Using the I3C® Target Module on 8-bit PIC® Microcontrollers

TB3340

Introduction

Author: Max Prasad, Microchip Technology Inc.

The Improved Inter-Integrated Circuit (I3C®) is a medium-speed, utility, and control bus interface for connecting peripherals to an application processor in several mobile, IoT, and automotive applications. It builds upon the traditional Inter-Integrated Circuit (I^2C) interface where devices on a bus communicate in a Controller/Target environment using a two-wire interface (serial clock and data). Refer to the latest version of the I3C specification from the MIPI® Alliance website for more information.

The I3C module on the 8-bit PIC® microcontrollers supports Target functionality only with no controller capabilities. Refer to the appropriate data sheet for device specific information. The key features supported are listed below:

- I3C Target in Single Data Rate (SDR) mode
- Bus transfers up to 12.5 MHz at 0.95-3.63V bus voltage range
- Dedicated SCL/SDA pads with high-speed input buffer selection options
- Dynamic Address Assignment
- Direct/Broadcast Common Command Codes (CCC)
- Private I3C and Legacy I^2C Read/Write Transfers
- In-Band Interrupt (IBI) and Hot-Join (HJ)
- Target Reset Pattern
- Built-in Error Detection and Recovery
- Direct Memory Access (DMA) Support
- Extensive General and Error Interrupt Support

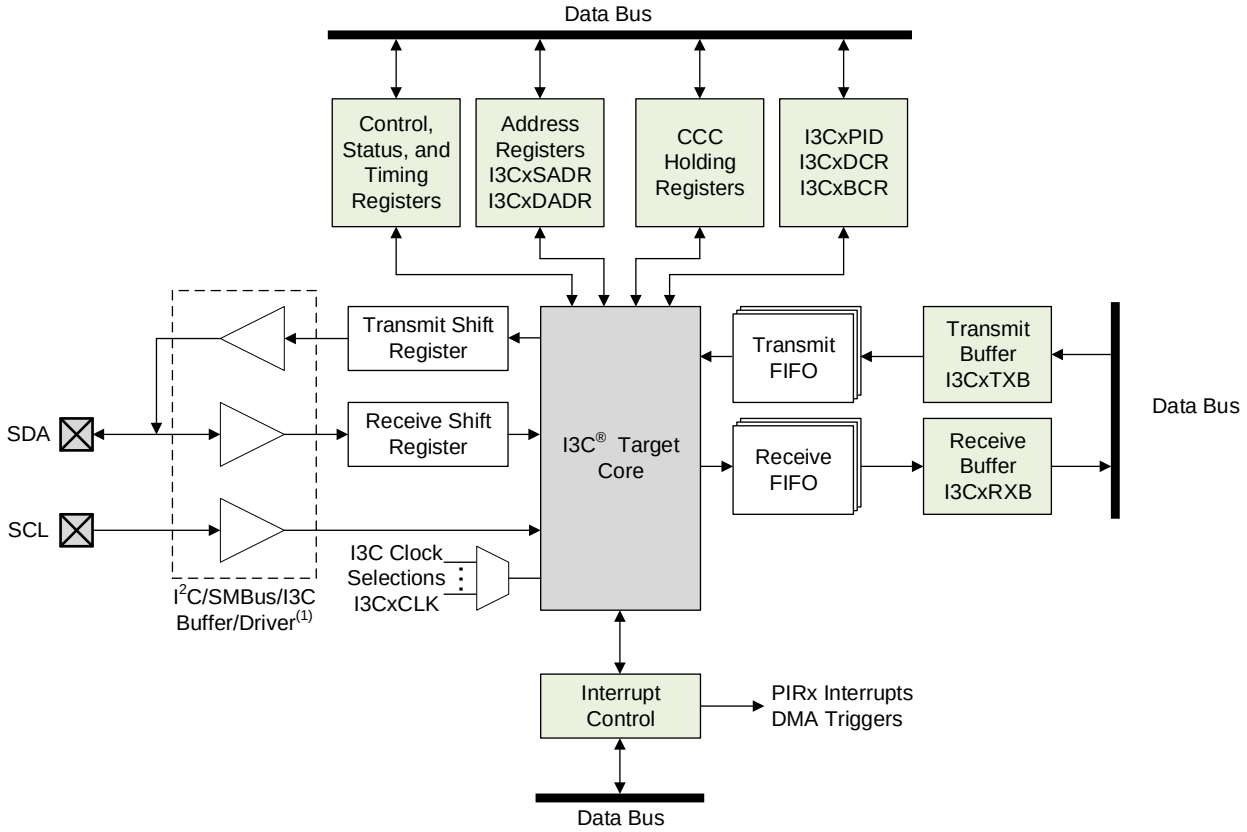
Table of Contents

1. Overview of the I3C Target Module

This section describes the implementation of the I3C Target module on the 8-bit PIC devices. Refer to the appropriate device data sheet for detailed information.

Figure 1-1 shows the block diagram of a typical I3C Target module.

Figure 1-1. I3C Block Diagram



Note 1: I^2C/SMBus/I3C Buffer/Driver selection can be changed based on the module's mode of operation.

1.1 Module Clock Selection

The I3C module requires a clock source for its internal state machine operation, which is selected using the I3CxCLK register. The user should choose a clock fast enough to keep up with the transactions on the bus for the desired packet length. In general, it is not recommended to operate the I3C module at a clock slower than the SCL bus frequency when the packet size is longer than the FIFO size. Doing so may result in transmission errors.

In addition, the I3C module clock also acts as a clock base for determining the bus idle, bus available, and bus timeout conditions on the bus.

1.2 SDA/SCL Pad Control

The I3C module can be connected to a pure or mixed I3C bus using dedicated SDA and SCL pins. Refer to the device data sheet for the location of I3C compatible SDA/SCL pads.

The I3C SDA and SCL pads are located in a separate Multi-Voltage I/O (MVIO) power domain and are powered by a separate V_{DDIOx} power pin. The V_{DDIOx} power pin should be connected to the same voltage as the desired I3C bus voltage on the SDA and SCL pins. This is because the device will drive the SDA line at the V_{DDIOx} voltage level in push-pull mode during I3C read transactions. The

individual SDA and SCL pins are designed to be fail-safe and high voltage input tolerant, meaning the device will not source or sink leakage current should the bus voltage become higher or lower than the V_{DDIOx} power pin.

The following selections must be made for the I3C module to operate properly:

- **MVIO Operating Mode:** The VDDIOxMD configuration bit must be set properly based on the VDDIOx power level. Select the Standard Operating Range for VDDIOxMD if V_{DDIOx} is in the 1.62V-3.63V range. Select the Low-voltage Operating Range for VDDIOxMD if V_{DDIOx} is in the 0.95V-1.62V range. Refer to the device data sheet for additional V_{DD} requirements when operating in the Low-voltage range.
- **Open-Drain Inputs:** The individual SDA and SCL pins must be configured as open-drain inputs using the Open-Drain Control (ODCONx) and Tri-State Control (TRISx) registers
- **Input Buffer Selection:** An appropriate input buffer must be selected for the I3C module using the I3CBUF bits in the RxyFEAT register. It must be noted that not all input buffers are operational at all voltage levels. It is recommended to use the I3C Fast ST Buffer when MVIO is operating in Standard range and the I3C Low-voltage Buffer when MVIO is operating in Low-voltage range.

⚠ CAUTION When the I3C module is enabled, the MVIO supply voltage on the corresponding V_{DDIOx} power pin should not exceed 3.63V, even though the individual I3C SDA and SCL pads on the MVIO domain are fail-safe and can support higher voltages.

1.3 Device Configuration

The device must be correctly configured to operate in an I3C environment. This section describes important selections that need to be properly configured, like device characteristics, speed limitations, and bus conditions.

1.3.1 Device Characteristics

There are three device characteristics that are required by the I3C protocol for a device to exist on an I3C bus.

1. **Bus Characteristics Register (BCR):** The BCR describes the role and capabilities of the I3C module. This can be configured using the I3CxBCR register. The I3C controller can query this value using the GETBCR Common Command Code (CCC). The BCR is also transmitted on the bus as part of ENTDAA-driven Dynamic Address Assignment procedure.
2. **Device Characteristics Register (DCR):** The DCR is a user-selectable value describing the compliant device type as per MIPI® specification. This can be configured using the I3CxDCR register. The I3C controller can query for this value using the GETDCR CCC. The DCR is also transmitted on the bus as part of ENTDAA-driven Dynamic Address Assignment procedure.
3. **Provisional ID (PID):** Every device on the I3C bus must have a unique 48-bit PID value, as defined by MIPI. This can be configured using the I3CxPID0 through I3CxPID5 registers. The I3C controller can query this value using the GETPID CCC. The PID is also transmitted on the bus as part of ENTDAA-driven Dynamic Address Assignment procedure.

1.3.2 Speed Limitations

Due to the high-speed nature of I3C transactions, the MIPI I3C Specification provides different ways for target devices on the bus to notify the controller of any delays in the user application. This can be done by setting the BCR0 bit in the I3CxBCR register. After the controller reads the I3CxBCR register during the Dynamic Address Assignment process, the controller can then read the I3CxMWS and I3CxMRS registers using the Get Maximum Data Speed (GETMXDS) CCC.

The Target can notify the controller of the following speed limitations:

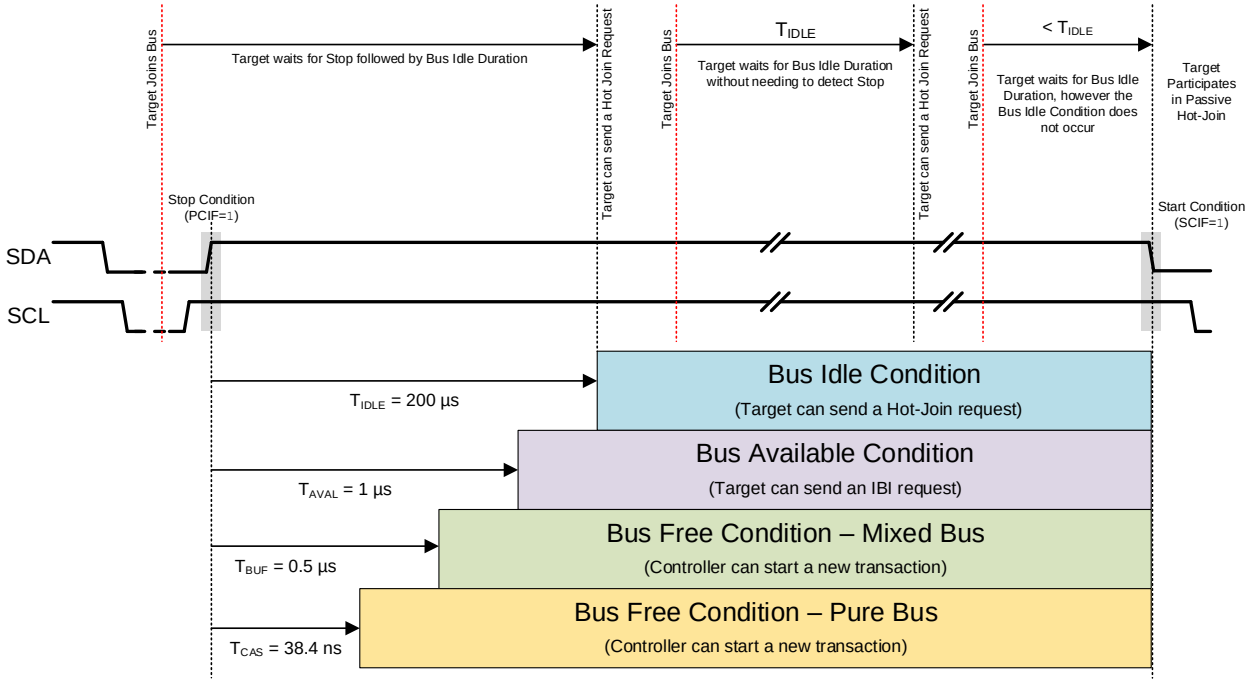
1. **Clock-to-Data Turnaround Time (T_{SCO}):** It is the time duration between reception of an SCL edge by the Target and the start of driving an SDA change. Refer to the device data sheet for exact T_{SCO} measurements. The user can notify the controller of this value through the I3CxMRS register.
2. **Maximum Read Turnaround Time (MRT):** Occasionally, it may happen that the controller requests specific data from the Target, and the application requires extra time to prepare the data before it can be read back by the controller. In such cases, the user can specify a maximum read turnaround time using the I3CxMRT register. When a non-zero value is specified in the I3CxMRT register, the MRS[6] bit in the I3CxMRS register can be used to notify the controller whether a Stop can be inserted between the request for data (by "writing" read index) and actual read of the data.
3. **Maximum Write/Read Speed (MWS/MRS):** While the I3C module can support transactions up to the maximum bus frequency of 12.5 MHz, the application may be slow to respond. In such cases the Target can notify the controller of the maximum write and read speeds using the I3CxMWS and I3CxMRS registers respectively.

1.3.3 Bus Conditions

The I3C bus is considered inactive after a Stop when there are no transactions happening on the bus for a certain time duration.

- A **Bus Free condition** happens after T_{BUF} time following a Stop on the bus, which is specified as 38.4 ns for a pure bus. The controller is allowed to begin a new transaction using a Start after this time. The BFREE bit is set in the I3C module when a Stop is detected on the bus, and the bit clears when a Start is detected.
- A **Bus Available condition** happens when a Bus Free condition is sustained for T_{AVAL} time, which is specified as 1 µs for a pure bus. The Target is allowed to request an In-Band Interrupt after this time. This time must be configured using the I3CxBAVL register for proper operation.
- A **Bus Idle condition** happens when a Bus Free condition is sustained for T_{IDLE} time, which is specified as 200 µs for a pure bus. The Target is allowed to request a Hot-Join after this time. This time must be configured using the I3CxBIDL register for proper operation.
- In addition, when there is no activity on the SCL line after a Start is detected (BFREE = 0), then an additional **Bus Time-out** timer can be configured using the I3CxBTO register to notify the Target of a stalled bus through the BTOIF bit. The feature can be enabled using the BTOEN bit.

Figure 1-2. I3C Bus Condition Timings



Note:
The diagram is not drawn to scale.

1.4 Dynamic Address Assignment

The Target device participates in Dynamic Address Assignment procedure depending on whether or not the device has been configured as Hot-Join capable while using the HJCAP bit.

When the Target is configured as Hot-Join capable (HJCAP = `1`), the I3C module responds to an Enter Dynamic Address Assignment (ENTDAA) CCC only after a Hot-Join is requested first. Refer to Hot-Join Transaction for more information.

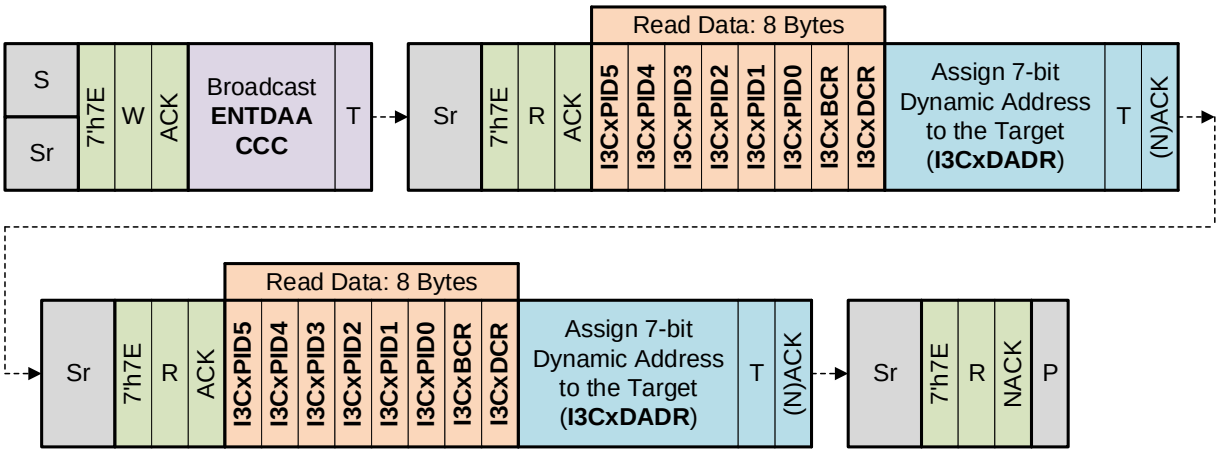
When the Target is configured as not Hot-Join capable (HJCAP = `0`), the I3C module responds to an ENTDAA CCC and participates in the Dynamic Address Assignment process if it does not have a Dynamic Address assigned. The entire Dynamic Address Assignment process happens automatically without any software intervention. Once the controller assigns the Target a Dynamic Address, it is stored in the I3CxDADR register, and the Dynamic Address Changed DACHIF flag is set as a notification.

Before a Dynamic Address is assigned, the module typically operates in I^2C mode (OPMD = `0b00`). In this mode, the module can perform Legacy I^2C Transactions and respond to certain Broadcast CCCs only. Once the Target device is assigned a Dynamic Address, the module begins operating in I3C SDR mode (OPMD = `0b01`). In this mode, the module can participate in all types of SDR transactions (except Legacy I^2C Transactions).

Important: If the Static Address SDR mode is enabled, the module can operate in I3C SDR mode (OPMD = `0b01`) even before a Dynamic Address is assigned.

The controller can clear the Dynamic Address stored in the Target device by sending the Reset Dynamic Address Assignment (RSTDAA) CCC. The controller can also assign a new Dynamic Address to the Target using the Set New Dynamic Address (SETNEWDA) CCC.

Figure 1-3. Dynamic Address Assignment Frame Format



1.5 Static Address SDR Mode

The Static Address SDR mode is a special mode that allows the module to operate in I3C SDR mode (OPMD = `0b01`) using the Static Address when a Dynamic Address is not assigned (or has been cleared). This special mode is activated by setting the SASDRMD bit during module initialization.

The module will continue to respond to ENTDAA CCC as usual for Dynamic Address Assignment. If a Dynamic Address is assigned in this mode, the module will participate in the I3C SDR transactions using both Static and Dynamic Addresses. The appropriate Static/Dynamic Address Match SADRIF/DADRIF bit will be set accordingly.

When an In-Band Interrupt is requested, the module will participate in address arbitration using the Dynamic Address first. If the module does not have a Dynamic Address assigned, Static Address is used for arbitration.

Tip: The I3C module on 8-bit PIC devices does not natively support Set All Addresses to Static Address (SETAASA) CCC. However, the Static Address SDR mode can be used with the Unsupported CCC UCCCIF flag to provide customized firmware support for SETAASA CCC.

1.6 Common Command Code (CCC)

Whether they are supported or not, the I3C module automatically participates in all CCC transactions. Refer to the appropriate device data sheet for a list of supported CCCs and associated registers.

When a CCC transaction happens on the bus, the CCC code is stored in the I3CxCCC register, and the SCCCIF/UCCCIF flags are set depending on whether the received CCC is supported or not. The flags are set for all Broadcast CCCs and only for those Direct CCCs for which an address match occurs.

Tip: The CCC code stored in the I3CxCCC register can be used with Unsupported CCC UCCCIF flag to provide custom firmware support for unsupported CCCs.

1.7 Private I3C Transaction

The controller can begin a Private I3C Transaction with the Target by addressing it directly using the Dynamic Address after a Start/Restart, or after transmitting the `7'h7E/W` I3C Broadcast Address. When a Dynamic Address match occurs, the DADRIF bit is set, and the RNW bits are changed to reflect whether the transaction is a read or a write request from the controller.

When a Dynamic Address match occurs, the module responds with an ACK/NACK as described below.

- If ACKP = `0`, then the module typically ACKs the request as follows:
  - Private Write: Module always ACKs regardless of the status of the Receive FIFO
  - Private Read: Module ACKs only when data is available in Transmit FIFO
- If ACKP = `1`, then the module typically NACKs the request, except when ACKPOS changes this behavior, as mentioned below
- If ACKPOS = `1`, the following immediate Private Transaction request will be ACK'd regardless of the status of the ACKP bit

Tip: The user can set ACKP = `1` to force a NACK on the bus for Private Transaction requests if the firmware is not ready to receive/send the data from/to the controller. When the firmware is ready, the user can set ACKPOS = `1` to receive/send the following data stream from/to the controller. Since the ACKPOS bit is cleared immediately after a one-time ACK, this mode is helpful when the firmware is speed limited and is unable to receive data from back-to-back Private Transaction requests.

1.7.1 Private Write Transaction

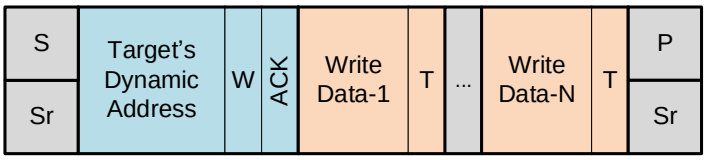
Once the Private Write request is ACK'd, the controller starts sending the 9-bit data words (8-bit data byte + 1-bit Parity T-bit). The received data bytes are stored in the Receive FIFO, which feeds into the I3CxRXB register for the user firmware/DMA to read. The transaction is completed when the controller sends a Stop or a Restart following the data, which is represented using the Transaction Complete TCOMPIF flag.

The flags on Table 1-1 provide additional buffer management abilities to the user.

Table 1-1. Receive Buffer/FIFO Management Flags and Commands

Flag		Description
RXBF	Receive Buffer Full	Data are available in the I3CxRXB register and ready to be read
I3CxRXIF	Receive Interrupt	System level interrupt for RXBF condition notifying the firmware/DMA to read the data from the I3CxRXB register
CLRRXB	Clear Receive Buffer + FIFO	Command to clear the I3CxRXB register and reset the Receive FIFO
RXREIF	Receive Read Error	Error interrupt signifying an attempt to read from the I3CxRXB register when data is not yet available from the Receive FIFO (RXBF = `0`)
RXOIF	Receive Overrun Error	Error interrupt signifying that a data byte has been lost due to controller writing to an already full Receive FIFO

Figure 1-4. Private Write Transfer Frame Format



1.7.2 Private Read Transaction

The user firmware/DMA writes the data byte to the I3CxTXB register, which feeds into the Transmit FIFO. Once the Private Read request is ACK'd, the Target starts sending the 9-bit data words (8-bit data byte + 1-bit End-of-Data T-bit) from the Transmit FIFO onto the bus. The transaction completes when either the Target or the controller ends the transaction as described below, followed by a Restart or a Stop, which is represented using the Transaction Complete TCOMPIF flag:

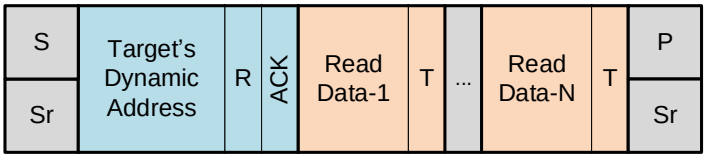
- The Target pulls the End-of-Data T-bit low after the Transmit FIFO becomes empty and all the data has been sent
- The controller aborts the transaction prematurely by pulling the End-of-Data T-bit low (a Restart condition on the bus), and the Abort Error ABEIF flag is set

The flags on Table 1-2 provide additional buffer management abilities to the user.

Table 1-2. Transmit Buffer/FIFO Management Flags and Commands

Flag		Description
TXBE	Transmit Buffer Empty	The I3CxTXB register is empty and ready to be written
I3CxTXIF	Transmit Interrupt	System level interrupt for TXBE condition notifying the firmware/DMA to write the data to the I3CxTXB register
CLRTXB	Clear Transmit Buffer + FIFO	Command to clear the I3CxTXB register and reset the Transmit FIFO
TXWEIF	Transmit Write Error	Error interrupt signifying an attempt to write to the I3CxTXB register when an existing data has not yet been moved to the Transmit FIFO (TXBE = 0)
TXUIF	Transmit Underrun Error	Error interrupt signifying that a read request has been made by the controller when the Transmit FIFO is empty
TXFNE	Transmit FIFO Not Empty	The Transmit FIFO (excludes the I3CxTXB register) status

Figure 1-5. Private Read Transfer Frame Format



1.7.3 Maximum Read/Write Length

The Maximum Read and Write Lengths can be specified using the I3CxMRL and I3CxMWL registers, respectively. The value in these registers can be updated by the user or by the controller using SETMRL/SETMWL CCCs. The controller can also read the value of these registers using the GETMRL/GETMWL CCCs. A value of zero in the registers signifies unlimited data length.

The I3C module responds according to the values in the registers automatically without any user intervention. During Private Write, when the controller exceeds the MWL length, the Target sets the Maximum Write Length Overflow MWLOEIF bit and stops receiving more data in the Receive FIFO even if the Receive FIFO is not full. A Receive Overrun RXOIF flag is also set. During Private Read, when the MRL length is reached, the Target pulls the End-of-Data T-bit low and ends the transaction even if Transmit FIFO has not been emptied (TXFNE = 1).

1.8 Legacy I^2C Transaction

The module participates in a Legacy I^2C Transaction anytime it does not have a Dynamic Address assigned and is not operating in Static Address SDR mode. The controller can begin an I^2C Transaction with the Target by addressing it directly using the Static Address after a Start/Restart. When a Static Address match occurs, the SADRIF bit is set, and the RNW bits are changed to reflect whether the transaction is a read or write request from the controller.

Important: It is imperative that the controller follows the I²C/SMBus timing and signaling in open-drain mode when conducting a Legacy I²C transaction on the I3C bus.

Tip: The Target can optionally select an I²C /SMBus-compatible input buffer using the RxyFEAT registers when operating in Legacy I²C mode. The Target can also optionally enable the 50 ns spike filters on the I²C/SMBus input buffers and set additional SDA hold times as desired using the I3CxI2CCON register.

As a Private I3C Write transaction, the Target's ACK/NACK response to an **I²C Write** request depends on the settings of the ACKPW and ACKPWOS bits. Once the I²C Write request is ACK'd, the controller sends an 8-bit data byte on the bus, and the Target responds with an ACK if data can be received in the Receive FIFO. The data are received in the Receive FIFO, which becomes available to be read from the I3CxRXB register.

Just like a Private I3C Read transaction, the Target's ACK/NACK response to an **I²C Read** request depends on the status of Transmit FIFO. Once the I²C Read request is ACK'd, the Target starts sending the 8-bit data byte on the bus, and the controller responds with an ACK/NACK, and the appropriate I2CACKIF/I2CNACKIF flags are set in the module.

All the flags and command bits for buffer management, as mentioned in the Private I3C Transaction section, are also available in the Legacy I²C mode. However, the Maximum Read/Write Length is an I3C only feature and does not apply to Legacy I²C Transactions on the bus.

Figure 1-6. Legacy I²C Write Transfer Frame Format

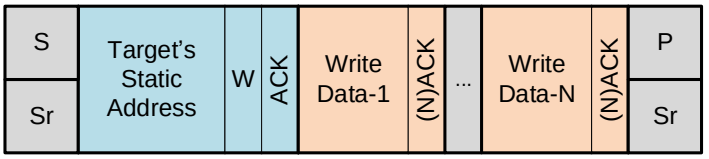
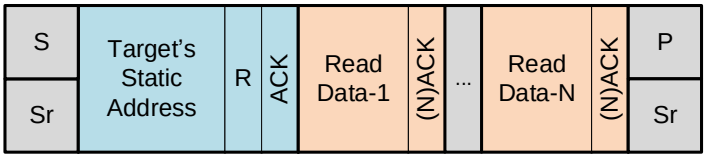


Figure 1-7. Legacy I²C Read Transfer Frame Format



1.9 In-Band Interrupt (IBI)

Before an IBI can be requested, the bus available time must be configured using the I3CxBAVL register to meet the minimum 1 µs specification. The module can perform an IBI request when IBI is enabled on the bus by the controller (IBIEN = `1`). The Target is operating in I3C SDR mode (OPMD = `0b01`).

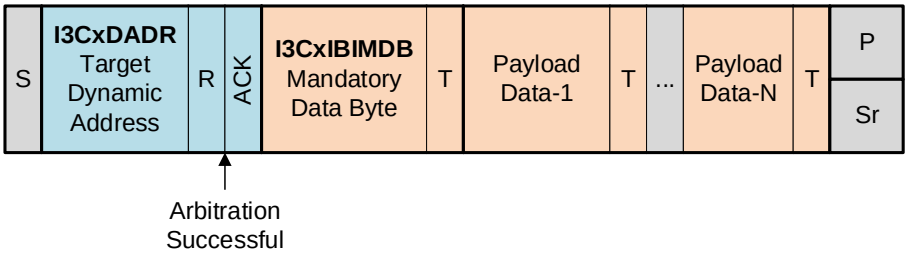
An IBI can be requested by setting the IBIREQ bit, after which the Target sends its Dynamic Address in read mode on the bus for arbitration after either one of the following conditions happens on the bus:

- Bus Available condition is detected on the bus, after which the Target issues a Start by pulling SDA low and participates in address arbitration (Standard IBI)
- Another device issues a Start on the bus before the Bus Available condition is detected, after which the Target participates in address arbitration as well (Passive IBI)

If the Target device wins arbitration and the controller ACKs, the IBI Mandatory Data Byte is transmitted on the bus from the I3CxIBIMDB register, followed by any payload data available in the Transmit FIFO. Just like a Private Read, the Target ends the transaction by pulling the End-of-Data T-bit low when the Transmit FIFO becomes empty, or the controller can abort the transaction by pulling the End-of-Data T-bit low (a Restart condition), in which case the Abort Error ABEIF flag is set. The Transaction Complete TCOMPIF flag is set when either of these conditions happens.

If the Target device loses arbitration or the controller NACKs, the Target attempts to request an IBI again, automatically, until the Arbitration Request Limit in the I3CxRETRY register has been reached, following which the IBI Error IBIEIF flag is set.

Figure 1-8. Successful IBI Transaction Frame Format



1.10 Hot-Join (HJ) Transaction

A Hot-Join (HJ) Transaction is a special type of IBI transaction that begins with the reserved Hot-Join Address `7'h02`. Before an HJ can be requested, the bus idle time must be configured using the I3CxBIDL register to meet the minimum 200 µs specification. The module can perform an HJ request when the Target is HJ capable (HJCAP = `1`), HJ is enabled on the bus by the controller (HJEN = `1`), and the Target does not have a Dynamic Address assigned.

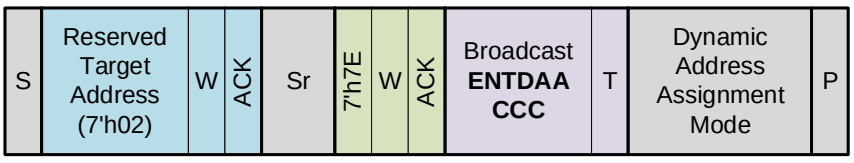
An HJ can be requested by setting the HJREQ bit, after which the Target sends the reserved HJ Address `7'h02` in read mode on the bus for arbitration after either one of the following conditions happens on the bus:

- Bus Idle condition is detected on the bus, after which the Target issues a Start by pulling SDA low and participates in address arbitration (Standard HJ)
- Another device issues a Start on the bus before the Bus Idle condition is detected, after which the Target participates in address arbitration (Passive HJ)

If the Target device wins arbitration and the controller ACKs, the Target automatically participates in the Dynamic Address Assignment process when it sees the next ENTDAA CCC on the bus. The controller may or may not immediately send the ENTDAA CCC. Once the Target receives a Dynamic Address, the Dynamic Address Changed DACHIF flag is set, and the HJREQ bit is cleared.

If the Target device loses arbitration or the controller NACKs, the Target attempts to request an HJ again, automatically, until the Arbitration Request Limit in the I3CxRETRY register has been reached, following which the HJ Error HJEIF flag is set and HJREQ bit is cleared.

Figure 1-9. Successful Hot-Join Frame Format



1.11 Target Reset

The controller typically configures one of the following three reset actions for the Target device using the RSTACT CCC:

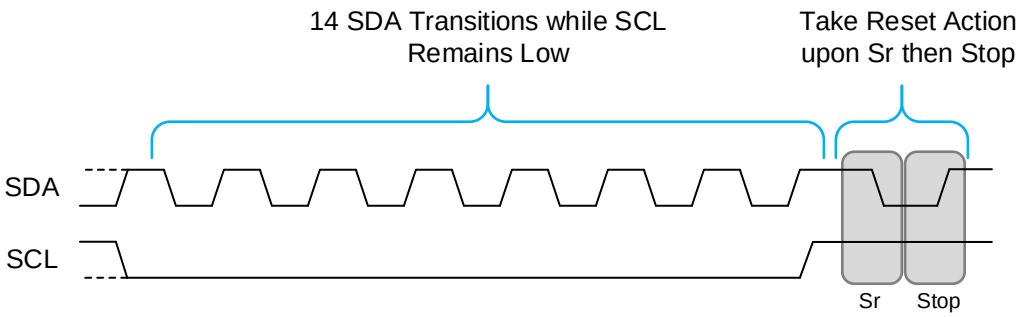
1. No Target Reset (RSTACT Defining Byte `0x00`).
2. Reset the I3C module only (RSTACT Defining Byte `0x01`).
3. Reset the entire device (RSTACT Defining Byte `0x02`).

The appropriate defining byte is stored in the I3CxRSTACT register. When the Target detects a Target Reset Pattern on the bus, the RSTDET bit is set, and the system-level I3CxRIF interrupt flag is set. The user firmware should read the RSTACT CCC defining byte from the I3CxRSTACT register and take the appropriate reset action. To reset only the I3C module, the user can set the Software Reset RST bit. The `RESET` instruction can be used to reset the entire device.

Important:

1. It is recommended to manually reset the I3CxRSTACT register to `0xFF` once the user has read it.
2. Refer to the device data sheet for the full version of the Target Reset operation flow.

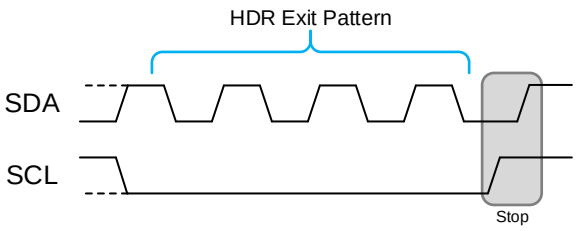
Figure 1-10. Target Reset Pattern



1.12 High Data Rate (HDR) Modes

The I3C module on 8-bit PIC devices does not support HDR modes. However, the module can detect HDR entry and exit patterns to ensure proper device operation. When the controller enters the HDR mode using any ENTHDRx CCCs, the device enters HDR mode (OPMD = `0b1x`). In this mode the device ignores all the HDR traffic on the bus until it detects an HDR Exit Pattern. Once the Target detects the HDR Exit Pattern on the bus, it reverts to operating in SDR mode (OPMD = `0b0x`).

Figure 1-11. HDR Exit Pattern with Stop



1.13 Error Detection and Recovery

The I3C Target module supports various error detection and recovery methods specified in the MIPI I3C Specification.

Table 1-3 summarizes the various supported methods.

Table 1-3. SDR Error Detection and Recovery Methods

Error Type	Description	Detection	Recovery
TE0	Invalid Broadcast Address	Invalid combination of `7'h7E/W`; TE0ERR/BUSEIF bits set	Target ignores all traffic on the bus and waits for an HDR Exit Pattern
TE1	Invalid CCC Code	Invalid parity after CCC code; TE1ERR/BUSEIF bits set	
TE2	Invalid Write Data	Invalid parity after Private/CCC Write data byte; TE2ERR/BUSEIF bits set	Target waits for the next Stop or Restart
TE3	Invalid Assigned Address during Dynamic Address Assignment	Invalid parity after assigned Dynamic Address; TE3ERR/BUSEIF bits set	Target NACKs and waits for next Restart to participate in Dynamic Address Assignment again
TE4	Illegally formatted data during Dynamic Address Assignment	Invalid `7'h7E/R` after Restart; TE4ERR/BUSEIF bits set	Target NACKs and waits for next Stop
TE5	Illegally formatted CCC frame	Monitor CCC frame for any discrepancy; TE5ERR/BUSEIF bits set	Target NACKs and waits for next Stop or Restart
TE6	Corrupted R/$\overline{W}$ bit during Private Transfer	Monitor data on SDA line during Private Read; TE6ERR/BUSEIF bits set	Target lets go of the bus and waits for next Stop or Restart

2. Using I3C with Direct Memory Access (DMA)

Since the I3C bus frequency can reach as high as 12.5 MHz, it is suggested that the user use DMA to read and write from the I3C Transmit and Receive Buffers to ensure that the application can keep up with the high-speed data transfer rates. This section describes setting up the DMA module on 8-bit PIC devices to operate with the I3C module. Refer to the appropriate device data sheet for information about setting up the I3C module in I3C/I^{2}C Target mode.

It is highly recommended for the user to read the **"DMA – Direct Memory Access"** chapter in the device data sheet to familiarize themselves with the DMA module and how it operates. The user can also refer to **"TB3242 – Configuring the DMA Peripheral"** for gaining better understanding of the DMA module on the Microchip website.

For user convenience, the MPLAB Code Configurator (MCC) Melody offers DMA-based drivers for the I3C peripheral. The API documentation for the I3C Target Driver can be found on the Microchip website and getting started code examples using the I3C drivers can be accessed using the MPLAB Discover link below.

2.1 Private I3C/I^{2}C Write Transactions using DMA

For a Private I3C/I^{2}C Write transaction, it is recommended for the user to configure the DMA so that the data is transferred from the I3CxRXB register to a software buffer in the GPR space. The DMA can be triggered every time a data byte is received in the I3CxRXB register, which is available as a system level I3CxRXIF Receive Interrupt. It is not required to enable the system level I3CxRXIF interrupt for it to be used as a DMA trigger.

The DMA can be configured to transfer a set number of bytes to read from the I3CxRXB register. If it is possible that the controller might send less data bytes than that, the DMA can optionally be configured to be stopped when a TCOMPIF interrupt flag is observed. The TCOMPIF interrupt is available through the system level I3Cx General Interrupt. Although it is not required to enable the system level I3Cx interrupt to be used as a DMA trigger, the module level TCOMPIF interrupt must be enabled for this function.

Table 2-1. DMA Settings for Private I3C/I^{2}C Write Transactions

DMA1 Parameter		Setting
SSA	Source Address	&I3CxRXB
SSZ	Source Size	1
SMR	Source Memory Region	SFR
SMODE	Source Address Mode	SPTR remains unchanged
DSA	Destination Address	Beginning of software receive buffer
DSZ	Destination Size	Number of bytes to read
DMODE	Destination Address Mode	DPTR is incremented
SIRQEN	Start Trigger Enable	Yes (enable when ready to receive data)
SIRQ	Start Trigger	I3CxRX (I3CxRXB is full)
SSTP	Source Counter Reload Stop	No – SIRQEN not cleared
DSTP	Destination Counter Reload Stop	Yes – SIRQEN is cleared when DCNT reloads
AIRQEN	Abort Trigger Enable	No (stops when SCNT reloads)(1)
AIRQ	Abort Trigger	N/A(1)

.....continued

DMA1 Parameter	Setting

Note:

1. A TCOMPIF-based abort trigger (through the system level I3Cx General Interrupt flag) can optionally be set to abort the DMA when the Controller ends the transaction prematurely.

How It Works:

1. After SIRQEN is set, the DMA is triggered as soon as data are available if the I3CxRXIF interrupt flag is set and the content of the I3CxRXB register is transferred to the first address in the software receive buffer. DPTR is incremented and now points to the following address in the software receive buffer. Because SSZ = `1`, the DMA stops after the first transfer. Since SSTP = `0`, SIRQEN remains set, and DMA waits for the next trigger.
2. When the data is available in I3CxRXB register again, the events in step 1 are repeated.
3. After the desired number of bytes have been transferred by the DMA (DSZ value), the DPTR reaches the end of the allocated software buffer, and DCNT reloads. Since DSTP = `1`, SIRQEN is cleared, and no further DMA transfers happen.
4. Optionally, if an abort trigger is configured using TCOMPIF, the DMA is aborted when the transaction is completed on the bus, regardless of the number of bytes transferred, which clears the SIRQEN bit.

Example 2-1. Example Code for Private I3C/I^2C Write Using DMA

```
// Example code for PIC18-Q20 device family

uint8_t rxData[SWBUF_SIZE];

// Private Write DMA1
void DMA1_Initialize(void)
{
    // Source and Destination settings
    DMASELECT = 0;              // DMA Instance
    DMAnSSA = &I3C1RXB;         // Source Address: Register in SFR space
    DMAnDSA = &rxData;          // Destination Address: Variable in GPR space
    DMAnSSZ = 1;                // Source Size: Source is only one register
    DMAnDSZ = SWBUF_SIZE;       // Destination Size: Size of variable in GPR
space
    DMAnSIRQ = 0x40;            // Start Trigger: I3C1RX
    DMAnAIRQ = 0x00;            // Abort Trigger: None

    DMAnCON1bits.SMODE = 0b00;  // Source Mode: Pointer unchanged
    DMAnCON1bits.SSTP = 0;      // Source Reload: No; SIRQEN not cleared
    DMAnCON1bits.SMR = 0b00;    // Source Region: SFR/GPR space
    DMAnCON1bits.DMODE = 0b01;  // Destination Mode: Pointer incremented
    DMAnCON1bits.DSTP = 1;      // Destination Reload: Yes; SIRQEN is cleared

    DMAnCON0bits.SIRQEN = 0;    // Start Trigger: Disabled (enable later when
ready)
    DMAnCON0bits.AIRQEN = 0;    // Abort Trigger: Disabled (stops when SCNT
reloads)
    DMAnCON0bits.DGO = 0;       // DMA Transaction: Do not start transaction
yet

    // Clear interrupt flags
    PIR0bits.DMA1DCNTIF = 0;    // Destination Count Interrupt
    PIR0bits.DMA1SCNTIF = 0;    // Source Count Interrupt
    PIR0bits.DMA1AIF = 0;       // Abort Interrupt
    PIR0bits.DMA1ORIF = 0;      // Overrun Interrupt

    // Enable appropriate interrupts as needed
    PIE0bits.DMA1DCNTIE = 1;    // Destination Count Interrupt (enabled)
    PIE0bits.DMA1SCNTIE = 0;    // Source Count Interrupt
    PIE0bits.DMA1AIE = 0;       // Abort Interrupt
    PIE0bits.DMA1ORIE = 0;      // Overrun Interrupt

    //Enable DMA
    DMAnCON0bits.EN = 1;
```

```
}

void I3C1_ReceiveDataWithDMA(void)
{
    // Clear RXB+RXFIFO if appropriate
    I3C1CON0bits.CLRRXB = 1;

    // Enable One-shot ACK (if ACKP = NACK by default)
    I3C1CON1bits.ACKPOS = 1;

    // Start DMA transfer with trigger
    DMASELECT = 0;
    DMAnCON0bits.SIRQEN = 1;
}

void __interrupt(irq(IRQ_DMA1DCNT)) DMA1_DCNT_ISR(void)
{
    PIR0bits.DMA1DCNTIF = 0;

    // Code execution reaches here when DMA has completed transfer
    // Received data is now available for processing in rxData[]
}
```

2.2 Private I3C/I²C Read and In-Band Interrupt Transactions using DMA

For a Private I3C/I²C Read transaction, it is recommended for the user to configure the DMA such that the data is transferred from a software buffer in the GPR space to the I3CxTXB register. The DMA can be triggered every time the I3CxTXB register is empty, which is available as a system level I3CxTXIF Transmit Interrupt. It is not required to enable the system level I3CxTXIF interrupt for it to be used as a DMA trigger.

The DMA can be configured to transfer a set number of bytes to write to the I3CxTXB register. If it is possible that the controller might read less data bytes than that, the DMA can optionally be configured to be stopped when a TCOMPIF interrupt flag is observed. The TCOMPIF interrupt is available through the system level I3Cx General Interrupt. Even though it is not required to enable the system level I3Cx interrupt to be used as a DMA trigger, the module level TCOMPIF interrupt must be enabled for this function.

For an IBI transaction, while the mandatory data byte is transmitted from the I3CxIBIMDB register, the payload is transmitted from the Transmit FIFO just like a Private Read transaction. The DMA configuration for sending the IBI payload from a software buffer is the same as that of a Private Read transaction.

Table 2-2. DMA Settings for Private I3C/I²C Read and IBI Transactions

DMA2 Parameter		Setting
SSA	Source Address	Beginning of software transmit buffer
SSZ	Source Size	Number of bytes to send
SMR	Source Memory Region	GPR
SMODE	Source Address Mode	SPTR is incremented
DSA	Destination Address	&I3CxTXB
DSZ	Destination Size	1
DMODE	Destination Address Mode	DPTR remains unchanged
SIRQEN	Start Trigger Enable	Yes (enable when ready to send data)
SIRQ	Start Trigger	I3CxTX (I3CxTXB is empty)
SSTP	Source Counter Reload Stop	Yes – SIRQEN is cleared when SCNT reloads
DSTP	Destination Counter Reload Stop	No – SIRQEN is not cleared
AIRQEN	Abort Trigger Enable	No (stops when DCNT reloads) (1)
AIRQ	Abort Trigger	N/A(1)

.....continued

DMA2 Parameter	Setting

Note:

1. A TCOMPIF-based abort trigger (through the system level I3Cx General Interrupt flag) can optionally be set to abort the DMA when the controller ends the transaction prematurely.

How It Works:

1. The DMA is triggered when SIRQEN and the I3CxTXIF interrupt flag are set. The content of the first address of the software transmit buffer is transferred to the I3CxTXB register. SPTR is incremented and now points to the following address in the software transmit buffer. Because DSZ = 1, the DMA stops after the first transfer. Since DSTP = 0, SIRQEN remains set, and DMA waits for the next trigger.
2. When the I3CxTXB register becomes empty again, the events in step 1 are repeated.
3. After the desired number of bytes have been transferred by the DMA (SSZ value), the SPTR reaches the end of the allocated software buffer, and SCNT reloads. Since SSTP = 1, SIRQEN is cleared, and no further DMA transfers happen.
4. Optionally, if an abort trigger is configured using TCOMPIF, the DMA is aborted when the transaction is completed on the bus, regardless of the number of bytes transferred, which clears the SIRQEN bit.

Example 2-2. Example Code for Private I3C/I²C/IBI Read Using DMA

```
// Example code for PIC18-Q20 device family

uint8_t txData[SWBUF_SIZE];

// Private Read/IBI DMA2
void DMA2_Initialize(void)
{
    // Source and Destination settings
    DMASELECT = 1;              // DMA Instance
    DMAnSSA = &txData;          // Source Address: Variable in GPR space
    DMAnDSA = &I3C1TXB;         // Destination Address: Register in SFR space
    DMAnSSZ = SWBUF_SIZE;       // Source Size: Size of variable in GPR space
    DMAnDSZ = 1;                // Destination Size: Only one register
    DMAnSIRQ = 0x41;            // Start Trigger: I3C1TX
    DMAnAIRQ = 0x00;            // Abort Trigger: None

    DMAnCON1bits.SMODE = 0b01;  // Source Mode: Pointer incremented
    DMAnCON1bits.SSTP = 1;      // Source Reload: Yes; SIRQEN is cleared
    DMAnCON1bits.SMR = 0b00;    // Source Region: SFR/GPR space
    DMAnCON1bits.DMODE = 0b00;  // Destination Mode: Pointer unchanged
    DMAnCON1bits.DSTP = 0;      // Destination Reload: No; SIRQEN not cleared

    DMAnCON0bits.SIRQEN = 0;    // Start Trigger: Disabled (enable later when
ready)
    DMAnCON0bits.AIRQEN = 0;    // Abort Trigger: Disabled (stops when DCNT
reloads)
    DMAnCON0bits.DGO = 0;       // DMA Transaction: Do not start transaction
yet

    // Clear interrupt flags
    PIR1bits.DMA2DCNTIF = 0;    // Destination Count Interrupt
    PIR1bits.DMA2SCNTIF = 0;    // Source Count Interrupt
    PIR1bits.DMA2AIF = 0;       // Abort Interrupt
    PIR1bits.DMA2ORIF = 0;      // Overrun Interrupt

    // Enable appropriate interrupts as needed
    PIE1bits.DMA2DCNTIE = 0;    // Destination Count Interrupt
    PIE1bits.DMA2SCNTIE = 1;    // Source Count Interrupt (enabled)
    PIE1bits.DMA2AIE = 0;       // Abort Interrupt
    PIE1bits.DMA2ORIE = 0;      // Overrun Interrupt

    //Enable DMA
    DMAnCON0bits.EN = 1;
```

```
}

void I3C1_SendDataWithDMA(void)
{
    // Initialize txData[] with data to be sent for Private Read (or payload
to be sent for IBI)
    initializeTxData();

    // Clear TXB+TXFIFO if appropriate
    I3C1CON0bits.CLRTXB = 1;

    // Enable One-shot ACK (if ACKP = NACK by default)
    I3C1CON1bits.ACKPOS = 1;

    // Start DMA transfer with trigger
    DMASELECT = 1;
    DMAnCON0bits.SIRQEN = 1;
}

void I3C1_RequestIBI(void)
{
    // Configure IBI mandatory data byte
    I3C1IBIMDB = 0xAA;

    // Configure DMA and send payload
    I3C1_SendDataWithDMA();

    // Request IBI
    I3C1CON0bits.IBIREQ = 1;
}

void __interrupt(irq(IRQ_DMA2SCNT)) DMA2_SCNT_ISR(void)
{
    PIR1bits.DMA2SCNTIF = 0;

    // Code execution reaches here when DMA has completed transfer
}
```

2.3 Setting an Appropriate DMA Priority

Once the DMA has been configured, it is important to set an appropriate priority to the DMA as described in the **"System Arbiter"** section of the **"PIC18 CPU"** chapter of the device data sheet. The system arbiter grants memory access to the DMA if the PRLOCKED bit is set. Setting and clearing this bit requires a special sequence as an extra precaution against unexpected changes.

Code Example 2-3 demonstrates setting the Priority Lock.

Example 2-3. Setting the DMA Priority Lock

```
void DMA_SetPriority(void)
{
    // This function is dependant on the PR1WAY CONFIG bit

    // Unlock first (if locked)
    PRLOCK = 0x55;
    PRLOCK = 0xAA;
    PRLOCKbits.PRLOCKED = 0;

    // Change priority as needed
    DMA1PR = 7;     // RX_DMA has highest priority
    DMA2PR = 7;     // TX_DMA has next highest priority

    // Lock again
    PRLOCK = 0x55;
    PRLOCK = 0xAA;
    PRLOCKbits.PRLOCKED = 1;
}
```

3. Use Cases

The 8-bit PIC microcontrollers contain a variety of core independent peripherals for serial communication (UART/SPI/I^{2}C), analog to digital conversion (ADC), and timer/counters. The I3C Target module can be used in conjunction with these peripherals in some sensor, IoT, and memory-related applications. Some key applications include:

- Multi-protocol Translator
- Multi-sensor Integration
- Solid State Drive (SSD) Sideband Communication
- Memory Module SidebandBus Communication

Find the MCC Melody code configuration here:

3.1 Multi-Protocol Translator

As many industries transition from traditional I^{2}C/SMBus-based communication to a faster I3C-based communication, system designers are looking for ways to maintain backward compatibility. While the I3C protocol is designed to be backward-compatible with I^{2}C/SMBus protocols, the presence of an I^{2}C/SMBus device on an I3C bus can severely impact the performance of the bus even when the controller optimizes the transactions to communicate to I3C devices.

In such scenarios, when it is not possible to have a pure I3C bus, the I3C module on the 8-bit PIC microcontrollers can be used to isolate the I^{2}C/SMBus devices from the I3C bus and act as a bridge device, as shown in Figure 3-1. This maintains the integrity of the pure I3C bus while still allowing the controller to communicate to the I^{2}C/SMBus devices through the PIC microcontroller.

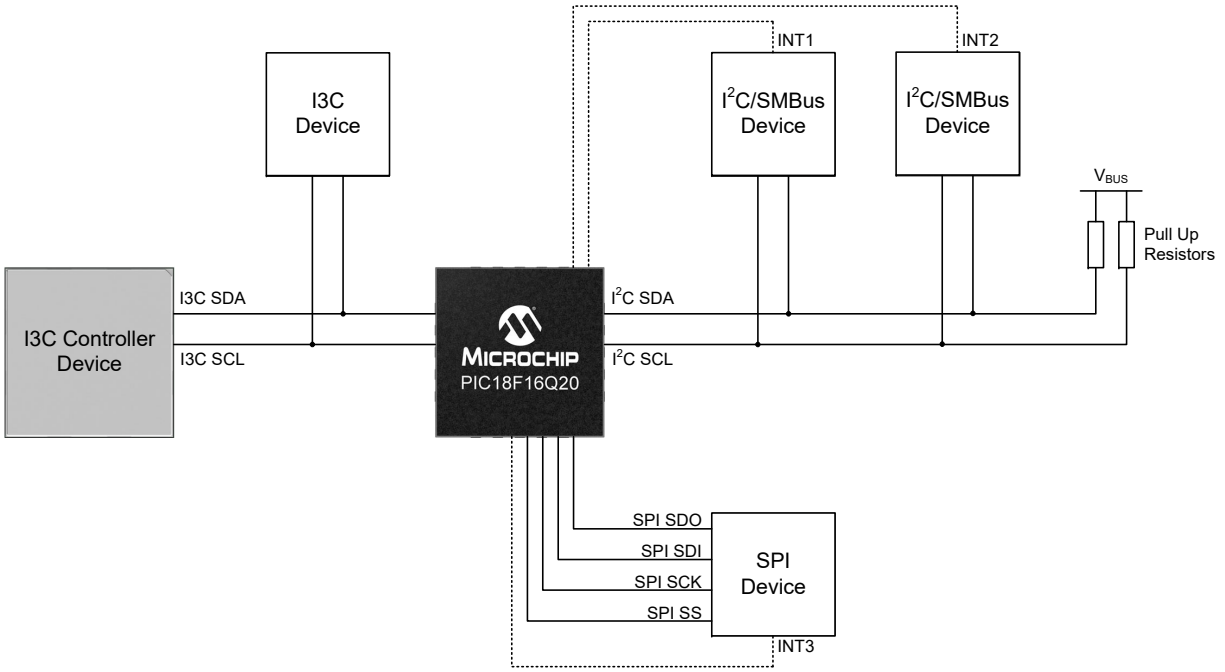
As an added benefit, since the I^{2}C/SMBus devices use external signals for interrupts, the PIC microcontroller can also consolidate the interrupts from all the I^{2}C/SMBus devices and relay them to the I3C controller using In-Band Interrupts without any extra pins/signals. In addition, the MVIO on the PIC microcontroller allows the I3C and I^{2}C/SMBus buses to operate at different bus voltages. For instance, the I3C bus can operate at 1V, whereas the I^{2}C/SMBus bus can continue to operate at a higher 3.3V to be compatible with the I^{2}C/SMBus devices.

Since the PIC microcontrollers also contain a SPI module, the device can also act as an I3C-to-SPI bridge if needed. This can be useful when an external SPI-based memory module or sensor needs to be added to the bus.

For a multi-protocol translator application the I3C module is configured in Target mode, whereas the I^{2}C/SPI modules are configured in Controller/Host mode. The user can write custom firmware or use DMA to read/write data across the I3C and I^{2}C/SPI modules and the GPR space.

Find the MCC Melody code configuration here:

Figure 3-1. Multi-protocol Translator using the I3C/I²C/SPI Modules on PIC18-Q20



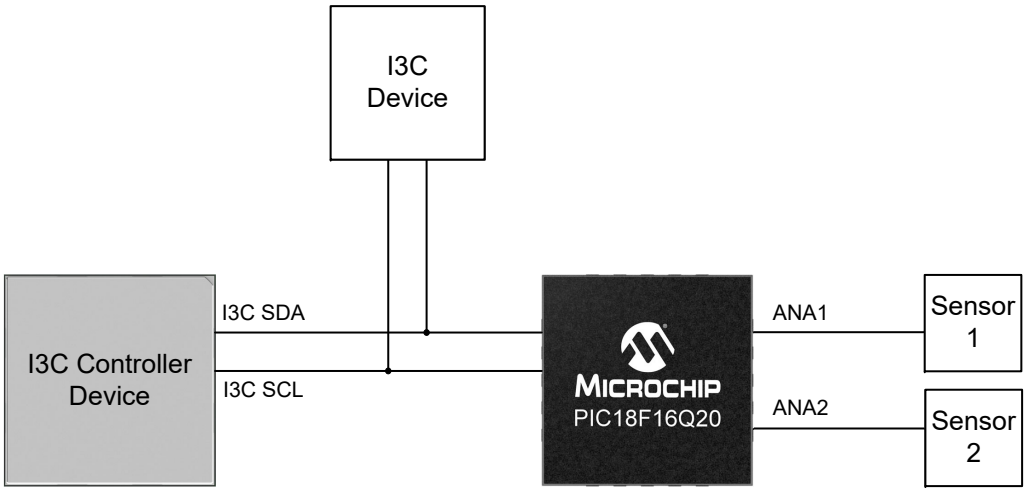
3.2 Multi-Sensor Integration

In applications requiring one or more analog sensors to be added to the I3C bus, the 8-bit PIC microcontroller can be used to add I3C functionality to the analog sensors. The sensors can interface with the PIC microcontroller through the analog pins, as shown in Figure 3-2. The on-board Analog-to-Digital Converter (ADC) converts the sensor readings into digital values, which can be passed onto the I3C module directly or stored in the internal memory to be retrieved later. The user can write custom firmware for the I3C Target module to read the converted sensor data and send it to the controller, when needed. Since the I3C module operates on an independent MVIO domain in the PIC microcontrollers, the analog sensors can operate in the entire voltage range and are not limited by the I3C bus voltage.

Find the MCC Melody code configuration here:

Click to view code example on MPLAB DISCOVER

Figure 3-2. Multi-sensor Integration Using the I3C Module on PIC18-Q20



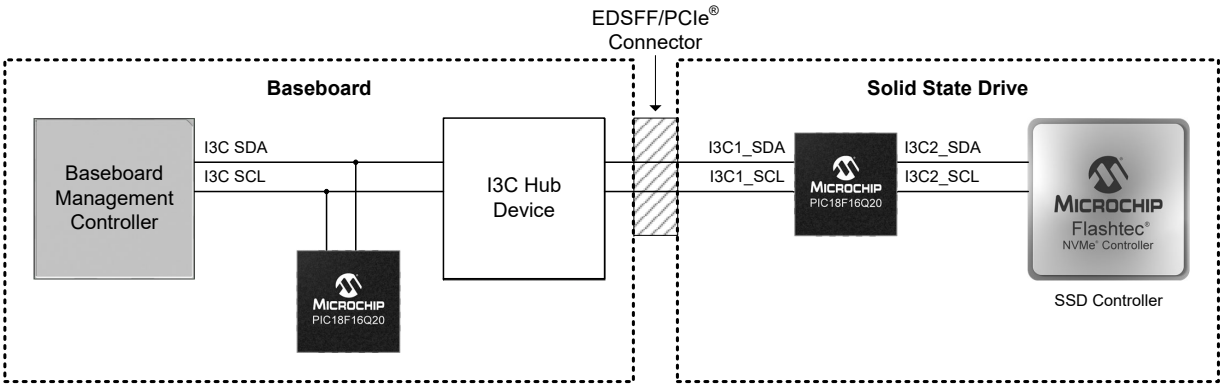
3.3 Solid State Drive (SSD) Sideband Communication

As more advanced SSDs appear in the datacenter market, the SNIA® Enterprise and Datacenter Standard Form Factor (EDSFF) has been adopted as the new SSD form factor, offering several advantages over the previous older U.2 and M.2 form factors. The EDSFF form factors use the NVMe® protocol over the PCIe® interface and dedicated connectors (SNIA SFF-TA-1002 specification). The EDSFF pinout and functions specification (SNIA SFF-TA-1009) has adopted the I3C protocol for sideband communication as a natural upgrade from the SMBus protocol. Refer to the SNIA website for SFF Specifications.

In a typical datacenter environment, the Baseboard Management Controller (BMC) uses the Management Component Transport Protocol (MCTP) to communicate with other management controllers/devices. MCTP is a media-independent protocol supporting I3C and SMBus as the underlying communication protocol. The 8-bit PIC microcontrollers with I^2C and I3C modules support both MCTP-over-I3C and MCTP-over-SMBus binding specifications, allowing it to be an ideal device for system management for providing next generation I3C support while still being SMBus compatible. Refer to the DMTF® website for MCTP Transport Binding Specifications.

The 8-bit PIC microcontrollers can have up to two I3C Target modules, which allows the PIC microcontroller to connect to the onboard SSD controller for system management in addition to being connected to the BMC through EDSFF/PCIe connector for sideband communication. One such example featuring various products from the Microchip portfolio is shown in Figure 3-3.

Figure 3-3. SSD Sideband Communication using the I3C Modules on PIC18-Q20

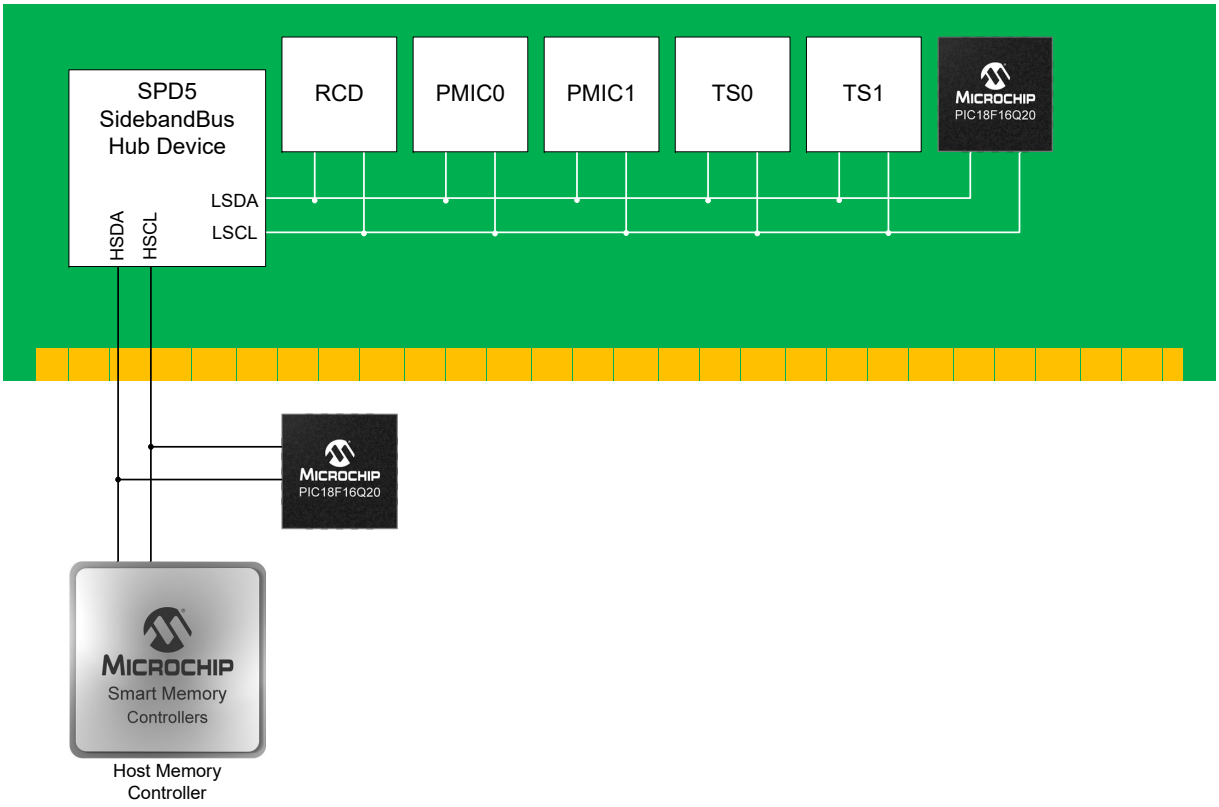


3.4 Memory Controller SidebandBus Communication

To cater to the system management needs for the next generation DDR5 memory modules and beyond, the JEDEC® group has introduced the JEDEC Module Sideband Bus (SidebandBus) based of the I3C protocol. The SidebandBus protocol can be downloaded from the JEDEC website. The 8-bit PIC microcontrollers with the I3C module can be connected to the SidebandBus as a generic I3C device that meets the interoperability considerations and the speed and voltage requirements, as mentioned in the JEDEC specification.

Many types of devices can be connected to the SidebandBus in a memory controller application, such as a Registering Clock Device (RCD), Power Management ICs (PMICs), Temperature Sensors (TS), and a Serial Presence Detect (SPD) device, which also acts as a SidebandBus hub. The 8-bit PIC microcontrollers can sit on the SidebandBus both on the Host bus (before the hub) or the Local bus (behind the hub) to assist with system and power management. One such example featuring various products from the Microchip portfolio in Registered Dual In-line Memory Modules (RDIMMs) is shown in Figure 3-4.

Figure 3-4. Memory Controller SidebandBus in RDIMMs Using PIC18-Q20



4. Conclusion

The I3C Target module on the PIC microcontroller is well suited for the next generation high speed applications requiring I3C functionality while maintaining backward compatibility with I^2C/SMBus. Due to the versatile nature of the PIC microcontrollers, they can provide a low-cost solution to assist with system and power management for SSD and memory module applications. Due to the presence of additional communication protocols, PIC microcontrollers can also act as a bridge device between I3C and I^2C/SMBus/SPI/Analog channels.

5. Revision History

Revision	Date	Description
A	09/2023	Initial document release.

Microchip Information

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **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 (FAQs), technical support requests, online discussion groups, Microchip design partner 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

Product Change Notification Service

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

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

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

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable". Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure

that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, CryptoMemory, CryptoRF, dsPIC, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AgileSwitch, ClockWorks, The Embedded Control Solutions Company, EtherSynch, Flashtec, Hyper Speed Control, HyperLight Load, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, TimeCesium, TimeHub, TimePictra, TimeProvider, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, Augmented Switching, BlueSky, BodyCom, Clockstudio, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, EyeOpen, GridTime, IdealBridge, IGaT, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, IntelliMOS, Inter-Chip Connectivity, JitterBlocker, Knob-on-Display, MarginLink, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, mSiC, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, Power MOS IV, Power MOS 7, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SmartHLS, SMART-I.S., storClad, SQI, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, Trusted Time, TSHARC, Turing, USBCheck, VariSense, VectorBlox, VeriPHY, ViewSpan, WiperLock, XpressConnect, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

ISBN: 978-1-6683-3050-0

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office	**Australia - Sydney**	**India - Bangalore**	**Austria - Wels**
2355 West Chandler Blvd.	Tel: 61-2-9868-6733	Tel: 91-80-3090-4444	Tel: 43-7242-2244-39
Chandler, AZ 85224-6199	**China - Beijing**	**India - New Delhi**	Fax: 43-7242-2244-393
Tel: 480-792-7200	Tel: 86-10-8569-7000	Tel: 91-11-4160-8631	**Denmark - Copenhagen**
Fax: 480-792-7277	**China - Chengdu**	**India - Pune**	Tel: 45-4485-5910
Technical Support:	Tel: 86-28-8665-5511	Tel: 91-20-4121-0141	Fax: 45-4485-2829
www.microchip.com/support	**China - Chongqing**	**Japan - Osaka**	**Finland - Espoo**
Web Address:	Tel: 86-23-8980-9588	Tel: 81-6-6152-7160	Tel: 358-9-4520-820
www.microchip.com	**China - Dongguan**	**Japan - Tokyo**	**France - Paris**
Atlanta	Tel: 86-769-8702-9880	Tel: 81-3-6880- 3770	Tel: 33-1-69-53-63-20
Duluth, GA	**China - Guangzhou**	**Korea - Daegu**	Fax: 33-1-69-30-90-79
Tel: 678-957-9614	Tel: 86-20-8755-8029	Tel: 82-53-744-4301	**Germany - Garching**
Fax: 678-957-1455	**China - Hangzhou**	**Korea - Seoul**	Tel: 49-8931-9700
Austin, TX	Tel: 86-571-8792-8115	Tel: 82-2-554-7200	**Germany - Haan**
Tel: 512-257-3370	**China - Hong Kong SAR**	**Malaysia - Kuala Lumpur**	Tel: 49-2129-3766400
Boston	Tel: 852-2943-5100	Tel: 60-3-7651-7906	**Germany - Heilbronn**
Westborough, MA	**China - Nanjing**	**Malaysia - Penang**	Tel: 49-7131-72400
Tel: 774-760-0087	Tel: 86-25-8473-2460	Tel: 60-4-227-8870	**Germany - Karlsruhe**
Fax: 774-760-0088	**China - Qingdao**	**Philippines - Manila**	Tel: 49-721-625370
Chicago	Tel: 86-532-8502-7355	Tel: 63-2-634-9065	**Germany - Munich**
Itasca, IL	**China - Shanghai**	**Singapore**	Tel: 49-89-627-144-0
Tel: 630-285-0071	Tel: 86-21-3326-8000	Tel: 65-6334-8870	Fax: 49-89-627-144-44
Fax: 630-285-0075	**China - Shenyang**	**Taiwan - Hsin Chu**	**Germany - Rosenheim**
Dallas	Tel: 86-24-2334-2829	Tel: 886-3-577-8366	Tel: 49-8031-354-560
Addison, TX	**China - Shenzhen**	**Taiwan - Kaohsiung**	**Israel - Ra'anana**
Tel: 972-818-7423	Tel: 86-755-8864-2200	Tel: 886-7-213-7830	Tel: 972-9-744-7705
Fax: 972-818-2924	**China - Suzhou**	**Taiwan - Taipei**	**Italy - Milan**
Detroit	Tel: 86-186-6233-1526	Tel: 886-2-2508-8600	Tel: 39-0331-742611
Novi, MI	**China - Wuhan**	**Thailand - Bangkok**	Fax: 39-0331-466781
Tel: 248-848-4000	Tel: 86-27-5980-5300	Tel: 66-2-694-1351	**Italy - Padova**
Houston, TX	**China - Xian**	**Vietnam - Ho Chi Minh**	Tel: 39-049-7625286
Tel: 281-894-5983	Tel: 86-29-8833-7252	Tel: 84-28-5448-2100	**Netherlands - Drunen**
Indianapolis	**China - Xiamen**		Tel: 31-416-690399
Noblesville, IN	Tel: 86-592-2388138		Fax: 31-416-690340
Tel: 317-773-8323	**China - Zhuhai**		**Norway - Trondheim**
Fax: 317-773-5453	Tel: 86-756-3210040		Tel: 47-72884388
Tel: 317-536-2380			**Poland - Warsaw**
Los Angeles			Tel: 48-22-3325737
Mission Viejo, CA			**Romania - Bucharest**
Tel: 949-462-9523			Tel: 40-21-407-87-50
Fax: 949-462-9608			**Spain - Madrid**
Tel: 951-273-7800			Tel: 34-91-708-08-90
Raleigh, NC			Fax: 34-91-708-08-91
Tel: 919-844-7510			**Sweden - Gothenberg**
New York, NY			Tel: 46-31-704-60-40
Tel: 631-435-6000			**Sweden - Stockholm**
San Jose, CA			Tel: 46-8-5090-4654
Tel: 408-735-9110			**UK - Wokingham**
Tel: 408-436-4270			Tel: 44-118-921-5800
Canada - Toronto			Fax: 44-118-921-5820
Tel: 905-695-1980			
Fax: 905-695-2078			