TB2669

I²C Slave Mode

Introduction

Author: Christopher Best, Microchip Technology Inc.

Inter-Integrated Circuit, more commonly referred to as I^{2}C, is a synchronous, two-wire, bidirectional serial communications bus. The I^{2}C module can be used to communicate with other I^{2}C compatible EEPROMs, display drivers, sensors, or other microcontroller devices. This technical brief discusses the features and functions of the stand-alone I^{2}C module in Slave mode. The stand-alone I^{2}C module should not be confused with the traditional Master Synchronous Serial Port (MSSP), which contained both I^{2}C and Serial Peripheral Interface (SPI) functions.

I^{2}C Module Modes and Features

The I^{2}C module provides the following operational modes and features:

- Master Mode
- Slave Mode with Byte NACKing
- Multi-Master Mode
- Dedicated Receive and Transmit Buffers
- Up to Four Dedicated Slave Address Registers**(1)**
- 7-Bit and 10-Bit Addressing with Masking
- General Call Addressing
- Interrupts
- Bus Collision Detection
- Bus Time-Out Detection with Programmable Sources
- SDA Hold Time Selection
- Programmable Bus-Free Time Selection
- I^{2}C, SMBus 2.0, and SMBus 3.0 Input Level Selection
- Direct Memory Access (DMA) Support**(2)**

Note:

1. Support for four dedicated slave registers is only available when in 7-bit Addressing mode. When in 10-bit Addressing mode, only two dedicated Address registers are available.
2. Direct Memory Access (DMA) is not available on all devices. Refer to the device data sheet to determine if the DMA is available.

Table of Contents

1. I^{2}C Specification

The I^{2}C specification was developed by Phillips Semiconductors to communicate between devices connected to a two-wire bus. Phillips recognized that there were many similarities between consumer electronics, industrial electronics, and telecommunications designs. Since the various designs often contained similar components, such as Analog-to-Digital Converters (ADCs), LCDs, or EEPROMs, Phillips determined that they could simplify system design and maximize hardware efficiency by creating a communication bus that could be used to transfer data between any device connected to the bus. This allowed designers to use devices from multiple manufacturers, or use one device in several designs. The specification also solved interfacing problems by creating a scheme that is now held as an industry standard, meaning any I^{2}C device could communicate with any other I^{2}C device without having to change the hardware or firmware of either device.

The I^{2}C specification defines the bus as a two-wire, bidirectional communications scheme. One line carries the Serial Data (SDA), and one line carries the Serial Clock (SCL). Each I^{2}C device has its unique address, either 7-bits or 10-bits in length. An I^{2}C device can operate as either a bus master, bus slave, or both, depending on the device and application. The specification defines the data transfer rates as follows:

- Standard mode – transfer rates up to 100 Kbits/s
- Fast mode – transfer rates up to 400 Kbits/s
- Fast mode Plus – transfer rates up to 1 Mbit/s
- High-Speed mode – transfer rates up to 3.4 Mbits/s

Microchip's I^{2}C module implements master and slave hardware that supports Standard mode, Fast mode, and Fast mode Plus. Throughout this technical brief, the I^{2}C Specification will be referred to so that the reader gains an understanding of both the I^{2}C module and the I^{2}C Specification.

I^{2}C Protocol Terminology

To properly understand the language used in the specification, the following is a list of terms commonly used by the specification and found throughout this technical brief:

Table 1-1. I^{2}C Bus Terminology

Term	Description
Transmitter	The device that shifts data out onto the bus
Receiver	The device that shifts data in from the bus
Master	The device that initiates data transfer, generates the clock signal, and terminates transmission
Slave	The device addressed by the master
Multi-master	A bus with more than one device that can initiate data transfers
Arbitration	Procedure that ensures that only one master at a time controls the bus
Synchronization	Procedure to synchronize the clocks of two or more devices on the bus
Idle	Both the SDA and SCK lines are in a logic high state; no activity on the bus
Active	Any time in which one or more master devices are controlling the bus

..........continued

Term	Description
Address Slave	Slave device that has received a matching address and is actively being clocked by a master
Matching Address	Address byte clocked into a slave that matches the value stored in one of the I2CxADR registers
Write Request	Master sends an address byte with the $R/\overline{W}$ bit cleared; master intends to write data to the slave
Read Request	Master sends an address byte with the $R/\overline{W}$ bit set; master intends to receive data from a slave
Clock Stretching	When a device holds the clock line low to pause communications
Bus Collision	Condition in which the expected data on SDA is a logic high, but is sampled as a logic low
Bus Time-Out	Condition in which a device on the bus is holding the bus for longer than a specified period

2. I^{2}C Module Overview

The I^{2}C module provides a synchronous serial interface between the microcontroller and other I^{2}C compatible devices using the two-wire bus. The two signal connections, Serial Clock (SCL) and Serial Data (SDA), are bidirectional open-drain lines, each requiring pull-up resistors to the supply voltage. Pulling the line to ground is considered a logic ‘0’, while allowing the line to float is considered a logic ‘1’.

Important: The voltage levels of the logic ‘0’ (low) and logic ‘1’ (high) are not fixed and are dependent on the bus supply voltage.

According to the I^{2}C specification, a logic input low level is up to 30% of V_{DD} ($V_{IL} \leq 0.3\ V_{DD}$), while a logic input high level is 70% to 100% of V_{DD} ($V_{IH} \geq 0.7\ V_{DD}$). Some legacy devices may use the previously defined fixed levels of V_{IL} = 1.5V and V_{IH} = 3.0V, but all new I^{2}C compatible devices require the use of the 30/70% specification.

All I^{2}C communication is performed using an 8-bit data word and a 1-bit Acknowledge sequence. All transactions on the bus are initiated and terminated by the master device. Depending on the direction of the data being transferred, there are four main operations performed by the I^{2}C module:

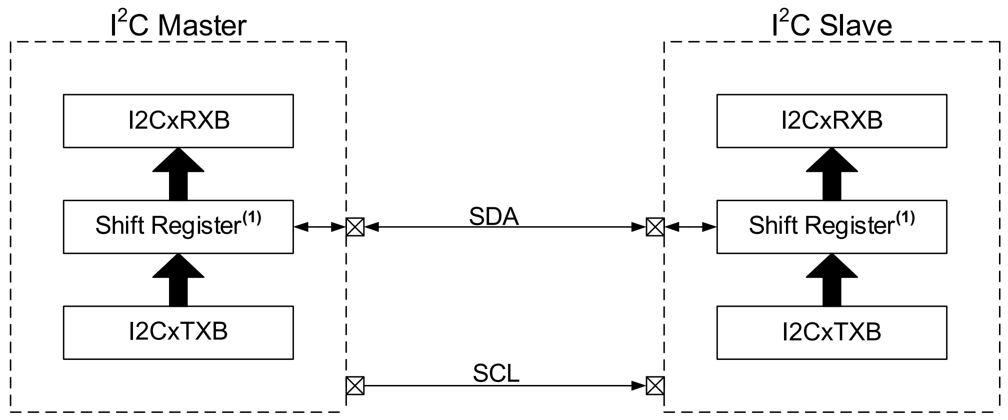
- Master Transmit – master is transmitting data to a slave
- Master Receive – master is receiving data from a slave
- Slave Transmit – slave is transmitting data to a master
- Slave Receive – slave is receiving data from a master

The I^{2}C interface allows for a multi-master bus, meaning that there can be several master devices present on one bus. A master can select a slave device by transmitting a unique address on the bus. When the address matches a slave’s address, the slave responds with an Acknowledge sequence ($\overline{\text{ACK}}$), and communication between the master and that slave can commence. All other devices connected to the bus must ignore any transactions not intended for them.

2.1 Dedicated Transmit/Receive Buffers

The I^{2}C module has two dedicated data buffers, one for transmission (I2CxTXB) and one for reception (I2CxRXB) (see Figure 2-1).

Figure 2-1. I²C Transmit (I2CxTXB) and Receive (I2CxRXB) Buffers



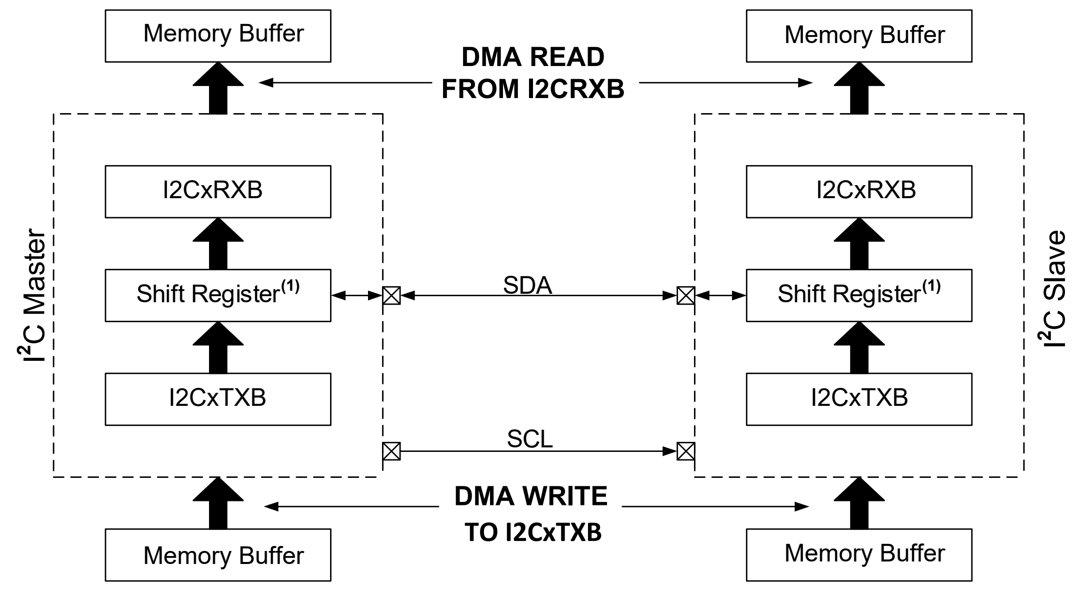
Note:

1. The Shift register is not accessible to the user.

The transmit buffer, I2CxTXB, is loaded from the software or from the Direct Memory Access (DMA) module (see Figure 2-2). When transmission begins, data loaded into the I2CxTXB is shifted into the transmit Shift register and out onto the bus. The I2CxTXB can be loaded when the Transmit Buffer Empty Status (TXBE) bit is set (TXBE = `1`), indicating that the buffer is empty. When the buffer is empty and the I2CxCNT register is not equal to '`0`', the I2C Transmit Interrupt Flag (I2CxTXIF) bit is set, and the generic I2C Interrupt Flag (I2CxIF) will also be set if the I2C Transmit Interrupt Enable (I2CxTXIE) bit is set. Loading a new byte of data into the I2CxTXB clears the I2CxTXIF flag bit. If user software attempts to load the I2CxTXB while it is full, the Transmit Write Error Status (TXWE) bit is set, a NACK is generated, and the new data is ignored. If the TXWE bit is set, the software must clear this bit before attempting to load the buffer again.

The receive buffer, I2CxRXB, holds one byte of data that is shifted in from the receive Shift register. User software or the DMA can read the byte through the I2CxRXB register (see Figure 2-2). When a new byte is received, the Receive Buffer Full Status (RXBF) bit and the I2C Receive Interrupt Flag (I2CxRXIF) bit are set, and the generic I2CxIF will also be set if the I2C Receive Interrupt Enable (I2CxRXIE) bit is set. Reading I2CxRXB clears both RXBF and I2CxRXIF. If the buffer is read while empty (RXBF = `0`), the Receive Read Error Status (RXRE) bit is set, and the module generates a NACK. User software must clear the RXRE bit to resume normal operation.

Figure 2-2. I²C Transmit/Receive Buffers with DMA



Note:

1. The Shift register is not accessible to the user.

Both transmit and receive buffers can be cleared by setting the Clear Buffer (CLRBF) bit of the I2CxSTAT1 register, which also clears both the I2CxTXIF and I2CxRXIF Interrupt Flags.

2.2 Address Buffers and Registers

The I²C module has two Address Buffer registers, I2CxADB0 and I2CxADB1, that can be used as receive buffers in Slave mode (see Table 2-2). This differs from the MSSP module in that the MSSP module only used the SSPBUF to receive or transmit an address (or data). The address buffers are enabled via the Address Buffer Disable (ABD) bit. When ABD is cleared, the address buffers are enabled; when the ABD is set, the address buffers are disabled.

When the ABD bit is cleared and in 7-bit Addressing mode, I2CxADB0 is loaded with the matching received slave address and the R/$\overline{W}$ bit, and I2CxADB1 is unused. In 10-bit Addressing mode, I2CxADB0 is loaded with the lower eight bits of the matching received address, and I2CxADB1 is loaded with the upper eight bits of the matching address, including the R/$\overline{W}$ bit.

When the ABD bit is set and in 7-bit Addressing mode, I2CxRXB is loaded with the matching received slave address and R/$\overline{W}$ bit, and neither I2CxADB0/1 are used. In 10-bit Addressing mode, I2CxRXB is loaded with both the high and low matching address bytes. In this case, user software must read I2CxRXB before the receive Shift register will load I2CxRXB with the matching lower address byte. I2CxADB0/1 are also unused in 10-bit Addressing mode.

Table 2-1. Address Buffer Direction for Slave Modes

Modes	MODE[2:0]	I2CADB0	I2CADB1
Slave (7-bit)	`000`	RX	Unused
	`001`		
Slave (10-bit)	`010`	RX	RX
	`011`		

The I^{2}C module has four additional Slave mode Address registers, I2CxADR0/1/2/3. These registers can hold up to four 7-bit slave addresses or up to two 10-bit slave addresses. The I2CxADR0/1/2/3 registers can also be used to form mask registers (see Table 2-2).

In 7-bit Addressing mode, all four registers can be used to store individual slave addresses. The first byte received in a transmission (address byte) is independently compared to each of the values in the I2CxADR registers. The LSb of the received address byte (R/$\overline{W}$ bit) is not used in determining a match. If address masking is desired, I2CxADR1 holds the value that is used to mask the address loaded in I2CxADR0, and I2CxADR3 holds the value used to mask I2CxADR2.

Table 2-2. Slave Mode Address Registers

Modes	MODE[2:0]	I2CADR0	I2CADR1	I2CADR2	I2CADR3
7-bit	`000`	7-bit address	7-bit address	7-bit address	7-bit address
7-bit w/ masking	`001`	7-bit address	7-bit mask for I2CxADR0	7-bit address	7-bit mask for I2CxADR2
10-bit	`010`	Lower address byte	Upper address byte	Lower address byte	Upper address byte
10-bit w/ masking	`011`	Lower address byte	Upper address byte	Lower address mask	Upper address mask

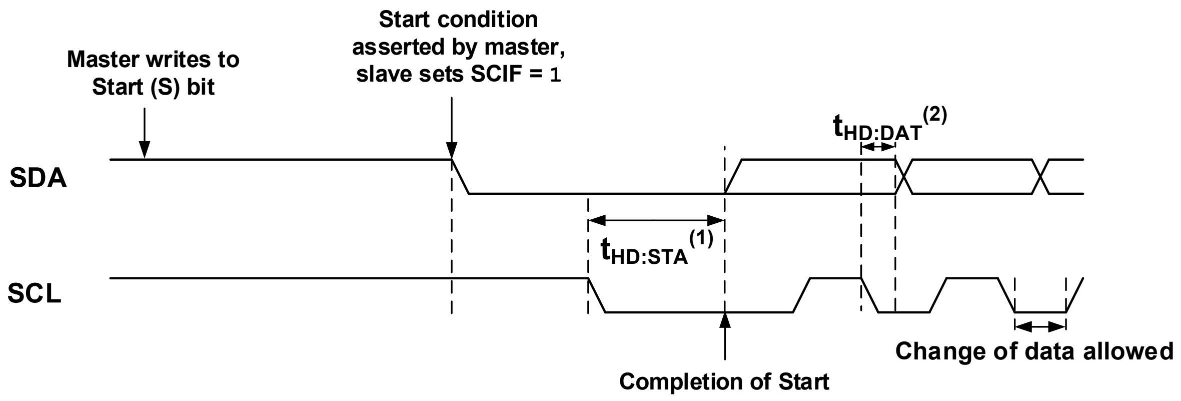
In 10-bit Addressing mode, I2CxADR0 and I2CxADR1 are combined to hold the 10-bit slave address, and I2CxADR2 and I2CxADR3 are combined to hold a second 10-bit slave address. I2CxADR0 and I2CxADR2 hold the lower eight bits of the 10-bit addresses, while I2CxADR1 and I2CxADR3 hold the upper two bits of the 10-bit addresses, the R/$\overline{W}$ bits, and the five digit '`11110`' code assigned to the five Most Significant bits of the address high bytes. If address masking is desired, I2CxADR0 and I2CxADR1 are combined to form the 10-bit slave address, and I2CxADR2 and I2CxADR3 are combined to form the 10-bit address mask. Address masking may be used to ignore the five-bit address code loaded into the Most Significant bits of the upper address byte.

Important: The '`11110`' code is specified by the I^{2}C Specification, but is not strictly enforced by Microchip. The user must ensure that the correct bit values are loaded into the 10-bit high address byte. If a master device has included that specific code in the address it intends to transmit, the slave must also include those bits in the slave address.

2.3 Start Condition

The I^{2}C Specification defines a Start condition as the transition of the SDA line from an Idle state (logic high level) to an Active state (logic low level) while the SCL line is idle (see Figure 2-3). The Start condition is always initiated by the master and signifies the beginning of a transmission.

Figure 2-3. Start Condition


Master writes to Start (S) bit
Start condition asserted by master, slave sets SCIF = 1
$t_{HD:DAT}$(2)
SDA
$t_{HD:STA}$(1)
SCL
Change of data allowed
Completion of Start

Note:

1. See device data sheet for Start condition hold time parameters.
2. SDA hold time are configured via the SDAHT[1:0] bits.

According to the I^{2}C Specification, a bus collision cannot occur on a Start condition. The Bus Free (BFRE) bit is used by the module hardware to indicate the status of the bus. The Bus Free Time Selection (BFRET[1:0]) bits define the amount of I^{2}C clock cycles that the master hardware must detect while the bus is idle before the BFRE bit is asserted. When the BFRE bit is set (BFRE = `1`), the bus is considered in an Idle state, and a master device may issue a Start condition. If there is more than one master on the bus (Multi-Master mode), and both attempt to issue a Start condition simultaneously, a bus collision will occur during the addressing phase of communication.

When the master asserts a Start condition, the slave hardware sets the Start Condition Interrupt Flag (SCIF), and if the Start Condition Interrupt Enable (SCIE) bit is set, the generic I2C Interrupt Flag (I2CxIF) bit is also set. The slave software must clear SCIF in order to clear I2CxIF and continue normal operation.

2.4 Repeated Start/Restart Condition

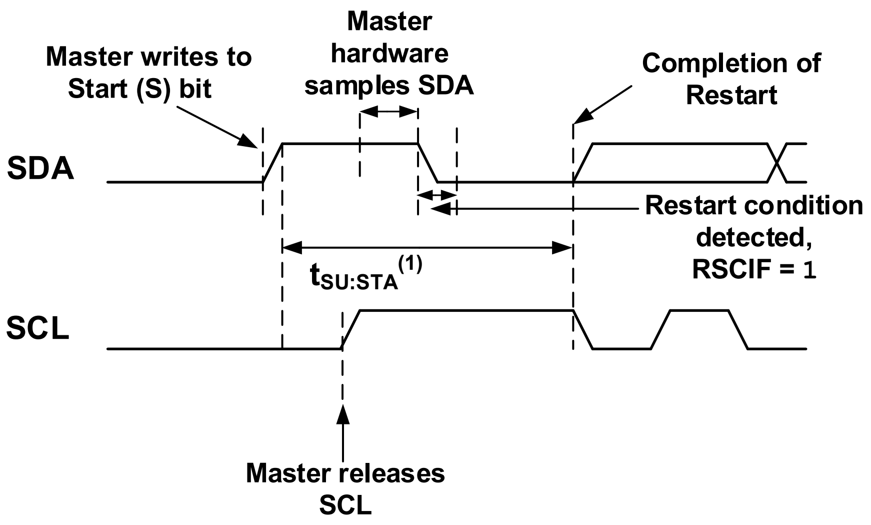
A Repeated Start or Restart condition is identical to a Start condition. A master device can issue a Restart condition instead of a Stop condition if it intends to hold the bus after completing the current data transfer. A Restart condition has the same effect on the slave as a Start condition would, resetting all slave logic and preparing it to receive an address. The Restart condition is always initiated by the master.

A Restart condition occurs when the Restart Enable (RSEN) bit is set, I2CxCNT is '`0`', and either the master hardware or the user software sets the Start bit.

When the Start bit is set, the master hardware releases SDA (SDA floats high) for $T_{SCL}/2$. Then, the hardware releases SCL for $T_{SCL}/2$, and samples SDA. If SDA is sampled low (while SCL is high), as bus collision has occurred, setting the Bus Collision Detect Interrupt Flag (BCLIF) bit and placing the master hardware in the Idle state. If SDA is sampled high (while SCL is also high), the master hardware issues a

Start condition. Once a Restart condition is detected on the bus, the slave hardware will set the Restart Condition Interrupt Flag (RSCIF) bit. See Figure 2-4 for more details.

Figure 2-4. Restart Condition


Master writes to Start (S) bit
Master hardware samples SDA
Completion of Restart
SDA
Restart condition detected, RSCIF = 1
$t_{SU:STA}$(1)
SCL
Master releases SCL

Note:

1. See device data sheet for Restart condition setup times.

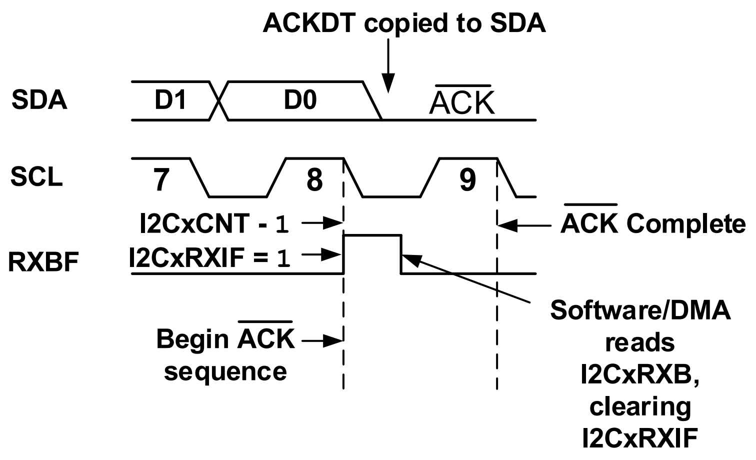
2.5 Acknowledge ($\overline{\text{ACK}}$)/Not Acknowledge (NACK) Sequence

The I^{2}C Specification defines the Acknowledge sequence ($\overline{\text{ACK}}$) as a logic low state of the SDA line during the 9th SCL pulse for any successfully transferred byte. During this time, the transmitter must relinquish control of the SDA line to the receiver. The receiver must then pull the SDA line low and keep it low during the high period of the 9th SCL pulse.

When the receiver has successfully received a matching address byte or a valid data byte, it will pull the SDA line low during the 9th SCL pulse, which indicates to the transmitter that is has successfully received the information and is ready for the next byte.

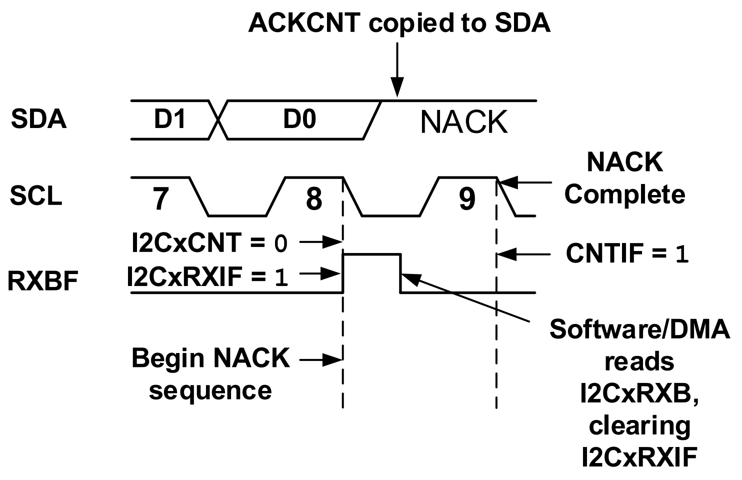
An Acknowledge sequence is enabled automatically by hardware following an address/data byte reception. On the 8th falling edge of SCL, the content of either the Acknowledge Data (ACKDT) bit or the Acknowledge End of Count (ACKCNT) bit is copied to the SDA output. When I2CxCNT is not '0', the value of the ACKDT bit is copied to SDA. When I2CxCNT is '0', the value of the ACKCNT bit is copied to SDA. In most applications, the value of ACKDT should be '0', which represents an $\overline{\text{ACK}}$ (see Figure 2-5).

Figure 2-5. Slave $\overline{\text{ACK}}$ (I2CxCNT != 0)



If the SDA line remains at a high logic level during the 9th SCL pulse, this is defined as a Not Acknowledge (NACK) sequence (see Figure 2-6).

Figure 2-6. Slave NACK (I2CxCNT = 0)



A NACK is generated when any of the following conditions occurs:

- No slave device is present on the bus that owns the transmitted address
- The receiver is busy and is not ready for communication
- The receiver gets data or commands that it cannot understand
- The receiver cannot receive any more data
- A master-receiver has received the requested data and is ready to terminate transmission
- An I^2C error condition has occurred
- The I2CxCNT register has reached a ‘0’ value and ACKCNT is set (ACKNT = 1).

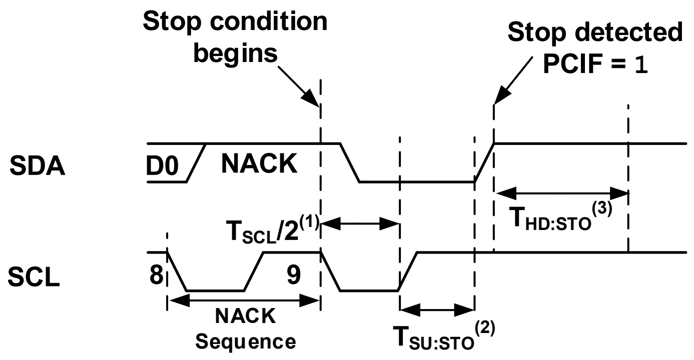
The master device can then decide to either generate a Stop condition to terminate the transfer, or issue a Restart condition to hold the bus and begin a new transfer.

2.6 Stop Condition

The I^{2}C Specification defines a Stop condition as the transition of the SDA line from an Active state to an Idle state while the SCL line is idle. The master will issue a Stop condition once the transactions have been completed and it is ready to release the control of the bus, or if a bus time-out occurs. If the SDA line transitions low and then high again while the SCL line is high, the Stop condition is ignored and a Start/Restart condition will be detected by the receiver (see Figure 2-7).

Important: At least one SCL low period must appear before a Stop condition is valid.

Figure 2-7. Stop Condition



Note:

1. At least one SCL low time must appear before a Stop is valid.
2. See device data sheet for Stop condition setup times.
3. See device data sheet for Stop condition hold times.

After the $\overline{\text{ACK}}$/NACK sequence of the final byte of the transmitted/received I^{2}C packet, hardware pulls the SCL line low for $T_{SCL}/2$, and then releases SCL. Hardware samples SCL to ensure a logic high level. SDA is then released, and the transition of SDA from low to high while SCL is high causes the Stop Condition Interrupt Flag (PCIF) bit to be set.

2.7 SDA and SCL Pins

The Serial Data (SDA) and Serial Clock (SCL) pins are used by the I^{2}C module to control the I^{2}C bus lines. Unlike previous versions of the MSSP, the SCL and SDA pins must be configured in open-drain operation by setting the appropriate bits in that port’s Open-Drain Control register (ODCONx). Also, unlike previous versions of the MSSP, the port’s Direction Control register (TRISx) must have the SDA and SCL pins configured as outputs by clearing the appropriate TRIS bits. Finally, slew rate control, internal pull-up resistor selection, and input threshold levels for each pin can be configured using the I^{2}C Pad Control (RxyI2C) register.

Important: Previous MSSP modules have recommended using external pull-up resistors rather than the internal weak pull-ups. However, the internal weak pull-ups may now be used, depending on the bus transmission frequency and capacitance. The internal pull-ups can be configured in the RxyI2C register.

The SDA and SCL pins are typically assigned to two I/O port pins, and must be enabled using the Peripheral Pin Select (PPS) module. The PPS module has two dedicated I^2C input registers: I2CxSCLPPS, which defines the SCL input pin, and I2CxDATPPS, which defines the SDA input pin. SDA and SCL outputs are also defined via the PPS module. The outputs use the RxyPPS registers to define the signal the pin will output.

Important: Both the SDA and SCL inputs and outputs must be defined, and must be assigned to the same pins. For example, if the SDA pin is assigned to pin RC4, both the I2CxDATPPS and the RC4PPS registers must be mapped to pin RC4. If both input and output signals are not mapped to the same pin, or if one of the signals are not mapped at all, no communication will take place.

The PPS module also allows for alternate pins to be used instead of the default pin locations. If an alternate pin location is desired, simply load the appropriate PPS registers with the new location.

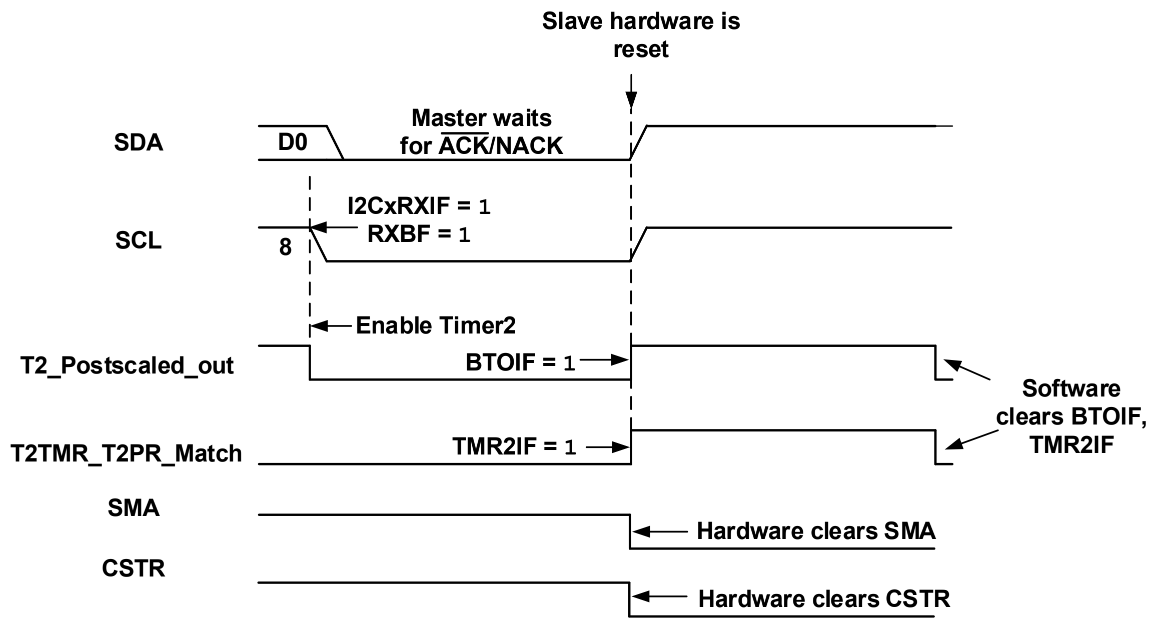
Important: Some devices allow the digital peripherals to be relocated to any pin, while other devices only allow the digital peripherals to be moved to pins within two I/O ports. Refer to the device data sheet PPS chapter for more details. Also, if the desired I^2C pin locations are moved from the default pins, the new location may not be configured for I^2C levels, and would require the Open-Drain (ODCONx), Slew Rate (SLRCONx), and Input Threshold Control (INLVLx) registers to be configured for I^2C.

2.8 Bus Time-Out

SMBus and PMBus protocols require a bus watchdog to prevent a stalled device from hanging the bus indefinitely. The I^2C module provides a bus time-out feature that can be used to reset the module if one of the bus devices is taking too long to respond. The I2C Bus Time-Out (I2CxBTO) register is used to select the time-out source for the module. When the time-out source expires, the I2CxBTO register notifies the module hardware and resets the module.

If the module is configured as a slave and a bus time-out event occurs while the slave is active (Slave Mode Active bit (SMA) = `1`), the SMA and Slave Clock Stretching (CSTR) bits are cleared, the module is reset, and the Bus Time-Out Interrupt Flag (BTOIF) bit is set (see Figure 2-8). If the Bus Time-Out Interrupt Enable (BTOIE) bit is set when the BTOIF bit becomes set, the generic I2C Error Interrupt Flag (I2CxEIF) is also set.

Figure 2-8. Slave Receive Bus Time-Out Event Example



2.9 Data Byte Count

The data byte count is the number of bytes in a complete I^2C packet. The I2CxCNT register is used to specify the length, in bytes, of the complete transaction. The value loaded into I2CxCNT will decrement each time a data byte is transmitted or received by the module.

When a byte transfer causes the I2CxCNT register to decrement to '0', the Byte Count Interrupt Flag (CNTIF) bit of the I2CxPIR register is set, and if the Byte Count Interrupt Enable (CNTIE) bit is also set, the general purpose I2C Interrupt Flag (I2CxIF) bit is set. The I2CxIF is a read-only bit and can only be cleared by clearing all Enabled Interrupt Flag bits in the I2CxPIR register.

The I2CxCNT register can be read at any time, but it is recommended that a double read is performed to ensure a valid read.

The I2CxCNT register can be written to, but care is required to prevent register corruption. If the I2CxCNT register is written to during the 8th falling SCL edge during reception, or during the 9th falling SCL edge during transmission, the register value may be corrupted. In Slave mode, I2CxCNT can be safely written to any time the slave is stretching the clock (CSTR = 1), or after a Stop condition has been received. If the I^2C packet is longer than 255 bytes, the I2CxCNT value can be updated mid-message to prevent the count from reaching '0'; however, the preventative measures listed above must be followed.

The I2CxCNT value can be automatically loaded when the Auto-Load I2C Count Register Enable (ACNT) bit of the I2CxCON2 register is set. When ACNT is set, the data byte following the address byte is loaded into I2CxCNT, and the value of the Acknowledge Data (ACKDT) bit is used for the $\overline{\text{ACK}}$ response.

When in either Slave-Read or Master-Write mode and the I2CxCNT value is not '0', the value of the ACKDT bit is used for the $\overline{\text{ACK}}$ response. When I2CCNT = 0, the value of the Acknowledge End of Count (ACKCNT) bit is used for the $\overline{\text{ACK}}$ response.

2.10 Clock Stretching

Clock stretching occurs when a slave device holds the SCL line low to pause bus communication. The slave may stretch the clock to allow more time to handle data or prepare a response for the master device. Clock stretching can be enabled by clearing the Clock Stretching Disable (CSD) bit of the I2CxCON1 register, and is only available in Multi-Master and Slave modes.

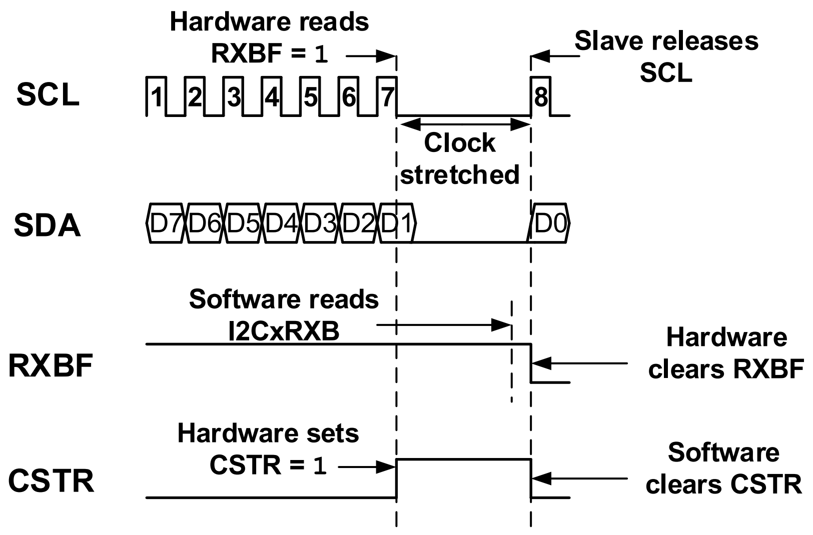
When clock stretching is enabled, the Slave Clock Stretching (CSTR) bit can be used to determine if the clock is currently being stretched, and to release the SCL line when the slave is ready to continue communication. This bit cannot be set by software, but can be cleared by software after the byte has been processed.

Clock Stretching for Buffer Operations

When enabled, clock stretching is forced during buffer read/write operations. This allows the slave time to either load the I2CxTXB with transmit data, or read from the I2CxRXB to clear the buffer.

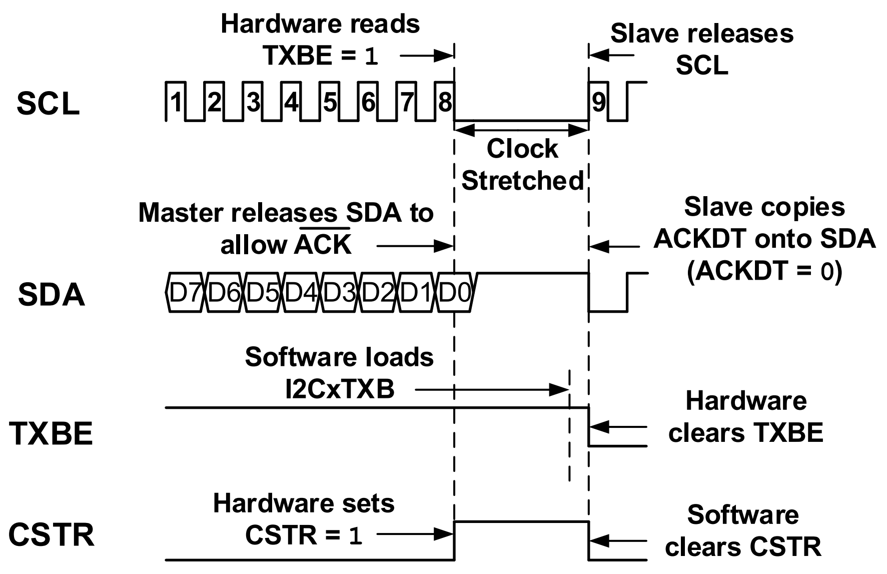
In Slave Receive mode, clock stretching helps prevent receive data overflows. When the first seven bits of the new byte are received into the receive Shift register while there is still data in I2CxRXB (RXBF = `1`), slave hardware automatically stretches the clock and sets CSTR. When the slave has read the byte, the slave software must clear the CSTR bit to release the clock and continue communication (see Figure 2-9).

Figure 2-9. Receive Buffer Clock Stretching



In Slave Transmit mode, clock stretching helps prevent transmit underflows. When the Transmit Buffer Empty Status (TXBE) bit is set and the I2CxCNT register is not equal to ‘`0`’, the slave hardware stretches the clock and sets CSTR upon the eighth falling SCL edge. This allows the slave software time to load the I2CxTXB with new data. Once the I2CxTXB is loaded, the slave software releases the clock by clearing CSTR (see Figure 2-10).

Figure 2-10. Transmit Buffer Clock Stretching



Clock Stretching for Other Slave Operations

In addition to the clock stretching features listed above, the module provides three Interrupt and Hold Enable features. When clock stretching is enabled (CSD = 0), the Interrupt and Hold Enable feature provides an interrupt and stretches the clock to allow time for address recognition, data processing, or $\overline{\text{ACK}}$/NACK response.

The Address Interrupt and Hold Enable feature will generate and interrupt and stretch the SCL signal when an incoming address match occurs. The feature is enabled via the Address Interrupt and Hold Enable (ADRIE) bit of the I2CxPIE register. When enabled, the Slave Clock Stretching (CSTR) bit and the Address Interrupt Flag (ADRIF) bit are set by hardware and the SCL line is stretched following the 8th falling SCL edge of a received matching address. Once the slave has completed processing the address, the software determines whether to send an $\overline{\text{ACK}}$ or a NACK back to the master. Slave software must clear both the ADRIF and CSTR bits to continue communication.

The Data Write Interrupt and Hold Enable feature provides an interrupt and stretches the SCL input after a received data byte. This feature is enabled by setting the Data Write Interrupt and Hold Enable (WRIE) bit of the I2CxPIE register. When enabled, hardware sets both the CSTR and the Data Write Interrupt Flag (WRIF) bits and stretches the SCL line after the 8th falling edge of SCL. Once completed, the slave software must clear both the CSTR and WRIF bits to resume communication.

The Acknowledge Status Time Interrupt and Hold Enable feature generates an interrupt and stretches the SCL signal after the Acknowledge phase of a transmission. The feature is enabled by setting the Acknowledge Status Time Interrupt and Hold Enable (ACKTIE) bit. When enabled, the module hardware sets both the CSTR and Acknowledge Status Time Interrupt Flag (ACKTIF) bit and stretches the clock after the 9th falling edge of SCL. This feature enables clock stretching for all address, read, or write transactions. Once completed, slave software must clear both the CSTR and ACKTIF bits to continue communication.

3. Interrupts

The stand-alone I^{2}C module contains additional interrupt features designed to assist with communication functions. In addition to the MSSP module's Start/Restart Condition (SCIF), Stop Condition (PCIF), Bus Collision (BCLIF), and transmit, receive, and acknowledge (SSPIF) interrupts, the stand-alone I^{2}C module adds an Address Match (ADRIF), Transmit Buffer Empty (TXBE), Receive Buffer Full (RXBF), Bus Time-Out (BTOIF), Data Byte Count (CNTIF), Acknowledge Status Time (ACKTIF), and Not Acknowledge Detect (NACKIF).

The stand-alone I^{2}C module incorporates a new register, the I2C Interrupt Flag register (I2CxPIR), which handles several I^{2}C related interrupts. Additionally, when any of the Flag bits in I2CxPIR becomes set and the associated Interrupt Enable bits in the I2CxPIE register are set, the generic I2C Interrupt Flag (I2CxIF) is also set. If the matching Interrupt Enable bit is set, an interrupt is generated whenever the Interrupt Flag bit is set. If the appropriate Interrupt Enable bit is cleared, the Interrupt flag will still be set when the Interrupt condition occurs, however, no interrupt will be triggered.

Important: The generic I2CxIF bit is read-only and is only cleared by hardware when all the bits in the I2CxPIR register are cleared.

The I2CxPIR contains the following Interrupt Flag bits:

- CNTIF: Byte Count Interrupt Flag
- ACKTIF: Acknowledge Status Time Interrupt Flag
- WRIF: Data Write Interrupt Flag
- ADRIF: Address Interrupt Flag
- PCIF: Stop Condition Interrupt Flag
- RSCIF: Restart Condition Interrupt Flag
- SCIF: Start Condition Interrupt Flag

The CNTIF becomes set (CNTIF = `1`) when the I2CxCNT register value reaches '`0`', indicating that all bytes in the data frame have been transmitted or received. CNTIF is set after the 9th falling edge of SCL when the I2CxCNT = `0`.

The ACKTIF becomes set (ACKTIF = `1`) after the 9th falling edge of SCL for any byte when the device is addressed as a slave in any I^{2}C Slave mode or I^{2}C Multi-Master mode whenever an $\overline{\text{ACK}}$ is detected.

The WRIF becomes set (WRIF = `1`) after the 8th falling edge of SCL when the module receives a data byte. This bit is only active in any I^{2}C Slave mode or I^{2}C Multi-Master mode. Once the data byte is received, the WRIF is set, as is the Receive Buffer Full (RXBF) Status bit, the I^{2}C Receive Interrupt Flag (I2CxRXIF) bit, and if the Data Write Interrupt and Hold Enable (WRIE) bit is set, the generic I2CxIF bit is also set. The WRIF bit is read/write and must be cleared by the user software, while the RXBF, I2CxRXIF, and I2CxIF are read-only, and are cleared by reading the I2CxRXB.

The ADRIF becomes set on the 8th falling edge of SCL after the module has received either a matching 7-bit address byte or the matching upper or lower bytes of a 10-bit address. This bit is only active in Slave mode or Multi-Master mode. Upon receiving a matching address byte, the ADRIF bit is set, and if the Address Interrupt and Hold Enable (ADRIE) bit is set, the generic I2CxIF bit is set.

The PCIF is set whenever a Stop condition is detected on the bus.

The RSCIF is set upon the detection of a Restart condition.

The SCIF is set upon the detection of a Start condition.

In addition to the I2CxPIR register, the stand-alone module incorporates the I2C Error register (I2CxERR). The I2CxERR register contains three Interrupt Flag bits that are used to detect bus errors. These bits are read/write and must be cleared by the user software. The I2CxERR register also includes the Enable bits for these three functions.

The I2CxERR register contains the following Interrupt Flag bits:

- BTOIF: Bus Time-Out Interrupt Flag
- BCLIF: Bus Collision Detect Interrupt Flag
- NACKIF: NACK Detect Interrupt Flag

The BTOIF is set when a bus time-out event occurs. The bus time-out time frame is controlled by the I2C Bus Time-Out (I2CxBTO) register. If a bus time-out event occurs and the module is configured as a slave and is active (SMA = `1`), BTOIF is set, the SMA and CSTR bits are cleared, and the module is reset. When the BTOIF becomes set and the Bus Time-Out Interrupt Enable (BTOIE) bit is set, the generic I2C Error Interrupt Flag (I2CxEIF) bit is also set.

Important: The I2CxEIF bit is read-only, and is cleared by hardware when all Error Interrupt Flag bits in the I2CxERR register are cleared.

The BCLIF is set whenever a bus collision is detected. A bus collision occurs any time the SDA input is sampled low while both the SDA and SCL outputs are high. When a bus collision event occurs, the BCLIF becomes set, and if the Bus Collision Detect Interrupt Enable (BCLIE) bit is set, the generic I2CxEIF bit is also set.

The NACKIF is set when either the master or slave is active (SMA = `1` || MMA = `1`) and a NACK is detected on the bus. A NACK response occurs on the 9th SCL pulse when the SDA line is released high. When the module is in Master mode, a NACK can be issued when the master has finished receiving data from the slave, or in the event it did not receive a byte. In Slave mode, the slave issues a NACK when it does not receive a matching address, or did not receive the last data byte. A NACK can also be automatically sent if any of the following bits are set, which will set the NACKIF, and if the NACK Detect Interrupt Enable (NACKIE) bit is set, the generic I2CxEIF is also set:

- TXWE: Transmit Write Error Status bit
- RXRE: Receive Read Error Status bit
- TXU: Transmit Underflow Status bit
- RXO: Receive Overflow Status bit

4. I^{2}C Slave Mode Operation

To begin any I^{2}C communication, the master hardware checks to ensure that the bus is in an Idle state, which means both the SCL and SDA lines are floating high. The master hardware monitors the Bus Free (BFRE) bit to be set, indicating the bus is idle. The master then transmits a Start condition, followed by the address of the slave it intends to communicate with. The slave address can be either 7-bit or 10-bit, depending on the application design.

In 7-bit Addressing mode, the Least Significant bit (LSb) acts as the Read/Write (R/$\overline{W}$) bit, while in 10-bit Addressing mode, the LSb of the address high byte is considered the R/$\overline{W}$ bit. When the R/$\overline{W}$ bit is set, the master intends to read from the slave. If the R/$\overline{W}$ bit is cleared, the master intends to write to the slave. If the addressed slave device exists on the bus, it must respond with an Acknowledge ($\overline{ACK}$) sequence.

The master then continues to either receive data from the slave, write data to the slave, or a combination of both. Data is always transmitted starting with the Most Significant bit (MSb) first. When the master intends to halt further transmission, it transmits a Stop condition, signaling to the slave that communication is to be terminated, or a Restart condition, signaling the bus that the current master wishes to hold the bus to communicate with the same or other slaves.

Slave mode is selected by configuring the MODE[2:0] bits of the I2CxCON0 register. There are four Slave mode configurations and two Multi-Master modes:

- I^{2}C Slave mode with 7-bit address
- I^{2}C Slave mode with 7-bit addresses with masking
- I^{2}C Slave mode with two 10-bit Address registers
- I^{2}C Slave mode with one 10-bit address with masking
- I^{2}C Multi-Master – Master mode with 7-bit address and Slave mode with two 7-bit addresses with masking
- I^{2}C Multi-Master – Master mode with 7-bit address and Slave mode with four 7-bit addresses.

The master device generates the SCL pulses, as well as the Start, Restart, and Stop conditions. Transmission always begins with a Start condition, and can end with either a Stop condition or a Restart condition. When the master has completed all transactions, and is ready to release the bus, it will generate a Stop condition. If the master wishes to stop communicating with one slave, but wants to hold the bus to address another slave, it issues a Restart condition. Control of the bus can only be asserted when the Bus Free (BFRE) bit of the I2CxSTAT0 register is set.

The slave waits until hardware detects a Start condition on the bus. Once a Start condition is detected, the slave waits for the incoming address information to be loaded into the receive Shift register. The address is then compared to the addresses loaded into the I2CxADR registers. If an address match is detected, the slave hardware transfers the address into either the I2CxADB0/1 register or the I2CxRXB register, depending on the state of the ABD bit. If there are no address matches, there is no response from the slave.

When the ABD bit is cleared, the Address Buffer registers, I2CxADB0 and I2CxADB1, are active and used to hold the incoming matching address.

In 7-bit Addressing mode, the incoming address is compared to the addresses stored in the I2CxADR registers. If an address matches, the matching address is loaded into the I2CxADB0 register. In 7-bit Address mode, I2CxADB1 is unused. In 10-bit Addressing mode, the first incoming address byte is the high byte of the 10-bit address, and is compared to the values in the I2CxADR1 and I2CxADR3 registers, and the matching upper address byte is loaded into I2CxADB1. If no match is detected, the module becomes idle. The second incoming byte is the lower byte of the 10-bit address, and is compared to the values in I2CxADR0 and I2CxADR2 registers, and the matching lower address byte is loaded into I2CxADB0.

When the ABD bit is set, the address buffers are inactive. In this case, the matching address is loaded into I2CxRXB. The slave software must read I2CxRXB to continue communication.

Once an address match occurs, the slave either continues to receive data from or transmits data to the master device.

5. Slave Mode Configuration and Operation

The steps listed below can be used to configure the I^{2}C module for Slave mode operation.

5.1 Initialization

To begin I^{2}C Slave mode communication, the following registers must be properly configured during initialization (see code example below).

I^{2}C Initialization Example

```
void I2C1_Initialize(void)
{
    I2C1ADR0 = 0x30;        // Load address registers with slave addresses
    I2C1ADR1 = 0x80;
    I2C1ADR2 = 0x50;
    I2C1ADR3 = 0xC0;
    I2C1CON0 = 0x00;        // CSTR Enable clocking; MODE four 7-bit address;
    I2C1CON1 = 0x80;        // CSD Clock Stretching enabled;
                            // ACKDT Acknowledge; ACKCNT Not Acknowledge;
    I2C1CON2 = 0x00;        // ABD enabled; ACNT disabled;
                            // SDAHT 300 ns hold time;
    I2C1CNT = 0x00;         // Clear count register
    PIR2bits.I2C1RXIF=0;    // Clear all I2C interrupt flags
    PIR3bits.I2C1TXIF=0;
    PIR3bits.I2C1EIF=0;
    I2C1ERRbits.NACKIF=0;
    PIR3bits.I2C1IF=0;
    I2C1PIRbits.PCIF=0;
    I2C1PIRbits.ADRIF=0;
    PIE2bits.I2C1RXIE=1;    // Enable I2C RX interrupt
    PIE3bits.I2C1TXIE=1;    // Enable I2C TX interrupt
    PIE3bits.I2C1EIE=1;     // Enable I2C error interrupt
    I2C1ERRbits.NACKIE=1;   // Enable I2C error interrupt for NACK
    PIE3bits.I2C1IE=1;      // Enable I2C interrupt
    I2C1PIEbits.PCIE=1;     // Enable I2C interrupt for stop condition
    I2C1PIEbits.ADRIE=1;    // Enable I2C interrupt for I2C address match
                            // condition
    I2C1PIR = 0;            // Clear all the error flags
    I2C1ERR = 0;
    I2C1CON0bits.EN = 1;    // Enable I2C module
}
void PIN_MANAGER_Initialize(void)
{
    LATC = 0x00;
    TRISC = 0xE7;           // RC3, RC4 outputs
    ANSELC = 0xE7;          // Disable analog input buffers
    WPUC = 0x00;            // RxyI2C value used instead
    ODCONC = 0x18;          // RC3, RC4 open-drain
    RC3PPS = 0x21;          // RC3->I2C1:SCL1;
    RC4PPS = 0x22;          // RC4->I2C1:SDA1;
    I2C1SCLPPS = 0x13;      // RC3->I2C1:SCL1;
    I2C1SDAPPS = 0x14;      // RC4->I2C1:SDA1;
}
```

I2CxCON0: The I2CxCON0 register contains the Module Enable (EN) bit and the Mode Select (MODE[2:0]) bits. The MODE[2:0] bits are used to select the communications mode, and the EN bit enables the Slave state machine hardware. MODE[2:0] bit settings should not be changed while the EN bit is set (module is enabled).

I2CxCON1: The I2CxCON1 register contains the Acknowledge End of Count (ACKCNT), Acknowledge Data (ACKDT), and Clock Stretching Disable (CSD) bits.

The ACKCNT bit reflects the value transmitted after the I2CxCNT register has reached ‘0’, signaling the end of the packet. When ACKCNT is cleared, the module will issue an $\overline{\text{ACK}}$; when set, the module issues a NACK. ACKCNT can be modified during run time, but should only be changed before the I2CxCNT reaches '`0`' and before an Acknowledge sequence is issued. If there are errors in either the I2CxERR or I2CxSTAT registers, the slave hardware automatically overrides this bit setting and generates a NACK.

The ACKDT bit reflects the value transmitted after a matching address is received, or after a byte is received while I2CxCNT is not '`0`'. When ACKDT is cleared, an $\overline{\text{ACK}}$ is issued; when ACKDT is set, a NACK is issued. The ACKDT bit value can be modified during run time, but should only be changed before an Acknowledge sequence is issued. If there are errors in either the I2CxERR or I2CxSTAT registers, the slave hardware automatically overrides this bit setting and generates a NACK.

The CSD bit enables/disables the slave's ability to stretch the SCL signal.

I2CxCON2: The I2CxCON2 register holds the Auto-Load I2C Count Register Enable (ACNT), General Call Address Enable (GCEN), Address Buffer Disable (ABD), and SDA Hold Time Selection (SDAHT[1:0]).

The ACNT bit enables/disables the auto-loading of the I2CxCNT register. Auto-loading of I2CxCNT can be useful when a slave device does not know the size of the data packet, or when the master needs to change the size of the packet to transmit.

When ACNT is set, the first byte following the matching address is used as the value that is loaded into the I2CxCNT register. For example, if the master device intends to transmit three data bytes to a slave, the byte following the address would have a value of '3', and would be loaded into the master device's I2CxCNT register during transmission.

When the byte is received by the slave device, it is loaded into the slave's I2CxCNT register. Of course, this assumes that both the master and the slave have the I2CxCNT register feature available, and both devices have ACNT set.

The ABD bit enables/disables the use of the dedicated Address Buffer registers. In Slave mode, a matching address received by the slave can be loaded into the I2CxADB0/1 registers. When ABD = `1`, the I2CxADB0/1 registers are ignored, and the matching slave address is loaded into the I2CxRXB receive buffer by hardware. User software must read I2CxRXB to continue communication.

When ABD = `0`, the matching received address data stored in I2CxADB0/1.

The SDAHT[1:0] bits are used to configure the amount of time the SDA line is held valid after the falling edge of SCL. The SDAHT[1:0] bits should be configured based on the bus capacitance; buses with larger capacitance may need longer hold times to ensure valid data.

I2CxBTO: The I2CxBTO register selects the timing source used for the Bus Time-Out feature. The current time-out sources are either a CLC or a Timer, and those modules must also be configured during initialization. The time-out source should be configured such that a device does not stall the bus for too long, but does not interfere with timely data processing or clock stretching.

I2CxERR: The I2CxERR register contains the Bus Time-Out Interrupt Enable (BTOIE), the Bus Collision Detect Interrupt Enable (BCLIE), and NACK Detect Interrupt Enable (NACKIE) bits. If these interrupts are not needed by the application, this register does not need to be explicitly initialized.

I2CxCNT: The I2CxCNT register is loaded with the number of data bytes present in a I^2C packet. The I2CxCNT can be loaded during initialization or run time directly, but it is recommended to write to this register only if the module is idle or during clock stretching. Writing at any other time may corrupt the register. I2CxCNT can also be automatically loaded during run time when the ACNT bit of I2CxCON2 is set. In this case, the first byte following the address byte(s) is loaded into I2CxCNT by module hardware. The I2CxCNT value should only include the number of data bytes in the packet, and not any address bytes.

I2CxPIE: The I2CxPIE register contains several I^{2}C specific Interrupt Enable bits. Initialization is only required if one or more of the following interrupts are necessary:

- Data Byte Count Interrupts
- Acknowledge Interrupt and Hold Enable
- Data Write Interrupt and Hold Enable
- Address Interrupt and Hold Enable
- Stop Condition Interrupts
- Restart Condition Interrupts
- Start Condition Interrupts

RxyI2C: The RxyI2C register controls the I^{2}C specific I/O pads. Most PIC® devices dedicate one or two pairs of I/O pins to the I^{2}C module. The RxyI2C register is used to configure the pin slew rate, input threshold level, and internal pull-up configurations.

The SLEW bit controls the slew rate. When SLEW is set, I^{2}C specific slew rate is enabled, which overrides the standard pin slew rate limiting, and the SLRCONx bit associated with the pin is ignored. When SLEW is cleared, the module uses the standard pad slew rate, which is enabled/disabled via the SLRCONx bit associated with the pin. Lower bus speeds may not need any slew rate limiting, while buses with higher speeds may need slew rate limiting.

The TH bits control the I^{2}C input threshold level. These bits can be configured to SMBus 3.0, SMBus 2.0, I^{2}C specific, or standard I/O input threshold levels to meet the specific protocol requirements. When either the SMBus 3.0, SMBus 2.0, or I^{2}C specific levels are selected, the INLVLx bit associated with the pin is ignored. If standard I/O threshold levels are selected, the INLVLx bit associated with the pin can be configured for either ST or TTL logic levels.

The PU bits are used to select the internal pull-up drive strength. The PU bits can be configured to increase the current drive of the pull-up, making the internal pull-ups strong enough to be used instead of external pull-up resistors. If external pull-ups are to be used in the application, the PU bits can be configured for standard weak pull-ups, which can be enabled/disabled via the WPUx bit associated with the pin.

TRISx Registers: The TRISx registers provide I/O direction support to PORT pins. When using the I^{2}C module, the TRISx bits associated with the SDA and SCL pins must be initialized clear (TRISxy = `0`). All previous I^{2}C module designs required the TRISx bits to be set. During run time, direction control is handled by the module hardware.

ODCONx: The I^{2}C module uses an open-drain circuit configuration. The ODCONx bits associated with the SCL and SDA pins must be configured for open-drain (ODCONxy = `1`).

I2C PPS Registers: The Peripheral Pin Select (PPS) feature allows digital signals to be moved from their default pin location to another location. To enable a digital peripheral's input and/or output signals, the appropriate PPS registers must be configured. When using the I^{2}C module, both the input PPS and output PPS registers must be configured due to the bidirectional nature of the I^{2}C bus. Both the input and output PPS registers for each I^{2}C signal must be routed to the same pin. In other words, if the I2CxSCLPPS input register is mapped to pin RC3, the RC3PPS register must also be mapped to pin RC3.

Input configuration is handled by the I2CxSCLPPS and I2CxSDAPPS registers. These registers must be mapped to the desired pins to enable the pin input drivers. Output configuration is handled by the RxyPPS registers. The 'xy' in the register name is a placeholder for the actual port and pin number. For example, if the SDA line is mapped to port pin RC4, the correct register name is RC4PPS. The PPS output registers must also be mapped to the desired pins to enable the pin output driver.

The PPS feature allows the I^{2}C pins to be moved from their default locations, but additional steps must be considered. The default I^{2}C pins use the RxyI2C register to define the slew rate, pull-up configuration, and input threshold levels. If the default pin locations are not used, additional registers, such as INLVLx, WPUx, and SLRCONx must also be configured.

6. Slave Mode Transmission

6.1 Slave Mode Transmission (7-Bit Addressing)

The following section describes the sequence of events when using the I^2C in Slave mode transmission.

1. Master device issues a Start condition. Once the Start is detected, the slave hardware sets the Start Condition Interrupt Flag (SCIF).
2. Master transmits the 7-bit slave address with $R/\overline{W}$ set.
3. The received address is compared to the values in the I2CxADR registers.
   If the slave 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 mode with masking, the received address is masked with the value in the I2CxADR1 and I2CxADR3 and compared to the value of I2CxADR0 and I2CxADR2.

   If a match occurs, the Slave Mode Active (SMA) bit is set, the $R/\overline{W}$ bit information (bit '`0`' of the matching address) is transferred to the Read Information (R) bit of the I2CxSTAT0 register, the Data (D) bit of the I2CxSTAT0 register is cleared, and the Address Interrupt Flag (ADRIF) bit is set. If the ABD is cleared, the matching address is copied into the I2CxADB0 register. If ABD is set, the matching address is copied into the receive buffer, I2CxRXB, setting the Receive Buffer Full (RXBF) and I^2C Receive Interrupt Flag (I2CxRXIF) bits. I2CRXIF is a read-only bit, and must be cleared by setting the Clear Buffer (CLRBF) bit or reading I2CxRXB.

   If there is not an address match, the module goes idle.
4. When ADRIF is set, if the Address Interrupt and Hold Enable (ADRIE) bit is set and the Clock Stretching Disable (CSD) bit is cleared, slave hardware sets the Slave Clock Stretching (CSTR) bit, and the generic I2CxIF Flag bit. This allows time for the slave to read either I2CxADB0 or I2CxRXB and selectively $\overline{ACK}$/NACK based on the received address. When the slave has finished processing the address, the software clears CSTR and ADRIF, which releases the clock and clears the I2CxIF Flag bit.
5. If TXBE = `1`, I2CxCNT has a non-zero value, and I2CxTXIF = `1`, the slave hardware sets CSTR, and waits for the software to load I2CxTXB. I2CxTXB must be loaded to clear I2CxTXIF and release SCL. I2CxCNT is decremented after the byte is transferred to the Shift register.
6. Master transmits the 9th clock pulse, and the slave hardware transfers the value of the ACKDT bit onto the SDA line. If there are pending errors, such as a receive overflow, the slave hardware automatically generates a NACK sequence, NACKIF is set, and the module goes idle. The slave hardware checks I2CxCNT. If I2CxCNT = `0`, the Byte Count Interrupt Flag (CNTIF) is set.
7. If the slave issued an $\overline{ACK}$ and the I2CxCNT is non-zero, the master transmits eight clock pulses, and the slave begins to shift out the data byte Most Significant Bit (MSb) first. If I2CxCNT = `0`, CNTIF is set.
8. Master receives data byte from slave, then transmits an $\overline{ACK}$/NACK sequence. The slave hardware copies the $\overline{ACK}$ value into the Acknowledge Status (ACKSTAT) bit, and sets ACKTIF. If the Acknowledge Time Interrupt and Hold Enable (ACKTIE) bit is set, the slave hardware sets the CSTR and I2CxIF bits. When the slave is ready, the software clears the CSTR and ACKTIF bits, which releases SCL and clears I2CxIF.
9. Steps 13-16 continue until the master has received all the requested data. Once all data is received, the master transmits a NACK condition followed by either a Stop condition or Restart condition. The slave hardware sets NACKIF and PCIF, and clears SMA.

6.2 Slave Mode Transmission (10-Bit Addressing)

The following section describes the sequence of events when using the I^{2}C in Slave mode transmission.

1. Master device issues a Start condition. Once the Start is detected, the slave hardware sets the Start Condition Interrupt Flag (SCIF).
2. Master transmits the 10-bit high address byte with $R/\overline{W}$ cleared.
3. The received address is compared to the values in the I2CxADR registers.
If the slave is configured in 10-bit Addressing mode (no masking), the received high address byte is compared to the values in the I2CxADR1 and I2CxADR3 registers. In 10-bit Addressing mode with masking, the received address is masked with the value of I2CxADR3, and then compared to the value of I2CxADR1. If a match occurs, the $R/\overline{W}$ bit information (bit '0' of the matching address) is transferred to the Read Information (R) bit of the I2CxSTAT0 register, the Data (D) bit of the I2CxSTAT0 register is cleared, and the Address Interrupt Flag (ADRIF) bit is set. If ABD is cleared, the matching address is copied into the I2CxADB1 register. If ABD is set, the matching address is copied into the receive buffer, I2CxRXB, setting the Receive Buffer Full (RXBF) and I2C Receive Interrupt Flag (I2CxRXIF) bits. I2CxRXIF is a read-only bit, and must be cleared by setting the Clear Buffer (CLRBF) bit or reading I2CxRXB. If there is not an address match, the module goes idle.
4. When ADRIF is set, if the Address Interrupt and Hold Enable (ADRIE) bit is set and the Clock Stretching Disable (CSD) bit is cleared, the slave hardware sets the Slave Clock Stretching (CSTR) bit and the generic I2CxIF Flag bit. This allows time for the slave to read either I2CxADB1 or I2CxRXB and selectively $\overline{ACK}$/NACK based on the received address. When the slave has finished processing the address, the software clears CSTR and ADRIF, which releases the clock and clears the I2CxIF bit.
5. Master transmits the 9th clock pulse, and the slave hardware transfers the value of the ACKDT bit onto the SDA line. If there are pending errors, such as a receive overflow, the slave hardware automatically generates a NACK sequence and the module goes idle.
6. The master transmits the lower address byte.
If the slave is configured in 10-bit Addressing mode (no masking), the received low address byte is compared to the values in the I2CxADR0 and I2CxADR2 registers. In 10-bit Addressing mode with masking, the received address is masked with the value of I2CxADR2, and then compared to the value of I2CxADR0. If a match occurs, the Slave Mode Active (SMA) bit is set, the $R/\overline{W}$ bit information is transferred to the R bit of the I2CxSTAT0 register, the D bit of the I2CxSTAT0 register is cleared, and the ADRIF bit is set. If ABD is cleared, the matching address is copied into the I2CxADB0 register. If ABD is set, the matching address is copied into the receive buffer, I2CxRXB, setting the RXBF and I2CxRXIF bits. I2CxRXIF is a read-only bit, and must be cleared by setting the CLRBF bit or reading I2CxRXB.
7. When ADRIF is set, if the ADRIE bit is set and the CSD bit is cleared, the slave hardware sets the CSTR bit and the generic I2CxIF bit. This allows time for the slave to read either I2CxADB0 or I2CxRXB and selectively $\overline{ACK}$/NACK based on the received address. When the slave has finished processing the address, the software clears CSTR and ADRIF, which releases the clock and clears the I2CxIF bit.
8. The master transmits the 9th clock pulse, and the slave hardware transfers the value of the ACKDT bit onto the SDA line. If there are pending errors, such as a receive overflow, the slave hardware automatically generates a NACK condition and the module goes idle.
9. After the 9th falling edge of the clock, the Acknowledge Status Time Interrupt and Hold (ACKTIF) bit is set. If the Acknowledge Time Interrupt and Hold Enable (ACKTIE) bit is set, the slave hardware

sets the CSTR and I2CxIF bits. When the slave is ready, the software clears the CSTR and ACKTIF bits, which releases SCL and clears I2CxIF.

10. The master issues a Restart condition, and once the slave detects the Restart, the hardware sets the Restart Condition Interrupt Flag (RSCIF), and if the Restart Condition Interrupt Enable (RSCIE) bit is set, I2CxIF is also set.
11. The master then transmits the high address byte again, but with the R/$\overline{W}$ bit set. If SMA = `1` and the high address matches, the R/$\overline{W}$ bit is copied into the R bit of I2CxSTAT0, and the D bit of I2CxSTAT0 is cleared. If ABD is cleared, the matching address is stored in I2CxADB1. If ABD is set, the matching address is copied into I2CxRXB, setting the RXBF and I2CxRXIF bits. I2CxRXIF is a read-only bit, and must be cleared by setting the CLRBF bit or reading I2CxRXB.
12. When ADRIF is set, if the ADRIE bit is set and the CSD bit is cleared, the slave hardware sets the CSTR bit and the generic I2CxIF bit. This allows time for the slave to read either I2CxADB1 or I2CxRXB and selectively $\overline{ACK}$/NACK based on the received address. When the slave has finished processing the address, the software clears CSTR and ADRIF, which releases the clock and clears the I2CxIF bit.
13. If TXBE = `1`, I2CxCNT has a non-zero value, and I2CxTXIF = `1`, the slave hardware sets CSTR, and waits for software to load I2CxTXB. I2CxTXB must be loaded to clear I2CxTXIF and release SCL. I2CxCNT is decremented after the byte is transferred to the Shift register.
14. Master transmits the 9th clock pulse, and the slave hardware copies the value of ACKDT onto SDA.
15. If the slave issued a NACK, the slave hardware sets the NACK Detect Interrupt Flag (NACKIF) and goes idle. If the slave issued an $\overline{ACK}$ and the I2CxCNT is non-zero, the master transmits eight clock pulses, and the slave begins to shift out the data byte Most Significant Bit (MSb) first. If I2CxCNT = `0`, CNTIF is set.
16. Master receives data byte from slave, then transmits an $\overline{ACK}$/NACK sequence. The slave hardware copies the $\overline{ACK}$ value into the Acknowledge Status (ACKSTAT) bit, and sets ACKTIF. If the ACKTIE bit is set, the slave hardware sets the CSTR and I2CxIF bits. When the slave is ready, the software clears the CSTR and ACKTIF bits, which releases SCL and clears I2CxIF.
17. Steps 13-16 continue until the master has received all the requested data. Once all data is received, the master transmits a NACK condition followed by either a Stop condition or Restart condition. The slave hardware sets NACKIF and PCIF, and clears SMA.

7. Slave Mode Reception

7.1 Slave Mode Reception (7-Bit Addressing)

The following section describes the sequence of events when using the I^{2}C in Slave mode reception.

1. Master device issues a Start condition. Once the Start is detected, the slave hardware sets the Start Condition Interrupt Flag (SCIF).
2. Master transmits the 7-bit slave address with R/$\overline{\text{W}}$ cleared.
3. The received address is compared to the values in the I2CxADR registers.
   If the slave 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 mode with masking, the received address is masked with the value of I2CxADR1 and I2CxADR3, and then compared to the value of I2CxADR0 and I2CxADR2.

   If a match occurs, the Slave Mode Active (SMA) bit is set, the R/$\overline{\text{W}}$ bit information (bit '`0`' of the matching address) is transferred to the Read Information (R) bit of the I2CxSTAT0 register, the Data (D) bit of the I2CxSTAT0 register is cleared, and the Address Interrupt Flag (ADRIF) bit is set. If the ABD is cleared, the matching address is copied into the I2CxADB0 register. If ABD is set, the matching address is copied into the receive buffer, I2CxRXB, setting the Receive Buffer Full (RXBF) and I2C Receive Interrupt Flag (I2CxRXIF) bits. I2CRXIF is a read-only bit, and must be cleared by setting the Clear Buffer (CLRBF) bit or reading I2CxRXB.

   If no address match occurs, the module stays idle.
4. When ADRIF is set, if the Address Interrupt and Hold Enable (ADRIE) bit is set and the Clock Stretching Disable (CSD) bit is cleared, slave hardware sets the Slave Clock Stretching (CSTR) bit and the generic I2CxIF Flag bit. This allows time for the slave to read either I2CxADB0 or I2CxRXB and selectively $\overline{\text{ACK}}$/NACK based on the received address. When the slave has finished processing the address, the software clears CSTR and ADRIF, which releases the clock and clears the I2CxIF bit.
5. Master transmits the 9th clock pulse, and the slave hardware transfers the value of the ACKDT bit onto the SDA line. If there are pending errors, such as a receive overflow, the slave hardware automatically generates a NACK condition, NACKIF is set, and the module goes idle. The slave hardware checks I2CxCNT. If I2CCNT = `0`, the Byte Count Interrupt Flag (CNTIF) is set.
6. Upon the falling edge of the 9th clock pulse, the Acknowledge Time Interrupt Flag (ACKTIF) bit is set, and if CSD = `0`, the slave hardware sets CSTR. This allows time for the slave to read the address from either I2CxADB0 or I2CxRXB. Once complete, the slave software clears CSTR to release the clock, and clears ACKTIF to continue communication.
7. If the slave transmits a NACK, the master hardware generates a Stop condition. The Stop Condition Interrupt Flag (PCIF) is set when the Stop condition is detected, and the slave goes idle. If the slave issued an $\overline{\text{ACK}}$, the master transmits the first 7 bits of the 8-bit data byte.
8. If previous data is still in the I2CxRXB register (RXBF = `1` and I2CRXIF = `1`) when the first seven bits of the new byte are received into the Shift register, the CSTR bit is set, and the clock is stretched after the 7th falling edge of SCL. This allows the slave software to read I2CxRXB, which clears the RXBF and I2CxRXIF bits, and prevents a receive buffer overflow. Once the RXBF bit is cleared, the software clears CSTR, which releases SCL.
9. Master hardware transmits the 8th bit of the current data byte into the slave's receive Shift register, then the slave hardware transfers the complete byte into I2CxRXB, sets the I2CxRXIF, the Data

Write Interrupt and Hold Flag (WRIF), the Data (D), and the RXBF bits. I2CxCNT is decremented by one. If the Data Write Interrupt and Hold is enabled (WRIE = 1), the hardware sets CSTR, allowing time for slave software to read I2CxRXB and decide the state of the ACKDT bit before clearing CSTR.

10. Master transmits the 9th clock pulse. If there are pending errors, such as a receive overflow, the slave hardware automatically generates a NACK sequence, NACKIF is set, and the module goes idle.
    If I2CxCNT is not ‘0’, the hardware transmits the value of the ACKDT bit as the $\overline{\text{ACK}}$ value to the master. It is up to the user to configure the ACKDT bit appropriately. In most cases, the ACKDT bit should be cleared, so that the master receives an $\overline{\text{ACK}}$ (logic low level on SDA during the 9th SCL pulse).

    If I2CxCNT = 0, the hardware transmits the value of the Acknowledge End of Count (ACKCNT) bit as the $\overline{\text{ACK}}$ value to the slave. It is up to the user to properly define the ACKCNT bit. In most cases, this bit should be set, indicating a NACK condition. When the master hardware detects the NACK on the bus, the master hardware will also generate a Stop condition. If the ACKCNT bit is cleared, an $\overline{\text{ACK}}$ will be issued, and the master hardware will not automatically generate the Stop condition.
11. Upon the falling edge of the 9th clock pulse, the ACKTIF bit is set, and if ACKTIE is also set, the generic I2CxIF bit is set, and if CSD is cleared, the slave hardware sets CSTR. This allows time for the slave to read the data from I2CxRXB. Once complete, the slave software clears CSTR to release the clock, and clears ACKTIF to continue communication.
12. Go to step 7 and continue until I2CxCNT = 0, or until the master issues a Stop condition.

7.2 Slave Mode Reception (10-Bit Addressing)

1. Master device issues a Start condition. Once the Start is detected, the slave hardware sets the Start Condition Interrupt Flag (SCIF).
2. Master transmits the 10-bit high address byte with R/$\overline{\text{W}}$ cleared.
3. The received address is compared to the values in the I2CxADR registers.
   If the slave is configured in 10-bit Addressing mode (no masking), the received high address byte is compared to the values in the I2CADR1 and I2CADR3 registers. In 10-bit Addressing mode with masking, the received address is masked with the value of I2CxADR3, and then compared to the value of I2CxADR1. If a match occurs, the R/$\overline{\text{W}}$ bit information (bit ‘0’ of the matching address) is transferred to the Read Information (R) bit of the I2CxSTAT0 register, the Data (D) bit of the I2CxSTAT0 register is cleared, and the Address Interrupt Flag (ADRIF) bit is set. If ABD is cleared, the matching address is copied into the I2CxADB1 register. If ABD is set, the matching address is copied into the receive buffer, I2CxRXB, setting the Receive Buffer Full (RXBF) and I2C Receive Interrupt Flag (I2CxRXIF) bits. I2CRXIF is a read-only bit, and must be cleared by setting the Clear Buffer (CLRBF) bit or reading I2CxRXB.

   If there is not an address match, the module goes idle.
4. When ADRIF is set, if the Address Interrupt and Hold Enable (ADRIE) bit is set and the Clock Stretching Disable (CSD) bit is cleared, the slave hardware sets the Slave Clock Stretching (CSTR) bit and the generic I2CxIF Flag bit. This allows time for the slave to read either I2CxADB1 or I2CxRXB and selectively $\overline{\text{ACK}}$/NACK based on the received address. When the slave has finished processing the address, the software clears CSTR and ADRIF, which releases the clock and clears the I2CxIF bit.

5. Master transmits the 9th clock pulse, and the slave hardware transfers the value of the ACKDT bit onto the SDA line. If there are pending errors, such as a receive overflow, the slave hardware automatically generates a NACK condition and the module goes idle.
6. The master transmits the lower address byte.
   If the slave is configured in 10-bit Addressing mode (no masking), the received low address byte is compared to the values in the I2CxADR0 and I2CxADR2 registers. In 10-bit Addressing mode with masking, the received address is masked with the value of I2CxADR1, and then compared to the value of I2CxADR0. If a match occurs, the Slave Mode Active (SMA) bit is set, the R/$\overline{\text{W}}$ bit information is transferred to the R bit of the I2CxSTAT0 register, the D bit of the I2CxSTAT0 register is cleared, and the ADRIF bit is set. If ABD is cleared, the matching address is copied into the I2CxADB0 register. If ABD is set, the matching address is copied into the receive buffer, I2CxRXB, setting the RXBF and I2CxRXIF bits. I2CxRXIF is a read-only bit, and must be cleared by setting the Clear Buffer (CLRBF) bit or reading I2CxRXB.
7. When ADRIF is set, if the ADRIE bit is set and the CSD bit is cleared, the slave hardware sets the CSTR bit and the generic I2CxIF Flag bit. This allows time for the slave to read either I2CxADB0 or I2CxRXB and selectively $\overline{\text{ACK}}$/NACK based on the received address. When the slave has finished processing the address, the software clears CSTR and ADRIF, which releases the clock and clears the I2CxIF bit.
8. The master transmits the 9th clock pulse, and the slave hardware transfers the value of the ACKDT bit onto the SDA line. If there are pending errors, such as a receive overflow, the slave hardware automatically generates a NACK condition and the module goes idle.
9. After the 9th falling edge of the clock, the Acknowledge Status Time Interrupt and Hold (ACKTIF) bit is set. If the Acknowledge Time Interrupt and Hold Enable (ACKTIE) bit is set, the slave hardware sets the CSTR and I2CxIF bits. When the slave is ready, the software clears the CSTR and ACKTIF bits, which releases SCL and clears I2CxIF.
10. If the slave transmits a NACK, the master hardware generates a Stop condition. The Stop Condition Interrupt Flag (PCIF) is set when the Stop condition is detected, and the slave goes idle. If the slave issued an $\overline{\text{ACK}}$, the master transmits the first 7 bits of the 8-bit data byte.
11. If previous data is still in the I2CxRXB register (RXBF = `1` and I2CxRXIF = `1`) when the first seven bits of the new byte are received into the Shift register, the CSTR bit is set, and the clock is stretched after the 7th falling edge of SCL. This allows the slave software to read I2CxRXB, which clears the RXBF and I2CxRXIF bits, and prevents a receive buffer overflow. Once the RXBF bit is cleared, the software releases SCL by clearing CSTR.
12. Master hardware transmits the 8th bit of the current data byte into the slave's receive Shift register, then the slave hardware transfers the complete byte into I2CxRXB, sets the I2CxRXIF, the Data Write Interrupt Flag (WRIF), the Data (D), and the RXBF bits. I2CxCNT is decremented by one. If the Data Write Interrupt and Hold is enabled (WRIE = `1`), the hardware sets CSTR, allowing time for the slave software to read I2CxRXB and decide the state of the ACKDT bit before clearing CSTR.
13. Master transmits the 9th clock pulse. If there are pending errors, such as a receive overflow, the slave hardware automatically generates a NACK condition, NACKIF is set, and the module goes idle.
    If I2CxCNT is not '`0`', the hardware transmits the value of the ACKDT bit as the $\overline{\text{ACK}}$ value to the master. It is up to the user to configure the ACKDT bit appropriately. In most cases, the ACKDT bit should be cleared, so that the master receives an $\overline{\text{ACK}}$ (logic low level on SDA during the 9th SCL pulse).

If I2CxCNT is '0', the hardware transmits the value of the Acknowledge End of Count (ACKCNT) bit as the $\overline{\text{ACK}}$ value to the slave. It is up to the user to properly define the ACKCNT bit. In most cases, this bit should be set, indicating a NACK condition. When the master hardware detects the NACK on the bus, the master hardware will also generate a Stop condition. If the ACKCNT bit is cleared, an $\overline{\text{ACK}}$ will be issued, and the master hardware will not automatically generate the Stop condition.

14. Upon the falling edge of the 9th clock pulse, the ACKTIF bit is set, and if ACKTIE is set, the generic I2CxIF bit is also set. If CSD = '0', the slave hardware sets CSTR. This allows time for the slave to read the data from I2CxRXB. Once complete, the slave software clears CSTR to release the clock, and clears ACKTIF to continue communication.
15. Go to step 10 and continue until I2CCNT = '0', or until the master issues a Stop condition.

8. External Pull-up Resistor Selection

The I^{2}C specification proposes two methods to determine the correct pull-up resistor size.

The first method calculates the maximum pull-up resistor size as a function of bus capacitance and rise time (see Equation 8-1). Bus capacitance is the total capacitance of the bus wires/traces, bus connection points, and bus pins, all of which must be considered when calculating the total bus capacitance. Rise time is the period in which the signal transitions from $V_{IL(MAX)}$ (0.3*V_{DD}) to $V_{IH(MIN)}$(0.7*V_{DD}). Rise time values are typically located in the device's data sheet.

Bus capacitance should be measured to achieve the most accurate pull-up values, but an estimated value, or the maximum allowable capacitance as defined by the I^{2}C specification, may also be used. The maximum allowable bus capacitance is specified to limit rise time decreases and allow operation at the rated frequency. The bus may operate at higher than allowable bus capacitance levels but at a lower frequency.

Equation 8-1. Maximum Pull-up Resistor Size

$$Rp(\text{max}) = \frac{t_{rise}}{0.8473 * C_{bus}}$$

$R_{p(max)}$ = Maximum pull-up value

t_{rise} = Maximum rise time

C_{bus} = Total bus capacitance

The second method calculates the minimum pull-up resistor size as a function of V_{DD} (see Equation 8-2). The supply voltage limits the minimum resistor value due to the specified minimum sink current of 3 mA for Standard mode (100 kHz) or Fast mode (400 kHz), or 20 mA for Fast mode Plus (1 MHz).

Equation 8-2. Minimum Pull-up Resistor Size

$$Rp(\text{min}) = \frac{V_{DD} - VOL(\text{max})}{I_{OL}}$$

$R_{p(min)}$ = Minimum pull-up value

V_{DD} = Supply voltage

$V_{OL(max)}$ = Maximum output low voltage

I_{OL} = Minimum sink current

9. Conclusion

This technical brief has covered the stand-alone I^{2}C module in Slave mode configuration. For more information, please visit www.microchip.com. For code examples, please visit www.microchip.com/mplab/mplab-xpress.

The Microchip Web Site

Microchip provides online support via our web site at http://www.microchip.com/. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Customer Change Notification Service

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at http://www.microchip.com/. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.

ISBN: 978-1-5224-4227-1

Quality Management System Certified by DNV

ISO/TS 16949
Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/
support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733

China - Beijing
Tel: 86-10-8569-7000

China - Chengdu
Tel: 86-28-8665-5511

China - Chongqing
Tel: 86-23-8980-9588

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

China - Nanjing
Tel: 86-25-8473-2460

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-3326-8000

China - Shenyang
Tel: 86-24-2334-2829

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

China - Wuhan
Tel: 86-27-5980-5300

China - Xian
Tel: 86-29-8833-7252

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444

India - New Delhi
Tel: 91-11-4160-8631

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880- 3770

Korea - Daegu
Tel: 82-53-744-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

Malaysia - Penang
Tel: 60-4-227-8870

Philippines - Manila
Tel: 63-2-634-9065

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

Thailand - Bangkok
Tel: 66-2-694-1351

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-67-3636

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-72884388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820