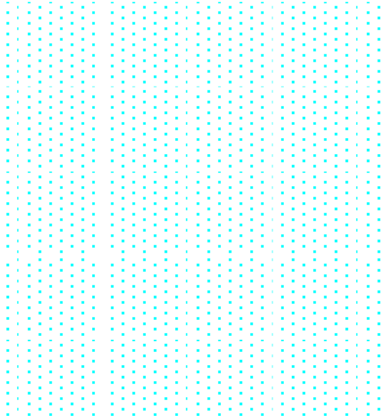
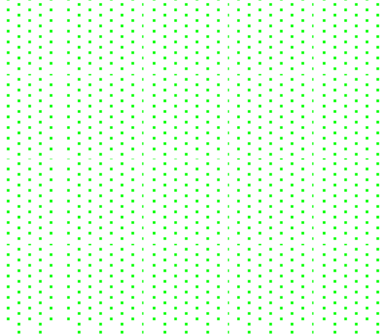
All transactions on the bus are initiated and terminated by the host device. Depending on the direction of the data being transferred, there are four main operations performed by the I2C module:

* + - Host Transmit: Host is transmitting data to a client
    - Host Receive: Host is receiving data from a client
    - Client Transmit: Client is transmitting data to a host
    - Client Receive: Client is receiving data from a host

The I2C interface allows for a multi-host bus, meaning that there can be several host devices present on the bus. A host can select a client device by transmitting a unique address on the bus. When the address matches a client’s address, the client responds with an Acknowledge (ACK) condition, and communication between the host and that client can commence. All other devices connected to the bus must ignore any transactions not intended for them.

The following figure shows a typical I2C bus configuration with one host and two clients.

**Figure 36-2.** I2C Host-Client Connections



SDA

SDA

SCK

I2C Client 1

SCK

I2C Host

SDA

SCK

I2C Client 2

Shift Register

Shift Register

Shift Register

Transmit Buffer

Receive Buffer

Transmit Buffer

Transmit Buffer

Receive Buffer

Receive Buffer

## Byte Format

As previously mentioned, all I2C communication is performed in 9-bit segments. The transmitting device sends a byte to a receiver, and once the byte is processed by the receiver, the receiver returns an Acknowledge bit. There are no limits to the amount of data bytes in a I2C transmission.

After the 8th falling edge of the SCL line, the transmitting device releases control of the SDA line to allow the receiver to respond with either an Acknowledge (ACK) sequence or a Not Acknowledge (NACK) sequence. At this point, if the receiving device is a client, it can hold the SCL line low (clock stretch) to allow itself time to process the incoming byte. Once the byte has been processed, the receiving device releases the SCL line, allowing the host device to provide the 9th clock pulse, within which the client responds with either an ACK or a NACK sequence. If the receiving device is a host, it may also hold the SCL line low until it has processed the received byte. Once the byte has been

processed, the host device will generate the 9th clock pulse and transmit the ACK or NACK sequence.

Data is valid to change only while the SCL signal is in a Low state, and sampled on the rising edge of SCL. Changes on the SDA line while the SCL line is high indicate either a Start or Stop condition.

## SDA and SCL Pins

The SDA and SCL pins must be configured as open-drain outputs. Open-drain configuration is accomplished by setting the appropriate bits in the Open-Drain Control (ODCONx) registers, while output direction configuration is handled by clearing the appropriate bits in the Tri-State Control (TRISx) registers. Input threshold, slew rate, and internal pull-up settings are configured using the RxyI2C registers. The RxyI2C registers are used exclusively on the default I2C pin locations, and provide the following selections:

* + - * Input threshold levels:
        + SMBus 3.0 (1.35V) input threshold
        + SMBus 2.0 (2.1V) input threshold
        + I2C-specific input thresholds
        + Standard GPIO input thresholds (controlled by the Input Level Control (INLVLx) registers)
      * Slew rate limiting:
        + I2C-specific slew rate limiting
        + Standard GPIO slew rate (controlled by the Slew Rate Control (SLRCONx) registers)
      * I2C pull-ups:
        + Programmable ten or two times the current of the standard internal pull-up
        + Standard GPIO pull-up (controlled by the Weak Pull-Up Control (WPUx) registers)

**Important:** The pin locations for SDA and SCL are remappable through the Peripheral Pin Select (PPS) registers. If new pin locations for SDA and SCL are desired, user software must configure the INLVLx, SLRCONx, ODCONx, and TRISx registers for each new pin location.

The RxyI2C registers cannot be used since they are dedicated to the default pin locations. Additionally, the internal pull-ups for non-I2C pins are not strong enough to drive the pins; therefore, external pull-up resistors must be used.

## 36.3.2.1 SDA Hold Time

SDA hold time refers to the amount of time between the low threshold region of the falling edge of SCL (VIL ≤ 0.3 VDD) and either the low threshold region of the rising edge of SDA (VIL ≤ 0.3 VDD) or the high threshold region of the falling edge of SDA (VIH ≥ 0.7 VDD) (see [Figure 36-3](#_bookmark0)). If the SCL fall time is long or close to the maximum allowable time set by the I2C Specification, data may be sampled in the undefined Logic state between the 70% and 30% region of the falling SCL edge, leading to data corruption. The I2C module offers selectable SDA hold times, which can be useful to ensure valid data transfers at various bus data rates and capacitance loads.

**Figure 36-3.** SDA Hold Time

**VIH 0.7 VDD**



**Change of**

**data allowed**

**VIL 0.3 VDD**

**VIH 0.7 VDD**

SDA Hold Time

## SCL

**SDA**

**VIL 0.3 VDD**

## Start Condition

All I2C transmissions begin with a Start condition. The Start condition is used to synchronize the SCL signals between the host and client devices. The I2C Specification defines a Start condition as a transition of the SDA line from a logic high level (Idle state) to a logic low level (Active state) while the SCL line is at a logic high (see [Figure 36-4](#_bookmark1)). A Start condition is always generated by the host, and is initiated by either writing to the Start ([S](#_bookmark17)) bit or by writing to the I2C Transmit Buffer ([I2CxTXB](#_bookmark87)) register, depending on the Address Buffer Disable ([ABD](#_bookmark34)) bit setting.

When the I2C module is configured in Host mode, module hardware waits until the bus is free (Idle state). Module hardware checks the Bus Free Status ([BFRE](#_bookmark38)) bit to ensure the bus is Idle before initiating a Start condition. When the BFRE bit is set, the bus is considered Idle, and indicates that the SCL and SDA lines have been in a Logic High state for the amount of I2C clock cycles as selected by the Bus Free Time Selection ([BFRET](#_bookmark36)) bits. When a Start condition is detected on the bus, module hardware clears the BFRE bit, indicating an active bus.

In Multi-Host mode, it is possible for two host devices to issue Start conditions at the same time. If two or more hosts initiate a Start at the same time, a bus collision will occur; however, the I2C Specification states that a bus collision cannot occur on a Start. In this case, the competing host devices must go through bus arbitration during the addressing phase.

The figure below shows a Start condition.

**Figure 36-4.** Start Condition

**Start**

**Condition**

## SDA

**SCL**

## Acknowledge Sequence

The 9th SCL pulse for any transferred address/data byte is reserved for the Acknowledge (ACK) sequence. During an Acknowledge sequence, the transmitting device relinquishes control of the SDA line to the receiving device. At this time, the receiving device must decide whether to pull the SDA line low (ACK) or allow the line to float high (NACK). Since the Acknowledge sequence is an active-low signal, pulling the SDA line low informs the transmitter that the receiver has successfully received the transmitted data.

The Acknowledge Data ([ACKDT](#_bookmark23)) bit holds the value to be transmitted during an Acknowledge sequence while the [I2CxCNT](#_bookmark74) register is nonzero (I2CxCNT != 0). When a client device receives a matching address, or a receiver receives valid data, the ACKDT bit is cleared by user software to indicate an ACK. If the client does not receive a matching address, user software sets the ACKDT bit, indicating a NACK. In Client or Multi-Host modes, if the Address Interrupt and Hold Enable ([ADRIE](#_bookmark61)) or Write Interrupt and Hold Enable ([WRIE](#_bookmark60)) bits are set, the clock is stretched after receiving a matching

address or after the 8th falling edge of SCL when a data byte is received. This allows user software time to determine the ACK/NACK response to send back to the transmitter.

The Acknowledge End of Count ([ACKCNT](#_bookmark22)) bit holds the value that will be transmitted once the [I2CxCNT](#_bookmark74) register reaches a zero value (I2CxCNT = 0). When the I2CxCNT register reaches a zero value, the ACKCNT bit can be cleared (ACKCNT = 0), indicating an ACK, or ACKCNT can be set (ACKCNT = 1), indicating a NACK.

**Important:** The [ACKCNT](#_bookmark22) bit is only used when the [I2CxCNT](#_bookmark74) register is zero, otherwise the [ACKDT](#_bookmark23) bit is used for ACK/NACK sequences.

In Host Write or Client Read modes, the Acknowledge Status ([ACKSTAT](#_bookmark24)) bit holds the result of the Acknowledge sequence transmitted by the receiving device. The ACKSTAT bit is cleared when the receiver sends an ACK, and is set when the receiver does not Acknowledge (NACK).

The Acknowledge Time Status ([ACKT](#_bookmark25)) bit indicates whether or not the bus is in an Acknowledge sequence. The ACKT bit is set during an ACK/NACK sequence on the 8th falling edge of SCL, and is cleared on the 9th rising edge of SCL, indicating that the bus is not in an ACK/NACK sequence.

Certain conditions will cause a NACK sequence to be sent automatically. A NACK sequence is generated by module hardware when any of the following bits are set:

* Transmit Write Error Status ([TXWE](#_bookmark44))
* Transmit Underflow Status ([TXU](#_bookmark28))
* Receive Read Error Status ([RXRE](#_bookmark46))
* Receive Overflow Status ([RXO](#_bookmark27))

**Important:** Once a NACK is detected on the bus, all subsequent Acknowledge sequences will consist of a NACK until all Error conditions are cleared.

The following figure shows ACK and NACK sequences.

**Figure 36-5.** ACK/NACK Sequences

8th falling edge

9th rising edge

8th falling edge

9th rising edge

**SCL**

**SDA**

**SCL**

**SDA**

Acknowledge

(ACK)

Not Acknowledge

(NACK)

## Restart Condition

A Restart condition is essentially the same as a Start condition – the SDA line transitions from an idle level to an active level while the SCL line is Idle – but may be used in place of a Stop condition

whenever the host device has completed its current transfer but wishes to keep control of the bus. A Restart condition has the same effect as a Start condition, resetting all client logic and preparing it to receive an address.

A Restart condition is also used when the host wishes to use a combined data transfer format. A combined data transfer format is used when a host wishes to communicate with a specific register address or memory location. In a combined format, the host issues a Start condition, followed by the client’s address, followed by a data byte which represents the desired client register or memory address. Once the client address and data byte have been acknowledged by the client, the host issues a Restart condition, followed by the client address. If the host wishes to write data to the client, the LSb of the client address, the Read/not Write (R/W) bit, will be clear. If the host wishes

to read data from the client, the R/W bit will be set. Once the client has acknowledged the second address byte, the host issues a Restart condition, followed by the upper byte of the client address with the R/W bit set. Client logic will then acknowledge the upper byte, and begin to transmit data to the host.

**Important:** In 10-bit Client mode, a Restart is required for the host to read data out of the client, regardless of which data transfer format is used – host read-only or combined. For example, if the host wishes to perform a bulk read, it will transmit the client’s 10-bit address with the R/W bit clear.

The figure below shows a Restart condition.

**Figure 36-6.** Restart Condition

**Restart**

**Condition**

## SDA

**SCL**

## Stop Condition

All I2C transmissions end with a Stop condition. A Stop condition occurs when the SDA line transitions from a logic low (active) level to a logic high (idle) level while the SCL line is at a logic high level. A Stop condition is always generated by the host device, and is generated by module hardware when a Not Acknowledge (NACK) is detected on the bus, a bus time-out event occurs, or when the I2C Byte Count ([I2CxCNT](#_bookmark74)) register reaches a zero count. A Stop condition may also be generated through software by setting the Stop ([P](#_bookmark26)) bit.

The figure below shows a Stop condition.

**Figure 36-7.** Stop Condition

## SCL

**Stop Condition**

**SDA**

## Bus Time-Out

The SMBus protocol requires a bus watchdog to prevent a stalled device from holding the bus indefinitely. The I2C Bus Time-Out Clock Source Selection ([I2CxBTOC](#_bookmark79)) register provides several clock sources that can be used as the time-out time base. The I2C Bus Time-Out ([I2CxBTO](#_bookmark75)) register is used to determine the actual bus time-out time period, as well as how the module responds to a time-out.

The bus time-out hardware monitors for the following conditions:

* + - * SCL = 0 (regardless of whether or not the bus is Active)
      * SCL = 1 and SDA = 0 while the bus is Active

If either of these conditions are true, an internal time-out counter increments, and continues to increment as long as the condition stays true, or until the time-out period has expired. If these conditions change (e.g. SCL = 1), the internal time-out counter is reset by module hardware.

The Bus Time-Out Clock Source Selection ([BTOC](#_bookmark80)) bits select the time-out clock source. If an oscillator is selected as the time-out clock source, such as the LFINTOSC, the time-out clock base period is approximately 1 ms. If a timer is selected as the time-out clock source, the timer can be configured to produce a variety of time periods.

**Remember:** The SMBus protocol dictates a 25 ms time-out for client devices and a 35 ms time-out for host devices.

The Time-Out Time Selection ([TOTIME](#_bookmark78)) bits and the Time-Out Prescaler Extension Enable ([TOBY32](#_bookmark77)) bit are used to determine the time-out period. The value written into TOTIME multiplies the base time-out clock period. For example, if a value of ‘35’ is written into the TOTIME bits, and the LFINTOSC is selected as the time-out clock source, the time-out period is approximately 35 ms (35 x 1 ms). If the TOBY32 bit is set (TOBY32 = 1), the time-out period determined by the TOTIME bits is multiplied by 32. If TOBY32 is clear (TOBY32 = 0), the time-out period determined by the TOTIME bits is used as the time-out period.

The examples below illustrate possible time-out configurations.

void **Init\_BTO\_35**(void)

{

I2C1BTOC = 0x06; I2C1BTObits.TOREC = 1;

I2C1BTObits.TOBY32 = 0;

*// Selections produce a 35 ms BTO period*

*// LFINTOSC as BTO clock source*

*// Reset I2C interface, set BTOIF*

*// BTO time = TOTIME \* T*BTOCLK

**Example 36-1.** 35 ms BTO Period Configuration

I2C1BTObits.TOTIME = 0x23; *// TOTIME = T*BTOCLK \* 35

*// = 1 ms \* 35 = 35 ms*

}

**Example 36-2.** 64 ms BTO Configuration

void **Init\_BTO\_64**(void) *// Selections produce a 64 ms BTO period*

{

I2C1BTOC = 0x06; *// LFINTOSC as BTO clock source* I2C1BTObits.TOREC = 1; *// Reset I2C interface, set BTOIF* I2C1BTObits.TOBY32 = 1; *// BTO time = TOTIME \* T*BTOCLK \* 32

*// = 2 ms \* 32 = 64 ms*

I2C1BTObits.TOTIME = 0x02; *// TOTIME = T*BTOCLK \* 2

*// = 1 ms \* 2 = 2 ms*

}

The Time-Out Recovery Selection ([TOREC](#_bookmark76)) bit determines how the module will respond to a bus time-out. When a bus time-out occurs and TOREC is set (TOREC = 1), the I2C module is reset and module hardware sets the Bus Time-Out Interrupt Flag ([BTOIF](#_bookmark66)). If the Bus Time-Out Interrupt Enable

([BTOIE](#_bookmark69)) is also set, an interrupt will be generated. If a bus time-out occurs and TOREC is clear (TOREC

= 0), the BTOIF bit is set, but the module is not reset.

If the module is configured in Client mode with [TOREC](#_bookmark76) set (TOREC = 1), and a bus time-out event occurs (regardless of the state of the Client Mode Active ([SMA](#_bookmark39)) bit), the module is immediately reset, the SMA and Client Clock Stretching ([CSTR](#_bookmark18)) bits are cleared, and the Bus Time-Out Interrupt Flag ([BTOIF](#_bookmark66)) bit is set.

If the module is configured in Client mode with [TOREC](#_bookmark76) clear (TOREC = 0), and a bus time-out event occurs (regardless of the state of the Client Mode Active ([SMA](#_bookmark39)) bit), the [BTOIF](#_bookmark66) bit is set, but user software must reset the module.

**Important:** It is recommended to set [TOREC](#_bookmark76) (TOREC = 1) when operating in Client mode.

If the module is configured in Host mode with [TOREC](#_bookmark76) set (TOREC = 1), and the bus time-out event occurs while the Host is active (Host Mode Active ([MMA](#_bookmark40)) = 1), the Host Data Ready ([MDR](#_bookmark19)) bit is cleared, the module will immediately attempt to transmit a Stop condition, and sets the [BTOIF](#_bookmark66) bit. Stop condition generation may be delayed if a client device is stretching the clock, but will resume once the clock is released, or if the client holding the bus also has a time-out event occur. The MMA bit is only cleared after the Stop condition has been generated.

If the module is configured in Host mode with [TOREC](#_bookmark76) clear (TOREC = 0), and the bus time-out event occurs while the Host is active (Host Mode Active ([MMA](#_bookmark40)) = 1), the [MDR](#_bookmark19) bit is cleared and the [BTOIF](#_bookmark66) bit is set, but user software must initiate the Stop condition by setting the [P](#_bookmark26) bit.

The figure below shows an example of a Bus Time-Out event when the module is operating in Host mode.

**Figure 36-8.** Host Mode Bus Time-Out Example

**Client releases SCL,**

**Host begins Stop**

**SDA D0**

**Host waits**

**for ACK/NACK**

**Stop detected**

**PCIF = 1**

**SCL 8**

**T2\_Postscaled\_out**

**I2CxTXIF = 1**

**TXBE = 1**

**Enable Timer2**

**BTOIF = 1**

**Host attempts to issue Stop ,**

**but must wait until SCL = 1**

**Software clears BTOIF,**

**T2TMR\_T2PR\_Match TMR2IF = 1**

**TMR2IF**

**MMA**

**Hardware clears MMA**

## Address Buffers

The I2C module has two address buffer registers, [I2CxADB0](#_bookmark81) and [I2CxADB1](#_bookmark82), which can be used as address receive buffers in Client mode, address transmit buffers in Host mode, or both address transmit and address receive buffers in 7-bit Multi-Host mode (see [Table 36-1](#_bookmark2)). The address buffers are enabled/disabled via the Address Buffer Disable ([ABD](#_bookmark34)) bit.

When the [ABD](#_bookmark34) bit is clear (ABD = 0), the buffers are enabled, which means:

* + - * In 7-bit Host mode, the desired client address with the R/W value is transmitted from the [I2CxADB1](#_bookmark82) register, bypassing the I2C Transmit Buffer ([I2CxTXB](#_bookmark87)). [I2CxADB0](#_bookmark81) is unused.
      * In 10-bit Host mode, [I2CxADB1](#_bookmark82) holds the upper bits and R/W value of the desired client address, while [I2CxADB0](#_bookmark81) holds the lower eight bits of the desired client address. Host hardware copies the contents of I2CxADB1 to the transmit shift register, and waits for an ACK from the client. Once the ACK is received, host hardware copies the contents of I2CxADB0 to the transmit shift register.
      * In 7-bit Client mode, a matching received address is loaded into [I2CxADB0](#_bookmark81), bypassing the I2C Receive Buffer ([I2CxRXB](#_bookmark88)). [I2CxADB1](#_bookmark82) is unused.
      * In 10-bit Client mode, [I2CxADB0](#_bookmark81) is loaded with the lower eight bits of the matching received address, while [I2CxADB1](#_bookmark82) is loaded with the upper bits and R/W value of the matching received address.
      * In 7-bit Multi-Host mode, the device can be both a host and a client depending on the sequence of events on the bus. When being addressed as a client, the matching received address with R/W value is stored into [I2CxADB0](#_bookmark81). When being used as a host, the desired client address and R/W value are loaded into the [I2CxADB1](#_bookmark82) register.

When the [ABD](#_bookmark34) bit is set (ABD = 1), the buffers are disabled, which means:

* + - * In Host mode, the desired client address is transmitted from the [I2CxTXB](#_bookmark87) register.
      * In Client mode, a matching received address is loaded into the [I2CxRXB](#_bookmark88) register.

**Table 36-1.** Address Buffer Direction

|  |  |  |
| --- | --- | --- |
| [**Mode**](#_bookmark20) | [**I2CxADB0**](#_bookmark81) | [**I2CxADB1**](#_bookmark82) |

|  |  |  |
| --- | --- | --- |
| Client (7-bit) | RX | Unused |
| Client (10-bit) | RX (address low byte) | RX (address high byte) |
| Host (7-bit) | Unused | TX |
| Host (10-bit) | TX (address low byte) | TX (address high byte) |
| Multi-Host (7-bit) | RX | TX |

## Transmit Buffer

The I2C module has a dedicated transmit buffer, [I2CxTXB](#_bookmark87), which is independent from the receive buffer.

The transmit buffer is loaded with an address byte (when [ABD](#_bookmark34) = 1), or a data byte, that is copied into the transmit shift register and transmitted onto the bus. When the [I2CxTXB](#_bookmark87) register does not

contain any transmit data, the Transmit Buffer Empty Status ([TXBE](#_bookmark45)) bit is set (TXBE = 1), allowing user software or the DMA to load a new byte into the buffer. When the TXBE bit is set and the [I2CxCNT](#_bookmark74) register is nonzero (I2CxCNT != 0), the I2C Transmit Interrupt Flag (I2CxTXIF) bit of the PIR registers is set, and can be used as a DMA trigger. A write to I2CxTXB will clear both the TXBE and I2CxTXIF bits. Setting the Clear Buffer ([CLRBF](#_bookmark47)) bit clears I2CxTXIF, the I2Cx Receive Buffer ([I2CxRXB](#_bookmark88)) and I2CxTXB.

If user software attempts to load [I2CxTXB](#_bookmark87) while it is full, the Transmit Write Error Status ([TXWE](#_bookmark44)) bit is set, a NACK is generated, and the new data is ignored. If TXWE is set, user software must clear the bit before attempting to load the buffer again.

When module hardware attempts to transfer the contents of [I2CxTXB](#_bookmark87) to the transmit shift register while I2CxTXB is empty ([TXBE](#_bookmark45) = 1), the Transmit Underflow Status ([TXU](#_bookmark28)) bit is set, I2CxTXB is loaded with 0xFF, and a NACK is generated.

**Important:** A transmit underflow can only occur when clock stretching is disabled (Clock Stretching Disable ([CSD](#_bookmark29)) bit = 1). Clock stretching prevents transmit underflows because the clock is stretched after the 8th falling SCL edge, and is only released upon the write of new data into [I2CxTXB](#_bookmark87).

## Receive Buffer

The I2C module has a dedicated receive buffer, [I2CxRXB](#_bookmark88), which is independent from the transmit buffer.

Data received through the shift register is transferred to [I2CxRXB](#_bookmark88) when the byte is complete. User software or the DMA can access the byte by reading the I2CxRXB register. When new data is loaded into I2CxRXB, the Receive Buffer Full Status ([RXBF](#_bookmark48)) bit is set, allowing user software or the DMA to read the new data. When the RXBF bit is set, the I2C Receive Interrupt Flag (I2CxRXIF) bit of the PIR registers is set, and can be used to trigger the DMA. A read of the I2CxRXB register will clear both RXBF and I2CxRXIF bits. Setting the [CLRBF](#_bookmark47) bit clears the I2CxRXIF bit, and the I2CxRXB and [I2CxTXB](#_bookmark87) registers.

If the buffer is read while empty ([RXBF](#_bookmark48) = 0), the Receive Read Error Status ([RXRE](#_bookmark46)) bit is set, and the module generates a NACK. User software must clear RXRE to resume normal operation.

When the module attempts to transfer the contents of the receive shift register to [I2CxRXB](#_bookmark88) while I2CxRXB is full ([RXBF](#_bookmark48) = 1), the Receive Overflow Status ([RXO](#_bookmark27)) bit is set, and a NACK is generated. The data currently stored in I2CxRXB remains unchanged, but the data in the receive shift register is lost.

**Important:** A receive overflow can only occur when clock stretching is disabled. Clock stretching prevents receive overflows because the receive shift register cannot receive any more data until user software or the DMA reads [I2CxRXB](#_bookmark88) and the SCL line is released.

## Clock Stretching

Clock stretching occurs when a client device holds the SCL line low to pause bus communication. A client device may stretch the clock to allow more time to process incoming data, prepare a response for the host device, or to prevent Receive Overflow or Transmit Underflow conditions. Clock stretching is enabled by clearing the Clock Stretch Disable ([CSD](#_bookmark29)) bit, and is only available in Client and Multi-Host modes.

When clock stretching is enabled ([CSD](#_bookmark29) = 0), the Client Clock Stretching ([CSTR](#_bookmark18)) bit can be used to determine if the clock is currently being stretched. While the client is actively stretching the clock,

CSTR is set by hardware (CSTR = 1). Once the client has completed its current transaction and

clock stretching is no longer required, either module hardware or user software must clear CSTR to release the clock and resume communication.

## Clock Stretching for Buffer Operations

When enabled ([CSD](#_bookmark29) = 0), clock stretching is forced during buffer read/write operations. This allows the client device time to either load [I2CxTXB](#_bookmark87) with transmit data, or read data from [I2CxRXB](#_bookmark88) to clear the buffer.

In Client Receive mode, clock stretching prevents receive data overflows. When the first seven bits of a new byte are received into the receive shift register while [I2CxRXB](#_bookmark88) is full ([RXBF](#_bookmark48) = 1), client hardware automatically stretches the clock and sets [CSTR](#_bookmark18). When the client has read the data in I2CxRXB, client hardware automatically clears CSTR to release the SCL line and continue communication (see [Figure](#_bookmark3) [36-9](#_bookmark3)).

**Figure 36-9.** Receive Buffer Clock Stretching

## SCL

**SDA**

###### Hardware reads

**RXBF = 1**



###### 1 2 3 4 5 6 7

**Clock stretched**

1

D7 D6 D5 D4 D3 D2 D

###### Client

**releases SCL**

###### 8

D0

## RXBF

**CSTR**

###### Software reads

**I2CxRXB**

###### Hardware sets

**CSTR = 1**

###### Hardware clears RXBF

**Hardware**

###### clears CSTR

In Client Transmit mode, clock stretching prevents transmit underflows. When [I2CxTXB](#_bookmark87) is empty ([TXBE](#_bookmark45) = 1) and the [I2CxCNT](#_bookmark74) register is nonzero (I2CxCNT != 0), client hardware stretches the clock

and sets [CSTR](#_bookmark18) upon the 8th falling SCL edge. Once the client has loaded new data into I2CxTXB, client hardware automatically clears CSTR to release the SCL line and allow further communication (see [Figure 36-10](#_bookmark4)).

**Figure 36-10.** Transmit Buffer Clock Stretching

## SCL

###### Hardware reads

**TXBE = 1**



###### 1 2 3 4 5 6 7

**Host releases SDA to**

###### allow ACK

**8**

###### Clock Stretched

**Client**

###### releases SCL

**9**

###### Client copies

**ACKDT onto**

## SDA

**TXBE**

## CSTR

D7 D6 D5 D4 D3 D2 D1 D0

###### Software loads I2CxTXB

**Hardware sets CSTR = 1**

###### SDA (ACKDT = 0)

**Hardware**

###### clears TXBE

**Hardware clears CSTR**

## Clock Stretching for Other Client Operations

The I2C module provides three Interrupt and Hold Enable features:

* + - * + Address Interrupt and Hold Enable
        + Data Write Interrupt and Hold Enable
        + Acknowledge Status Time Interrupt and Hold Enable

When clock stretching is enabled ([CSD](#_bookmark29) = 0), the Interrupt and Hold Enable features provide an interrupt response, and stretches the clock to allow time for address recognition, data processing, or an ACK/NACK response.

The Address Interrupt and Hold Enable feature will generate an interrupt event and stretch the SCL line when a matching address is received. This feature is enabled by setting the Address Interrupt and Hold Enable ([ADRIE](#_bookmark61)) bit. When enabled (ADRIE = 1), the [CSTR](#_bookmark18) bit and the Address Interrupt Flag ([ADRIF](#_bookmark53)) bit are set by module hardware, and the SCL line is stretched following the 8th falling SCL edge of a received matching address. Once the client has completed processing the address, software determines whether to send an ACK or a NACK back to the host device. Client software must clear both the ADRIF and CSTR bits to resume communication.

**Important:** In 10-bit Client Addressing mode, clock stretching occurs only after the client receives a matching low address byte, or a matching high address byte with the R/W bit = 1 (Host read) while the Client Mode Active ([SMA](#_bookmark39)) bit is set (SMA = 1). Clock stretching does not occur after the client receives a matching high address byte with the R/W bit = 0 (Host write).

The Data Write Interrupt and Hold Enable feature provides an interrupt event and stretches the SCL signal after the client receives a data byte. This feature is enabled by setting the Data Write Interrupt and Hold Enable ([WRIE](#_bookmark60)) bit. When enabled (WRIE = 1), module hardware sets both the [CSTR](#_bookmark18) bit

and the Data Write Interrupt Flag ([WRIF](#_bookmark52)) bit and stretches the SCL line after the 8th falling edge of SCL. Once the client has read the new data, software determines whether to send an ACK or a NACK back to the host device. Client software must clear both the CSTR and WRIF bits to resume communication.

The Acknowledge Status Time Interrupt and Hold Enable feature generates an interrupt event and stretches the SCL line after the acknowledgement phase of a transaction. This feature is enabled by setting the Acknowledge Status Time Interrupt and Hold Enable ([ACKTIE](#_bookmark59)) bit. When enabled (ACKTIE

= 1), module hardware sets the [CSTR](#_bookmark18) bit and the Acknowledge Status Time Interrupt Flag ([ACKTIF](#_bookmark51)) bit and stretches the clock after the 9th falling edge of SCL for all address, read, or write operations. Client software must clear both the ACKTIF and CSTR bits to resume communication.

## Data Byte Count

The data byte count refers to the number of data bytes in a complete I2C packet. The data byte count does not include address bytes. The I2C Byte Count ([I2CxCNT](#_bookmark74)) register is used to specify the length, in bytes, of the complete transaction. The value loaded into I2CxCNT will be decremented by module hardware each time a data byte is transmitted or received by the module.

**Important:** The [I2CxCNT](#_bookmark74) register will not decrement past a zero value.

When a byte transfer causes the [I2CxCNT](#_bookmark74) register to decrement to ‘0’, the Byte Count Interrupt Flag ([CNTIF](#_bookmark50)) bit is set, and if the Byte Count Interrupt Enable ([CNTIE](#_bookmark58)) is set, the general purpose I2C Interrupt Flag (I2CxIF) bit of the Peripheral Interrupt Registers (PIR) is also set. If the I2C Interrupt Enable (I2CxIE) bit of the Peripheral Interrupt Enable (PIE) registers is set, module hardware will generate an interrupt event.

**Important:** The I2CxIF bit is read-only and can only be cleared by clearing all the interrupt flag bits of the [I2CxPIR](#_bookmark49) register.

The [I2CxCNT](#_bookmark74) register can be read at any time, but it is recommended that a double read is performed to ensure a valid count value.

The [I2CxCNT](#_bookmark74) register can be written to; however, care is required to prevent register corruption. If the I2CxCNT register is written to during the 8th falling SCL edge of a reception, or during the 9th falling SCL edge of a transmission, the register value may be corrupted. In Client mode, I2CxCNT can be safely written to any time the clock is not being stretched ([CSTR](#_bookmark18) = 0), or after a Stop condition has been received (Stop Condition Interrupt Flag ([PCIF](#_bookmark54)) = 1). In Host mode, I2CxCNT can be safely written

to any time the Host Data Ready ([MDR](#_bookmark19)) or Bus Free ([BFRE](#_bookmark38)) bits are set. If the I2C packet is longer than

65,536 bytes, the I2CxCNT register can be updated mid-message to prevent the count from reaching zero; however, the preventative measures listed above must be followed.

When in either Client Read or Host Write mode and the [I2CxCNT](#_bookmark74) value is nonzero (I2CxCNT != 0), the value of the [ACKDT](#_bookmark23) bit is used as the acknowledgement response. When I2CxCNT reaches zero (I2CxCNT = 0), the value of the Acknowledge End of Count ([ACKCNT](#_bookmark22)) bit is used for the acknowledgement response.

In Host read or write operations, when the [I2CxCNT](#_bookmark74) register is clear (I2CxCNT = 0) and the Restart Enable ([RSEN](#_bookmark16)) bit is clear, host hardware automatically generates a Stop condition upon the 9th falling edge of SCL. When I2CxCNT is clear (I2CxCNT = 0) and RSEN is set (RSEN = 1), host hardware

will stretch the clock while it waits for the Start ([S](#_bookmark17)) bit to be set (S = 1). When the Start bit has been set, module hardware transmits a Restart condition followed by the address of the client it wishes to communicate with.

## 36.3.12.1 Auto-Load I2CxCNT

The [I2CxCNTL](#_bookmark74) register can be automatically loaded. Auto-loading of the I2CxCNTL register is enabled when the Auto-Load I2C Count Register Enable ([ACNT](#_bookmark31)) bit is set (ACNT = 1).

In Host Transmit mode, the first byte following either the 7-bit or 10-bit client address is transferred from [I2CxTXB](#_bookmark87) into both [I2CxCNTL](#_bookmark74) and the transmit shift register.

In Host Reception mode, the first byte received from the client are loaded into both [I2CxCNTL](#_bookmark74) and [I2CxRXB](#_bookmark88). The value of the Acknowledge Data ([ACKDT](#_bookmark23)) bit is used as the host’s acknowledgement response to prevent a false NACK from being generated before the I2CxCNTL register is updated with the new count value.

In Client Reception mode, the first byte received after receiving a matching 7-bit or 10-bit address is loaded into both [I2CxCNTL](#_bookmark74) and [I2CxRXB](#_bookmark88), and the value of the [ACKDT](#_bookmark23) bit is used as the client’s acknowledgement response.

In Client Transmit mode, the first byte loaded into [I2CxTXB](#_bookmark87) following the reception of a matching 7-bit or 10-bit address is transferred into both [I2CxCNTL](#_bookmark74) and the transmit shift register.

**Important:** It is not necessary to preload the [I2CxCNT](#_bookmark74) register when using the auto-load feature. If no value is loaded by the 9th falling SCL edge following an address transmission or reception, the Byte Count Interrupt Flag ([CNTIF](#_bookmark50)) will be set by module hardware, and must be cleared by software to prevent an interrupt event before I2CxCNTL is updated.

Alternatively, I2CxCNTL can be preloaded with a nonzero value to prevent the CNTIF from being set. In this case, the preloaded value will be overwritten once the new count value has been loaded into I2CxCNTL.

## DMA Integration

The I2C module can be used with the DMA for data transfers. The DMA can be triggered through software via the DMA Transaction (DGO) bit, or through the use of the following hardware triggers:

* + - * I2C Transmit Interrupt Flag (I2CxTXIF)
      * I2C Receive Interrupt Flag (I2CxRXIF)
      * I2C Interrupt Flag (I2CxIF)
      * I2C Error Interrupt Flag (I2CxEIF)

For I2C communication, the I2CxTXIF is commonly used as the hardware trigger source for host or client transmission, and I2CxRXIF is commonly used as the hardware trigger source for host or client reception.

## 7-Bit Host Transmission

When address buffers are enabled ([ABD](#_bookmark34) = 0), [I2CxADB1](#_bookmark82) is loaded with the client address, and [I2CxCNT](#_bookmark74) is loaded with a count value. At this point, [I2CxTXB](#_bookmark87) does not contain data, and the Transmit Buffer Empty ([TXBE](#_bookmark45)) bit is set (TXBE = 1). The I2CxTXIF bit is not set since it can only be set when the Host Mode Active ([MMA](#_bookmark40)) and TXBE bits are set. Once software sets the Start ([S](#_bookmark17)) bit, the MMA bit is set and hardware transmits the client address. Upon the 8th falling SCL edge, since TXBE = 1, the Host Data Request ([MDR](#_bookmark19)) and I2CxTXIF bits are set, and hardware stretches the clock while the DMA loads I2CxTXB with data. Once the DMA loads I2CxTXB, the TXBE, MDR and I2CxTXIF bits are cleared by hardware, and the DMA waits for the next occurrence of I2CxTXIF being set.

When address buffers are disabled ([ABD](#_bookmark34) = 1), software must load [I2CxTXB](#_bookmark87) with the client address to begin transmission. This is because I2CxTXIF can only be set when [MMA](#_bookmark40) = 1, and since a Start has

not occurred, MMA = 0. Once the address has been transmitted, I2CxTXIF will be set, triggering the DMA to load I2CxTXB with data.

## 10-Bit Host Transmission

When address buffers are enabled ([ABD](#_bookmark34) = 0), [I2CxADB1](#_bookmark82) is loaded with the client high address, [I2CxADB0](#_bookmark81) is loaded with the client low address, and [I2CxCNT](#_bookmark74) is loaded with a count value. Once software sets the Start ([S](#_bookmark17)) bit, the [MMA](#_bookmark40) bit is set and hardware transmits the 10-bit client address. Upon the 8th falling SCL edge of the transmitted address low byte, since [TXBE](#_bookmark45) = 1, the [MDR](#_bookmark19) and I2CxTXIF bits are set, and hardware stretches the clock while the DMA loads [I2CxTXB](#_bookmark87) with data. Once the DMA loads I2CxTXB, the TXBE, MDR and I2CxTXIF bits are cleared by hardware, and the DMA waits for the next occurrence of I2CxTXIF being set.

When address buffers are disabled ([ABD](#_bookmark34) = 1), software must load [I2CxTXB](#_bookmark87) with the client high address to begin transmission. Once the client high address has been transmitted, I2CxTXIF will be set, triggering the DMA to load I2CxTXB with client low address. Once the DMA loads I2CxTXB with the client low address, the [TXBE](#_bookmark45), [MDR](#_bookmark19) and I2CxTXIF bits are cleared by hardware, and the DMA waits for the next occurrence of I2CxTXIF being set.

## 7/10-Bit Host Reception

In both 7-bit and 10-bit Host Receive modes, the state of the [ABD](#_bookmark34) bit is ignored. Once the complete 7-bit or 10-bit address has been received by the client, the client will transmit a data byte. Once the byte has been received by the host, hardware sets the I2CxRXIF bit, which triggers the DMA to read [I2CxRXB](#_bookmark88). Once the DMA has read I2CxRXB, I2CxRXIF is cleared by hardware and the DMA waits for the next occurrence of I2CxRXIF being set.

## 7-Bit Client Transmission

In 7-bit Client Transmission mode, the state of [ABD](#_bookmark34) is ignored. If the client receives the matching 7-bit address and [TXBE](#_bookmark45) is set, I2CxTXIF is set by hardware, triggering the DMA to load data into

[I2CxTXB](#_bookmark87). Once the data is transmitted from I2CxTXB, I2CxTXIF is set by hardware, triggering the DMA to once again load I2CxTXB with data. The DMA will continue to load data into I2CxTXB until [I2CxCNT](#_bookmark74) reaches a zero value. Once I2CxCNT reaches zero and the data is transmitted from I2CxTXB, I2CxTXIF will not be set, and the DMA will stop loading data.

## 10-Bit Client Transmission

In 10-bit Client Transmission mode, the state of [ABD](#_bookmark34) is ignored. If there is no data in [I2CxTXB](#_bookmark87) after the client has received the address high byte with the R/W bit set, hardware sets I2CxTXIF, triggering the DMA to load I2CxTXB. The DMA will continue to load data into I2CxTXB until [I2CxCNT](#_bookmark74) reaches a zero value. Once I2CxCNT reaches zero and the data is transmitted from I2CxTXB, I2CxTXIF will not be set, and the DMA will stop loading data.

## 7/10-Bit Client Reception

When address buffers are enabled ([ABD](#_bookmark34) = 0), client hardware loads [I2CxADB0](#_bookmark81)/[1](#_bookmark82) with the matching address, while all data is received by [I2CxRXB](#_bookmark88). Once the client loads I2CxRXB with a received data byte, hardware sets I2CxRXIF, which triggers the DMA to read I2CxRXB. The DMA will continue to read I2CxRXB whenever I2CxRXIF is set.

When address buffers are disabled ([ABD](#_bookmark34) = 1), the client loads I2CxRXB with the matching address byte(s) as they are received. Each received address byte sets I2CxRXIF, which triggers the DMA to read I2CxRXB. The DMA will continue to read I2CxRXB whenever I2CxRXIF is set.

## Interrupts

The I2C module offers several interrupt features designed to assist with communication functions. The interrupt hardware contains four high-level interrupts and several condition-specific interrupts.

## High-Level Interrupts

Module hardware provides four high-level interrupts:

* + - * + Transmit
        + Receive
        + General Purpose
        + Error

These flag bits are read-only bits, and cannot be cleared by software.

The I2C Transmit Interrupt Flag (I2CxTXIF) bit is set when the [I2CxCNT](#_bookmark74) register is nonzero (I2CxCNT != 0), and the transmit buffer, [I2CxTXB](#_bookmark87), is empty as indicated by the Transmit Buffer Empty Status ([TXBE](#_bookmark45)) bit (TXBE = 1). If the I2C Transmit Interrupt Enable (I2CxTXIE) bit is set, an interrupt event will occur when the I2CxTXIF bit becomes set. Writing new data to [I2CxTXB](#_bookmark87), or setting the Clear Buffer ([CLRBF](#_bookmark47)) bit, will clear the interrupt condition. The I2CxTXIF bit is also used by the DMA as a trigger source.

**Important:** I2CxTXIF can only be set when either the Client Mode Active ([SMA](#_bookmark39)) or Host Mode Active ([MMA](#_bookmark40)) bits are set, and the [I2CxCNT](#_bookmark74) register is nonzero (I2CxCNT != 0). The SMA bit is only set after an address has been successfully acknowledged by a client device, which prevents false interrupts from being triggered on address reception. The MMA bit is set once the host completes the transmission of a Start condition.

The I2C Receive Interrupt Flag (I2CxRXIF) bit is set when the receive shift register has loaded new data into the receive buffer, [I2CxRXB](#_bookmark88). When new data is loaded into I2CxRXB, the Receive Buffer Full Status ([RXBF](#_bookmark48)) bit is set (RXBF = 1), which also sets I2CxRXIF. If the I2C Receive Interrupt Enable (I2CxRXIE) bit is set, an interrupt event will occur when the I2CxRXIF bit becomes set. Reading data from [I2CxRXB](#_bookmark88), or setting the [CLRBF](#_bookmark47) bit, will clear the interrupt condition. The I2CxRXIF bit is also used by the DMA as a trigger source.

**Important:** I2CxRXIF can only be set when either the Client Mode Active ([SMA](#_bookmark39)) or Host Mode Active ([MMA](#_bookmark40)) bits are set.

The I2C Interrupt Flag (I2CxIF) is the general purpose interrupt. I2CxIF is set whenever any of the interrupt flag bits contained in the I2C Peripheral Interrupt ([I2CxPIR](#_bookmark49)) Register and the associated interrupt enable bits contained in the I2C Peripheral Interrupt Enable ([I2CxPIE](#_bookmark57)) Register are set. If I2CxIF becomes set while the I2C Interrupt Enable (I2CxIE) bit is set, an interrupt event will occur. I2CxIF is cleared by module hardware when all enabled interrupt flag bits in I2CxPIR are clear.

The I2C Error Interrupt Flag (I2CxEIF) is set whenever any of the interrupt flag bits contained in the I2C Error ([I2CxERR](#_bookmark65)) Register and their associated interrupt enable bits are set. If I2CxEIF becomes set while the I2C Error Interrupt Enable (I2CxEIE) bit is set, an interrupt event will occur. I2CxEIF is cleared by hardware when all enabled error interrupt flag bits in the I2CxERR register are clear.

## Condition-Specific Interrupts

In addition to the high-level interrupts, module hardware provides several condition-specific interrupts.

The I2C Peripheral Interrupt ([I2CxPIR](#_bookmark49)) Register contains the following interrupt flag bits:

* + - * + [CNTIF](#_bookmark50): Byte Count Interrupt Flag
        + [ACKTIF](#_bookmark51): Acknowledge Status Time Interrupt Flag
        + [WRIF](#_bookmark52): Data Write Interrupt Flag
        + [ADRIF](#_bookmark53): Address Interrupt Flag
        + [PCIF](#_bookmark54): Stop Condition Interrupt Flag
        + [RSCIF](#_bookmark55): Restart Condition Interrupt Flag
        + [SCIF](#_bookmark56): Start Condition Interrupt Flag

When any of the flag bits in [I2CxPIR](#_bookmark49) becomes set and the associated interrupt enable bits in [I2CxPIE](#_bookmark57) are set, the generic I2CxIF is also set. If the generic I2CxIE bit is set, an interrupt event is generated whenever one of the I2CxPIR flag bits becomes set. If the I2CxIE bit is clear, the I2CxPIR flag bit will still be set by hardware; however, no interrupt event will be triggered.

[CNTIF](#_bookmark50) becomes set (CNTIF = 1) when the [I2CxCNT](#_bookmark74) register value reaches zero, indicating that all data bytes in the I2C packet have been transmitted or received. CNTIF is set after the 9th falling SCL edge when I2CxCNT reaches zero (I2CxCNT = 0).

[ACKTIF](#_bookmark51) is set (ACKTIF = 1) by the 9th falling edge of SCL for any byte when the device is addressed as a client in any Client or Multi-Host mode. If the Acknowledge Interrupt and Hold Enable ([ACKTIE](#_bookmark59)) bit is set and ACKTIF becomes set:

* + - * + If an ACK is detected, clock stretching is also enabled ([CSTR](#_bookmark18) = 1)
        + If a NACK is detected, no clock stretching occurs (CSTR = 0)

[WRIF](#_bookmark52) is set (WRIF = 1) after the 8th falling edge of SCL when the module receives a data byte in Client or Multi-Host modes. Once the data byte is received, WRIF is set, as is the Receive Buffer Full Status ([RXBF](#_bookmark48)) and the I2CxRXIF bits, and if the Data Write Interrupt and Hold Enable ([WRIE](#_bookmark60)) bit is set, the generic I2CxIF bit is also set. WRIF is a read/write bit and must be cleared in software, while the [RXBF](#_bookmark48), I2CxRXIF and I2CxIF bits are read-only and are cleared by reading [I2CxRXB](#_bookmark88) or by setting the Clear Buffer bit ([CLRBF](#_bookmark47) = 1).

[ADRIF](#_bookmark53) is set on the 8th falling edge of SCL after the module has received a matching 7-bit address, after receiving a matching 10-bit upper address byte, and after receiving a matching 10-bit lower address byte in Client or Multi-Host modes. Upon receiving a matching 7-bit address or 10-bit upper address, the address is copied to [I2CxADB0](#_bookmark81), the R/W bit setting is copied to the Read Information

([R](#_bookmark41)) bit, the Data ([D](#_bookmark42)) bit is cleared, and the ADRIF bit is set. If the Address Interrupt and Hold Enable ([ADRIE](#_bookmark61)) bit is set, I2CxIF is set, and the clock will be stretched while the module determines whether to ACK or NACK the transmitter. Upon receiving the matching 10-bit lower address, the address is copied to [I2CxADB1](#_bookmark82), and the ADRIF bit is set. If ADRIE is also set, the clock is stretched while the module determines the ACK/NACK response to return to the transmitter.

[PCIF](#_bookmark54) is set whenever a Stop condition is detected on the bus. [RSCIF](#_bookmark55) is set upon the detection of a Restart condition.

[SCIF](#_bookmark56) is set upon the detection of a Start condition.

In addition to the [I2CxPIR](#_bookmark49) register, the I2C Error ([I2CxERR](#_bookmark65)) register contains three interrupt flag bits that are used to detect bus errors. These read/write bits are set by module hardware, but must be cleared by user software. The I2CxERR register also includes the interrupt enable bits for these three Error conditions, and when set, will cause an interrupt event whenever the associated interrupt flag bit becomes set.

[I2CxERR](#_bookmark65) contains the following interrupt flag bits:

* + - * + [BTOIF](#_bookmark66): Bus Time-Out Interrupt Flag
        + [BCLIF](#_bookmark67): Bus Collision Interrupt Flag
        + [NACKIF](#_bookmark68): NACK Detect Interrupt Flag

[BTOIF](#_bookmark66) is set when a bus time-out occurs. The bus time-out period is configured using one of the time-out sources selected by the I2C Bus Time-Out Clock Source Selection ([I2CxBTOC](#_bookmark79)) register.

If the module is configured in Client mode with [TOREC](#_bookmark76) set (TOREC = 1), and a bus time-out event occurs (regardless of the state of the Client Mode Active ([SMA](#_bookmark39)) bit), the module is immediately reset,

the SMA and Client Clock Stretching ([CSTR](#_bookmark18)) bits are cleared, and the BTOIF bit is set. If the Bus Time-Out Interrupt Enable ([BTOIE](#_bookmark69)) bit is set, the generic I2C Error Interrupt Flag (I2CxEIF) bit is set.

If the module is configured in Client mode with [TOREC](#_bookmark76) clear (TOREC = 0), and a bus time-out event occurs (regardless of the state of the Client Mode Active ([SMA](#_bookmark39)) bit), the BTOIF bit is set, but user software must reset the module. If the Bus Time-Out Interrupt Enable ([BTOIE](#_bookmark69)) bit is set, the generic I2C Error Interrupt Flag (I2CxEIF) bit is set.

If the module is configured in Host mode with [TOREC](#_bookmark76) set (TOREC = 1), and the bus time-out event occurs while the Host is active (Host Mode Active ([MMA](#_bookmark40)) = 1), the Host Data Ready ([MDR](#_bookmark19)) bit is cleared, the module will immediately attempt to transmit a Stop condition, and sets the BTOIF bit. Stop condition generation may be delayed if a client device is stretching the clock, but will resume once the clock is released, or if the client holding the bus also has a time-out event occur. The MMA bit is only cleared after the Stop condition has been generated. If the Bus Time-Out Interrupt Enable ([BTOIE](#_bookmark69)) bit is set, the generic I2C Error Interrupt Flag (I2CxEIF) bit is set.

If the module is configured in Host mode with [TOREC](#_bookmark76) clear (TOREC = 0), and the bus time-out event occurs while the Host is active (Host Mode Active ([MMA](#_bookmark40)) = 1), the [MDR](#_bookmark19) bit is cleared and the BTOIF bit is set, but user software must initiate the Stop condition by setting the [P](#_bookmark26) bit. If the Bus Time-Out Interrupt Enable ([BTOIE](#_bookmark69)) bit is set, the generic I2C Error Interrupt Flag (I2CxEIF) bit is set.

[BCLIF](#_bookmark67) is set upon the detection of a bus collision. A bus collision occurs any time the SDA line is sampled at a logic low while the module expects both SCL and SDA lines to be at a high logic level. When a bus collision occurs, BCLIF is set, and if the Bus Collision Detect Interrupt Enable ([BCLIE](#_bookmark70)) bit is set, I2CxEIF is also set, and the module is reset.

[NACKIF](#_bookmark68) is set when either the host or client is active ([SMA](#_bookmark39) = 1 || [MMA](#_bookmark40) = 1) and a NACK response is detected on the bus. A NACK response occurs during the 9th SCL pulse in which the SDA line is released to a logic high. In Host mode, a NACK can be issued when the host has finished receiving

data from a client, or when the host receives incorrect data. In Client mode, a NACK is issued when the client does not receive a matching address, or when it receives incorrect data. A NACK can also be automatically issued when any of the following bits becomes set, which will also set NACKIF and I2CxEIF:

* + - * + [TXWE](#_bookmark44): Transmit Write Error Status
        + [RXRE](#_bookmark46): Receive Read Error Status
        + [TXU](#_bookmark28): Transmit Underflow Status
        + [RXO](#_bookmark27): Receive Overflow Status

**Important:** The I2CxEIF bit is read-only, and is only cleared by hardware after all enabled [I2CxERR](#_bookmark65) error flags have been cleared.

## Operation in Sleep

The I2C module can operate while in Sleep mode.

In Client mode, the module can transmit and receive data as long as the system clock source operates in Sleep. If the generic I2C Interrupt Enable (I2CxIE) bit is set and the client receives or transmits a complete byte, I2CxIF is set and the device wakes up from Sleep.

In Host mode, both the system clock and the selected [I2CxCLK](#_bookmark72) source must be able to operate in Sleep. If the I2CxIE bit is set and the I2CxIF bit becomes set, the device wakes from Sleep.

# I2C Operation

All I2C communication is performed in 9-bit segments consisting of an 8-bit address/data segment followed by a 1-bit acknowledgement segment. Address and data bytes are transmitted with the

Most Significant bit (MSb) first. Interaction between the I2C module and other devices on the bus is controlled and monitored through several I2C Control, Status, and Interrupt registers.

To begin any I2C communication, mater hardware checks to ensure that the bus is in an Idle state as indicated by the Bus Free Status ([BFRE](#_bookmark38)) bit. When BFRE = 1, both SDA and SCL lines are floating to

a logic high and the bus is considered ‘Idle’. When the host detects an Idle bus, it transmits a Start condition, followed by the address of the client it intends to communicate with. The client address can be either 7-bit or 10-bit, depending on the application design.

In 7-bit Addressing mode, the Least Significant bit (LSb) of the 7-bit client address is reserved for the Read/not Write (R/W) bit, while in 10-bit Addressing mode, the LSb of the high address byte is reserved as the R/W bit. If the R/W bit is clear (R/W = 0), the host intends to read information from the client. If R/W is set (R/W = 1), the host intends to write information to the client. If the addressed client exists on the bus, it must respond with an Acknowledgement (ACK) sequence.

Once a client has been successfully addressed, the host will continue to receive data from the client, write data to the client, or a combination of both. Data is always transmitted Most Significant bit (MSb) first. When the host has completed its transactions, it can either issue a Stop condition, signaling to the client that communication is to be terminated, or a Restart condition, informing the bus that the current host wishes to hold the bus to communicate with the same or other client devices.

## I2C Client Mode Operation

The I2C module provides four Client Operation modes as selected by the I2C Mode Select ([MODE](#_bookmark20)) bits:

* + - * I2C Client mode with recognition of up to four 7-bit addresses
      * I2C Client mode with recognition of up to two masked 7-bit addresses
      * I2C Client mode with recognition of up to two 10-bit addresses
      * I2C Client mode with recognition of one masked 10-bit address

During operation, the client device waits until module hardware detects a Start condition on the bus. Once the Start condition is detected, the client waits for the incoming address information to be received by the receive shift register. The address is then compared to the addresses stored in the I2C Address 0/1/2/3 registers ([I2CxADR0](#_bookmark83), [I2CxADR1](#_bookmark84), [I2CxADR2](#_bookmark85), [I2CxADR3](#_bookmark86)), and if an address match is detected, client hardware transfers the matching address into either the [I2CxADB0](#_bookmark81)/[I2CxADB1](#_bookmark82) registers or the [I2CxRXB](#_bookmark88) register, depending on the state of the Address Buffer Disable ([ABD](#_bookmark34)) bit. If there are no address matches, there is no response from the client.

## Client Addressing Modes

The [I2CxADR0](#_bookmark83), [I2CxADR1](#_bookmark84), [I2CxADR2](#_bookmark85) and [I2CxADR3](#_bookmark86) registers contain the client’s addresses. The first byte (7-bit mode) or first and second bytes (10-bit mode) following a Start or Restart condition are compared to the values stored in the I2CxADR registers (see [Figure 36-11](#_bookmark5)). If an address match occurs, the valid address is transferred to the [I2CxADB0](#_bookmark81)/[I2CxADB1](#_bookmark82) registers or [I2CxRXB](#_bookmark88) register, depending on the Addressing mode and the state of the [ABD](#_bookmark34) bit.

**Table 36-2.** I2C Address Registers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mode** | [**I2CxADR0**](#_bookmark83) | [**I2CxADR1**](#_bookmark84) | [**I2CxADR2**](#_bookmark85) | [**I2CxADR3**](#_bookmark86) |
| 7-bit | 7-bit address | 7-bit address | 7-bit address | 7-bit address |
| 7-bit w/ masking | 7-bit address | 7-bit mask for I2CxADR0 | 7-bit address | 7-bit mask for I2CxADR2 |
| 10-bit | Address low byte | Address high byte | Address low byte | Address high byte |
| 10-bit w/ masking | Address low byte | Address high byte | Address low byte mask | Address high byte mask |

In 7-bit Address mode, the received address byte is compared to all four I2CxADR registers independently to determine a match. The R/W bit is ignored during address comparison. If a match occurs, the matching received address is transferred from the receive shift register to either the

[I2CxADB0](#_bookmark81) register (when [ABD](#_bookmark34) = 0) or to the [I2CxRXB](#_bookmark88) register (when [ABD](#_bookmark34) = 1), and the value of the R/W bit is loaded into the Read Information ([R](#_bookmark41)) bit.

In 7-bit Address with Masking mode, [I2CxADR0](#_bookmark83) holds one client address and [I2CxADR1](#_bookmark84) holds the mask value for I2CxADR0, while [I2CxADR2](#_bookmark85) holds a second client address and [I2CxADR3](#_bookmark86) holds the mask value for I2CxADR2. A zero bit in a mask register means that the associated bit in the address register is a ‘don’t care’, which means that the particular address bit is not used in the address comparison between the received address in the shift register and the address stored in either I2CxADR0 or I2CxADR2 (see [Figure 36-11](#_bookmark5)).

**Figure 36-11.** 7-Bit Address with Masking Example

###### I2CxRSR (receive shift register)

**Bits compared**

###### I2CxADR0 (Client Address)

**I2CxADR1 (Address Mask)**

**R/W**

**X X**

**7-bit address**

**X**

**0**

**0**

**1**

**0**

**0**

**1**

**1**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **1** | **0** | **0** | **1** | **1** | **1** | **X** |

**Bits ignored**

**(masked)**

## Mask bit = 0: associated

**address bit is ignored**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **1** | **1** | **1** | **1** | **0** | **0** | **X** |

In 10-bit Address mode, [I2CxADR0](#_bookmark83) and [I2CxADR1](#_bookmark84), and [I2CxADR2](#_bookmark85) and [I2CxADR3](#_bookmark86) are combined to create two 10-bit addresses. I2CxADR0 and I2CxADR2 hold the lower eight bits of the address, while I2CxADR1 and I2CxADR3 hold the upper two bits of the address, the R/W bit, and the five-digit ‘11110’ code assigned to the five Most Significant bits of the high address byte.

**Important:** The ‘11110’ code is specified by the I2C Specification, but is not supported by Microchip. It is up to the user to ensure the correct bit values are loaded into the address high byte. If a host device has included the five-digit code in the address it intends to transmit, the client must also include those bits in client address.

The upper received address byte is compared to the values in [I2CxADR1](#_bookmark84) and [I2CxADR3](#_bookmark86). If a match occurs, the address is stored in either [I2CxADB1](#_bookmark82) (when [ABD](#_bookmark34) = 0) or in [I2CxRXB](#_bookmark88) (when ABD = 1), and the value of the R/W bit is transferred into the [R](#_bookmark41) bit. The lower received address byte is compared to the values in [I2CxADR0](#_bookmark83) and [I2CxADR2](#_bookmark85), and if a match occurs, the address is stored in either [I2CxADB0](#_bookmark81) (when ABD = 0) or in I2CxRXB (when ABD = 1).

In 10-bit Address with Masking mode, [I2CxADR0](#_bookmark83) and [I2CxADR1](#_bookmark84) are combined to form the 10-bit address, while [I2CxADR2](#_bookmark85) and [I2CxADR3](#_bookmark86) are combined to form the 10-bit mask. The upper received address byte is compared to the masked value in I2CxADR1. If a match occurs, the address is stored in either [I2CxADB1](#_bookmark82) (when [ABD](#_bookmark34) = 0) or in [I2CxRXB](#_bookmark88) (when ABD = 1), and the value of the R/W bit is transferred into the [R](#_bookmark41) bit. The lower received address byte is compared to the value in I2CxADR0, and if a match occurs, the address is stored in either [I2CxADB0](#_bookmark81) (when ABD = 0) or in I2CxRXB (when ABD = 1).

## General Call Addressing Support

The I2C Specification reserves the address 0x00 as the General Call address. The General Call

address is used to address all client modules connected to the bus at the same time. When a host

issues a General Call, all client devices may respond with an ACK. The General Call Enable ([GCEN](#_bookmark32)) bit determines whether client hardware will respond to a General Call address. When GCEN is set (GCEN = 1), client hardware will respond to a General Call with an ACK, and when GCEN is clear (GCEN = 0), the General Call is ignored, and the client responds with a NACK.

When the module receives a General Call, the [ADRIF](#_bookmark53) bit is set and the address is stored in [I2CxADB0](#_bookmark81). If the [ADRIE](#_bookmark61) bit is set, the module will generate an interrupt and stretch the clock after the 8th falling edge of SCL. This allows the client to determine the acknowledgement response to return to the host (see [Figure 36-12](#_bookmark6)).

**Important:** When using the General Call addressing feature, loading the I2CxADR0/1/2/3 registers with the 0x00 address is not recommended. Additionally, client hardware only supports General Call addressing in 7-bit Addressing modes.

**Figure 36-12.** General Call Addressing

**SDA**

**Start (S)**

**General Call Address (0x00)**

**ACK**

**D7 D6 D5 D4 D3 D2 D1 D0**

**ACK**

**SCL**

**1 2 3 4 5 6 7 8**

**9 1 2 3 4 5 6 7 8 9**

**ADRIF**

**Cleared by software**

**General Call address**

**loaded into I2CxADB0**

## Client Operation in 7-Bit Addressing Modes

The upper seven bits of an address byte are used to determine a client’s address, while the LSb of the address byte is reserved as the Read/not Write (R/W) bit. When R/W is set (R/W = 1), the host device intends to read data from the client. When R/W is clear (R/W = 0), the host device

intends to write data to the client. When an address match occurs, the R/W bit is copied to the Read Information ([R](#_bookmark41)) bit, and the 7-bit address is copied to [I2CxADB0](#_bookmark81).

## Client Transmission (7-Bit Addressing Mode)

The following section describes the sequence of events that occur when the module is transmitting data in 7-bit Addressing mode:

The host device issues a Start condition. Once the Start condition has been detected, client hardware sets the Start Condition Interrupt Flag ([SCIF](#_bookmark56)) bit. If the Start Condition Interrupt Enable ([SCIE](#_bookmark64)) bit is also set, the generic I2CxIF is also set.

Host hardware transmits the 7-bit client address with the R/W bit set, indicating that it intends to read data from the client.

The received address is compared to the values in the I2CxADR registers. If the client is configured in 7-bit Addressing mode (no masking), the received address is independently compared to each of the I2CxADR0/1/2/3 registers. In 7-bit Addressing with Masking mode, the received address is compared to the masked value of [I2CxADR0](#_bookmark83) and [I2CxADR2](#_bookmark85).

If an address match occurs:

The Client Mode Active ([SMA](#_bookmark39)) bit is set by module hardware.

The R/W bit value is copied to the Read Information ([R](#_bookmark41)) bit by module hardware.

The Data ([D](#_bookmark42)) bit is cleared by hardware, indicating the last received byte was an address.

The Address Interrupt Flag ([ADRIF](#_bookmark53)) bit is set. If the Address Interrupt and Hold Enable ([ADRIE](#_bookmark61)) bit is set, and the Clock Stretching Disable ([CSD](#_bookmark29)) bit is clear, hardware sets the Client Clock Stretching ([CSTR](#_bookmark18)) bit and the generic I2CxIF bit. This allows time for the client to read either [I2CxADB0](#_bookmark81) or [I2CxRXB](#_bookmark88) and selectively ACK/NACK based on the received address. When the client has finished processing the address, software must clear CSTR to resume operation.

The matching received address is loaded into either the [I2CxADB0](#_bookmark81) register or into the [I2CxRXB](#_bookmark88) register as determined by the Address Buffer Disable ([ABD](#_bookmark34)) bit. When ABD is clear (ABD = 0), the matching address is copied to I2CxADB0. When ABD is set (ABD = 1), the matching address is copied to I2CxRXB, which also sets the Receive Buffer Full Status ([RXBF](#_bookmark48)) bit and the I2C Receive Interrupt Flag (I2CxRXIF) bit. I2CxRXIF is a read-only bit, and must be cleared by either reading I2CxRXB or by setting the Clear Buffer ([CLRBF](#_bookmark47)) bit (CLRBF = 1).

If no address match occurs, the module remains Idle.

If the Transmit Buffer Empty Status ([TXBE](#_bookmark45)) bit is set (TXBE = 1), [I2CxCNT](#_bookmark74) has a nonzero value (I2CxCNT != 0), and the I2C Transmit Interrupt Flag (I2CxTXIF) is set (I2CxTXIF = 1), client hardware sets [CSTR](#_bookmark18), stretches the clock (when [CSD](#_bookmark29) = 0), and waits for software to load [I2CxTXB](#_bookmark87) with

data. I2CxTXB must be loaded to clear I2CxTXIF. Once data is loaded into I2CxTXB, hardware automatically clears CSTR to resume communication.

The host device transmits the 9th clock pulse, and client hardware transfers the value of the [ACKDT](#_bookmark23) bit onto the SDA line. If there are pending errors, such as a receive overflow ([RXO](#_bookmark27) = 1), client hardware automatically generates a NACK condition. [NACKIF](#_bookmark68) is set, and the module goes Idle.

Upon the 9th falling SCL edge, the data byte in [I2CxTXB](#_bookmark87) is transferred to the transmit shift register, and [I2CxCNT](#_bookmark74) is decremented by one. Additionally, the Acknowledge Status Time Interrupt Flag ([ACKTIF](#_bookmark51)) bit is set. If the Acknowledge Status Time Interrupt and Hold Enable ([ACKTIE](#_bookmark59)) bit is also set, the generic I2CxIF is set, and if client hardware generated an ACK, the [CSTR](#_bookmark18) bit is also set and the clock is stretched (when [CSD](#_bookmark29) = 0). If a NACK was generated, the CSTR bit remains unchanged. Once complete, software must clear CSTR and ACKTIF to release the clock and continue operation.

If the client generated an ACK and [I2CxCNT](#_bookmark74) is nonzero, host hardware transmits eight clock pulses, and client hardware begins to shift the data byte out of the shift register starting with the Most Significant bit (MSb).

After the 8th falling edge of SCL, client hardware checks the status of [TXBE](#_bookmark45) and [I2CxCNT](#_bookmark74). If TXBE is set and I2CxCNT has a nonzero count value, hardware sets [CSTR](#_bookmark18) and the clock is stretched (when [CSD](#_bookmark29) = 0) until software loads [I2CxTXB](#_bookmark87) with new data. Once I2CxTXB has been loaded, hardware clears TXBE, I2CxTXIF, and CSTR to resume communication.

Once the host hardware clocks in all eight data bits, it transmits the 9th clock pulse along with the ACK/NACK response back to the client. Client hardware copies the ACK/NACK value to the Acknowledge Status ([ACKSTAT](#_bookmark24)) bit and sets [ACKTIF](#_bookmark51). If [ACKTIE](#_bookmark59) is also set, client hardware sets the generic I2CxIF bit and [CSTR](#_bookmark18), and stretches the clock (when [CSD](#_bookmark29) = 0). Software must clear CSTR to resume operation.

After the 9th falling edge of SCL, data currently loaded in [I2CxTXB](#_bookmark87) is transferred to the transmit shift register, setting both [TXBE](#_bookmark45) and I2CxTXIF. [I2CxCNT](#_bookmark74) is decremented by one. If I2CxCNT is zero (I2CxCNT = 0), [CNTIF](#_bookmark50) is set.

If [I2CxCNT](#_bookmark74) is nonzero and the host issued an ACK on the last byte ([ACKSTAT](#_bookmark24) = 0), the host transmits eight clock pulses, and client hardware begins to shift data out of the shift register.

Repeat steps 8 – 11 until the host has received all the requested data ([I2CxCNT](#_bookmark74) = 0). Once all data has been received, the host issues a NACK, followed by either a Stop or Restart condition. Once the NACK has been received by the client, hardware sets [NACKIF](#_bookmark68) and clears [SMA](#_bookmark39). If

the NACK Detect Interrupt Enable ([NACKIE](#_bookmark71)) bit is also set, the generic I2C Error Interrupt Flag (I2CxEIF) is set. If the host issued a Stop condition, client hardware sets the Stop Condition Interrupt Flag ([PCIF](#_bookmark54)). If the host issued a Restart condition, client hardware sets the Restart Condition Interrupt Flag ([RSCIF](#_bookmark55)). If the associated interrupt enable bits are also set, the generic I2CxIF is also set.

**Important:** I2CxEIF is read-only, and is cleared by hardware when all enable interrupt flag bits in [I2CxERR](#_bookmark65) are cleared.

**Start**

**R/W**

**ACK**

**(from**

**host)**

**NACK**

**(from**

**host)**

**Stop**

Rev. I2C Slave 5/8/2019

**Hardware sets PCIF**

**SDA**

A7 A6 A5 A4 A3 A2 A1 **1** D7 D6 D5 D4 D3 D2 D1 D0

D7 D6 D5 D4 D3 D2 D1 D0

**SCL**

**7-bit address**

1 2 3 4 5 6 7 8

**ACK (from client)**

9 1 2 3 4 5 6 7 8

9 1 2 3 4 5 6 7 8 9

**Hardware sets NACKIF**

**SMA**

**R**

**D**

**Hardware sets SMA**

**Hardware copies R/W value to R bit**

**Hardware clears D bit, last byte was address**

**Hardware sets D bit, last byte was data**

**Hardware clears**

**SMA**

**Hardware sets CNTIF**

**I2CxCNT**

**ACKSTAT**

**TXBE**

###### 0x02

**Before Start, software loads data into I2CxTXB**

###### 0x01

**Host's ACK copied to ACKSTAT**

###### 0x00

**Data byte transferred to shift register,**

**I2CxTXIF NOT set**

**Host's NACK copied to ACKSTAT**

**I2CxTXIF set,**

**data byte transferred to shift register**

**Software loads data into I2CxTXB, clearing I2CxTXIF**

**PIC18F06/16Q41**

**I2C - Inter-Integrated Circuit Module**

Data Sheet

© 2020-2023 Microchip Technology Inc. and its subsidiaries

DS40002214F - 626

**Start**

**R/W**

**ACK**

**(from**

**host)**

**NACK**

**(from**

**host)**

**Stop**

**SDA**

A7 A6 A5 A4 A3 A2 A1 **1**

**7-bit address**

D7 D6 D5 D4 D3 D2 D1 D0

**ACK (from client)**

D7 D6 D5 D4 D3 D2 D1 D0

**SCL**

1 2 3

4 5 6 7 8

9 1 2 3

4 5 6

7 8 9

1 2 3

4 5 6 7 8 9

**ADRIF**

**Hardware sets ADRIF**

**Software clears ADRIF**

**Hardware sets**

**CNTIF**

**CSTR**

**ACKSTAT**

**Hardware sets CSTR**

**Software clears CSTR**

**Host’s ACK copied to ACKSTAT**

**Host’s NACK**

**copied to ACKSTAT**

**I2CxCNT**

**TXBE**

###### 0x02

**Before Start, software loads data into I2CxTXB**

###### 0x01

**0x00**

**Data byte transferred to shift register, I2CxTXIF NOT set**

**I2CxTXIF set,**

**data byte transferred to shift register**

**Software loads data into I2CxTXB, clearing I2CxTXIF**

**PIC18F06/16Q41**

**I2C - Inter-Integrated Circuit Module**

Data Sheet

© 2020-2023 Microchip Technology Inc. and its subsidiaries

DS40002214F - 627

**Start**

**R/W**

**ACKDT value**

**copied to SDA**

**ACKCNT value copied to SDA**

**Stop**

**SDA**

A7 A6 A5 A4 A3 A2 A1 **0**

**7-bit address**

D7 D6 D5 D4 D3 D2 D1 D0

**ACK**

D7 D6 D5 D4 D3 D2 D1 D0

**NACK**

**SCL**

1 2 3 4 5 6 7 8 9

1 2 3 4 5

6 7 8 9

1 2 3

4 5 6 7 8 9

**RXBF**

**Matching received**

**address loaded**

**into I2CxADB0**

**I2CxRXIF set, data byte transferred to I2CxRXB**

**Software reads I2CxRXB, clearing I2CxRXIF**

**CSTR**

**Hardware sets**

**CSTR**

**Software clears**

**CSTR**

**ACKT**

**Hardware sets**

**ACKT**

**Hardware clears**

**ACKT**

**ACKTIF**

**Hardware sets**

**ACKTIF**

**Software clears**

**ACKTIF**

**Hardware sets CNTIF**

**I2CxCNT**

**0x02 0x01 0x00**

**PIC18F06/16Q41**

**I2C - Inter-Integrated Circuit Module**

Data Sheet

© 2020-2023 Microchip Technology Inc. and its subsidiaries

DS40002214F - 628

## Client Reception (7-Bit Addressing Mode)

The following section describes the sequence of events that occur when the module is receiving data in 7-bit Addressing mode:

The host issues a Start condition. Once the Start is detected, client hardware sets the Start Condition Interrupt Flag ([SCIF](#_bookmark56)) bit. If the Start Condition Interrupt Enable ([SCIE](#_bookmark64)) bit is also set, the generic I2CxIF bit is also set.

The host transmits the 7-bit client address with the R/W bit clear, indicating that it intends to write data to the client.

The received address is compared to the values in the I2CxADR registers. If the client is configured in 7-bit Addressing mode (no masking), the received address is independently compared to each of the I2CxADR0/1/2/3 registers. In 7-bit Addressing with Masking mode, the received address is compared to the masked value of [I2CxADR0](#_bookmark83) and [I2CxADR2](#_bookmark85).

If an address match occurs:

The Client Mode Active ([SMA](#_bookmark39)) bit is set by module hardware.

The R/W bit value is copied to the Read Information ([R](#_bookmark41)) bit by module hardware.

The Data ([D](#_bookmark42)) bit is cleared (D = 0) by hardware, indicating the last received byte was an address.

The Address Interrupt Flag ([ADRIF](#_bookmark53)) bit is set (ADRIF = 1). If the Address Interrupt and Hold Enable ([ADRIE](#_bookmark61)) bit is set (ADRIE = 1), and the Clock Stretching Disable ([CSD](#_bookmark29)) bit is clear (CSD = 0), hardware sets the Client Clock Stretching ([CSTR](#_bookmark18)) bit and the generic I2CxIF bit. This allows time for the client to read either [I2CxADB0](#_bookmark81) or [I2CxRXB](#_bookmark88) and selectively ACK/NACK based on the received address. When the client has finished processing the address, software must clear CSTR to resume operation.

The matching received address is loaded into either the [I2CxADB0](#_bookmark81) register or into the [I2CxRXB](#_bookmark88) register as determined by the Address Buffer Disable ([ABD](#_bookmark34)) bit. When ABD is clear (ABD = 0), the matching address is copied to I2CxADB0. When ABD is set (ABD = 1), the matching address is copied to I2CxRXB, which also sets the Receive Buffer Full Status ([RXBF](#_bookmark48)) bit and the I2C Receive Interrupt Flag (I2CxRXIF) bit. I2CxRXIF is a read-only bit, and must be cleared by either reading I2CxRXB or by setting the Clear Buffer ([CLRBF](#_bookmark47)) bit (CLRBF = 1).

If no address match occurs, the module remains Idle.

The host device transmits the 9th clock pulse, and client hardware transfers the value of the [ACKDT](#_bookmark23) bit onto the SDA line. If there are pending errors, such as a receive overflow ([RXO](#_bookmark27) = 1), client hardware automatically generates a NACK condition. [NACKIF](#_bookmark68) is set, and the module goes Idle.

Upon the 9th falling SCL edge, the Acknowledge Status Time Interrupt Flag ([ACKTIF](#_bookmark51)) bit is set. If the Acknowledge Interrupt and Hold Enable ([ACKTIE](#_bookmark59)) bit is also set, the generic I2CxIF is set, and if client hardware generated an ACK, the [CSTR](#_bookmark18) bit is also set and the clock is stretched (when [CSD](#_bookmark29)

= 0). If a NACK was generated, the CSTR bit remains unchanged. Once complete, software must clear CSTR and ACKTIF to release the clock and continue operation.

If client hardware generated a NACK, host hardware generates a Stop condition, the Stop Condition Interrupt Flag ([PCIF](#_bookmark54)) bit is set when client hardware detects the Stop condition, and the client goes Idle. If an ACK was generated, host hardware transmits the first seven bits of the 8-bit data byte.

If data remains in [I2CxRXB](#_bookmark88) ([RXBF](#_bookmark48) = 1 and I2CxRXIF = 1) when the first seven bits of the new byte are received by the shift register, [CSTR](#_bookmark18) is set, and if [CSD](#_bookmark29) is clear, the clock is stretched after the 7th falling edge of SCL. This allows time for the client to read I2CxRXB, which clears RXBF and I2CxRXIF, and prevents a receive buffer overflow. Once RXBF and I2CxRXIF are cleared, hardware releases SCL.

Host hardware transmits the 8th bit of the current data byte into the client receive shift register. Client hardware then transfers the complete byte into [I2CxRXB](#_bookmark88) on the 8th falling edge of SCL, and sets the following bits:

I2CxRXIF

I2CxIF

Data Write Interrupt Flag ([WRIF](#_bookmark52))

Data ([D](#_bookmark42))

[RXBF](#_bookmark48)

[I2CxCNT](#_bookmark74) is decremented by one. If the Data Write Interrupt and Hold Enable ([WRIE](#_bookmark60)) is set (WRIE = 1), hardware sets [CSTR](#_bookmark18) (when [CSD](#_bookmark29) = 0) and stretches the clock, allowing time for client software to read [I2CxRXB](#_bookmark88) and determine the state of the [ACKDT](#_bookmark23) bit that is transmitted back to the

host. Once the client determines the Acknowledgement response, software clears CSTR to allow further communication.

Host hardware transmits the 9th clock pulse. If there are pending errors, such as receive buffer overflow, client hardware automatically generates a NACK condition, sets [NACKIF](#_bookmark68), and the module goes Idle. If [I2CxCNT](#_bookmark74) is nonzero (I2CxCNT != 0), client hardware transmits the value

of [ACKDT](#_bookmark23) as the acknowledgement response to the host. It is up to software to configure ACKDT appropriately. In most cases, the ACKDT bit must be clear (ACKDT = 0) so that the host receives an ACK response (logic low level on SDA during the 9th clock pulse).

If [I2CxCNT](#_bookmark74) is zero (I2CxCNT = 0), client hardware transmits the value of the Acknowledge End of Count ([ACKCNT](#_bookmark22)) bit as the Acknowledgement response, rather than the value of [ACKDT](#_bookmark23). It is

up to software to configure ACKCNT appropriately. In most cases, ACKCNT must be set (ACKCNT

= 1), which represents a NACK condition. When host hardware detects a NACK on the bus, it will generate a Stop condition. If ACKCNT is clear (ACKCNT = 0), an ACK will be issued, and host hardware will not issue a Stop condition.

Upon the 9th falling edge of SCL, the [ACKTIF](#_bookmark51) bit is set. If [ACKTIE](#_bookmark59) is also set, the generic I2CxIF is set, and if [CSD](#_bookmark29) is clear, client hardware sets [CSTR](#_bookmark18) and stretches the clock. This allows time for software to read [I2CxRXB](#_bookmark88). Once complete, software must clear both CSTR and ACKTIF to release the clock and continue communication.

Repeat steps 6 -10 until the host has transmitted all the data ([I2CxCNT](#_bookmark74) = 0), or until the host issues a Stop or Restart condition.

**Start**

###### R/W

**ACKDT value**

**copied to SDA**

**ACKCNT value**

**copied to SDA**

**Stop**

**Hardware sets PCIF**

## SDA

A7 A6 A5 A4 A3 A2 A1 **0**

###### 7-bit address

D7 D6 D5 D4 D3 D2 D1 D0

###### ACK

D7 D6 D5 D4 D3 D2 D1 D0

**NACK**

## SCL

**RXBF**

1 2 3 4 5 6 7 8

**Matching received**

**address loaded**

**into I2CxADB0**

9 1 2 3 4 5 6 7 8

**I2CxRXIF set, data byte transferred to I2CxRXB**

9 1 2

3 4 5 6 7 8 9

**Software reads I2CxRXB, clearing I2CxRXIF**

**Hardware sets NACKIF**

## SMA

**Hardware sets**

**SMA**

**Hardware clears**

**SMA**

## R

**Hardware clears D bit, last byte was address**

**Hardware copies**

**R/W value to R bit**

**D**

**Hardware sets D bit, last byte was data**

**Hardware sets**

**CNTIF**

## I2CxCNT

###### 0x02 0x01 0x00

**PIC18F06/16Q41**

**I2C - Inter-Integrated Circuit Module**

Data Sheet

© 2020-2023 Microchip Technology Inc. and its subsidiaries

DS40002214F - 631

**Start**

###### R/W

**ACKDT value**

**copied to SDA**

**ACKCNT value**

**copied to SDA**

**Stop**

## SDA

A7 A6 A5 A4 A3 A2 A1 **0**

###### 7-bit address

D7 D6 D5 D4 D3 D2 D1 D0

**ACK**

D7 D6 D5 D4 D3 D2 D1 D0

**NACK**

## SCL

1 2 3 4 5 6 7 8

9 1 2

3 4 5

6 7 8 9

1 2 3

4 5 6 7 8 9

## ADRIF

**Matching received address loaded into I2CxADB0**

## CSTR

**RXBF**

**Hardware sets**

**ADRIF and CSTR**

**Software clears**

**ADRIF and CSTR**

**I2CxRXIF set,**

**data byte transferred to**

**I2CxRXB**

**Software reads**

**I2CxRXB, clearing**

**I2CxRXIF**

## I2CxCNT

###### 0x02 0x01

**0x00**

## CNTIF

**Hardware sets**

**CNTIF**

**PIC18F06/16Q41**

**I2C - Inter-Integrated Circuit Module**

Data Sheet

© 2020-2023 Microchip Technology Inc. and its subsidiaries

DS40002214F - 632

**Start**

**R/W**

**ACKDT value**

**copied to SDA**

**ACKCNT value copied to SDA**

**Stop**

**SDA**

A7 A6 A5 A4 A3 A2 A1 **0**

**7-bit address**

D7 D6 D5 D4 D3 D2 D1 D0

**ACK**

D7 D6 D5 D4 D3 D2 D1 D0

**NACK**

**SCL**

1 2 3 4 5 6 7 8 9

1 2 3 4 5

6 7 8 9

1 2 3

4 5 6 7 8 9

**RXBF**

**Matching received**

**address loaded**

**into I2CxADB0**

**I2CxRXIF set, data byte transferred to I2CxRXB**

**Software reads I2CxRXB, clearing I2CxRXIF**

**CSTR**

**Hardware sets**

**CSTR**

**Software clears**

**CSTR**

**ACKT**

**Hardware sets**

**ACKT**

**Hardware clears**

**ACKT**

**ACKTIF**

**Hardware sets**

**ACKTIF**

**Software clears**

**ACKTIF**

**Hardware sets CNTIF**

**I2CxCNT**

**0x02 0x01 0x00**

**PIC18F06/16Q41**

**I2C - Inter-Integrated Circuit Module**

Data Sheet

© 2020-2023 Microchip Technology Inc. and its subsidiaries

DS40002214F - 633

**Start**

###### R/W

**ACKDT value**

**copied to SDA ACKCNT value**

**copied to SDA**

**Stop**

## SDA

A7 A6 A5 A4 A3 A2 A1 **0**

###### 7-bit address

D7 D6 D5 D4 D3 D2 D1 D0

**ACK**

D7 D6 D5 D4 D3 D2 D1 D0

**NACK**

## SCL

1 2 3 4 5 6 7 8

9 1 2 3 4 5 6 7 8

9 1 2 3

4 5 6 7 8 9

## WRIF

**Matching received address loaded into I2CxADB0**

## CSTR

**Hardware sets**

**CSTR and WRIF**

**Software clears**

**CSTR and WRIF**

## RXBF

**I2CxRXIF set, data byte transferred to I2CxRXB**

**Software reads I2CxRXB, clearing I2CxRXIF**

## I2CxCNT

**0x02 0x01 0x00**

## CNTIF

**Hardware sets**

**CNTIF**

**PIC18F06/16Q41**

**I2C - Inter-Integrated Circuit Module**

Data Sheet

© 2020-2023 Microchip Technology Inc. and its subsidiaries

DS40002214F - 634

## Client Operation in 10-Bit Addressing Modes

In 10-bit Addressing modes, the first two bytes following a Start condition form the 10-bit address (see [Figure 36-20](#_bookmark7)). The first byte (address high byte) holds the upper two address bits, the R/W bit, and a five digit code (11110) as defined by the I2C Specification. The second byte (address low byte) holds the lower eight address bits. In all 10-bit Addressing modes, the R/W value contained in the first byte must always be zero (R/W = 0). If the host intends to read data from the client, it must issue a Restart condition, followed by the address high byte with R/W set (R/W = 1).

The first byte is compared to the values in the [I2CxADR1](#_bookmark84) and [I2CxADR3](#_bookmark86) registers in 10-bit Addressing mode, or to the masked value of I2CxADR1 in 10-bit Addressing with Masking mode. The second byte is compared to the values in the [I2CxADR0](#_bookmark83) and [I2CxADR2](#_bookmark85) registers in 10-bit Addressing mode, or to the masked value of I2CxADR0 in 10-bit Addressing with Masking mode. If an address high byte match occurs, the high address byte is copied to [I2CxADB1](#_bookmark82) and the R/W bit value is copied to the Read Information ([R](#_bookmark41)) bit, and if an address low byte match occurs, the low address byte is copied to [I2CxADB0](#_bookmark81).

**Figure 36-20.** Upper and Lower 10-Bit Address Bytes

## Address High Byte Address Low Byte

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **1** | **1** | **1** | **0** | **A9** | **A8** |  |  |
| **R/W** | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **A7** | **A6** | **A5** | **A4** | **A3** | **A2** | **A1** | **A0** |
|  |  |  |  |  |  |  |  |

## 5-digit address code

**Upper two**

## address bits

**Lower eight**

## address bits

* + - * 1. **Client Transmission (10-Bit Addressing Mode)**

The following section describes the sequence of events that occur when the module is transmitting data in 10-bit Addressing mode:

The host device issues a Start condition. Once the Start condition has been detected, client hardware sets the Start Condition Interrupt Flag ([SCIF](#_bookmark56)) bit. If the Start Condition Interrupt Enable ([SCIE](#_bookmark64)) bit is also set, the generic I2CxIF is also set.

Host hardware transmits the 10-bit high address byte with the R/W bit clear (R/W = 0).

Client hardware compares the received address to the values in the I2CxADR registers. If the client is configured in 10-bit Addressing mode (no masking), the received high address byte is compared to the values in [I2CxADR1](#_bookmark84) and [I2CxADR3](#_bookmark86). In 10-bit Addressing with Masking mode, the received high address byte is compared to the masked value of I2CxADR1.

If an address match occurs:

The R/W value is copied to the Read Information ([R](#_bookmark41)) bit by module hardware.

The Data ([D](#_bookmark42)) bit is cleared by hardware.

The Address Interrupt Flag ([ADRIF](#_bookmark53)) bit is set (ADRIF = 1).

The matching address is loaded into either the [I2CxADB1](#_bookmark82) register or into the [I2CxRXB](#_bookmark88) register as determined by the Address Buffer Disable ([ABD](#_bookmark34)) bit. When ABD is clear (ABD

= 0), the matching address is copied to I2CxADB1. When ABD is set (ABD = 1), the matching address is copied to I2CxRXB, which also sets the Receive Buffer Full Status ([RXBF](#_bookmark48)) bit and the I2C Receive Interrupt Flag (I2CxRXIF) bit.

**Important:** Regardless of whether the Address Interrupt and Hold Enable ([ADRIE](#_bookmark61)) bit is set, clock stretching does not occur when the R/W bit is clear in 10-bit Addressing modes.

If no address match occurs, the module remains Idle.

The host device transmits the 9th clock pulse, and client hardware transfers the value of the [ACKDT](#_bookmark23) bit onto the SDA line. If there are pending errors, such as a receive buffer overflow ([RXO](#_bookmark27) = 1), client hardware generates a NACK and the module goes Idle.

The host device transmits the low address byte. If the client is configured in 10-bit Addressing mode (no masking), the received low address byte is compared to the values in [I2CxADR0](#_bookmark83) and [I2CxADR2](#_bookmark85). In 10-bit Addressing with Masking mode, the received low address byte is compared to the masked value of I2CxADR0.

If a match occurs:

The Client Mode Active ([SMA](#_bookmark39)) bit is set by module hardware.

[ADRIF](#_bookmark53) is set. If [ADRIE](#_bookmark61) is set, and the Clock Stretching Disable ([CSD](#_bookmark29)) bit is clear, hardware sets the Client Clock Stretching ([CSTR](#_bookmark18)) bit and the generic I2CxIF bit. This allows time for the client to read either [I2CxADB0](#_bookmark81) or [I2CxRXB](#_bookmark88) and selectively ACK/NACK based on the received address. When the client has finished processing the address, software must clear CSTR to resume operation.

The matching received address is loaded into either the [I2CxADB0](#_bookmark81) register or into the [I2CxRXB](#_bookmark88) register as determined by the [ABD](#_bookmark34) bit. When ABD is clear (ABD = 0), the matching address is copied to I2CxADB0. When ABD is set (ABD = 1), the matching address is copied to I2CxRXB, which also sets [RXBF](#_bookmark48) and I2CxRXIF. I2CxRXIF is a read-only bit, and must be cleared by either reading I2CxRXB or by setting the Clear Buffer ([CLRBF](#_bookmark47)) bit (CLRBF = 1).

If no match occurs, the module goes Idle.

The host device transmits the 9th clock pulse, and client hardware transfers the value of the [ACKDT](#_bookmark23) bit onto the SDA line. If there are pending errors, such as a receive buffer overflow ([RXO](#_bookmark27) = 1), client hardware generates a NACK and the module goes Idle.

After the 9th falling edge of SCL, the Acknowledge Status Time Interrupt Flag ([ACKTIF](#_bookmark51)) bit is set. If the Acknowledge Time Interrupt and Hold Enable ([ACKTIE](#_bookmark59)) bit is also set, the generic I2CxIF is set, and if client hardware generated an ACK, the [CSTR](#_bookmark18) bit is also set and the clock is stretched (when [CSD](#_bookmark29) = 0). If a NACK was generated, the CSTR bit remains unchanged. Once completed, software must clear CSTR and ACKTIF to release the clock and resume operation.

Host hardware issues a Restart condition (cannot be a Start condition), and once the client detects the Restart, hardware sets the Restart Condition Interrupt Flag ([RSCIF](#_bookmark55)). If the Restart Condition Interrupt Enable ([RSCIE](#_bookmark63)) bit is also set, the generic I2CxIF is also set.

Host hardware transmits the client’s high address byte with R/W set. If the received high address byte matches:

The R/W bit value is copied to the [R](#_bookmark41) bit.

The SMA bit is set.

The [D](#_bookmark42) bit is cleared, indicating the last byte as an address.

[ADRIF](#_bookmark53) is set. If [ADRIE](#_bookmark61) is set, and the [CSD](#_bookmark29) bit is clear, hardware sets [CSTR](#_bookmark18) and the generic I2CxIF bit. This allows time for the client to read either [I2CxADB1](#_bookmark82) or [I2CxRXB](#_bookmark88) and selectively ACK/NACK based on the received address. When the client has finished processing the address, software must clear CSTR to resume operation.

The matching received address is loaded into either the [I2CxADB1](#_bookmark82) register or into the [I2CxRXB](#_bookmark88) register as determined by the [ABD](#_bookmark34) bit. When ABD is clear (ABD = 0), the matching address is copied to I2CxADB1. When ABD is set (ABD = 1), the matching address is copied to

I2CxRXB, which also sets [RXBF](#_bookmark48) and I2CxRXIF. I2CxRXIF is a read-only bit, and must be cleared by either reading I2CxRXB or by setting [CLRBF](#_bookmark47) (CLRBF = 1).

If the address does not match, the module goes Idle.

If the Transmit Buffer Empty Status ([TXBE](#_bookmark45)) bit is set (TXBE = 1), [I2CxCNT](#_bookmark74) has a nonzero value (I2CxCNT != 0), and the I2C Transmit Interrupt Flag (I2CxTXIF) is set (I2CxTXIF = 1), client hardware sets [CSTR](#_bookmark18), stretches the clock (when [CSD](#_bookmark29) = 0), and waits for software to load [I2CxTXB](#_bookmark87) with

data. I2CxTXB must be loaded to clear I2CxTXIF. Once data is loaded into I2CxTXB, hardware automatically clears CSTR to resume communication.

The host device transmits the 9th clock pulse, and client hardware transfers the value of the [ACKDT](#_bookmark23) bit onto the SDA line. If there are pending errors, such as a receive overflow ([RXO](#_bookmark27) = 1), client hardware automatically generates a NACK condition. [NACKIF](#_bookmark68) is set, and the module goes Idle.

Upon the 9th falling SCL edge, the data byte in [I2CxTXB](#_bookmark87) is transferred to the transmit shift register, and [I2CxCNT](#_bookmark74) is decremented by one. Additionally, the [ACKTIF](#_bookmark51) bit is set. If the [ACKTIE](#_bookmark59) bit is also set, the generic I2CxIF is set, and if client hardware generated an ACK, the [CSTR](#_bookmark18) bit is also set and the clock is stretched (when [CSD](#_bookmark29) = 0). If a NACK was generated, the CSTR bit

remains unchanged. Once complete, software must clear CSTR and ACKTIF to release the clock and continue operation.

If the client generated an ACK and [I2CxCNT](#_bookmark74) is nonzero, host hardware transmits eight clock pulses, and client hardware begins to shift the data byte out of the shift register starting with the Most Significant bit (MSb).

After the 8th falling edge of SCL, client hardware checks the status of [TXBE](#_bookmark45) and [I2CxCNT](#_bookmark74). If TXBE is set and I2CxCNT has a nonzero count value, hardware sets [CSTR](#_bookmark18) and the clock is stretched (when [CSD](#_bookmark29) = 0) until software loads [I2CxTXB](#_bookmark87) with new data. Once I2CxTXB has been loaded, hardware clears CSTR to resume communication.

Once the host hardware clocks in all eight data bits, it transmits the 9th clock pulse along with the ACK/NACK response back to the client. Client hardware copies the ACK/NACK value to the Acknowledge Status ([ACKSTAT](#_bookmark24)) bit and sets [ACKTIF](#_bookmark51). If [ACKTIE](#_bookmark59) is also set, client hardware sets the generic I2CxIF bit and [CSTR](#_bookmark18), and stretches the clock (when [CSD](#_bookmark29) = 0). Software must clear CSTR to resume operation.

After the 9th falling edge of SCL, data currently loaded in [I2CxTXB](#_bookmark87) is transferred to the transmit shift register, setting both [TXBE](#_bookmark45) and I2CxTXIF. [I2CxCNT](#_bookmark74) is decremented by one. If I2CxCNT is zero (I2CxCNT = 0), [CNTIF](#_bookmark50) is set.

If [I2CxCNT](#_bookmark74) is nonzero and the host issued an ACK on the last byte ([ACKSTAT](#_bookmark24) = 0), the host transmits eight clock pulses, and client hardware begins to shift data out of the shift register.

Repeat Steps 13-17 until the host has received all the requested data ([I2CxCNT](#_bookmark74) = 0). Once all data is received, host hardware transmits a NACK condition, followed by either a Stop or

Restart condition. Once the NACK has been received by the client, hardware sets [NACKIF](#_bookmark68) and clears [SMA](#_bookmark39). If the NACK Detect Interrupt Enable ([NACKIE](#_bookmark71)) bit is also set, the generic I2C Error Interrupt Flag (I2CxEIF) is set. If the host issued a Stop condition, client hardware sets the Stop Condition Interrupt Flag ([PCIF](#_bookmark54)). If the host issued a Restart condition, client hardware sets the Restart Condition Interrupt Flag ([RSCIF](#_bookmark55)) bit. If the associated interrupt enable bits are also set, the generic I2CxIF is also set.

**Figure 36-21.** 10-Bit Client Mode Transmission

###### R/W

**ACK (from client)**

**Restart R/W**

**NACK**

**(from host )**



**Stop**

## SDA

**1 1 1 1 0** A9 A8 **0** A7 A6 A5 A4 A3 A2 A1 A0

**1 1 1 1 0** A9 A8 **1**

D7 D6 D5 D4 D3 D2 D1 D0

###### High address

**Low address**

###### High address

**ACK (from client)**

## SCL

1 2 3

4 5 6 7 8

9 1 2 3 4

5 6 7 8 9

1 2 3 4 5

6 7 8

9 1 2 3 4 5

6 7 8 9

## SMA

**R**

**Hardware sets SMA**

**Hardware copies R/W**

**value to R bit**

**Hardware clears SMA**

**Hardware sets NACKIF, CNTIF**

**Hardware copies R/W value to R bit**

## D I2CxCNT

#### ACKSTAT

**Hardware clears D bit for address bytes**

###### 0x01

**Hardware sets D bit, last byte was data**

###### 0x00

**Host's NACK copied to ACKSTAT**

## TXBE

**Before Start, software loads data into I2CxTXB**

**Data byte transferred to shift register,**

**I2CxTXIF NOT set**

**PIC18F06/16Q41**

**I2C - Inter-Integrated Circuit Module**

Data Sheet

© 2020-2023 Microchip Technology Inc. and its subsidiaries

DS40002214F - 638

## Client Reception (10-Bit Addressing Mode)

The following section describes the sequence of events that occur when the module is receiving data in 7-bit Addressing mode:

The host issues a Start condition. Once the Start is detected, client hardware sets the Start Condition Interrupt Flag ([SCIF](#_bookmark56)) bit. If the Start Condition Interrupt Enable ([SCIE](#_bookmark64)) bit is also set, the generic I2CxIF bit is also set.

Host hardware transmits the address high byte with the R/W bit clear (R/W = 0).

The received high address byte is compared to the values in the I2CxADR registers. If the client is configured in 10-bit Addressing mode (no masking), the received high address byte is compared to the values in the [I2CxADR1](#_bookmark84) and [I2CxADR3](#_bookmark86) registers. If the client is configured in 10-bit Addressing with Masking mode, the received high address byte is compared to the masked value in the I2CxADR1 register.

If a high address match occurs:

The R/W bit value is copied to the Read Information ([R](#_bookmark41)) bit by module hardware.

The Data ([D](#_bookmark42)) bit is cleared ([D](#_bookmark42) = 0) by hardware, indicating the last received byte was an address.

The Address Interrupt Flag ([ADRIF](#_bookmark53)) bit is set (ADRIF = 1). It is important to note that regardless of whether the Address Interrupt and Hold Enable ([ADRIE](#_bookmark61)) bit is set, clock stretching does not occur when the R/W bit is clear in 10-bit Addressing modes.

The matching address is loaded into either the [I2CxADB1](#_bookmark82) register or into the [I2CxRXB](#_bookmark88) register as determined by the Address Buffer Disable ([ABD](#_bookmark34)) bit. When ABD is clear (ABD

= 0), the matching address is copied to I2CxADB1. When ABD is set (ABD = 1), the matching address is copied to I2CxRXB, which also sets the Receive Buffer Full Status ([RXBF](#_bookmark48)) bit and the I2C Receive Interrupt Flag (I2CxRXIF) bit.

If no address match occurs, the module remains Idle.

The host device transmits the 9th clock pulse, and client hardware transfers the value of the [ACKDT](#_bookmark23) bit onto the SDA line. If there are pending errors, such as a receive buffer overflow ([RXO](#_bookmark27) = 1), client hardware generates a NACK and the module goes Idle.

The host device transmits the low address byte. If the client is configured in 10-bit Addressing mode (no masking), the received low address byte is compared to the values in [I2CxADR0](#_bookmark83) and [I2CxADR2](#_bookmark85). In 10-bit Addressing with Masking mode, the received low address byte is compared to the masked value of I2CxADR0.

If a match occurs:

The Client Mode Active ([SMA](#_bookmark39)) bit is set by module hardware.

[ADRIF](#_bookmark53) is set. If [ADRIE](#_bookmark61) is set, and the Clock Stretching Disable ([CSD](#_bookmark29)) bit is clear, hardware sets the Client Clock Stretching ([CSTR](#_bookmark18)) bit and the generic I2CxIF bit. This allows time for the client to read either [I2CxADB0](#_bookmark81) or [I2CxRXB](#_bookmark88) and selectively ACK/NACK based on the received address. When the client has finished processing the address, software must clear CSTR to resume operation.

The matching received address is loaded into either the [I2CxADB0](#_bookmark81) register or into the [I2CxRXB](#_bookmark88) register as determined by the [ABD](#_bookmark34) bit. When ABD is clear (ABD = 0), the matching address is copied to I2CxADB0. When ABD is set (ABD = 1), the matching address is copied to I2CxRXB, which also sets the [RXBF](#_bookmark48) and the I2CxRXIF bits. I2CxRXIF is a read-only bit, and

must be cleared by either reading I2CxRXB or by setting the Clear Buffer ([CLRBF](#_bookmark47)) bit (CLRBF =

1).

If no match occurs, the module goes Idle.

The host device transmits the 9th clock pulse, and client hardware transfers the value of the [ACKDT](#_bookmark23) bit onto the SDA line. If there are pending errors, such as a receive buffer overflow ([RXO](#_bookmark27) = 1), client hardware generates a NACK and the module goes Idle.

After the 9th falling edge of SCL, the Acknowledge Status Time Interrupt Flag ([ACKTIF](#_bookmark51)) bit is set. If the Acknowledge Time Interrupt and Hold Enable ([ACKTIE](#_bookmark59)) bit is also set, the generic I2CxIF is set, and if client hardware generated an ACK, the [CSTR](#_bookmark18) bit is also set and the clock is stretched (when [CSD](#_bookmark29) = 0). If a NACK was generated, the [CSTR](#_bookmark18) bit remains unchanged. Once completed, software must clear CSTR and ACKTIF to release the clock and resume operation.

If client hardware generated a NACK, host hardware generates a Stop condition, the Stop Condition Interrupt Flag ([PCIF](#_bookmark54)) is set when client hardware detects the Stop condition, and the client goes Idle. If an ACK was generated, host hardware transmits the first seven bits of the 8-bit data byte.

If data remains in [I2CxRXB](#_bookmark88) ([RXBF](#_bookmark48) = 1 and I2CxRXIF = 1) when the first seven bits of the new

byte are received by the shift register, [CSTR](#_bookmark18) is set, and if [CSD](#_bookmark29) is clear, the clock is stretched after the 7th falling edge of SCL. This allows time for the client to read I2CxRXB, which clears [RXBF](#_bookmark48) and I2CxRXIF, and prevents a receive buffer overflow. Once I2CxRXB has been read, RXBF and I2CxRXIF are cleared, and hardware releases SCL.

Host hardware transmits the 8th bit of the current data byte into the client receive shift register. Client hardware then transfers the complete byte into [I2CxRXB](#_bookmark88) on the 8th falling edge of SCL, and sets the following bits:

I2CxRXIF

I2CxIF

Data Write Interrupt Flag ([WRIF](#_bookmark52))

Data ([D](#_bookmark42))

[RXBF](#_bookmark48)

[I2CxCNT](#_bookmark74) is decremented by one. If the Data Write Interrupt and Hold Enable ([WRIE](#_bookmark60)) bit is set (WRIE = 1), hardware sets [CSTR](#_bookmark18) (when [CSD](#_bookmark29) = 0) and stretches the clock, allowing time for client software to read [I2CxRXB](#_bookmark88) and determine the state of the [ACKDT](#_bookmark23) bit that is transmitted back to the host. Once the client determines the Acknowledgement response, software clears CSTR to allow further communication.

Upon the 9th falling edge of SCL, the [ACKTIF](#_bookmark51) bit is set. If [ACKTIE](#_bookmark59) is also set, the generic I2CxIF is set, and if [CSD](#_bookmark29) is clear, client hardware sets [CSTR](#_bookmark18) and stretches the clock. This allows time for software to read [I2CxRXB](#_bookmark88). Once complete, software must clear both CSTR and ACKTIF to release the clock and continue communication.

Repeat Steps 8 – 11 until the host has transmitted all the data ([I2CxCNT](#_bookmark74) = 0), or until the host issues a Stop or Restart condition.

**Figure 36-22.** 10-Bit Client Mode Reception

**Start**

**R/W**

**ACKDT value**

**copied to SDA**

**ACKCNT value**

**copied to SDA**

**Stop**

**Hardware sets PCIF**

**SDA**

**1 1 1 1 0** A9 A8 **0**

**High address**

A7 A6 A5 A4 A3 A2 A1 A0

**ACK**

D7 D6 D5 D4 D3 D2 D1 D0

**NACK**

**SCL**

1 2 3 4

5 6 7 8

9 1 2 3 4 5 6 7 8

9 1 2

3 4 5

6 7 8 9

**Hardware sets**

**RXBF**

**Hardware sets**

**SMA**

**Matching received High address loaded into I2CxADB1**

**Matching received Low address loaded into I2CxADB0**

**I2CxRXIF set,**

**data byte transferred to**

**I2CxRXB**

**NACKIF**

**SMA**

**Hardware clears**

**SMA**

**Hardware copies**

**R R/W value to R bit**

**D**

**Hardware sets D bit, last byte was data**

**Hardware clears D bit, last byte was address**

**Hardware sets**

**CNTIF**

**I2CxCNT**

**0x01 0x00**

**PIC18F06/16Q41**

**I2C - Inter-Integrated Circuit Module**

Data Sheet

© 2020-2023 Microchip Technology Inc. and its subsidiaries

DS40002214F - 641

## I2C Host Mode Operation

The I2C module provides two Host Operation modes as selected by the I2C Mode Select ([MODE](#_bookmark20)) bits:

* + - * I2C Host mode with 7-bit addressing
      * I2C Host mode with 10-bit addressing

To begin any I2C communication, host hardware checks to ensure that the bus is in an Idle state, which means both the SCL and SDA lines are floating in a high Logic state as indicated by the Bus Free Status ([BFRE](#_bookmark38)) bit.

Once Host hardware has determined that the bus is free ([BFRE](#_bookmark38) = 1), it examines the state of the Address Buffer Disable ([ABD](#_bookmark34)) bit. The ABD bit determines whether the I2CxADB registers are used.

When [ABD](#_bookmark34) is clear (ABD = 0), address buffers [I2CxADB0](#_bookmark81) and [I2CxADB1](#_bookmark82) are active. In 7-bit Addressing mode, software loads I2CxADB1 with the 7-bit client address and R/W bit setting, and also loads [I2CxTXB](#_bookmark87) with the first byte of data . In 10-bit Addressing mode, software loads I2CxADB1 with the address high byte and I2CxADB0 with the address low byte, and also loads I2CxTXB with the first data byte. Software must issue a Start condition to initiate communication with the client.

When [ABD](#_bookmark34) is set (ABD = 1), the address buffers are inactive. In this case, communication begins as soon as software loads the client address into [I2CxTXB](#_bookmark87). Writes to the Start ([S](#_bookmark17)) bit are ignored.

In 7-bit Addressing mode, the Least Significant bit (LSb) of the 7-bit address byte acts as the Read/not Write (R/W) information bit, while in 10-bit Addressing mode, the LSb of the address high byte is reserved as the R/W bit. When R/W is set, the host intends to read data from the client (see the figure below). When R/W is clear, the host intends to write data to the client (see the figure below). The host may also wish to read or write data to a specific location, such as writing to a specific EEPROM location. In this case, the host issues a Start condition, followed by

the client’s address with the R/W bit clear. Once the client acknowledges the address, the first data byte following the 7-bit or 10-bit address is used as the client’s specific register location. If the host intends to read data from the specific location, it must issue a Restart condition, followed by the client address with the R/W bit set (see the figure below). If the addressed client device exists on the bus, it must respond with an Acknowledge (ACK) sequence.

Once a client has acknowledged its address, the host begins to receive data from the client or transmits data to the client. Data is always transmitted Most Significant bit (MSb) first. When the host wishes to halt further communication, it transmits either a Stop condition, signaling to the client that communication is to be terminated, or a Restart condition, informing the bus that the current host wishes to hold the bus to communicate with the same or other client devices.

**Figure 36-23.** 7-Bit Host Read Diagram

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S** | **A6** | **A5** | **A4** | **A3** | **A2** | **A1** | **A0** | **1** | **A** | **D7** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** | **N** | **S**  **T**  **O P** |

**7-bit address**  **Data**

## R/W ACK NACK

**(from client) (from host)**

**Figure 36-24.** 7-Bit Host Read Diagram (from a specific memory/register location)

**(from client)**

**A R A6**

**S**

**A**

**7-bit address**

**R/W**

**Register**

**Address**

**7-bit address**

**R/W ACK**

**(from client)**

**Data**

**NACK**

**(from host )**

**ACK**

**ACK**

**Restart**

**S T O P**

**D0 N**

**D1**

**D2**

**D3**

**D4**

**D5**

**D6**

**D7**

**1**

**A0**

**A1**

**A2**

**A3**

**A4**

**A5**

**RA0**

**RA1**

**RA2**

**RA3**

**RA4**

**RA5**

**RA6**

**A RA7**

**0**

**A0**

**A1**

**A2**

**A3**

**A4**

**A5**

**S A6**

**(from client)**

**condition**

**Figure 36-25.** 10-Bit Host Read Diagram

**Address**

**high byte**

**0**

**R/W**

**Restart**

**condition Address**

**high byte**

**R**

**A8**

**A9**

**0**

**1**

**1**

**1**

**A1**

**A2**

**A3**

**A4**

**A5**

**A6**

**A0**

**A A7**

**R/W**

**S**

**D1**

**D2**

**D3**

**D4**

**D5**

**D6**

**D0**

**A D7**

**1**

**S**

**A8**

**A9**

**0**

**1**

**1**

**1**

**1**

**5-digit**

**address**

**10-bit**

**address**

**Address low byte**

**A S 1**

**5-digit**

**address**

**10-bit**

**address**

**T O P**

**Data**

**N**

**code**

**MSb s**

**ACK ACK**

**code**

**MSb s**

**NACK**

**)**

**(from client)**

**(from client)**

**ACK**

**(from client)**

**(from host**

**Figure 36-26.** 7-Bit Host Write Diagram

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S** | **A6** | **A5** | **A4** | **A3** | **A2** | **A1** | **A0** | **0** | **A** | **D7** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** | **N** | **S T O**  **P** |

## R/W ACK NACK

**7-bit address**

**Data**

**(from client)**

**(from client)**

**Figure 36-27.** 7-Bit Host Write Diagram (to a specific memory/register location)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S** | **A6** | **A5** | **A4** | **A3** | **A2** | **A1** | **A0** | **0** | **A** | **RA7** | **RA6** | **RA5** | **RA4** | **RA3** | **RA2** | **RA1** | **RA0** | **A** | **D7** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** | **N** | **S**  **T O P** |

**R/W**

**7-bit address**

**ACK**

**Register**

**Address**

**ACK**

**Data**

**NACK**

**(from client)**

**(from client) (from client)**

**Figure 36-28.** 10-Bit Host Write Diagram

**Address**

**high byte**

**R/W**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S** | **1** | **1** | **1** | **1** | **0** | **A9** | **A8** | **0** | **A** | **A7** | **A6** | **A5** | **A4** | **A3** | **A2** | **A1** | **A0** | **A** | **D7** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** | **N** | **S**  **T O P** |

### 5-digit address code

**10-bit**

### address

**MSb s**

### Address low

**byte**

### ACK

**Data**

### NACK

**ACK**

**(from client)**

**(from client) (from client)**

## Bus Free Time

The Bus Free Status ([BFRE](#_bookmark38)) bit indicates the activity status of the bus. When BFRE is set (BFRE = 1), the bus is in an Idle state (both SDA and SCL are floating high), and any host device residing on the bus can compete for control of the bus. When BFRE is clear (BFRE = 0), the bus is in an Active state, and any attempts by a host to control the bus will cause a collision.

The Bus Free Time ([BFRET](#_bookmark36)) bits determine the length of time, in terms of I2C clock pulses, before the bus is considered Idle. Once module hardware detects logic high levels on both SDA and SCL, it monitors the I2C clock signal, and when the desired number of pulses have occurred, module hardware sets [BFRE](#_bookmark38). The [BFRET](#_bookmark36) bits are also used to ensure that the module meets the minimum Stop hold time as defined by the I2C Specification.

## Host Clock Timing

The Serial Clock (SCL) signal is generated by module hardware via the I2C Clock Selection ([I2CxCLK](#_bookmark72)) Register, the I2C Baud Rate Prescaler ([I2CxBAUD](#_bookmark73)) Register, and the Fast Mode Enable ([FME](#_bookmark33)) bit.

The figure below illustrates the SCL clock generation.

**Figure 36-29.** SCL Clock Generation

###### CLK[4:0]

**I2CxBAUD FME**



**FSCL**

**1**

**0**

FPRECLK = I2CxCLK (BAUD + 1)

**11111**

**10111**

**10110**

**.**

**.**

**.**

**00010**

**00001**

**00000**

[I2CxCLK](#_bookmark72) contains several clock source selections. The clock source selections typically include variants of the system clock and timer resources.

**Important:** When using a timer as the clock source, the timer must also be configured. Additionally, when using the HFINTOSC as a clock source, it is important to understand that the HFINTOSC frequency selected by the OSCFRQ register is used as the clock source. The clock divider selected by the NDIV bits is not used. For example, if OSCFRQ selects 4 MHz as the HFINTOSC clock frequency, and the NDIV bits select a divide by four scaling factor, the I2C Clock Frequency will be 4 MHz and not 1 MHz since the divider is ignored.

[I2CxBAUD](#_bookmark73) is used to determine the prescaler (clock divider) for the I2CxCLK source. The [FME](#_bookmark33) bit acts as a secondary divider to the prescaled clock source.

When [FME](#_bookmark33) is clear (FME = 0), one SCL period (TSCL) is equal to five clock periods of the prescaled [I2CxCLK](#_bookmark72) source. In other words, the prescaled I2CxCLK source is divided by five. For example, if the HFINTOSC (set to 4 MHz) clock source is selected, I2CxBAUD is loaded with a value of ‘7’, and the FME bit is clear, the actual SCL frequency is 100 kHz (see the equation below).

**Equation 36-1.** SCL Frequency (FME = 0)

Example:

* I2CxCLK: HFINTOSC (4 MHz)
* I2CxBAUD: 7
* FME: FME = 0

fSCL =

FME

4 MHz

= 8

fI2CxCLK BAUD + 1

5

= 100 kHz

When [FME](#_bookmark33) is clear, host hardware uses the first prescaled [I2CxCLK](#_bookmark72) source period to drive SCL low (see [Figure 36-30](#_bookmark8)). During the second period, hardware verifies that SCL is in fact low. During the third period, hardware releases SCL, allowing it to float high. Host hardware then uses the fourth and fifth periods to sample SCL to verify that SCL is high. If a client is holding SCL low (clock stretch) during the fourth and/or fifth period, host hardware samples each successive prescaled I2CxCLK

period until a high level is detected on SCL. Once the high level is detected, host hardware samples SCL during the next two I2CxCLK periods to verify that SCL is high.

**Figure 36-30.** SCL Timing (FME = 0)

**I2C**

**Host drives**

**SCL low**

**Host releases SCL**

**Host drives**

**SCL low**

**Host releases SCL, but client stretches clock**

**Client**

**releases SCL**

**Host drives**

**SCL low**

**Prescaled**

**Clock**

**SCL**

**1 2 3 4 5**

**TSCL**

**1 2**

**3 4 5**

**TSCL**

**1 2**

**Host samples SCL to ensure SCL is low**

**Host samples SCL to ensure SCL is high**

**Host samples SCL to ensure SCL is low**

**Host samples**

**SCL for high**

**Host must detect SCL high twice**

When [FME](#_bookmark33) is set (FME = 1), one SCL period (TSCL) is equal to four clock periods of the prescaled [I2CxCLK](#_bookmark72) source. In other words, the prescaled I2CxCLK source is divided by four. Using the example from above, if the HFINTOSC (4 MHz) clock source is selected, I2CxBAUD is loaded with a value of ‘7’, and the FME bit is set, the actual SCL frequency is 125 kHz (see the equation below).

**Equation 36-2.** SCL Frequency (FME = 1)

Example:

* [I2CxCLK](#_bookmark72): HFINTOSC (4 MHz)
* [I2CxBAUD](#_bookmark73): 7
* [FME](#_bookmark33): FME = 1

fSCL =

FME

4 MHz

= 8

fI2CxCLK BAUD + 1

##### 4

= 125 kHz

When [FME](#_bookmark33) is set, host hardware uses the first prescaled [I2CxCLK](#_bookmark72) source period to drive SCL low (see [Figure 36-31](#_bookmark9)). During the second prescaled period, hardware verifies that SCL is in fact low. During the third period, hardware releases SCL, allowing it to float high. Host hardware then uses the fourth period to sample SCL to verify that SCL is high. If a client is holding SCL low (clock stretch) during the fourth period, host hardware samples each successive prescaled I2CxCLK period until a high level is detected on SCL. Once the high level is detected, host hardware samples SCL during the next period to verify that SCL is high.

**Figure 36-31.** SCL Timing (FME = 1)

**I2C**

**Host drives SCL low**

**Host releases SCL**

**Host drives SCL low**

**Host releases SCL, but client stretches clock**

**Client**

**releases SCL**

**Host**

**drives SCL low**

**1 2 3 4 1 2 3 4 1 2 3 4**

**TSCL TSCL**

**TSCL**

**Prescaled Clock**

**SCL**

**Host samples SCL to ensure SCL is low**

**Host samples SCL to ensure SCL is high**

**Host samples SCL to ensure SCL is low**

**Host samples SCL**

**for high**

**Host must detect SCL high**

## Start Condition Timing

A Start condition is initiated by either writing to the Start ([S](#_bookmark17)) bit (when [ABD](#_bookmark34) = 0), or by writing to [I2CxTXB](#_bookmark87) (when ABD = 1). When the Start condition is initiated, host hardware verifies that the bus is Idle, then begins to count the number of [I2CxCLK](#_bookmark72) periods as determined by the Bus Free Time Status ([BFRET](#_bookmark36)) bits. Once the Bus Free Time period has been reached, hardware sets [BFRE](#_bookmark38) (BFRE = 1), the Start condition is asserted on the bus, which pulls the SDA line low, and the Start Condition Interrupt Flag ([SCIF](#_bookmark56)) bit is set (SCIF = 1). Host hardware then waits one full SCL period (TSCL) before pulling the SCL line low, signaling the end of the Start condition. At this point, hardware loads the transmit shift register from either [I2CxADB0](#_bookmark81)/[I2CxADB1](#_bookmark82) (ABD = 0) or I2CxTXB (ABD = 1).

The figure below shows an example of a Start condition.

**Figure 36-32.** Start Condition Timing

**Write to START (S) bit**

**BFRE = 1**

**SCIF = 1**

**Start condition asserted**

**t (2)**

**HD:DAT**

**SDA**

**SCL**

**I2CxCLK**

**(1)**

**HD:STA**

**t**

**Change of data allowed**

**(FME = 1)**

**3 4 1**

**2 3 4**

**1 2 3 4 1 2**

**3 4 1**

**2 3 4 1**

**2 3 4**

**1 2 3 4**

**1 2 3**

**BFRET = 00**

**(8 - I2C Clock Pulses)**

**Completion of Start**

**If ABD = 0: Hardware loads I2C Shift register from I2CxADB0/1**

**If ABD = 1: Hardware loads I2C Shift register with I2CxTXB**

###### Important:

* + - * 1. See the device data sheet for Start condition hold time parameters.
        2. SDA hold times are configured via the [SDAHT](#_bookmark35) bits.

## Acknowledge Sequence Timing

The 9th SCL pulse for any transferred address/data byte is reserved for the Acknowledge (ACK) sequence. During an Acknowledge sequence, the transmitting device relinquishes control of the SDA line to the receiving device. At this time, the receiving device must decide whether to pull the SDA line low (ACK) or allow the line to float high (NACK).

An Acknowledge sequence is enabled automatically by module hardware following an address/data byte reception. On the 8th falling edge of SCL, the value of either the [ACKDT](#_bookmark23) or [ACKCNT](#_bookmark22) bits are copied to the SDA output, depending on the state of [I2CxCNT](#_bookmark74). When I2CxCNT holds a nonzero value (I2CxCNT != 0), the value of ACKDT is copied to SDA (see [Figure 36-33](#_bookmark10)). When I2CxCNT reaches a zero count (I2CxCNT = 0), the value of ACKCNT is copied to SDA (see [Figure 36-34](#_bookmark11)). In most applications, the value of ACKDT needs to be zero (ACKDT = 0), which represents an ACK, while the value of ACKCNT needs to be one (ACKCNT = 1), which represents a NACK.

**Figure 36-33.** Acknowledge (ACK) Sequence Timing

**ACKDT copied to SDA**

## SDA

###### D1 D0

**ACK**

**ACK Complete**

## SCL

**8**  **9**

###### I2CxCNT = 1

**7**

**Software/DMA**

###### reads I2CxRXB,

**RXBF I2CxRXIF = 1**

###### clearing I2CxRXIF

**and RXBF**

## I2CxCLK

**4 1 2 3**

**4 1 2 3 4 1**

###### Begin ACK sequence

**Figure 36-34.** Not Acknowledge (NACK) Sequence Timing

**ACKCNT copied to SDA**

## SDA

**D1**

**D0**

**NACK**

**SCL**

**8**  **9**

**7**

###### I2CxCNT = 0

**NACK Complete**

###### CNTIF = 1

**RXBF I2CxRXIF = 1**

**Software/DMA reads I2CxRXB, clearing I2CxRXIF and RXBF**

## I2CxCLK

**4 1 2**

**3 4 1 2 3 4 1**

###### Begin NACK

**sequence**

## Restart Condition Timing

A Restart condition is identical to a Start condition. A host device may issue a Restart instead of a Stop condition if it intends to hold the bus after completing the current data transfer. A Restart condition occurs when the Restart Enable ([RSEN](#_bookmark16)) bit is set (RSEN = 1), either [I2CxCNT](#_bookmark74) is zero

(I2CxCNT = 0) or [ACKSTAT](#_bookmark24) is set (ACKSTAT = 1), and either host hardware ([ABD](#_bookmark34) = 1) or user software (ABD = 0) sets the Start ([S](#_bookmark17)) bit.

When the Start bit is set, host hardware releases SDA (SDA floats high) for half of an SCL clock period (TSCL/2), and then releases SCL for another half of an SCL period, then samples SDA (see [Figure](#_bookmark12)

[36-35](#_bookmark12)). If SDA is sampled low while SCL is sampled high, a bus collision has occurred. In this case, the Bus Collision Detect Interrupt Flag ([BCLIF](#_bookmark67)) is set, and if the Bus Collision Detect Interrupt Enable ([BCLIE](#_bookmark70)) bit is also set, the generic I2CxEIF is set and the module goes Idle. If SDA is sampled high while SCL is also sampled high, host hardware issues a Start condition. Once the Restart condition is detected on the bus, the Restart Condition Interrupt Flag ([RSCIF](#_bookmark55)) is set by hardware, and if the Restart Condition Interrupt Enable ([RSCIE](#_bookmark63)) bit is set, the generic I2CxIF is also set.

**Figure 36-35.** Restart Condition Timing

**Write to Start (S)**

**bit**

**Hardware**

**samples SDA**

**Completion of**

**Repeated Start**

**SDA**

**t (1) 2**

**SCL**

**I2CxCLK**

**1 2 3 4**

**TSCL/2**

**1 2**

**TSCL/2**

**3 4 1**

**SU:STA**

**2 3 4**

**1 2 3 4 1 2**

**If ABD = 0: I C Shift register loaded from I2CxADB0/1**

**If ABD = 1: I2C Shift register loaded from I2CxTXB**

**Host releases SDA**

**Host releases SCL**

**Repeated Start condition detected RSCIF = 1**

###### Important:

* + - * 1. See the device data sheet for Restart condition setup times.

## Stop Condition Timing

A Stop condition occurs when SDA transitions from an Active state to an Idle state while SCL is Idle. Host hardware will issue a Stop condition when it has completed its current transmission and is ready to release control of the bus. A Stop condition is also issued after an Error condition occurs, such as a bus time-out, or when a NACK condition is detected on the bus. User software may also generate a Stop condition by setting the Stop ([P](#_bookmark26)) bit.

After the ACK/NACK sequence of the final byte of the transmitted/received packet, hardware pulls SCL low for half of an SCL period (TSCL/2) (see [Figure 36-36](#_bookmark13)). After the half SCL period, hardware releases SCL, then samples SCL to ensure it is in an Idle state (SCL = 1). Host hardware then waits the duration of the Stop condition setup time (TSU:STO) and releases SDA, setting the Stop Condition Interrupt Flag ([PCIF](#_bookmark54)). If the Stop Condition Interrupt Enable ([PCIE](#_bookmark62)) bit is also set, the generic I2CxIF is also set.

**Important:** At least one SCL low period must appear before a Stop condition is valid. If the SDA line transitions low, then high again, while SCL is high, the Stop condition is ignored, and a Start condition will be detected by the receiver.

**Figure 36-36.** Stop Condition Timing

## SDA

**Stop condition**

**begins**

**Stop detected**

**PCIF = 1**

**D0**

**NACK**

**TSU:ST (2)**

**O**

**TSCL/2(1)**

**THD:ST (2)**

**O**

**8**

**9**

**NACK SEQUENCE**

**1 2 3 4 1 2 3 4 1 2 3 4 1 2**

**SCL**

## I2CxCLK

###### Important:

* + - * 1. At least one SCL low period must appear before a Stop is valid.
        2. See the device data sheet Electrical Specifications for Stop condition setup and hold times.

## Host Operation in 7-Bit Addressing Modes

In Host 7-bit Addressing modes, the client’s 7-bit address and R/W bit value are loaded into either [I2CxADB1](#_bookmark82) or [I2CxTXB](#_bookmark87), depending on the Address Buffer Disable ([ABD](#_bookmark34)) bit setting. When the host wishes to read data from the client, software must set the R/W bit (R/W = 1). When the host wishes to write data to the client, software must clear the R/W bit (R/W = 0).

## Host Transmission (7-Bit Addressing Mode)

The following section describes the sequence of events that occur when the module is transmitting data in 7-bit Addressing mode:

Depending on the configuration of the Address Buffer Disable ([ABD](#_bookmark34)) bit, one of two methods may be used to begin communication:

When [ABD](#_bookmark34) is clear (ABD = 0), the address buffer, [I2CxADB1](#_bookmark82), is enabled. In this case, the 7-bit client address and R/W bit are loaded into I2CxADB1, with the R/W bit clear (R/W = 0). The number of data bytes are loaded into [I2CxCNT](#_bookmark74), and the first data byte is loaded

into [I2CxTXB](#_bookmark87). After these registers are loaded, software must set the Start ([S](#_bookmark17)) bit to begin communication. Once the S bit is set, host hardware waits for the Bus Free ([BFRE](#_bookmark38)) bit to be set before transmitting the Start condition to avoid bus collisions.

When [ABD](#_bookmark34) is set (ABD = 1), the address buffer is disabled. In this case, the number of data bytes are loaded into [I2CxCNT](#_bookmark74), and the client’s 7-bit address and R/W bit are loaded into [I2CxTXB](#_bookmark87). A write to I2CxTXB will cause host hardware to automatically issue a Start condition once the bus is Idle ([BFRE](#_bookmark38) = 1). Software writes to the Start bit are ignored.

Host hardware waits for [BFRE](#_bookmark38) to be set, then shifts out the Start condition. Module hardware sets the Host Mode Active ([MMA](#_bookmark40)) bit and the Start Condition Interrupt Flag ([SCIF](#_bookmark56)). If the Start Condition Interrupt Enable ([SCIE](#_bookmark64)) bit is set, the generic I2CxIF is also set.

Host hardware transmits the 7-bit client address and R/W bit.

If upon the 8th falling edge of SCL, [I2CxTXB](#_bookmark87) is empty (Transmit Buffer Empty Status ([TXBE](#_bookmark45)) = 1), [I2CxCNT](#_bookmark74) is nonzero (I2CxCNT != 0), and the Clock Stretching Disable ([CSD](#_bookmark29)) bit is clear (CSD = 0):

* The I2C Transmit Interrupt Flag (I2CxTXIF) is set. If the I2C Transmit Interrupt Enable (I2CxTXIE) bit is also set, the generic I2CxIF is also set.
* The Host Data Request ([MDR](#_bookmark19)) bit is set, and the clock is stretched, allowing time for software to load [I2CxTXB](#_bookmark87) with new data. Once I2CxTXB has been written, hardware releases SCL and clears MDR.

Hardware transmits the 9th clock pulse and waits for an ACK/NACK response from the client. If the host receives an ACK, module hardware transfers the data from [I2CxTXB](#_bookmark87) into the transmit shift register, and [I2CxCNT](#_bookmark74) is decremented by one. If the host receives a NACK, hardware will attempt to issue a Stop condition. If the clock is currently being stretched by a client, the host must wait until the bus is free before issuing the Stop.

Host hardware checks [I2CxCNT](#_bookmark74) for a zero value. If I2CxCNT is zero:

If [ABD](#_bookmark34) is clear (ABD = 0), host hardware issues a Stop condition, or sets [MDR](#_bookmark19) if the Restart Enable ([RSEN](#_bookmark16)) bit is set and waits for software to set the Start bit to issue a Restart condition. [CNTIF](#_bookmark50) is set.

If [ABD](#_bookmark34) is set (ABD = 1), host hardware issues a Stop condition, or sets [MDR](#_bookmark19) if [RSEN](#_bookmark16) is set and waits for software to load [I2CxTXB](#_bookmark87) with a new client address. [CNTIF](#_bookmark50) is set.

Host hardware transmits the data byte.

If upon the 8th falling edge of SCL [I2CxTXB](#_bookmark87) is empty ([TXBE](#_bookmark45) = 1), [I2CxCNT](#_bookmark74) is nonzero ([I2CxCNT](#_bookmark74) !=

1. , and [CSD](#_bookmark29) is clear (CSD = 0):
   * I2CxTXIF is set. If the I2CxTXIE bit is also set, the generic I2CxIF is also set.
   * The [MDR](#_bookmark19) bit is set, and the clock is stretched, allowing time for software to load [I2CxTXB](#_bookmark87) with new data. Once I2CxTXB has been written, hardware releases SCL and clears MDR.

If [TXBE](#_bookmark45) is set (TXBE = 1) and [I2CxCNT](#_bookmark74) is zero (I2CxCNT = 0):

* + I2CxTXIF is NOT set.
  + [CNTIF](#_bookmark50) is set.
  + Host hardware issues a Stop condition, setting [PCIF](#_bookmark54).

Repeat Steps 5 – 8 until all data has been transmitted.

**Figure 36-37.** 7-Bit Host Mode Transmission

**Start**

###### R/W ACK (from client)

**Stop**

**Hardware sets PCIF**

## SDA

A7 A6 A5 A4 A3 A2 A1 **0** D7 D6 D5 D4 D3 D2 D1 D0

###### 7-bit address

D7 D6 D5 D4 D3 D2 D1 D0

## SCL

1 2 3 4 5 6 7 8

9 1 2 3 4 5 6 7 8

9 1 2

3 4 5

6 7 8 9

**MMA**

#### ACKSTAT

**Hardware sets MMA**

**on detection of Start**

**Client's ACK copied**

**to ACKSTAT**

**Hardware clears**

**MMA**

**Hardware sets CNTIF; RSTEN = 0, so**

**host issues**

**Stop**

## I2CxCNT

**TXBE**

###### 0x02

**Before Start, software loads data into I2CxTXB**

###### 0x01

**0x00**

**Data byte transferred to shift register,**

**I2CxTXIF NOT set**

**I2CxTXIF set,**

**data byte transferred to shift register**

**Software loads data into**

**I2CxTXB, clearing I2CxTXIF**

**PIC18F06/16Q41**

**I2C - Inter-Integrated Circuit Module**

Data Sheet

© 2020-2023 Microchip Technology Inc. and its subsidiaries

DS40002214F - 653

## Host Reception (7-Bit Addressing Mode)

The following section describes the sequence of events that occur when the module is receiving data in 7-bit Addressing mode:

Depending on the configuration of the Address Buffer Disable ([ABD](#_bookmark34)) bit, one of two methods may be used to begin communication:

When [ABD](#_bookmark34) is clear (ABD = 0), the address buffer, [I2CxADB1](#_bookmark82), is enabled. In this case, the 7-bit client address and R/W bit are loaded into I2CxADB1, with the R/W bit set (R/W = 1). The number of expected received data bytes are loaded into [I2CxCNT](#_bookmark74). After these registers are loaded, software must set the Start ([S](#_bookmark17)) bit to begin communication. Once the S bit is set, host hardware waits for the Bus Free ([BFRE](#_bookmark38)) bit to be set before transmitting the Start condition to avoid bus collisions.

When [ABD](#_bookmark34) is set (ABD = 1), the address buffer is disabled. In this case, the number of expected received data bytes are loaded into [I2CxCNT](#_bookmark74), and the client’s 7-bit address and R/W bit are loaded into [I2CxTXB](#_bookmark87). A write to I2CxTXB will cause host hardware to automatically issue a Start condition once the bus is Idle ([BFRE](#_bookmark38) = 1). Software writes to the Start bit are ignored.

Host hardware waits for [BFRE](#_bookmark38) to be set, then shifts out the Start condition. Module hardware sets the Host Mode Active ([MMA](#_bookmark40)) bit and the Start Condition Interrupt Flag ([SCIF](#_bookmark56)). If the Start Condition Interrupt Enable ([SCIE](#_bookmark64)) bit is set, the generic I2CxIF is also set.

Host hardware transmits the 7-bit client address and R/W bit.

Host hardware samples SCL to determine if the client is stretching the clock, and continues to sample SCL until the line is sampled high.

Host hardware transmits the 9th clock pulse, and receives the ACK/NACK response from the client.

If an ACK is received, host hardware receives the first seven bits of the data byte into the receive shift register.

If a NACK is received, hardware sets the NACK Detect Interrupt Flag ([NACKIF](#_bookmark68)), and:

[ABD](#_bookmark34) = 0: Host generates a Stop condition, or sets the [MDR](#_bookmark19) bit (if [RSEN](#_bookmark16) is also set) and waits for software to set the Start bit to generate a Restart condition.

[ABD](#_bookmark34) = 1: Host generates a Stop condition, or sets the [MDR](#_bookmark19) bit (if [RSEN](#_bookmark16) is also set) and waits for software to load a new address into [I2CxTXB](#_bookmark87). Software writes to the Start bit are ignored.

If the NACK Detect Interrupt Enable ([NACKIE](#_bookmark71)) is also set, hardware sets the generic I2CxEIF bit.

If previous data remains in the I2C Receive Buffer ([I2CxRXB](#_bookmark88)) when the first seven bits of the new byte are received into the receive shift register ([RXBF](#_bookmark48) = 1), the [MDR](#_bookmark19) bit is set (MDR = 1), and the clock is stretched after the 7th falling edge of SCL. This allows the host time to read I2CxRXB, which clears the RXBF bit, and prevents receive buffer overflows. Once RXBF is clear, hardware releases SCL.

The host clocks in the 8th bit of the data byte into the receive shift register, then transfers the full byte into [I2CxRXB](#_bookmark88). Host hardware sets the I2C Receive Interrupt Flag (I2CxRXIF) and [RXBF](#_bookmark48), and if the I2C Receive Interrupt Enable (I2CxRXIE) is set, the generic I2CxIF is also set. Finally, [I2CxCNT](#_bookmark74) is decremented by one.

Host hardware checks [I2CxCNT](#_bookmark74) for a zero value.

If I2CxCNT is nonzero (I2CxCNT != 0), hardware transmits the value of the Acknowledge Data ([ACKDT](#_bookmark23)) bit as the acknowledgement response to the client. It is up to user software to properly configure ACKDT. In most cases, ACKDT must be clear (ACKDT = 0), which indicates an ACK response.

If [I2CxCNT](#_bookmark74) is zero (I2CxCNT = 0), hardware transmits the value of the Acknowledge End of Count ([ACKCNT](#_bookmark22)) bit as the acknowledgement response to the client. [CNTIF](#_bookmark50) is set, and host hardware either issues a Stop condition or a Restart condition. It is up to user software to properly

configure ACKCNT. In most cases, ACKCNT must be set (ACKCNT = 1), which indicates a NACK response. When hardware detects a NACK on the bus, it automatically issues a Stop condition. If a NACK is not detected, the Stop will not be generated, which may lead to a stalled Bus condition.

Host hardware receives the first seven bits of the next data byte into the receive shift register.

Repeat Steps 6 – 9 until all expected bytes have been received.

**Figure 36-38.** 7-Bit Host Mode Reception**r**

**Start**

###### R/W

**ACK (from**

###### host)



**NACK (from**

###### host)

**Stop**

**Hardware sets PCIF**

## SDA

A7 A6 A5 A4 A3 A2 A1 **1**

###### 7-bit address

D7 D6 D5 D4 D3 D2 D1 D0

###### ACK (from client)

D7 D6 D5 D4 D3 D2 D1 D0

## SCL

1 2 3 4 5 6 7 8

9 1 2 3 4 5 6 7 8

9 1 2

3 4 5

6 7 8 9

## MMA

#### ACKDT

**Hardware sets MMA**

**on detection of Start**

**Host's ACK**

**copied from ACKDT**

**Hardware clears**

**MMA**

**Hardware sets CNTIF; RSTEN = 0, so**

**host issues**

**Stop**

#### ACKCNT

**Host's NACK copied**

**from ACKCNT**

## I2CxCNT

**RXBF**

###### 0x02

**I2CxRXIF is set**

###### 0x01

**Software reads I2CxRXB,**

**clearing I2CxRXIF**

**0x00**

**PIC18F06/16Q41**

**I2C - Inter-Integrated Circuit Module**

Data Sheet

© 2020-2023 Microchip Technology Inc. and its subsidiaries

DS40002214F - 656

## Host Operation in 10-Bit Addressing Modes

In Host 10-bit Addressing modes, the client’s 10-bit address and R/W bit value are loaded into either the [I2CxADB0](#_bookmark81) and [I2CxADB1](#_bookmark82) registers (when [ABD](#_bookmark34) = 0), or [I2CxTXB](#_bookmark87) (when ABD = 1). When the host intends to read data from the client, it must first transmit the full 10-bit address with the R/W bit clear (R/W = 0), issue a Restart condition, then transmit the address high byte with the R/W bit set (R/W = 1). When the host intends to write data to the client, it must transmit the full 10-bit address with the R/W bit clear (R/W = 0).

## Host Transmission (10-Bit)

The following section describes the sequence of events that occur when the module is transmitting data in 10-bit Addressing mode:

Depending on the configuration of the Address Buffer Disable ([ABD](#_bookmark34)) bit, one of two methods may be used to begin communication:

When [ABD](#_bookmark34) is clear (ABD = 0), the address buffers, [I2CxADB0](#_bookmark81) and [I2CxADB1](#_bookmark82), are enabled. In this case, the address high byte is loaded into I2CxADB1 with the R/W bit clear, while the address low byte is loaded into I2CxADB0. [I2CxCNT](#_bookmark74) is loaded with the total number of data bytes to transmit, and the first data byte is loaded into [I2CxTXB](#_bookmark87). After these registers are loaded, software must set the Start bit to begin communication.

When [ABD](#_bookmark34) is set (ABD = 1), the address buffers are disabled. In this case, [I2CxCNT](#_bookmark74) must be loaded with the total number of bytes to transmit prior to loading [I2CxTXB](#_bookmark87) with the address high byte and R/W bit. A write to I2CxTXB forces module hardware to issue a Start condition automatically; software writes to the [S](#_bookmark17) bit are ignored.

Host hardware waits for [BFRE](#_bookmark38) to be set, then shifts out the Start condition. Module hardware sets the Host Mode Active ([MMA](#_bookmark40)) bit and the Start Condition Interrupt Flag ([SCIF](#_bookmark56)). If the Start Condition Interrupt Enable ([SCIE](#_bookmark64)) bit is also set, the generic I2CxIF is also set.

Host hardware transmits the address high byte and R/W bit from [I2CxADB1](#_bookmark82).

Host hardware transmits the 9th clock pulse and shifts in the ACK/NACK response from the client.

If the host receives a NACK, it issues a Stop condition. If the host receives and ACK and:

[ABD](#_bookmark34) = 0: Hardware transmits the address low byte from [I2CxADB0](#_bookmark81).

[ABD](#_bookmark34) = 1: Hardware sets I2CxTXIF and the Host Data Request ([MDR](#_bookmark19)) bit and waits for software to load [I2CxTXB](#_bookmark87) with the address low byte. Software must load I2CxTXB to resume communication.

If upon the 8th falling edge of SCL [I2CxTXB](#_bookmark87) is empty ([TXBE](#_bookmark45) = 1), [I2CxCNT](#_bookmark74) is nonzero (I2CxCNT !=

1. , and the Clock Stretching Disable ([CSD](#_bookmark29)) bit is clear (CSD = 0):
   * I2CxTXIF is set. If the I2C Transmit Interrupt Enable (I2CxTXIE) bit is also set, the generic I2CxIF is also set.
   * [MDR](#_bookmark19) bit is set, and the clock is stretched, allowing time for software to load [I2CxTXB](#_bookmark87) with the address low byte. Once I2CxTXB has been written, hardware releases SCL and clears MDR.

Hardware transmits the 9th clock pulse and waits for an ACK/NACK response from the client. If the host receives an ACK, module hardware transfers the data from [I2CxTXB](#_bookmark87) into the transmit shift register, and [I2CxCNT](#_bookmark74) is decremented by one. If the host receives a NACK, hardware will attempt to issue a Stop condition. If the clock is currently being stretched by a client, the host must wait until the bus is free before issuing the Stop.

Host hardware checks [I2CxCNT](#_bookmark74) for a zero value. If I2CxCNT is zero:

If [ABD](#_bookmark34) is clear (ABD = 0), host hardware issues a Stop condition, or sets [MDR](#_bookmark19) if the Restart Enable ([RSEN](#_bookmark16)) bit is set and waits for software to set the Start bit to issue a Restart condition. [CNTIF](#_bookmark50) is set.

If [ABD](#_bookmark34) is set (ABD = 1), host hardware issues a Stop condition, or sets [MDR](#_bookmark19) if [RSEN](#_bookmark16) is set and waits for software to load [I2CxTXB](#_bookmark87) with a new client address. [CNTIF](#_bookmark50) is set.

Host hardware transmits the data byte.

If upon the 8th falling edge of SCL [I2CxTXB](#_bookmark87) is empty ([TXBE](#_bookmark45) = 1), [I2CxCNT](#_bookmark74) is nonzero (I2CxCNT !=

1. , and [CSD](#_bookmark29) is clear (CSD = 0):
   * The I2CxTXIF bit is set. If the I2CxTXIE bit is also set, the generic I2CxIF is also set.
   * The [MDR](#_bookmark19) bit is set, and the clock is stretched, allowing time for software to load [I2CxTXB](#_bookmark87) with new data. Once I2CxTXB has been written, hardware releases SCL and clears MDR.

If [TXBE](#_bookmark45) is set (TXBE = 1) and [I2CxCNT](#_bookmark74) is zero (I2CxCNT = 0):

* + I2CxTXIF is NOT set.
  + [CNTIF](#_bookmark50) is set.
  + Host hardware issues a Stop condition, setting [PCIF](#_bookmark54).

Repeat Steps 6 – 9 until all data has been transmitted.

**Figure 36-39.** 10-Bit Host Mode Transmission

**Start**

###### R/W ACK (from client)



**Stop**

## SDA

###### 1 1 1 1

**0** A9 A8 **0**

A7 A6 A5 A4 A3 A2 A1 A0

D7 D6 D5 D4 D3 D2 D1 D0

**Hardware sets PCIF**

**High address**

## SCL

1 2 3 4 5 6 7 8

9 1 2 3 4 5 6 7 8

9 1 2

3 4 5

6 7 8 9

## MMA

#### ACKSTAT

**Hardware sets MMA**

**on detection of Start**

**Client's ACK copied to ACKSTAT**

**Hardware clears**

**MMA**

**Hardware sets**

**CNTIF;**

**RSTEN = 0, so**

**host issues**

**Stop**

## I2CxCNT

###### 0x01

**0x00**

## TXBE

**Before Start, software loads data into I2CxTXB**

**Data byte transferred to shift register,**

**I2CxTXIF NOT set**

**PIC18F06/16Q41**

**I2C - Inter-Integrated Circuit Module**

Data Sheet

© 2020-2023 Microchip Technology Inc. and its subsidiaries

DS40002214F - 659

## Host Reception (10-Bit Addressing Mode)

The following section describes the sequence of events that occur when the module is receiving data in 10-bit Addressing mode:

Depending on the configuration of the Address Buffer Disable ([ABD](#_bookmark34)) bit, one of two methods may be used to begin communication:

When [ABD](#_bookmark34) is clear (ABD = 0), the address buffers, [I2CxADB0](#_bookmark81) and [I2CxADB1](#_bookmark82), are enabled. In this case, the address high byte and R/W bit are loaded into I2CxADB1, with R/W clear (R/W

= 0). The address low byte is loaded into I2CxADB0, and the Restart Enable ([RSEN](#_bookmark16)) bit is set by software. After these registers are loaded, software must set the Start ([S](#_bookmark17)) bit to begin communication. Once the S bit is set, host hardware waits for the Bus Free ([BFRE](#_bookmark38)) bit to be set before transmitting the Start condition to avoid bus collisions.

When [ABD](#_bookmark34) is set (ABD = 1), the address buffers are disabled. In this case, the number of expected received bytes are loaded into [I2CxCNT](#_bookmark74), the address high byte and R/W bit are loaded into [I2CxTXB](#_bookmark87), with R/W clear (R/W = 0). A write to I2CxTXB will cause host hardware to automatically issue a Start condition once the bus is Idle ([BFRE](#_bookmark38) = 1). Software writes to the Start bit are ignored.

Host hardware waits for [BFRE](#_bookmark38) to be set, then shifts out the Start condition. Module hardware sets the Host Mode Active ([MMA](#_bookmark40)) bit and the Start Condition Interrupt Flag ([SCIF](#_bookmark56)). If the Start Condition Interrupt Enable ([SCIE](#_bookmark64)) bit is set, the generic I2CxIF is also set.

Host hardware transmits the address high byte and R/W bit.

Host hardware samples SCL to determine if the client is stretching the clock, and continues to sample SCL until the line is sampled high.

Host hardware transmits the 9th clock pulse, and receives the ACK/NACK response from the client.

If a NACK was received, the NACK Detect Interrupt Flag ([NACKIF](#_bookmark68)) is set and the host immediately issues a Stop condition.

If an ACK was received, module hardware transmits the address low byte.

Host hardware samples SCL to determine if the client is stretching the clock, and continues to sample SCL until the line is sampled high.

Host hardware transmits the 9th clock pulse, and receives the ACK/NACK response from the client.

If an ACK was received, hardware sets [MDR](#_bookmark19), and waits for hardware or software to set the Start bit.

If a NACK is received, hardware sets [NACKIF](#_bookmark68), and:

[ABD](#_bookmark34) = 0: Host generates a Stop condition, or sets the [MDR](#_bookmark19) bit (if [RSEN](#_bookmark16) is also set) and waits for software to set the Start bit to generate a Restart condition.

[ABD](#_bookmark34) = 1: Host generates a Stop condition, or sets the MDR bit (if RSEN is also set) and waits for software to load a new address into [I2CxTXB](#_bookmark87). Software writes to the Start bit are ignored.

If the NACK Detect Interrupt Enable ([NACKIE](#_bookmark71)) is also set, hardware sets the generic I2CxEIF bit.

Software loads [I2CxCNT](#_bookmark74) with the expected number of received bytes.

If [ABD](#_bookmark34) is clear (ABD = 0), software sets the Start bit. If ABD is set (ABD = 1), software writes the address high byte with R/W bit into [I2CxTXB](#_bookmark87), with R/W set (R/W = 1).

Host hardware transmits the Restart condition, which sets the Restart Condition Interrupt Flag ([RSCIF](#_bookmark55)) bit. If the Restart Condition Interrupt Enable ([RSCIE](#_bookmark63)) bit is set, the generic I2CxIF is set by hardware.

Host hardware transmits the high address byte and R/W bit.

Host hardware samples SCL to determine if the client is stretching the clock, and continues to sample SCL until the line is sampled high.

Host hardware transmits the 9th clock pulse, and receives the ACK/NACK response from the client.

If an ACK is received, host hardware receives the first seven bits of the data byte into the receive shift register.

If a NACK is received, and:

[ABD](#_bookmark34) = 0: Host generates a Stop condition, or sets the [MDR](#_bookmark19) bit (if [RSEN](#_bookmark16) is also set) and waits for software to set the Start bit to generate a Restart condition.

ABD = 1: Host generates a Stop condition, or sets the MDR bit (if RSEN is also set) and waits for software to load a new address into [I2CxTXB](#_bookmark87). Software writes to the Start bit are ignored.

If previous data is currently in [I2CxRXB](#_bookmark88) ([RXBF](#_bookmark48) = 1) when the first seven bits are received by the receive shift register, hardware sets [MDR](#_bookmark19), and the clock is stretched after the 7th falling edge of SCL. This allows software to read I2CxRXB, which clears the RXBF bit, and prevents a receive buffer overflow. Once the RXBF bit is cleared, hardware releases SCL.

Host hardware clocks in the 8th bit of the data byte into the receive shift register, then transfers the complete byte into [I2CxRXB](#_bookmark88), which sets the I2CxRXIF and [RXBF](#_bookmark48) bits. If I2CxRXIE is also set, hardware sets the generic I2CxIF bit. [I2CxCNT](#_bookmark74) is decremented by one.

Hardware checks [I2CxCNT](#_bookmark74) for a zero value.

If [I2CxCNT](#_bookmark74) is nonzero (I2CxCNT != 0), hardware transmits the value of the Acknowledge Data ([ACKDT](#_bookmark23)) bit as the acknowledgement response to the client. It is up to user software to properly configure ACKDT. In most cases, ACKDT must be clear (ACKDT = 0), which indicates an ACK response.

If [I2CxCNT](#_bookmark74) is zero (I2CxCNT = 0), hardware transmits the value of the Acknowledge End of Count ([ACKCNT](#_bookmark22)) bit as the acknowledgement response to the client. [CNTIF](#_bookmark50) is set, and host hardware either issues a Stop condition or a Restart condition. It is up to user software to properly configure ACKCNT. In most cases, ACKCNT must be set (ACKCNT = 1), which indicates a NACK response. When hardware detects a NACK on the bus, it automatically issues a Stop condition. If a NACK is not detected, the Stop will not be generated, which may lead to a stalled Bus condition.

Host hardware receives the first seven bits of the next data byte into the receive shift register.

Repeat Steps 14 – 17 until all expected bytes have been received.

**PIC18F06/16Q41**

**I2C - Inter-Integrated Circuit Module**

Data Sheet

© 2020-2023 Microchip Technology Inc. and its subsidiaries

DS40002214F - 662

**Figure 36-40.** 10-Bit Host Mode Reception



**R/W**

**ACK (from client)**

A7 A6 A5 A4 A3 A2 A1 A0

**Restart**

**R/W**

**1**

**1**

**1 1 0** A9 A8 **1**

**Low address**

**High address**

**NACK**

**(from**

**host )**

D7 D6 D5 D4 D3 D2 D1 D0

**ACK (from client)**

**Stop**

**SDA**

**1 1 1 1 0** A9 A8 **0**

**High address**

**SCL**

1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9

1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9

**MMA**

**Hardware sets MMA**

**on detection of Start**

**MDR**

**Software sets Start,**

**clearing MDR**

**Hardware sets NACKIF, CNTIF**

**RSTEN**

**Software sets RSTEN**

**before setting Start**

**Software clears RSTEN before setting Start**

**I2CxCNT**

**0x00**

**0x00**

**ACKSTAT**

**Client's ACK copied**

**to ACKSTAT**

**Received data transferred to I2CxRXB, I2CxRXIF**

**RXBF**

**is set**

**0x01**

**Software loads I2CxCNT before setting Start**

**Hardware sets**

**RSCIF**

**Hardware sets MDR, wait for Start**

## I2C Multi-Host Mode Operation

In Multi-Host mode, multiple host devices reside on the same bus. A single device, or all devices, may act as both a host and a client. Control of the bus is achieved through clock synchronization and bus arbitration.

The Bus Free ([BFRE](#_bookmark38)) bit is used to determine if the bus is free. When BFRE is set (BFRE = 1), the bus is in an Idle state, allowing a host device to take control of the bus.

In Multi-Host mode, the Address Interrupt and Hold Enable ([ADRIE](#_bookmark61)) bit must be set (ADRIE = 1), and the Clock Stretching Disable ([CSD](#_bookmark29)) bit must be clear (CSD = 0), for a host device to be addressed as a client.

When a matching address is received into the receive shift register, the [SMA](#_bookmark39) bit is set, and the Address Interrupt Flag ([ADRIF](#_bookmark53)) bit is set. Since [ADRIE](#_bookmark61) is also set, hardware sets the Client Clock Stretching ([CSTR](#_bookmark18)) bit, and hardware stretches the clock to allow time for software to respond to the host device being addressed as a client. Once the address has been processed, software must clear CSTR to resume communication.

**Important:** Client hardware has priority over host hardware in Multi-Host mode. Host mode communication can only be initiated when [SMA](#_bookmark39) = 0.

## Multi-Host Mode Clock Synchronization

In a multi-host system, each host may begin to generate a clock signal as soon as the bus is Idle. Clock synchronization allows all devices on the bus to use a single SCL signal.

When a high-to-low transition on SCL occurs, all active host devices begin SCL low period timing, with their clocks held low until their low hold time expires and the High state is reached. If one host’s clock signal is still low, SCL will be held low until that host reaches its High state. During this time, all other host devices are held in a Wait state (see [Figure 36-41](#_bookmark14)).

Once all hosts have counted off their low period times, SCL is released high, and all host devices begin counting their high periods. The first host to complete its high period pulls the SCL line low again.

This means that when the clocks are synchronized, the SCL low period is determined by the host with the longest SCL low period, while the SCL high period is determined by the host device with the shortest SCL high period.

**Important:** The I2C Specification does not require the SCL signal to have a 50% duty cycle. In other words, one host’s clock signal may have a low time that is 60% of the SCL period and a high time that is 40% of the SCL period, while another host may be 50/50. This creates a timing difference between the two clock signals, which may result in data loss.

**Figure 36-41.** Clock Synchronization During Arbitration

## Host 1

**SCL**

## Host 2

**SCL**

## Actual bus

**SCL**

###### Wait state

**Host 1 pulls**

###### SCL low first

**Host 1 releases SCL,**

###### but Host 2 continues to pull SCL low

**Host 2**

**releases SCL**

## Multi-Host Mode Bus Arbitration

When the bus is Idle, any host device may attempt to take control of the bus. Two or more host devices may issue a Start condition within the minimum hold time (THD:STA), which triggers a valid Start on the bus. The host devices must then compete using bus arbitration to determine who takes control of the bus and completes their transaction.

Bus arbitration takes place bit by bit, and it may be possible for two hosts who have identical messages to complete the entire transaction without either device losing arbitration.

During every bit period, while SCL is high each host device compares the actual signal level of SDA to the signal level the host actually transmitted. SDA sampling is performed during the SCL high period because the SDA data must be stable during this period; therefore, the first host to detect a low signal level on SDA while it expects a high signal level loses arbitration. In this case, the ‘losing’ host device detects a bus collision and sets the Bus Collision Detect Interrupt Flag ([BCLIF](#_bookmark67)), and if the Bus Collision Detect Interrupt Enable ([BCLIE](#_bookmark70)) bit is set, the generic I2CxEIF is also set.

Arbitration can be lost in any of the following states:

* + - * + Address transfer
        + Data transfer
        + Start condition
        + Restart condition
        + Acknowledge sequence
        + Stop condition

If a collision occurs during the data transfer phase, the transmission is halted and both SCL and SDA are released by hardware. If a collision occurs during a Start, Restart, Acknowledge, or Stop, the operation is aborted and hardware releases SCL and SDA. If a collision occurs during the addressing phase, the host that ‘wins’ arbitration may be attempting to address the ‘losing’ host as a client. In this case, the host that lost arbitration must switch to its Client mode and check to see if an address matches.

**Important:** The I2C Specification states that a bus collision cannot occur during a Start condition. If a collision occurs during a Start, [BCLIF](#_bookmark67) will be set during the addressing phase.

User software must clear [BCLIF](#_bookmark67) to resume operation.

**Figure 36-42.** Bus Collision

#### SDA

**SCL**

**Change of**

**Host 1**

**releases SDA**

**Host 2 pulls**

**SDA low**

**Host 1 samples SDA**

**low, but expects SDA high**



**Hardware sets**

**BCLIF**

**Expected SDA**

**(SDA = 1)**

**Actual SDA**

**(SDA = 0)**

#### BCLIF

**data allowed**

**Figure 36-43.** Multi-Host Mode Transmission

**Start Stop**

## SCL

1 2 3 4 5 6 7 8

9 1 2 3 4 5 6 7 8

9 1 2

3 4 5

6 7 8 9

## SDA

###### (Host 1 )

1 1 0

**Host 1 wins arbitration, continues transmitting**

1 0 0 0 **0** D7 D6 D5 D4 D3 D2 D1 D0

D7 D6 D5 D4 D3 D2 D1 D0

**Hardware sets**

**PCIF**

## SDA

**(Host 2 )**

## MMA

**(Host 1)**

## MMA

###### (Host 2 )

1 1 1

###### R/W ACK (from client)

**Host 2 loses arbitration, hardware releases SDA**

**Host 2 loses arbitration, hardware clears MMA**

**Hardware sets** **CNTIF;**

**RSTEN = 0, so**

**host issues Stop**

**Hardware clears**

**MMA**

## BCLIF

**Host 2 loses arbitration, hardware sets BCLIF (software must clear BCLIF to**

**resume communication)**

**ACKSTAT**

**Client's ACK copied to ACKSTAT**

## I2CxCNT

**TXBE**

**Software loads data**

**0x02**

**0x01**

**0x00**

**Data byte transferred to shift register,**

**I2CxTXIF set,**

**data byte transferred to shift register**

**into I2CxTXB, clearing I2CxTXIF**

**I2CxTXIF NOT set**

**PIC18F06/16Q41**

**I2C - Inter-Integrated Circuit Module**

Data Sheet

© 2020-2023 Microchip Technology Inc. and its subsidiaries

DS40002214F - 666

# Register Definitions: I2C Control

Long bit name prefixes for the I2C peripherals are shown in the following table. Refer to the **“Long Bit Names”** section in the **“Register and Bit Naming Conventions”** chapter for more information.

**Table 36-3.** I2C Long Bit Name Prefixes

|  |  |
| --- | --- |
| **Peripheral** | **Bit Name Prefix** |
| I2C1 | I2C1 |

## I2CxCON0

**Name:** I2CxCON0

**Address:** 0x0294 I2C Control Register 0

Bit 7 6 5 4 3 2 1 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EN | RSEN | S | CSTR | MDR | MODE[2:0] |

Access R/W R/W R/W/HS/HC R/C/HS/HC R R/W R/W R/W Reset 0 0 0 0 0 0 0 0

**Bit 7 – EN** I2C Module Enable**(1,2)**

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | The I2C module is enabled |
| 0 | The I2C module is disabled |

**Bit 6 – RSEN** Restart Enable (*used only when MODE = 1xx*)

|  |  |
| --- | --- |
| **Value** | **Description** |
| 1 | Hardware sets MDR on 9th falling SCL edge (when [I2CxCNT](#_bookmark74) = 0 or [ACKSTAT](#_bookmark24) = 1) |
| 0 | Hardware issues Stop condition on 9th falling SCL edge (when I2CxCNT = 0 or ACKSTAT = 1) |

**Bit 5 – S** Host Start (*used only when MODE = 1xx*)

|  |  |  |
| --- | --- | --- |
| **Value Condition Description** | | |
| 1 | [MMA](#_bookmark40) = 0: | Set by write to [I2CxTXB](#_bookmark87) or S bit, hardware issues Start condition |
| 0 | MMA = 0: | Cleared by hardware after sending Start condition |
| 1 | MMA = 1 and MDR = 1: | Set by write to [I2CxTXB](#_bookmark87) or S bit, communication resumes with a Restart condition |
| 0 | MMA = 1 and MDR = 1: | Cleared by hardware after sending Restart condition |

**Bit 4 – CSTR** Client Clock Stretching**(3)**

|  |  |  |
| --- | --- | --- |
| **Value Condition Description** | | |
| 1 |  | Clock is held low (clock stretching) |
| 0 |  | Enable clocking, SCL control is released |
|  | [SMA](#_bookmark39) = 1 and [RXBF](#_bookmark48) = 1**(6)**: | Set by hardware on 7th falling SCL edge  User must read [I2CxRXB](#_bookmark88) and clear CSTR to release SCL |
|  | SMA = 1 and [TXBE](#_bookmark45) = 1 and [I2CxCNT](#_bookmark74) != 0: | Set by hardware on 8th falling SCL edge  User must write to [I2CxTXB](#_bookmark87) and clear CSTR to release SCL |
|  | when [ADRIE](#_bookmark61) = 1**(4)**: | Set by hardware on 8th falling edge of matching received address User must clear CSTR to release SCL |
|  | SMA = 1 and [WRIE](#_bookmark60) = 1: | Set by hardware on 8th falling SCL edge of received data byte User must clear CSTR to release SCL |
|  | SMA = 1 and [ACKTIE](#_bookmark59) = 1: | Set by hardware on 9th falling SCL edge User must clear CSTR to release SCL |

**Bit 3 – MDR** Host Data Request (*Host pause*)

|  |  |  |
| --- | --- | --- |
| **Value Condition Description** | | |
| 1 |  | Host state machine pauses until data is read/written (SCL is held low) |
| 0 |  | Host clocking of data is enabled |
|  | [MMA](#_bookmark40) = 1 and [RXBF](#_bookmark48) = 1 (*pause for RX*): | Set by hardware on 7th falling SCL edge User must read [I2CxRXB](#_bookmark88) to release SCL |
|  | [MMA](#_bookmark40) = 1 and [TXBE](#_bookmark45) = 1 and [I2CxCNT](#_bookmark74) != 0 (*pause for TX*): | Set by hardware on the 8th falling SCL edge User must write to [I2CxTXB](#_bookmark87) to release SCL |
|  | RSEN = 1 and [MMA](#_bookmark40) = 1 and ([I2CxCNT](#_bookmark74) = 0 or [ACKSTAT](#_bookmark24) = 1) (*pause for Restart*): | Set by hardware on 9th falling SCL edge  User must set S bit or write to [I2CxTXB](#_bookmark87) to release SCL and issue a Restart condition |

**Bits 2:0 – MODE[2:0]** I2C Mode Select

|  |  |
| --- | --- |
| **Value** | **Description** |
| 111 | I2C Multi-Host mode (SMBus 2.0 Host)**(5)** |
| 110 | I2C Multi-Host mode (SMBus 2.0 Host)**(5)** |
| 101 | I**2**C Host mode, 10-bit address |
| 100 | I**2**C Host mode, 7-bit address |
| 011 | I**2**C Client mode, one 10-bit address with masking |
| 010 | I**2**C Client mode, two 10-bit addresses |
| 001 | I**2**C Client mode, two 7-bit addresses with masking |
| 000 | I**2**C Client mode, four 7-bit addresses |

###### Notes:

* + - 1. SDA and SCL pins must be configured as open-drain I/Os and use either internal or external pull-up resistors.
      2. SDA and SCL signals must configure both the input and output PPS registers for each signal.
      3. CSTR can be set by multiple hardware sources; all sources must be addressed by user software before the SCL line can be released.
      4. [SMA](#_bookmark39) is set on the same SCL edge as CSTR for a matching received address.
      5. In this mode, [ADRIE](#_bookmark61) needs to be set, allowing an interrupt to clear the [BCLIF](#_bookmark67) condition and the ACK of a matching address.
      6. In 10-bit Client mode (when [ABD](#_bookmark34) = 1), CSTR will be set when the high address has not been read from [I2CxRXB](#_bookmark88) before the low address is shifted in.

## I2CxCON1

**Name:** I2CxCON1

**Address:** 0x0295 I2C Control Register 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ACKCNT | ACKDT | ACKSTAT | ACKT | P | RXO | TXU | CSD |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  |  |  |  |  |  |  |  |  |
| Access | R/W | R/W | R | R | R/S/HC | R/W/HS | R/W/HS | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – ACKCNT** Acknowledge End of Count**(2)**

|  |  |  |
| --- | --- | --- |
| **Value** | **Condition** | **Description** |
| 1 | [I2CxCNT](#_bookmark74) = 0 | Not Acknowledge (NACK) copied to SDA output |
| 0 | I2CxCNT = 0 | Acknowledge (ACK) copied to SDA output |

**Bit 6 – ACKDT** Acknowledge Data**(1,2)**

|  |  |  |
| --- | --- | --- |
| **Value** | **Condition** | **Description** |
| 1 | Matching received address | Not Acknowledge (NACK) copied to SDA output |
| 0 | Matching received address | Acknowledge (ACK) copied to SDA output |
| 1 | [I2CxCNT](#_bookmark74) != 0 | Not Acknowledge (NACK) copied to SDA output |
| 0 | I2CxCNT != 0 | Acknowledge (ACK) copied to SDA output |

**Bit 5 – ACKSTAT** Acknowledge Status (*Transmission only*)

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Acknowledge was not received for the most recent transaction |
| 0 | Acknowledge was received for the most recent transaction |

**Bit 4 – ACKT** Acknowledge Time Status

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Indicates that the bus is in an Acknowledge sequence, set on the 8th falling SCL edge |
| 0 | Not in an Acknowledge sequence, cleared on the 9th rising SCL edge |

**Bit 3 – P** Host Stop**(4)**

|  |  |  |
| --- | --- | --- |
| **Value Condition Description** | | |
| 1 | [MMA](#_bookmark40) = 1 | Initiate a Stop condition |
| 0 | MMA = 1 | Cleared by hardware after sending Stop |

**Bit 2 – RXO** Receive Overflow Status (*used only when* [MODE](#_bookmark20) *= 0xx or MODE = 11x*)**(3)**

|  |  |
| --- | --- |
| **Value** | **Description** |
| 1 | Set when [SMA](#_bookmark39) = 1 and a host receives data when [RXBF](#_bookmark48) = 1 |
| 0 | No client receive Overflow condition |

**Bit 1 – TXU** Transmit Underflow Status (*used only when* [MODE](#_bookmark20) *= 0xx or* [MODE](#_bookmark20) *= 11x*)**(3)**

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Set when [SMA](#_bookmark39) = 1 and a host transmits data when [TXBE](#_bookmark45) = 1 |
| 0 | No client transmit Underflow condition |

**Bit 0 – CSD** Clock Stretching Disable (*used only when* [MODE](#_bookmark20) *= 0xx or* [MODE](#_bookmark20) *= 11x*)

|  |  |
| --- | --- |
| **Value** | **Description** |
| 1 | When [SMA](#_bookmark39) = 1, the [CSTR](#_bookmark18) bit will not be set |
| 0 | Client clock stretching proceeds normally |

###### Notes:

* + - 1. Software writes to ACKDT must be followed by a minimum SDA setup time before clearing [CSTR](#_bookmark18).
      2. A NACK may still be generated by hardware when bus errors are present as indicated by the [I2CxSTAT1](#_bookmark43) or [I2CxERR](#_bookmark65) registers.
      3. This bit can only be set when [CSD](#_bookmark29) = 1.
      4. If SCL is high (SCL = 1) when this bit is set, the current clock pulse will complete (SCL = 0) with

the proper SCL/SDA timing required for a valid Stop condition; any data in the transmit or receive shift registers will be lost.

## I2CxCON2

**Name:** I2CxCON2

**Address:** 0x0296 I2C Control Register 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ACNT | GCEN | FME | ABD | SDAHT[1:0] | BFRET[1:0] |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  |  |  |  |  |  |  |  |  |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – ACNT** Auto-Load I2C Count Register Enable

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | The first transmitted/received byte after the address is automatically loaded into the [I2CxCNT](#_bookmark74) register |
| 0 | Auto-load of I2CxCNT is disabled |

**Bit 6 – GCEN** General Call Address Enable (*used when* [MODE](#_bookmark20) *= 00x or MODE = 11x*)

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | General Call Address (0x00) causes an address match event |
| 0 | General Call Addressing is disabled |

**Bit 5 – FME** Fast Mode Enable

|  |  |
| --- | --- |
| **Value** | **Description** |
| 1 | SCL frequency (FSCL) = FI2CxCLK/4 |
| 0 | SCL frequency (FSCL) = FI2CxCLK/5 |

**Bit 4 – ABD** Address Buffer Disable

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Address buffers are disabled.  Received address is loaded into [I2CxRXB](#_bookmark88), address to transmit is loaded into [I2CxTXB](#_bookmark87). |
| 0 | Address buffers are enabled.  Received address is loaded into [I2CxADB0](#_bookmark81)/[I2CxADB1](#_bookmark82), address to transmit is loaded into I2CxADB0/I2CxADB1. |

**Bits 3:2 – SDAHT[1:0]** SDA Hold Time Selection

|  |  |
| --- | --- |
| **Value Description** | |
| 11 | Reserved |
| 10 | Minimum of 30 ns hold time on SDA after the falling SCL edge |
| 01 | Minimum of 100 ns hold time on SDA after the falling SCL edge |
| 00 | Minimum of 300 ns hold time on SDA after the falling SCL edge |

**Bits 1:0 – BFRET[1:0]** Bus Free Time Selection

|  |  |
| --- | --- |
| **Value** | **Description** |
| 11 | 64 [I2CxCLK](#_bookmark72) pulses |
| 10 | 32 I2CxCLK pulses |
| 01 | 16 I2CxCLK pulses |
| 00 | 8 I2CxCLK pulses |

**Name:** I2CxSTAT0

**Address:** 0x0298 I2C Status Register 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| BFRE | SMA | MMA | R | D |  |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  |  |  |  |  |  |  |  |  |
| Access | R | R | R | R | R |  |  |  |
| Reset | 0 | 0 | 0 | 0 | 0 |  |  |  |

**Bit 7 – BFRE** Bus Free Status**(2)**

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Indicates an Idle bus; both SCL and SDA have been high for the time selected by the [BFRET](#_bookmark36) bits |
| 0 | Bus is not Idle |

**Bit 6 – SMA** Client Mode Active Status

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Client mode is active.  Set after the 8th falling SCL edge of a received matching 7-bit client address.  Set after the 8th falling SCL edge of a matching received 10-bit client **low** address.  Set after the 8th falling SCL edge of a received matching 10-bit client **high w/read** address, only after a previous received matching **high and low w/write** address. |
| 0 | Client mode is not active.  Cleared when any Restart/Stop condition is detected on the bus.  Cleared by the [BTOIF](#_bookmark66) and [BCLIF](#_bookmark67) conditions. |

**Bit 5 – MMA** Host Mode Active Status

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Host mode is active.  Set when Host state machine asserts a Start condition. |
| 0 | Host mode is not active.  Cleared when [BCLIF](#_bookmark67) is set.  Cleared when Stop condition is issued.  Cleared for the [BTOIF](#_bookmark66) condition after the host successfully shifts out a Stop condition. |

**Bit 4 – R** Read Information**(1)**

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Indicates that the last matching received address was a Read request |
| 0 | Indicates that the last matching received address was a Write request |

**Bit 3 – D** Data

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Indicates that the last byte received or transmitted was data |
| 0 | Indicates that the last byte received or transmitted was an address |

###### Notes:

* + - 1. This bit holds the R/W bit information following the last received address match. Addresses transmitted by the host do not affect the host’s R bit, and addresses appearing on the bus without a match do not affect the R bit.
      2. [I2CxCLK](#_bookmark72) must have a valid clock source selected for this bit to function.

**Name:** I2CxSTAT1

**Address:** 0x0299 I2C Status Register 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TXWE |  | TXBE |  | RXRE | CLRBF |  | RXBF |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  |  |  |  |  |  |  |  |  |
| Access | R/W/HS |  | R |  | R/W/HS | R/S |  | R |
| Reset | 0 |  | 1 |  | 0 | 0 |  | 0 |

**Bit 7 – TXWE** Transmit Write Error Status**(1)**

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | A new byte of data was written into [I2CxTXB](#_bookmark87) when it was full (*must be cleared by software*) |
| 0 | No transmit write error occurred |

**Bit 5 – TXBE** Transmit Buffer Empty Status**(2)**

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | [I2CxTXB](#_bookmark87) is empty (*cleared by writing to the I2CxTXB register*) |
| 0 | I2CxTXB is full |

**Bit 3 – RXRE** Receive Read Error Status(1)

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | A byte of data was read from [I2CxRXB](#_bookmark88) when it was empty (*must be cleared by software*) |
| 0 | No receive overflow occurred |

**Bit 2 – CLRBF** Clear Buffer(3)

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Setting this bit clears/empties the receive and transmit buffers, causing a Reset of RXBF and TXBE Setting this bit clears the I2CxRXIF and I2CxTXIF interrupt flags |

**Bit 0 – RXBF** Receive Buffer Full Status(2)

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | [I2CxRXB](#_bookmark88) is full (*cleared by reading the I2CxRXB register*) |
| 0 | I2CxRXB is empty |

###### Notes:

1. This bit, when set, will cause a NACK to be issued.
2. Used as a trigger source for DMA operations.
3. This bit is special function; it can only be set by user software and always reads ‘0’.

## I2CxPIR

**Name:** I2CxPIR

**Address:** 0x029A

I2C Interrupt Flag Register

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CNTIF | ACKTIF |  | WRIF | ADRIF | PCIF | RSCIF | SCIF |

**PIC18F06/16Q41**

**I2C - Inter-Integrated Circuit Module**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  |  |  |  |  |  |  |  |  |
| Access | R/W/HS | R/W/HS |  | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS |
| Reset | 0 | 0 |  | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – CNTIF** Byte Count Interrupt Flag**(1)**

|  |  |
| --- | --- |
| **Value** | **Description** |
| 1 | Set on the 9th falling SCL edge when [I2CxCNT](#_bookmark74) = 0 |
| 0 | I2CxCNT value is not zero |

**Bit 6 – ACKTIF** Acknowledge Status Time Interrupt Flag (*used only when* [MODE](#_bookmark20) *= 0xx or MODE = 11x*)**(1,2)**

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Acknowledge sequence detected, set on the 9th falling SCL edge for any byte when addressed as a client |
| 0 | Acknowledge sequence not detected |

**Bit 4 – WRIF** Data Write Interrupt Flag (*used only when* [MODE](#_bookmark20) *= 0xx or MODE = 11x*)**(1)**

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Data byte detected, set on the 8th falling SCL edge for a received data byte |
| 0 | Data byte not detected |

**Bit 3 – ADRIF** Address Interrupt Flag (*used only when* [MODE](#_bookmark20) *= 0xx or MODE = 11x*)**(1)**

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Address detected, set on the 8th falling SCL edge for a matching received address byte |
| 0 | Address not detected |

**Bit 2 – PCIF** Stop Condition Interrupt Flag**(1)**

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Stop condition detected |
| 0 | Stop condition not detected |

**Bit 1 – RSCIF** Restart Condition Interrupt Flag**(1)**

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Restart condition detected |
| 0 | Restart condition not detected |

**Bit 0 – SCIF** Start Condition Interrupt Flag**(1)**

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Start condition detected |
| 0 | Start condition not detected |

###### Notes:

* + - 1. Enabled interrupt flags are OR’ed to produce the PIRx[I2CxIF] bit.
      2. ACKTIF is not set by a matching 10-bit high address byte with the R/W bit clear. It is only set after the matching low address byte is shifted in.

## I2CxPIE

**Name:** I2CxPIE

**Address:** 0x029B

I2C Interrupt and Hold Enable Register

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CNTIE | ACKTIE |  | WRIE | ADRIE | PCIE | RSCIE | SCIE |

**PIC18F06/16Q41**

**I2C - Inter-Integrated Circuit Module**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  |  |  |  |  |  |  |  |  |
| Access | R/W | R/W |  | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 |  | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – CNTIE** Byte Count Interrupt Enable**(1)**

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Enables Byte Count interrupts |
| 0 | Disables Byte Count interrupts |

**Bit 6 – ACKTIE** Acknowledge Status Time Interrupt and Hold Enable**(1,2)**

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Enables Acknowledge Status Time Interrupt and Hold condition |
| 0 | Disables Acknowledge Status Time Interrupt and Hold condition |

**Bit 4 – WRIE** Data Write Interrupt and Hold Enable**(1,3)**

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Enables Data Write Interrupt and Hold condition |
| 0 | Disables Data Write Interrupt and Hold condition |

**Bit 3 – ADRIE** Address Interrupt and Hold Enable**(1,4)**

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Enables Address Interrupt and Hold condition |
| 0 | Disables Address Interrupt and Hold condition |

**Bit 2 – PCIE** Stop Condition Interrupt Enable**(1)**

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Enables interrupt on the detection of a Stop condition |
| 0 | Disables interrupt on the detection of a Stop condition |

**Bit 1 – RSCIE** Restart Condition Interrupt Enable**(1)**

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Enables interrupt on the detection of a Restart condition |
| 0 | Disables interrupt on the detection of a Restart condition |

**Bit 0 – SCIE** Stop Condition Interrupt Enable**(1)**

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Enables interrupt on the detection of a Start condition |
| 0 | Disables interrupt on the detection of a Start condition |

###### Notes:

* + - 1. Enabled interrupt flags are OR’ed to produce the PIRx[I2CxIF] bit.
      2. When ACKTIE is set (ACKTIE = 1) and [ACKTIF](#_bookmark51) becomes set (ACKTIF = 1), if an ACK is generated, [CSTR](#_bookmark18) is also set. If a NACK is generated, CSTR remains unchanged.
      3. When WRIE is set (WRIE = 1) and [WRIF](#_bookmark52) becomes set (WRIF = 1), [CSTR](#_bookmark18) is also set.
      4. When ADRIE is set (ADRIE = 1) and [ADRIF](#_bookmark53) becomes set (ADRIF = 1), [CSTR](#_bookmark18) is also set.

## I2CxERR

**Name:** I2CxERR

**Address:** 0x0297 I2C Error Register

**PIC18F06/16Q41**

**I2C - Inter-Integrated Circuit Module**

Bit 7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | BTOIF | BCLIF | NACKIF |  | BTOIE | BLCIE | NACKIE |

Access R/W/HS R/W/HS R/W/HS R/W R/W R/W Reset 0 0 0 0 0 0

**Bit 6 – BTOIF** Bus Time-Out Interrupt Flag**(1,2)**

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Bus time-out event occurred |
| 0 | No bus time-out event occurred |

**Bit 5 – BCLIF** Bus Collision Detect Interrupt Flag**(1)**

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Bus collision detected |
| 0 | No bus collision occurred |

**Bit 4 – NACKIF** NACK Detect Interrupt Flag**(1,3,4)**

|  |  |
| --- | --- |
| **Value** | **Description** |
| 1 | NACK detected on the bus (when [SMA](#_bookmark39) = 1 or [MMA](#_bookmark40) = 1) |
| 0 | No NACK detected on the bus |

**Bit 2 – BTOIE** Bus Time-Out Interrupt Enable

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Enable bus time-out interrupts |
| 0 | Disable bus time-out interrupts |

**Bit 1 – BLCIE** Bus Collision Detect Interrupt Enable

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Enable Bus Collision interrupts |
| 0 | Disable Bus Collision interrupts |

**Bit 0 – NACKIE** NACK Detect Interrupt Enable

|  |  |
| --- | --- |
| **Value Description** | |
| 1 | Enable NACK detect interrupts |
| 0 | Disable NACK detect interrupts |

###### Notes:

* + - 1. Enabled error interrupt flags are OR’ed to produce the PIRx[I2CxEIF] bit.
      2. User software must select the bus time-out source in the [I2CxBTOC](#_bookmark79) register.
      3. NACKIF is also set when any of the [TXWE](#_bookmark44), [RXRE](#_bookmark46), [TXU](#_bookmark28), or [RXO](#_bookmark27) bits are set.
      4. NACKIF is not set for the NACK response to a nonmatching client address.

## I2CxCLK

**Name:** I2CxCLK

**Address:** 0x029E

I2C Clock Selection Register

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | CLK[3:0] |

**PIC18F06/16Q41**

**I2C - Inter-Integrated Circuit Module**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  |  |  |  |  |  |  |  |  |
| Access  Reset |  |  |  |  | R/W 0 | R/W 0 | R/W 0 | R/W 0 |

**Bits 3:0 – CLK[3:0]** I2C Clock Selection

###### Table 36-4.

|  |  |
| --- | --- |
| **CLK** | **Selection** |
| 1111 - 1110 | Reserved |
| 1101 | CLC4\_OUT |
| 1100 | CLC3\_OUT |
| 1011 | CLC2\_OUT |
| 1010 | CLC1\_OUT |
| 1001 | SMT1\_OUT |
| 1000 | TMR4\_Postscaler\_OUT |
| 0111 | TMR2\_Postscaler\_OUT |
| 0110 | TMR0\_OUT |
| 0101 | EXTOSC |
| 0100 | Clock Reference Output |
| 0011 | MFINTOSC (500 kHz) |
| 0010 | HFINTOSC |
| 0001 | FOSC (System Clock) |
| 0000 | FOSC/4 |

**Name:** I2CxBAUD

**Address:** 0x029D

I2C Baud Rate Prescaler

Bit 7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | BAUD[7:0] | |  | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – BAUD[7:0]** Baud Rate Prescaler Selection

|  |  |
| --- | --- |
| **Value Description** | |
| n | Prescaled I2C Clock Frequency (FPRECLK) = I2CxCLK  BAUD + 1 |

**Note:** It is recommended to write this register only when the module is Idle ([MMA](#_bookmark40) = 0 or [SMA](#_bookmark39) = 0), or when the module is clock stretching ([CSTR](#_bookmark18) = 1 or [MDR](#_bookmark19) = 1).

**Name:** I2CxCNT

**Address:** 0x028C

I2C Byte Count Register**(1,2)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 15 | 14 | 13 | 12 |  | 11 | 10 | 9 | 8 |
|  |  |  |  |  | CNT[15:8] |  |  |  |  |
| Access | R/W | R/W | R/W | R/W |  | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 |  | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 |  | 3 | 2 | 1 | 0 |
|  |  |  |  |  | CNT[7:0] |  |  |  |  |
| Access | R/W | R/W | R/W | R/W |  | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 |  | 0 | 0 | 0 | 0 |

**Bits 15:0 – CNT[15:0]** Byte Count

|  |  |
| --- | --- |
| **Condition Description** | |
| **If receiving data:** | Count value decremented on 8th falling SCL edge when a new byte is loaded into [I2CxRXB](#_bookmark88) |
| **If transmitting data:** | Count value is decremented on the 9th falling SCL edge when a new byte is moved from [I2CxTXB](#_bookmark87) |

###### Notes:

* + - 1. It is recommended to write this register only when the module is Idle ([MMA](#_bookmark40) = 0 or [SMA](#_bookmark39) = 0), or when the module is clock stretching ([CSTR](#_bookmark18) = 1 or [MDR](#_bookmark19) = 1).
      2. [CNTIF](#_bookmark50) is set on the 9th falling SCL edge when I2CxCNT = 0.

## I2CxBTO

**Name:** I2CxBTO

**Address:** 0x029C

I2C Bus Time-Out Register**(1)**

|  |  |  |
| --- | --- | --- |
| TOREC | TOBY32 | TOTIME[5:0] |

**PIC18F06/16Q41**

**I2C - Inter-Integrated Circuit Module**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  |  |  |  |  |  |  |  |  |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – TOREC** Time-Out Recovery Selection

|  |  |
| --- | --- |
| **Value** | **Description** |
| 1 | A BTO event will reset the I2C module and set BTOIF |
| 0 | A BTO event will set BTOIF, but will not reset the I2C module |

**Bit 6 – TOBY32** Time-Out Prescaler Extension Enable**(2)**

|  |  |
| --- | --- |
| **Value** | **Description** |
| 1 | BTO time = TOTIME \* TBTOCLK |
| 0 | BTO time = TOTIME \* TBTOCLK \* 32 |

**Bits 5:0 – TOTIME[5:0]** Time-Out Time Selection

|  |  |  |
| --- | --- | --- |
| **Value** | **Condition** | **Description** |
| n | TOBY32 = 1 | Time-out is TOTIME periods of the prescaled BTO clock (TOTIME = n \* TBTOCLK) |
| n | TOBY32 = 0 | Time-out is TOTIME periods of the prescaled BTO clock multiplied by 32 (TOTIME = n \* TBTOCLK \* 32) |

###### Notes:

1. It is recommended to write this register only when the module is Idle ([MMA](#_bookmark40) = 0 or [SMA](#_bookmark39) = 0), or when the module is clock stretching ([CSTR](#_bookmark18) = 1 or [MDR](#_bookmark19) = 1).
2. When TOBY32 is set (TOBY32 = 1) and the LFINTOSC, MFINTOSC, or SOSC is selected as the BTO clock source, the time-out time (TOTIME) will be approximately in milliseconds.

## I2CxBTOC

**Name:** I2CxBTOC

**Address:** 0x029F

I2C Bus Time-Out Clock Source Selection

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | BTOC[2:0] |

**PIC18F06/16Q41**

**I2C - Inter-Integrated Circuit Module**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  |  |  |  |  |  |  |  |  |
| Access  Reset |  |  |  |  |  | R/W 0 | R/W 0 | R/W 0 |

**Bits 2:0 – BTOC[2:0]** Bus Time-Out Clock Source Selection

###### Table 36-5.

|  |  |
| --- | --- |
| **BTOC** | **Selection** |
| 111 | Reserved |
| 110 | Reserved |
| 101 | SOSC |
| 100 | MFINTOSC (32 kHz) |
| 011 | LFINTOSC |
| 010 | TMR4\_postscaled |
| 001 | TMR2\_postscaled |
| 000 | Reserved |

**Name:** I2CxADB0

**Address:** 0x028E

I2C Address Buffer 0 Register**(1)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 7 | 6 | 5 | 4 |  | 3 | 2 | 1 | 0 |
|  |  |  |  |  | ADB[7:0] |  |  |  |  |
| Access | R/W | R/W | R/W | R/W |  | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 |  | 0 | 0 | 0 | 0 |

**Bits 7:0 – ADB[7:0]** I2C Address Buffer 0

|  |  |
| --- | --- |
| **Condition** | **Description** |
| 7-bit Client/Multi-Host modes *(*[MODE](#_bookmark20) *= 00x or 11x)*: | **ADB[7:1]:** Received matching 7-bit client address  **ADB[0]:** Received R/W value from 7-bit address |
| 10-bit Client modes *(*[MODE](#_bookmark20) *= 01x)*: | **ADB[7:0]:** Received matching lower eight bits of 10-bit client address |
| 7-bit Host mode *(*[MODE](#_bookmark20) *= 100)*: | Unused in this mode |
| 10-bit Host mode *(*[MODE](#_bookmark20) *= 101)*: | **ADB[7:0]:** Eight Least Significant bits of the 10-bit client address |

###### Note:

* + - 1. This register is read-only except in Host 10-bit Address mode *(*[MODE](#_bookmark20) *= 101)*.

**Name:** I2CxADB1

**Address:** 0x028F

I2C Address Buffer 1 Register**(1)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 7 | 6 | 5 | 4 |  | 3 | 2 | 1 | 0 |
|  |  |  |  |  | ADB[7:0] |  |  |  |  |
| Access | R/W | R/W | R/W | R/W |  | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 |  | 0 | 0 | 0 | 0 |

**Bits 7:0 – ADB[7:0]** I2C Address Buffer 1

|  |  |
| --- | --- |
| **Condition** | **Description** |
| 7-bit Client modes *(*[MODE](#_bookmark20) *= 00x)*: | Unused in this mode |
| 10-bit Client modes *(*[MODE](#_bookmark20) *= 01x)*: | **ADB[7:1]:** Received matching 10-bit client address high byte  **ADB[0]:** Received R/W value from 10-bit high address byte |
| 7-bit Host mode *(*[MODE](#_bookmark20) *= 100)*: | **ADB[7:1]:** 7-bit client address  **ADB[0]:** R/W value |
| 10-bit Host mode *(*[MODE](#_bookmark20) *= 101)*: | **ADB[7:1]:** 10-bit client high address byte  **ADB[0]:** R/W value |
| 7-bit Multi-Host modes *(*[MODE](#_bookmark20) *= 11x)*: | **ADB[7:1]:** 7-bit client address  **ADB[0]:** R/W value |

###### Note:

1. This register is read-only in 7-bit Client Address modes *(*[MODE](#_bookmark20) *= 0xx)*.

**Name:** I2CxADR0

**Address:** 0x0290 I2C Address 0 Register

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 7 | 6 | 5 | 4 |  | 3 | 2 | 1 | 0 |
|  |  |  |  |  | ADR[7:0] |  |  |  |  |
| Access | R/W | R/W | R/W | R/W |  | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 |  | 1 | 1 | 1 | 1 |

**Bits 7:0 – ADR[7:0]** I2C Client Address 0

|  |  |
| --- | --- |
| **Condition** | **Description** |
| 7-bit Client/Multi-Host modes *(*[MODE](#_bookmark20) *= 00x or 11x)*: | **ADR[7:1]:** 7-bit client address  **ADR[0]:** Unused; bit state is ‘don’t care’ |
| 10-bit Client modes *(*[MODE](#_bookmark20) *= 01x)*: | **ADR[7:0]:** Eight Least Significant bits of first 10-bit address |

**Name:** I2CxADR1

**Address:** 0x0291 I2C Address 1 Register

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 0 |
|  |  |  |  | ADR[6:0] |  |  |  |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bits 7:1 – ADR[6:0]** I2C Client Address 1

|  |  |
| --- | --- |
| **Condition** | **Description** |
| 7-bit Client/Multi-Host modes *(*[MODE](#_bookmark20) *= 000 or 110)*: | 7-bit client address 1 |
| 7-bit Client/Multi-Host modes with Masking *(*[MODE](#_bookmark20) *= 011*  *or 111)*: | 7-bit client address mask for I2CxADR0 |
| 10-bit Client mode *(*[MODE](#_bookmark20) *= 010)*: | **ADR[7:3]:** Bit pattern (11110) as defined by the I2C Specification**(1)**  **ADR[2:1]:** Two Most Significant bits of first 10-bit address |
| 10-bit Client mode with Masking *(*[MODE](#_bookmark20) *= 011)*: | **ADR[7:3]:** Bit pattern (11110) as defined by the I2C Specification**(1)**  **ADR[2:1]:** Two Most Significant bits of 10-bit address |

###### Note:

1. The ‘11110’ bit pattern used in the 10-bit address high byte is defined by the I2C Specification. It is up to the user to define these bits. These bit values are compared to the received address by hardware to determine a match. The bit pattern transmitted by the host must be the same as the client address’s bit pattern used for comparison or a match will not occur.

**Name:** I2CxADR2

**Address:** 0x0292 I2C Address 2 Register

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 7 | 6 | 5 | 4 |  | 3 | 2 | 1 | 0 |
|  |  |  |  |  | ADR[7:0] |  |  |  |  |
| Access | R/W | R/W | R/W | R/W |  | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 |  | 1 | 1 | 1 | 1 |

**Bits 7:0 – ADR[7:0]** I2C Client Address 2

|  |  |
| --- | --- |
| **Condition** | **Description** |
| 7-bit Client/Multi-Host modes *(*[MODE](#_bookmark20) *= 000 or 110)*: | **ADR[7:1]:** 7-bit client address 2  **ADR[0]:** Unused; bit state is ‘don’t care’ |
| 7-bit Client/Multi-Host modes with Masking *(*[MODE](#_bookmark20) *= 001 or*  *111)*: | **ADR[7:1]:** 7-bit client address  **ADR[0]:** Unused; bit state is ‘don’t care’ |
| 10-bit Client mode *(*[MODE](#_bookmark20) *= 010)*: | **ADR[7:0]:** Eight Least Significant bits of the second 10-bit address |
| 10-bit Client mode with Masking *(*[MODE](#_bookmark20) *= 011)*: | **ADR[7:0]:** Eight Least Significant bits of 10-bit address mask |

**Name:** I2CxADR3

**Address:** 0x0293

I2C Address 3 Register**(1)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 0 |
|  |  |  |  | ADR[6:0] |  |  |  |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bits 7:1 – ADR[6:0]** I2C Client Address 3

|  |  |
| --- | --- |
| **Name** | **Description** |
| 7-bit Client/Multi-Host modes *(*[MODE](#_bookmark20) *= 000 or 110)*: | 7-bit client address 3 |
| 7-bit Client/Multi-Host modes with Masking *(*[MODE](#_bookmark20) *= 001*  *or 111)*: | 7-bit client address mask for I2CxADR2 |
| 10-bit Client mode *(*[MODE](#_bookmark20) *= 010)*: | **ADR[7:3]:** Bit pattern (11110) as defined by the I2C Specification**(1)**  **ADR[2:1]:** Two Most Significant bits of second 10-bit address |
| 10-bit Client mode with Masking *(*[MODE](#_bookmark20) *= 011)*: | **ADR[7:3]:** Bit pattern (11110) as defined by the I2C Specification**(1)**  **ADR[2:1]:** Two Most Significant bits of 10-bit address mask |

###### Note:

1. The ‘11110’ bit pattern used in the 10-bit address high byte is defined by the I2C Specification. It is up to the user to define these bits. These bit values are compared to the received address by hardware to determine a match. The bit pattern transmitted by the host must be the same as the client address’s bit pattern used for comparison or a match will not occur.

**Name:** I2CxTXB

**Address:** 0x028B

I2C Transmit Buffer Register**(1)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 7 | 6 | 5 | 4 |  | 3 | 2 | 1 | 0 |
|  |  |  |  |  | TXB[7:0] |  |  |  |  |
| Access | W | W | W | W |  | W | W | W | W |
| Reset | x | x | x | x |  | x | x | x | x |

**Bits 7:0 – TXB[7:0]** I2C Transmit Buffer

**Note:** This register is write-only. Reading this register will return a value of 0x00.

**Name:** I2CxRXB

**Address:** 0x028A I2C Receive Buffer**(1)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 7 | 6 | 5 | 4 |  | 3 | 2 | 1 | 0 |
|  |  |  |  |  | RXB[7:0] |  |  |  |  |
| Access | R | R | R | R |  | R | R | R | R |
| Reset | x | x | x | x |  | x | x | x | x |

**Bits 7:0 – RXB[7:0]** I2C Receive Buffer

**Note:** This register is read-only. Writes to this register are ignored.

# Register Summary - I2C

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Address** | **Name** | **Bit Pos.** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 0x00  ...  0x0289 | Reserved |  |  |  |  |  |  |  |  |  |
| 0x028A | [I2C1RXB](#_bookmark88) | 7:0 | RXB[7:0] | | | | | | | |
| 0x028B | [I2C1TXB](#_bookmark87) | 7:0 | TXB[7:0] | | | | | | | |
| 0x028C | [I2C1CNT](#_bookmark74) | 7:0 | CNT[7:0] | | | | | | | |
| 15:8 | CNT[15:8] | | | | | | | |
| 0x028E | [I2C1ADB0](#_bookmark81) | 7:0 | ADB[7:0] | | | | | | | |
| 0x028F | [I2C1ADB1](#_bookmark82) | 7:0 | ADB[7:0] | | | | | | | |
| 0x0290 | [I2C1ADR0](#_bookmark83) | 7:0 | ADR[7:0] | | | | | | | |
| 0x0291 | [I2C1ADR1](#_bookmark84) | 7:0 | ADR[6:0] | | | | | | |  |
| 0x0292 | [I2C1ADR2](#_bookmark85) | 7:0 | ADR[7:0] | | | | | | | |
| 0x0293 | [I2C1ADR3](#_bookmark86) | 7:0 | ADR[6:0] | | | | | | |  |
| 0x0294 | [I2C1CON0](#_bookmark15) | 7:0 | EN | RSEN | S | CSTR | MDR | MODE[2:0] | | |
| 0x0295 | [I2C1CON1](#_bookmark21) | 7:0 | ACKCNT | ACKDT | ACKSTAT | ACKT | P | RXO | TXU | CSD |
| 0x0296 | [I2C1CON2](#_bookmark30) | 7:0 | ACNT | GCEN | FME | ABD | SDAHT[1:0] | | BFRET[1:0] | |
| 0x0297 | [I2C1ERR](#_bookmark65) | 7:0 |  | BTOIF | BCLIF | NACKIF |  | BTOIE | BLCIE | NACKIE |
| 0x0298 | [I2C1STAT0](#_bookmark37) | 7:0 | BFRE | SMA | MMA | R | D |  |  |  |
| 0x0299 | [I2C1STAT1](#_bookmark43) | 7:0 | TXWE |  | TXBE |  | RXRE | CLRBF |  | RXBF |
| 0x029A | [I2C1PIR](#_bookmark49) | 7:0 | CNTIF | ACKTIF |  | WRIF | ADRIF | PCIF | RSCIF | SCIF |
| 0x029B | [I2C1PIE](#_bookmark57) | 7:0 | CNTIE | ACKTIE |  | WRIE | ADRIE | PCIE | RSCIE | SCIE |
| 0x029C | [I2C1BTO](#_bookmark75) | 7:0 | TOREC | TOBY32 | TOTIME[5:0] | | | | | |
| 0x029D | [I2C1BAUD](#_bookmark73) | 7:0 | BAUD[7:0] | | | | | | | |
| 0x029E | [I2C1CLK](#_bookmark72) | 7:0 |  |  |  |  | CLK[3:0] | | | |
| 0x029F | [I2C1BTOC](#_bookmark79) | 7:0 |  |  |  |  |  | BTOC[2:0] | | |