TUSB3410 USB to Serial Port Controller

1 Device Overview

1.1 Features

- Fully Compliant With USB 2.0 Full-Speed Specifications: TID#40340262
- Supports 12-Mbps USB Data Rate (Full Speed)
- Supports USB Suspend, Resume, and Remote Wake-Up Operations
- Configurable to Bus-Powered and Self-Powered Operation
- Supports a Total of Three Input and Three Output (Interrupt, Bulk) Endpoints
- Integrated 8052 Microcontroller With:
  - 256 × 8 RAM for Internal Data
  - 10K × 8 ROM (With USB and I²C Bootloader)
  - 16K × 8 RAM for Code Space Loadable From Host or I²C Port
  - 2K × 8 Shared RAM Used for Data Buffers and Endpoint Descriptor Blocks (EDBs)
  - Master I²C Controller for EEPROM Device Access
  - MCU Operates at 24 MHz, Providing 2-MIPS Operation
  - 128-ms Watchdog Timer
- Enhanced UART Features:
  - Software and Hardware Flow Control
  - Automatic RS-485 Bus Transceiver Control, With and Without Echo
  - Selectable IrDA Mode for Up to 115.2-kbps Transfer
  - Software-Selectable Baud Rate From 50 BPS to 921.6 kbps
  - Programmable Serial-Interface Characteristics
    - 5-, 6-, 7-, or 8-Bit Characters
    - Even, Odd, or No Parity-bit Generation and Detection
    - 1-, 1.5-, or 2-Stop Bit Generation
  - Line Break Generation and Detection
  - Internal Test and Loopback Capabilities
  - Modem Control Functions (CTS, RTS, DSR, RI and DCD)
  - Internal Diagnostic Capability
    - Loopback Control for Communications Link-Fault Isolation
    - Break, Parity, Overrun, Framing-Error Simulation

1.2 Applications

- Modems
- Peripherals:
  Printers, Handheld Devices, and so on
- Medical Meters
- DSP and µC Interface

1.3 Description

The TUSB3410 device provides bridging between a USB port and an enhanced UART serial port. The device contains an 8052 microcontroller unit (MCU) with 16KB of RAM that can be loaded from the host or from the external onboard memory through an I²C. The device also contains 10KB of ROM that allows the MCU to configure the USB port at boot time. The ROM code also contains an I²C bootloader. All device functions (such as the USB command decoding, UART setup, and error reporting) are managed by the internal MCU firmware in unison with the PC host.

Device Information

PART NUMBER	PACKAGE	BODY SIZE
TUSB3410	VQFN (32)	5.00 mm × 5.00 mm
	LQFP (32)	7.00 mm × 7.00 mm

1. For all available packages, see the orderable addendum at the end of the data sheet.

1.4 Functional Block Diagram

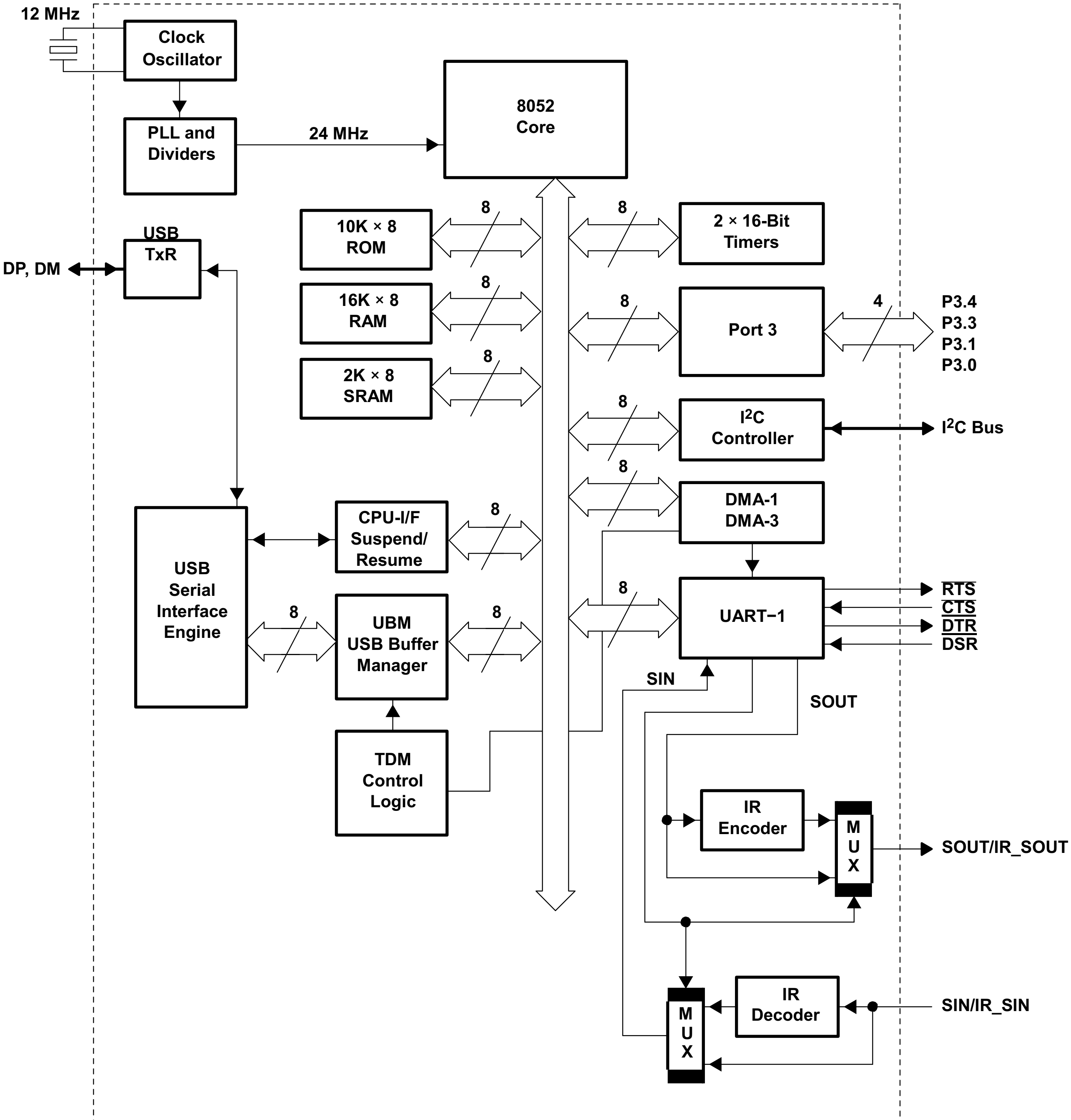


Table of Contents

2 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

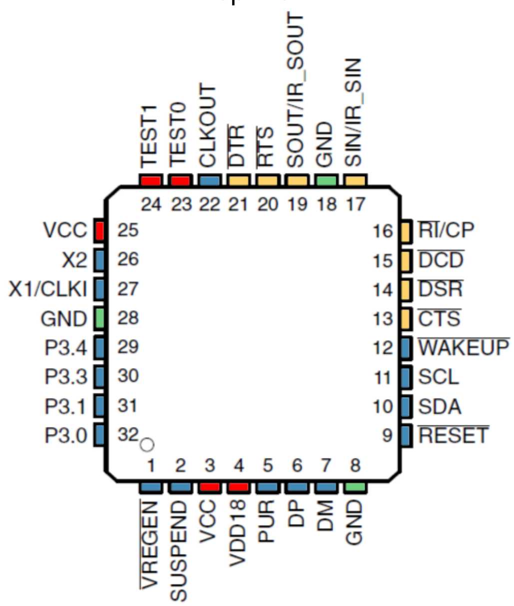
Changes from Revision I (November 2015) to Revision J	**Page**
• Changed pin 21 From: DTR To: active low $\overline{\text{DTR}}$ in the *Pin Functions* table	5
• Changed the description of bit 7 CONT in *USBCTL: USB Control Register (Addr:FFFCh)*, CONT= 0 From: enabled To: disables, CONT= 1 From: disbaled To: enabled	40

Changes from Revision H (April 2013) to Revision I	**Page**
• Added *Pin Configuration and Functions* section, *ESD Ratings* table, *Thermal Information* table, *Typical Characteristics* section, *Feature Description* section, *Device Functional Modes*, *Application and Implementation* section, *Power Supply Recommendations* section, *Layout* section, *Device and Documentation Support* section, and *Mechanical, Packaging, and Orderable Information* section	1
• Deleted *Ordering Information* table.	1

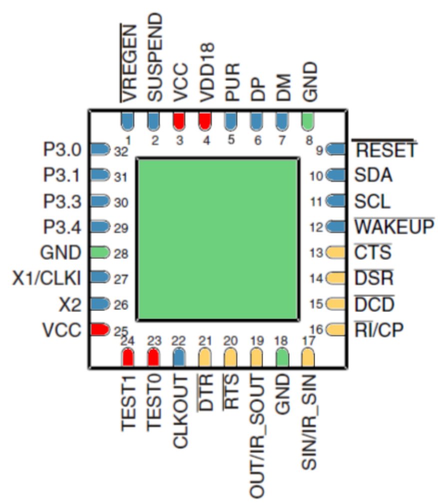
3 Pin Configuration and Functions

3.1 Pin Diagrams

RHB Package
32-Pin VQFN
Top View



VF Package
32-Pin LQFP
Bottom View



Pin Functions

PIN NAME	PIN NO.	I/O	DESCRIPTION
CLKOUT	22	O	Clock output (controlled by bits 2 (CLKOUTEN) and 3(CLKSLCT) in the MODECNFG register (see (1) and Section 5.5.5.5)
$\overline{\text{CTS}}$	13	I	UART: Clear to send(2)
$\overline{\text{DCD}}$	15	I	UART: Data carrier detect(2)
DM	7	I/O	Upstream USB port differential data minus
DP	6	I/O	Upstream USB port differential data plus
$\overline{\text{DSR}}$	14	I	UART: Data set ready(2)
$\overline{\text{DTR}}$	21	O	UART: Data terminal ready(1)
GND	8, 18, 28	GND	Digital ground
P3.0	32	I/O	General-purpose I/O 0 (port 3, terminal 0)(3)(4)(5)
P3.1	31	I/O	General-purpose I/O 1 (port 3, terminal 1)(3)(4)(5)
P3.3	30	I/O	General-purpose I/O 3 (port 3, terminal 3)(3)(4)(5)
P3.4	29	I/O	General-purpose I/O 4 (port 3, terminal 4)(3)(4)(5)
PUR	5	O	Pullup resistor connection(6)
$\overline{\text{RESET}}$	9	I	Device master reset input(2)
$\overline{\text{RI}}$/CP	16	I	UART: Ring indicator(2)
$\overline{\text{RTS}}$	20	O	UART: Request to send(1)
SCL	11	O	Master I^2C controller: clock signal(1)
SDA	10	I/O	Master I^2C controller: data signal(1)(4)
SIN/IR_SIN	17	I	UART: Serial input data / IR Serial data input(7)
SOUT/IR_SOUT	19	O	UART: Serial output data / IR Serial data output(8)
SUSPEND	2	O	Suspend indicator terminal(3). When this terminal is asserted high, the device is in suspend mode.
TEST0	23	I	Test input (for factory test only). This terminal must be tied to VCC through a 10-kΩ resistor.
TEST1	24	I	Test input (for factory test only)(4). This terminal must be tied to VCC through a 10-kΩ resistor.
VCC	3, 25	PWR	3.3 V
VDD18	4	PWR	1.8-V supply. An internal voltage regulator generates this supply voltage when terminal $\overline{\text{VREGEN}}$ is low. When $\overline{\text{VREGEN}}$ is high, 1.8 V must be supplied externally.
$\overline{\text{VREGEN}}$	1	I	This active-low terminal is used to enable the 3.3-V to 1.8-V voltage regulator.
$\overline{\text{WAKEUP}}$	12	I	Remote wake-up request terminal. When low, wakes up system(4)
X1/CLKI	27	I	12-MHz crystal input or clock input
X2	26	O	12-MHz crystal output

(1) 3-state CMOS output (±4-mA drive and sink)
(2) TTL-compatible, hysteresis input
(3) 3-state CMOS output (±12-mA drive and sink)
(4) TTL-compatible, hysteresis input, with internal 100-µA active pullup resistor
(5) The MCU treats the outputs as open drain types in that the output can be driven low continuously, but a high output is driven for two clock cycles and then the output is high impedance.
(6) 3-state CMOS output (±8-mA drive and sink)
(7) TTL-compatible input without hysteresis, with internal 100-µA active pullup resistor
(8) Normal or IR mode: 3-state CMOS output (±4-mA drive and sink)

4 Specifications

4.1 Absolute Maximum Ratings

over operating free-air temperature range (unless otherwise noted)[1]

			MIN	MAX	UNIT
V_{CC}	Supply voltage		−0.5	3.6	V
V_I	Input voltage		−0.5	V_{CC} + 0.5	V
V_O	Output voltage		−0.5	V_{CC} + 0.5	V
I_{IK}	Input clamp current			±20	mA
I_{OK}	Output clamp current			±20	mA
T_{stg}	Storage temperature	Industrial	–65	150	°C
		Standard	–55	150	

(1) Stresses beyond those listed under *Absolute Maximum Ratings* may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under *Recommended Operating Conditions* is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

4.2 ESD Ratings

				VALUE	UNIT
V_{ESD}	Electrostatic discharge (ESD) performance	Human Body Model (HBM), per ANSI/ESDA/JEDEC JS001[1]		±2000	V
		Charged Device Model (CDM), per JESD22-C101[2]	All pins	±500	V

(1) JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process.
(2) JEDEC document JEP157 states that 250-V CDM allows safe manufacturing with a standard ESD control process.

4.3 Recommended Operating Conditions

			MIN	TYP	MAX	UNIT
V_{CC}	Supply voltage		3	3.3	3.6	V
V_I	Input voltage		0		V_{CC}	V
V_{IH}	High-level input voltage	TTL	2		V_{CC}	V
		CMOS	0.7 × V_{CC}		V_{CC}	
V_{IL}	Low-level input voltage	TTL	0		0.8	V
		CMOS	0		0.2 × V_{CC}	
T_A	Operating temperature	Commercial range	0		70	°C
		Industrial range	–40		85	°C

4.4 Thermal Information

THERMAL METRIC[1]		TUSB3410 RHB (VQFN) 32 PINS	TUSB3410 VF (LQFP) 32 PINS	UNIT
$R_{\theta JA}$	Junction-to-ambient thermal resistance	32.1	70.5	°C/W
$R_{\theta JC(top)}$	Junction-to-case (top) thermal resistance	24.6	31.4	°C/W
$R_{\theta JB}$	Junction-to-board thermal resistance	6.5	28.3	°C/W
ψ_{JT}	Junction-to-top characterization parameter	0.2	2.2	°C/W
ψ_{JB}	Junction-to-board characterization parameter	6.5	28.2	°C/W
$R_{\theta JC(bot)}$	Junction-to-case (bottom) thermal resistance	24.6	31.4	°C/W

(1) For more information about traditional and new thermal metrics, see the *Semiconductor and C Package Thermal Metrics* application report.

4.5 Electrical Characteristics

T_A = 25°C, V_{CC} = 3.3 V ±5%, V_{SS} = 0 V

PARAMETER			TEST CONDITIONS	MIN	TYP	MAX	UNIT
V_{OH}	High-level output voltage	TTL	I_{OH} = –4 mA	V_{CC} – 0.5			V
		CMOS		V_{CC} – 0.5			
V_{OL}	Low-level output voltage	TTL	I_{OL} = 4 mA			0.5	V
		CMOS				0.5	
V_{IT+}	Positive threshold voltage	TTL	$V_I = V_{IH}$			1.8	V
		CMOS				0.7 × V_{CC}	
V_{IT-}	Negative threshold voltage	TTL	$V_I = V_{IH}$	0.8		1.8	V
		CMOS		0.2 × V_{CC}			
V_{hys}	Hysteresis ($V_{IT+} - V_{IT-}$)	TTL	$V_I = V_{IH}$	0.3		0.7	V
		CMOS		0.17 × V_{CC}		0.3 × V_{CC}	
I_{IH}	High-level input current	TTL	$V_I = V_{IH}$			±20	µA
		CMOS				±1	
I_{IL}	Low-level input current	TTL	$V_I = V_{IL}$			±20	µA
		CMOS				±1	
I_{OZ}	Output leakage current (Hi-Z)		$V_I = V_{CC}$ or V_{SS}			±20	µA
I_{OL}	Output low drive current			0.1			mA
I_{OH}	Output high drive current			0.1			mA
I_{CC}	Supply current (operating)		Serial data at 921.6 k			15	mA
	Supply current (suspended)					200	µA
	Clock duty cycle (1)				50%		
	Jitter specification (1)					±100	ppm
C_I	Input capacitance					18	pF
C_O	Output capacitance					10	pF

(1) Applies to all clock outputs

4.6 Timing and Switching Characteristics Information

4.6.1 Wakeup Timing ($\overline{\text{WAKEUP}}$ or $\overline{\text{RI}}$/CP Transitions)

The TUSB3410 device can be brought out of the suspended state, or woken up, by a command from the host. The TUSB3410 device also supports remote wakeup and can be awakened by either of two input signals. A low pulse on the $\overline{\text{WAKEUP}}$ terminal or a low-to-high transition on the $\overline{\text{RI}}$/CP terminal wakes up the device.

NOTE

For reliable operation, either condition must persist for approximately 3-ms minimum, which allows time for the crystal to power up because in the suspend mode, the crystal interface is powered down. The state of the $\overline{\text{WAKEUP}}$ or $\overline{\text{RI}}$/CP terminal is then sampled by the clock to verify there was a valid wake-up event.

4.6.2 Reset Timing

There are three requirements for the reset signal timing. First, the minimum reset pulse duration is 100 µs. At power up, this time is measured from the time the power ramps up to 90% of the nominal V_{CC} until the reset signal exceeds 1.2 V. The second requirement is that the clock must be valid during the last 60 µs of the reset window. The third requirement is that, according to the USB specification, the device must be ready to respond to the host within 100 ms. This means that within the 100-ms window, the device must come out of reset, load any pertinent data from the I^2C EEPROM device, and transfer execution to the application firmware if any is present. Because the latter two events can require significant time, the amount of which can change from system to system, TI recommends having the device come out of reset within 30 ms, leaving 70 ms for the other events to complete. This means the reset signal must rise to 1.8 V within 30 ms.

These requirements are depicted in Figure 4-1. When using a 12-MHz crystal, the clock signal may take several milliseconds to ramp up and become valid after power up. Therefore, the reset window may need to be elongated up to 10 ms or more to ensure that there is a 60-µs overlap with a valid clock.

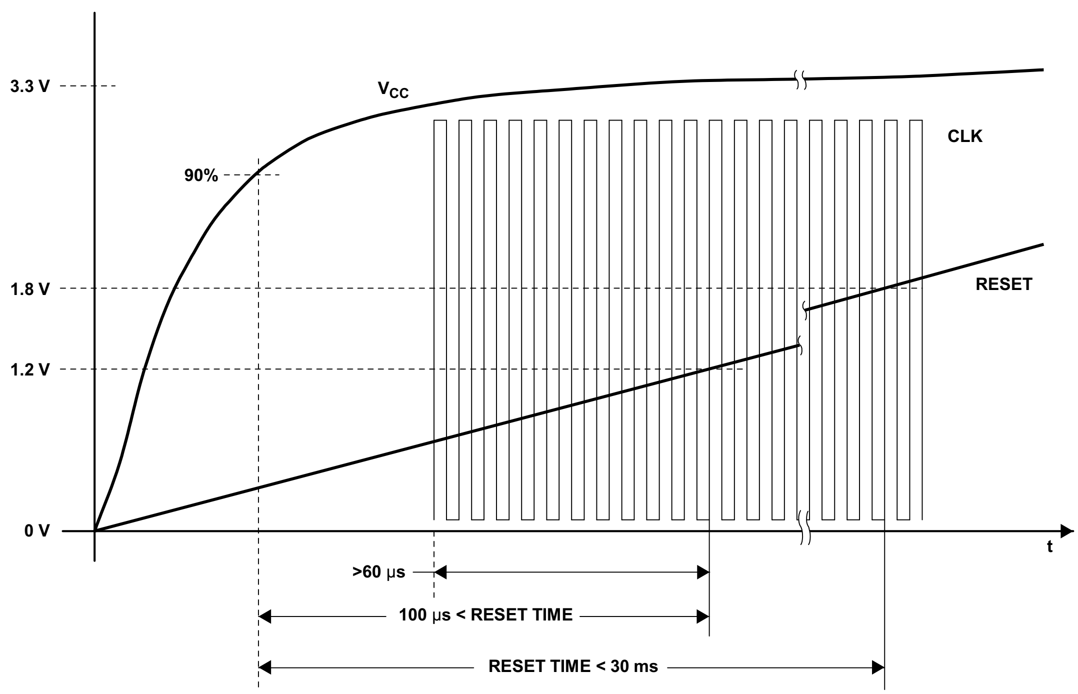


Figure 4-1. Reset Timing

4.7 Typical Characteristics

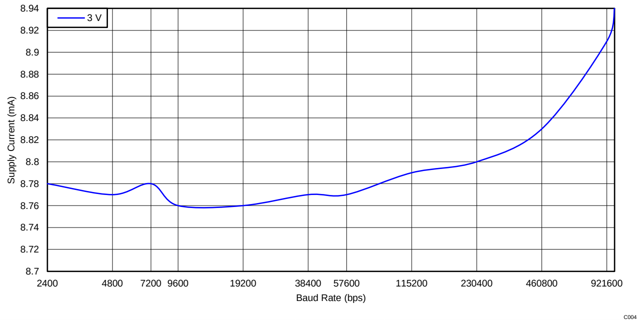


Figure 4-2. Supply Current at 3 V

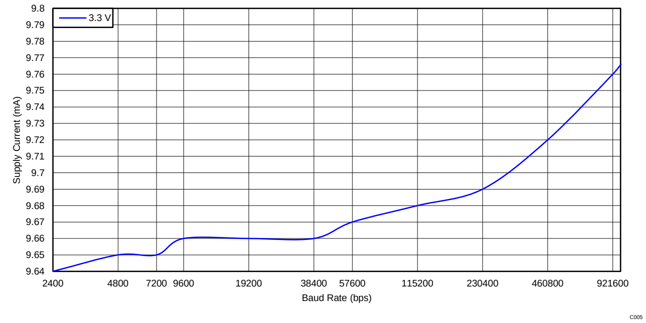


Figure 4-3. Supply Current at 3.3 V

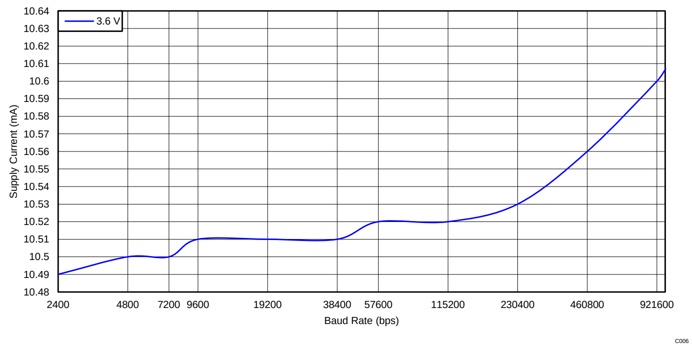


Figure 4-4. Supply Current at 3.6 V

5 Detailed Description

5.1 Overview

The TUSB3410 device provides bridging between a USB port and an enhanced UART serial port. The TUSB3410 device contains all the necessary logic to communicate with the host computer using the USB bus. It contains an 8052 microcontroller unit (MCU) with 16K bytes of RAM that can be loaded from the host or from the external on-board memory through an I^2C bus. It also contains 10K bytes of ROM that allow the MCU to configure the USB port at boot time. The ROM code also contains an I^2C bootloader. All device functions, such as the USB command decoding, UART setup, and error reporting, are managed by the internal MCU firmware under the auspices of the PC host.

The TUSB3410 device can be used to build an interface between a legacy serial peripheral device and a PC with USB ports, such as a legacy-free PC. When configured, data flows from the host to the TUSB3410 device through USB OUT commands and then out from the TUSB3410 device on the SOUT line. Conversely, data flows into the TUSB3410 device on the SIN line and then into the host through USB IN commands.

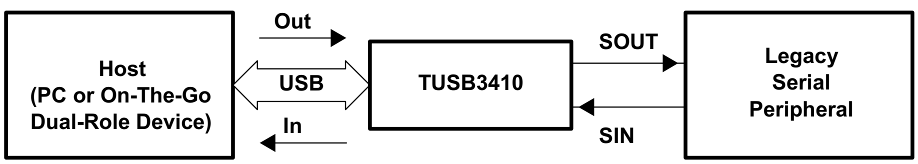


Figure 5-1. Data Flow

5.2 Functional Block Diagram

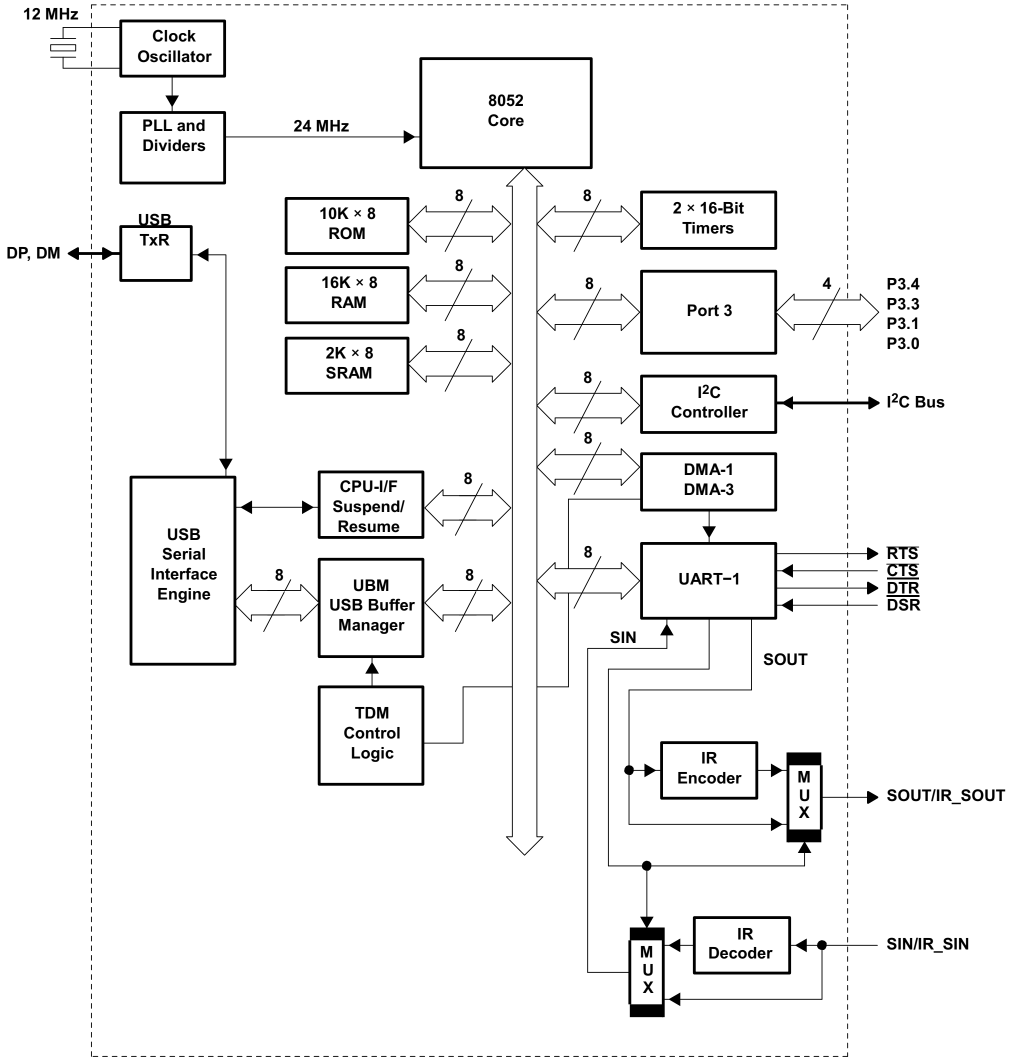


Figure 5-2. USB-to-Serial (Single Channel) Controller Block Diagram

5.3 Device Functional Modes

The TUSB3410 device controls its USB interface in response to USB commands, and this action is independent of the serial port mode selected. On the other hand, the serial port can be configured in three different modes.

As with any interface device, data movement is the main function of the TUSB3410 device, but typically the initial configuration and error handling consume most of the support code. The following sections describe the various modes the device can be used in and the means of configuring the device.

5.3.1 USB Interface Configuration

The TUSB3410 device contains onboard ROM microcode, which enables the MCU to enumerate the device as a USB peripheral. The ROM microcode can also load application code into internal RAM from either external memory through the I^{2}C bus or from the host through the USB.

5.3.1.1 External Memory Case

After reset, the TUSB3410 device is disconnected from the USB. Bit 7 (CONT) in the USBCTL register (see Section 5.5.5.4) is cleared. The TUSB3410 device checks the I^2C port for the existence of valid code; if it finds valid code, then the device uploads the code from the external memory device into the RAM program space. When loaded, the TUSB3410 device connects to the USB by setting the CONT bit; then, enumeration and configuration are performed. This is the most likely use of the device.

5.3.1.2 Host Download Case

If the valid code is not found at the I^2C port, then the TUSB3410 device connects to the USB by setting bit 7 (CONT) in the USBCTL register (see Section 5.5.5.4), and then an enumeration and default configuration are performed. The host can download additional microcode into RAM to tailor the application. Then, the MCU causes a disconnect and reconnect by clearing and setting the CONT bit, which causes the TUSB3410 device to be re-enumerated with a new configuration.

5.3.2 USB Data Movement

From the USB perspective, the TUSB3410 device looks like a USB peripheral device. It uses endpoint zero as its control endpoint, as do all USB peripherals. It also configures up to three input and three output endpoints, although most applications use one bulk input endpoint for data in, one bulk output endpoint for data out, and one interrupt endpoint for status updates. The USB configuration likely remains the same regardless of the serial port configuration.

Most data is moved from the USB side to the UART side and from the UART side to the USB side using on-chip DMA transfers. Some special cases may use programmed I/O under control of the MCU.

5.3.3 Serial Port Setup

The serial port requires a few control registers to be written to configure its operation. This configuration likely remains the same regardless of the data mode used. These registers include the line control register that controls the serial word format and the divisor registers that control the baud rate.

These registers are usually controlled by the host application.

5.3.4 Serial Port Data Modes

The serial port can be configured in three different, although similar, data modes: the RS-232 data mode, the RS-485 data mode, and the IrDA data mode. Similar to the USB mode, when configured for a specific application, it is unlikely that the mode would be changed. The different modes affect the timing of the serial input and output or the use of the control signals. However, the basic serial-to-parallel conversion of the receiver and parallel-to-serial conversion of the transmitter remain the same in all modes. Some features are available in all modes, but are only applicable in certain modes. For instance, software flow control through Xoff/Xon characters can be used in all modes, but would usually only be used in RS-232 or IrDA mode because the RS-485 mode is half-duplex communication. Similarly, hardware flow control through $\overline{RTS}$/$\overline{CTS}$ (or $\overline{DTR}$/$\overline{DSR}$) handshaking is available in RS-232 or IrDA mode. However, this would probably be used only in RS-232 mode, because in IrDA mode only the SIN and SOUT paths are optically coupled.

5.3.4.1 RS-232 Data Mode

The default mode is called the RS-232 mode and is typically used for full duplex communication on SOUT and SIN. In this mode, the modem control outputs ($\overline{RTS}$ and $\overline{DTR}$) communicate to a modem or are general outputs. The modem control inputs ($\overline{CTS}$, $\overline{DSR}$, $\overline{DCD}$, and $\overline{RI}$/CP) communicate to a modem or are general inputs. Alternatively, $\overline{RTS}$ and $\overline{CTS}$ (or $\overline{DTR}$ and $\overline{DSR}$) can throttle the data flow on SOUT and SIN to prevent receive FIFO overruns. Finally, software flow control through Xoff/Xon characters can be used for the same purpose (see Section 5.2).

This mode represents the most general-purpose applications, and the other modes are subsets of this mode.

5.3.4.2 RS-485 Data Mode

The RS-485 mode is very similar to the RS-232 mode in that the SOUT and SIN formats remain the same. Because RS-485 is a bus architecture, it is inherently a single duplex communication system. The TUSB3410 device in RS-485 mode controls the $\overline{\text{RTS}}$ and $\overline{\text{DTR}}$ signals such that either can enable an RS-485 driver or RS-485 receiver. When in RS-485 mode, the enable signals for transmitting are automatically asserted whenever the DMA is set up for outbound data.

NOTE

The receiver can be left enabled while the driver is enabled to allow an echo if desired, but when receive data is expected, the driver must be disabled. This precludes use of hardware flow control, because this is a half-duplex operation, it would not be effective. Software flow control is supported, but may be of limited value.

The RS-485 mode is enabled by setting bit 7 (485E) in the FCRL register (see Section 5.5.7.4), and bit 1 (RCVE) in the MCR register (see Section 5.5.7.6) allows the receiver to eavesdrop while in the RS-485 mode.

5.3.4.3 IrDA Data Mode

The IrDA mode encodes SOUT and decodes SIN in the manner prescribed by the IrDA standard, up to 115.2 kbps. Connection to an external IrDA transceiver is required. Communications is usually full duplex. Generally, in an IrDA system, only the SOUT and SIN paths are connected so hardware flow control is usually not an option. Software flow control is supported (see Section 5.2).

The IrDA mode is enabled by setting bit 6 (IREN) in the USBCTL register (see Section 5.5.5.4).

The IR encoder and decoder circuitry work with the UART to change the serial bit stream into a series of pulses and back again. For every zero bit in the outbound serial stream, the encoder sends a low-to-high-to-low pulse with the duration of 3/16 of a bit frame at the middle of the bit time. For every one bit in the serial stream, the output remains low for the entire bit time.

The decoding process consists of receiving the signal from the IrDA receiver and converting it into a series of zeroes and ones. As the converse to the encoder, the decoder converts a pulse to a zero bit and the lack of a pulse to a one bit.

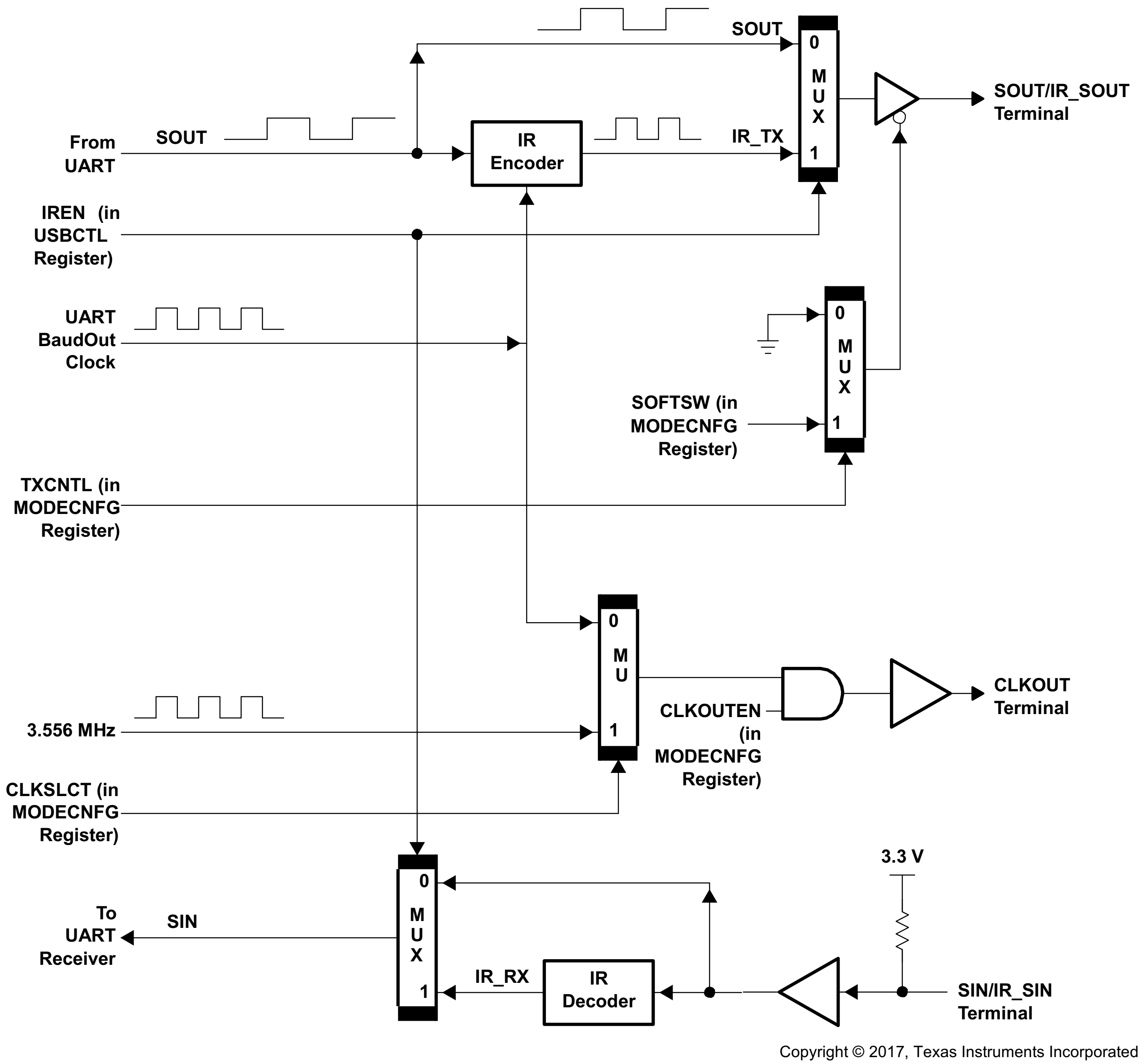


Figure 5-3. RS-232 and IR Mode Select

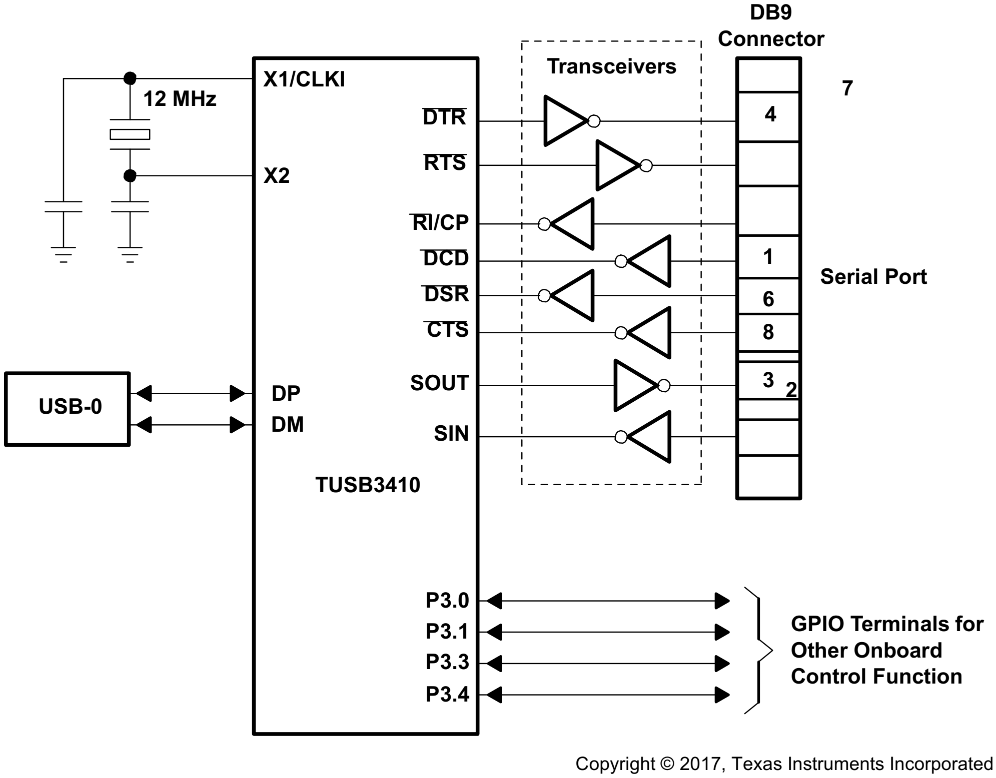


Figure 5-4. USB-to-Serial Implementation (RS-232)

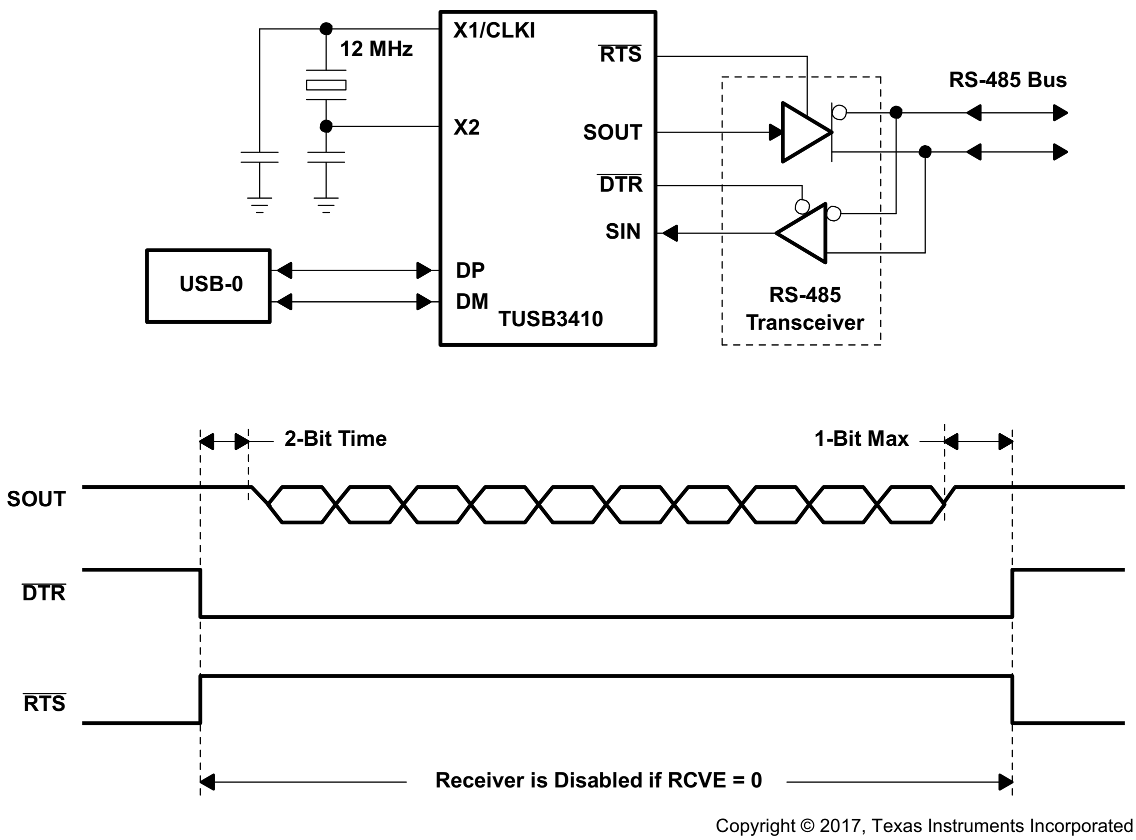


Figure 5-5. RS-485 Bus Implementation

5.4 Processor Subsystems

5.4.1 DMA Controller

5.4.1.1 Bulk Data I/O Using the EDB

The UBM (USB buffer manager) and the DMAC (DMA controller) access the EDB to fetch buffer parameters for IN and OUT transactions (IN and OUT are with respect to host). In this discussion, it is assumed that:

- The MCU initialized the EDBs
- DMA-continuous mode is being used
- Double buffering is being used
- The X/Y toggle is controlled by the UBM

5.4.1.1.1 IN Transaction (TUSB3410 to Host)

1. The MCU initializes the IEDB (64-byte packet, and double buffering is used) and the following DMA registers:
   - **DMACSR3:** Defines the transaction time-out value.
   - **DMACDR3:** Defines the IEDB being used and the DMA mode of operation (continuous mode). Once this register is set with EN = 1, the transfer starts.
2. The DMA transfers data from the UART to the X buffer. When a block of 64 bytes is transferred, the DMA updates the byte count and sets NAK to 0 in the input endpoint byte count register (indicating to the UBM that the X buffer is ready to be transferred to host). The UBM starts X-buffer transfer to host using the byte-count value in the input endpoint byte count register and toggles the X/Y bit. The DMA continues transferring data from a device to Y buffer. At the end of the block transfer, the DMA updates the byte count and sets NAK to 0 in the input endpoint byte count register (indicating to the UBM that the Y buffer is ready to be transferred to host). The DMA continues the transfer from the device to host, alternating between X and Y buffers without MCU intervention.
3. Transfer termination: The DMA/UBM continues the data transfer, alternating between the X and Y buffers. Termination of the transfer can happen under the following conditions:
   - **Stop Transfer:** The host notifies the MCU (through control-end-point) to stop the transfer. Under this condition, the MCU sets bit 7 (EN) to 0 in the DMACDR register.
   - **Partial Packet:** The device receiver has no data to be transferred to host. Under this condition, the byte-count value is less than 64 when the transaction timer time-out occurs. When the DMA detects this condition, it sets bit 1 (TXFT) to 1 and bit 0 (OVRUN) to 0 in the DMACSR3 register, updates the byte count and NAK bit in the input endpoint byte count register, and interrupts the MCU. The UBM transfers the partial packet to host.
   - **Buffer Overrun:** The host is busy, X and Y buffers are full (X-NAK = 0 and Y-NAK = 0), and the DMA cannot write to these buffers. The transaction time-out stops the DMA transfer, the DMA sets bit 1 (TXFT) to 1 and bit 0 (OVRUN) to 1 in the DMACSR3 register, and interrupts the MCU.
   - **UART Error Condition:** When receiving from a UART, a receiver-error condition stops the DMA and sets bit 1 (TXFT) to 1 and bit 0 (OVRUN) to 0 in the DMACSR3 register, but the EN bit remains set at 1. Therefore, the DMA does not interrupt the MCU. However, the UART generates a status interrupt, notifying the MCU that an error condition has occurred.

5.4.1.1.2 OUT Transaction (Host to TUSB3410)

1. The MCU initializes the OEDB (64-byte packet, and double buffering is used) and the following DMA registers:
   – **DMACSR1:** Provides an indication of a partial packet.
   – **DMACDR1:** Defines the output endpoint being used, and the DMA mode of operation (continuous mode). When the EN bit is set to 1 in this register, the transfer starts.
2. The UBM transfers data from host to X buffer. When a block of 64 bytes is transferred, the UBM updates the byte count and sets NAK to 1 in the output endpoint byte count register (indicating to DMA that the X buffer is ready to be transferred to the UART). The DMA starts X buffer transfer using the byte-count value in the output endpoint byte count register. The UBM continues transferring data from host to Y buffer. At the end of the block transfer, the UBM updates the byte count and sets NAK to 1 in the output endpoint byte count register (indicating to DMA that the Y buffer is ready to be transferred to device). The DMA continues the transfer from the X and Y buffers to the device, alternating between X and Y buffers without MCU intervention.
3. Transfer termination: The DMA/UBM continues the data transfer alternating between X and Y buffers. The termination of the transfer can happen under the following conditions:
   – **Stop Transfer:** The host notifies the MCU (through control-end point) to stop the transfer. Under this condition, the MCU sets EN to 0 in the DMACDR1 register.
   – **Partial Packet:** UBM receives a partial packet from host. Under this condition, the byte-count value is less than 64. When the DMA detects this condition, it transfers the partial packet to the device, sets PPKT to 1, updates NAK to 0 in the output endpoint byte count register and interrupts MCU.

5.4.2 UART

5.4.2.1 UART Data Transfer

Figure 5-6 illustrates the data transfer between the UART and the host using the DMA controller and the USB buffer manager (UBM). A buffer of 512 bytes is reserved for buffering the UART channel (transmit and receive buffers). The UART channel has 64 bytes of double-buffer space (X and Y buffer). When the DMA writes to the X buffer, the UBM reads from the Y buffer. Similarly, when the DMA reads from the X buffer, the UBM writes to the Y buffer. The DMA channel is configured to operate in the continuous mode (by setting bit 5 (CNT) in the DMACDR registers = 1). Once the MCU enables the DMA, data transfer toggles between the UMB and the DMA without MCU intervention. See Section 5.4.1.1.1 for DMA transfer-termination condition.

5.4.2.1.1 Receiver Data Flow

The UART receiver has a 32-byte FIFO. The receiver FIFO has two trigger levels. One is the high-level mark (HALT), which is set to 12 bytes, and the other is the low-level mark (RESUME), which is set to 4 bytes. When the HALT mark is reached, either the $\overline{\text{RTS}}$ terminal goes high or Xoff is transmitted (depending on the auto setting). When the FIFO reaches the RESUME mark, then either the $\overline{\text{RTS}}$ terminal goes low or Xon is transmitted.

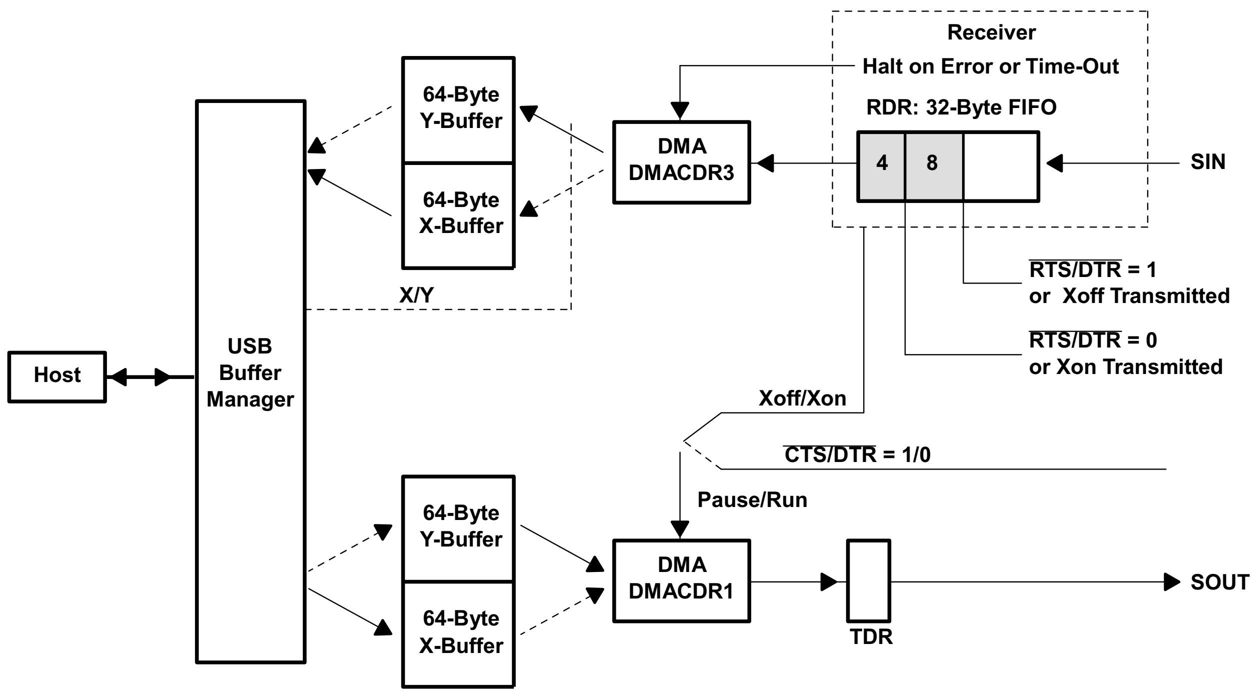


Figure 5-6. Receiver and Transmitter Data Flow

5.4.2.1.2 Hardware Flow Control

Figure 5-7 illustrates the connection necessary to achieve hardware flow control. The $\overline{\text{CTS}}$ and $\overline{\text{RTS}}$ signals are provided for this purpose. Auto $\overline{\text{CTS}}$ and auto $\overline{\text{RTS}}$ (and Xon/Xoff) can be enabled and disabled independently by programming the UART flow control register (FCRL).

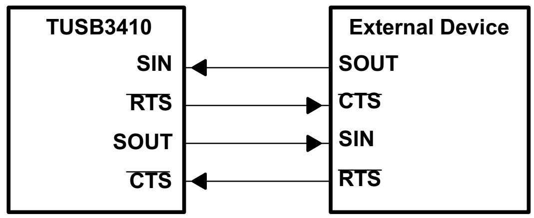


Figure 5-7. Auto Flow Control Interconnect

5.4.2.1.3 Auto $\overline{\text{RTS}}$ (Receiver Control)

In this mode, the $\overline{\text{RTS}}$ output terminal signals the receiver-FIFO status to an external device. The $\overline{\text{RTS}}$ output signal is controlled by the high- and low-level marks of the FIFO. When the high-level mark is reached, $\overline{\text{RTS}}$ goes high, signaling to an external sending device to halt its transfer. Conversely, when the low-level mark is reached, $\overline{\text{RTS}}$ goes low, signaling to an external sending device to resume its transfer.

Data transfer from the FIFO to the X and Y buffer is performed by the DMA controller. See Section 5.4.1.1.1 for DMA transfer-termination condition.

5.4.2.1.4 Auto $\overline{\text{CTS}}$ (Transmitter Control)

In this mode, the $\overline{\text{CTS}}$ input terminal controls the transfer from internal buffer (X or Y) to the TDR. When the DMA controller transfers data from the Y buffer to the TDR and the $\overline{\text{CTS}}$ input terminal goes high, the DMA controller is suspended until $\overline{\text{CTS}}$ goes low. Meanwhile, the UBM is transferring data from the host to the X buffer. When $\overline{\text{CTS}}$ goes low, the DMA resumes the transfer. Data transfer continues alternating between the X and Y buffers, without MCU intervention. See Section 5.4.1.1.2 for DMA transfer-termination condition.

5.4.2.1.5 Xon/Xoff Receiver Flow Control

To enable Xon/Xoff flow control, certain bits within the modem control register must be set as follows: MCR bit 5 = 1 and MCR bits 6 and 7 = 00. In this mode, the Xon/Xoff bytes are transmitted to an external sending device to control the transmission of the device. When the high-level mark (of the FIFO) is reached, the Xoff byte is transmitted, signaling to an external sending device to halt its transfer. Conversely, when the low-level mark is reached, the Xon byte is transmitted, signaling to an external sending device to resume its transfer. The data transfer from the FIFO to X and Y buffer is performed by the DMA controller.

5.4.2.1.6 Xon/Xoff Transmit Flow Control

To enable Xon/Xoff flow control, certain bits within the modem control register must be set as follows: MCR bit 5 = 1 and MCR bits 6 and 7 = 00. In this mode, the incoming data are compared to the XON and XOFF registers. If a match to XOFF is detected, the DMA is paused. If a match to XON is detected, the DMA resumes. Meanwhile, the UBM is transferring data from the host to the X-buffer. The MCU does not switch the buffers unless the Y buffer is empty and the X-buffer is full. When Xon is detected, the DMA resumes the transfer.

5.4.3 I²C Port

5.4.3.1 Random-Read Operation

A random read requires a dummy byte-write sequence to load in the data word address. Once the device-address word and the data-word address are clocked out and acknowledged by the device, the MCU starts a current-address sequence. The following describes the sequence of events to accomplish this transaction.

5.4.3.1.1 Device Address + EPROM [High Byte]

1. The MCU clears bit 1 (SRD) within the I2CSTA register. This forces the I²C controller not to generate a stop condition after the contents of the I2CDAI register are received.
2. The MCU clears bit 0 (SWR) within the I2CSTA register. This forces the I²C controller not to generate a stop condition after the contents of the I2CDAO register are transmitted.
3. The MCU writes the device address (bit 0 (R/W) = 0) to the I2CADR register (write operation)
4. The MCU writes the high byte of the EEPROM address into the I2CDAO register (this starts the transfer on the SDA line).
5. Bit 3 (TXE) in the I2CSTA register is automatically cleared (indicates busy) by writing data to the I2CDAO register.
6. The contents of the I2CADR register are transmitted to EEPROM (preceded by start condition on SDA).
7. The contents of the I2CDAO register are transmitted to EEPROM (EPROM address).
8. Bit 3 (TXE) in the I2CSTA register is set and interrupts the MCU, indicating that the I2CDAO register has been transmitted.
9. A stop condition is not generated.

5.4.3.1.2 EPROM [Low Byte]

1. The MCU writes the low byte of the EEPROM address into the I2CDAO register.
2. Bit 3 (TXE) in the I2CSTA register is automatically cleared (indicates busy) by writing to the I2CDAO register.
3. The contents of the I2CDAO register are transmitted to the device (EEPROM address).
4. Bit 3 (TXE) in the I2CSTA register is set and interrupts the MCU, indicating that the I2CDAO register has been transmitted.
5. This completes the dummy write operation. At this point, the EEPROM address is set and the MCU can do either a single- or a sequential-read operation.

5.4.3.2 Current-Address Read Operation

When the EEPROM address is set, the MCU can read a single byte by executing the following steps:

1. The MCU sets bit 1 (SRD) in the I2CSTA register to 1. This forces the I²C controller to generate a stop condition after the I2CDAI-register contents are received.
2. The MCU writes the device address (bit 0 (R/W) = 1) to the I2CADR register (read operation).
3. The MCU writes a dummy byte to the I2CDAO register (this starts the transfer on SDA line).
4. Bit 7 (RXF) in the I2CSTA register is cleared (RX is empty).
5. The contents of the I2CADR register are transmitted to the device (preceded by start condition on SDA).
6. The data from EEPROM are latched into the I2CDAI register (stop condition is transmitted).
7. Bit 7 (RXF) in the I2CSTA register is set and interrupts the MCU, indicating that the data are available.
8. The MCU reads the I2CDAI register. This clears bit 7 (RXF) in the I2CSTA register.

5.4.3.3 Sequential-Read Operation

When the EEPROM address is set, the MCU can execute a sequential read operation by executing the following steps (this example illustrates a 32-byte sequential read):

5.4.3.3.1 Device Address

1. The MCU clears bit 1 (SRD) in the I2CSTA register. This forces the I^{2}C controller to not generate a stop condition after the I2CDAI register contents are received.
2. The MCU writes the device address (bit 0 (R/W) = 1) to the I2CADR register (read operation).
3. The MCU writes a dummy byte to the I2CDAO register (this starts the transfer on the SDA line).
4. Bit 7 (RXF) in the I2CSTA register is cleared (RX is empty).
5. The contents of the I2CADR register are transmitted to the device (preceded by start condition on SDA).

5.4.3.3.2 N-Byte Read (31 Bytes)

1. The data from the device is latched into the I2CDAI register (stop condition is not transmitted).
2. Bit 7 (RXF) in the I2CSTA register is set and interrupts the MCU, indicating that data is available.
3. The MCU reads the I2CDAI register. This clears bit 7 (RXF) in the I2CSTA register.
4. This operation repeats 31 times.

5.4.3.3.3 Last-Byte Read (Byte 32)

1. MCU sets bit 1 (SRD) in the I2STA register to 1. This forces the I^{2}C controller to generate a stop condition after the I2CDAI register contents are received.
2. The data from the device is latched into the I2CDAI register (stop condition is transmitted).
3. Bit 7 (RXF) in the I2CSTA register is set and interrupts the MCU, indicating that data is available.
4. The MCU reads the I2CDAI register. This clears bit 7 (RXF) in the I2CSTA register.

5.4.3.4 Byte-Write Operation

The byte-write operation involves three phases: device address + EPROM [high byte] phase, EPROM [low byte] phase, and EPROM [DATA] phase. The following describes the sequence of events to accomplish the byte-write transaction.

5.4.3.4.1 Device Address + EPROM [High Byte]

1. The MCU sets clears the SWR bit in the I2CSTA register. This forces the I^{2}C controller to not generate a stop condition after the contents of the I2CDAO register are transmitted.
2. The MCU writes the device address (bit 0 (R/W) = 0) to the I2CADR register (write operation).
3. The MCU writes the high byte of the EEPROM address into the I2CDAO register (this starts the transfer on the SDA line).
4. Bit 3 (TXE) in the I2CSTA register is cleared (indicates busy).
5. The contents of the I2CADR register are transmitted to the device (preceded by start condition on SDA).
6. The contents of the I2CDAO register are transmitted to the device (EEPROM high address).
7. Bit 3 (TXE) in the I2CSTA register is set and interrupts the MCU, indicating that the I2CDAO register contents have been transmitted.

5.4.3.4.2 EPROM [Low Byte]

1. The MCU writes the low byte of the EEPROM address into the I2CDAO register.
2. Bit 3 (TXE) in the I2CSTA register is cleared (indicating busy).
3. The contents of the I2CDAO register are transmitted to the device (EEPROM address).
4. Bit 3 (TXE) in the I2CSTA register is set and interrupts the MCU, indicating that the I2CDAO register contents have been transmitted.

5.4.3.4.3 EPROM [DATA]

1. The MCU sets bit 0 (SWR) in the I2CSTA register. This forces the I^2C controller to generate a stop condition after the contents of the I2CDAO register are transmitted.
2. The data to be written to the EPROM is written by the MCU into the I2CDAO register.
3. Bit 3 (TXE) in the I2CSTA register is cleared (indicates busy).
4. The contents of the I2CDAO register are transmitted to the device (EEPROM data).
5. Bit 3 (TXE) in the I2CSTA register is set and interrupts the MCU, indicating that the I2CDAO register contents have been transmitted.
6. The I^2C controller generates a stop condition after the contents of the I2CDAO register are transmitted.

5.4.3.5 Page-Write Operation

The page-write operation is initiated in the same way as byte write, with the exception that a stop condition is not generated after the first EPROM [DATA] is transmitted. The following describes the sequence of writing 32 bytes in page mode.

5.4.3.5.1 Device Address + EPROM [High Byte]

1. The MCU clears bit 0 (SWR) in the I2CSTA register. This forces the I^2C controller to not generate a stop condition after the contents of the I2CDAO register are transmitted.
2. The MCU writes the device address (bit 0 (R/W) = 0) to the I2CADR register (write operation).
3. The MCU writes the high byte of the EEPROM address into the I2CDAO register.
4. Bit 3 (TXE) in the I2CSTA register is cleared (indicating busy).
5. The contents of the I2CADR register are transmitted to the device (preceded by start condition on SDA).
6. The contents of the I2CDAO register are transmitted to the device (EEPROM address).
7. Bit 3 (TXE) in the I2CSTA register is set and interrupts the MCU, indicating that the I2CDAO register contents have been transmitted.

5.4.3.5.2 EPROM [Low Byte]

1. The MCU writes the low byte of the EEPROM address into the I2CDAO register.
2. Bit 3 (TXE) in the I2CSTA register is cleared (indicates busy).
3. The contents of the I2CDAO register are transmitted to the device (EEPROM address).
4. Bit 3 (TXE) in the I2CSTA register is set and interrupts the MCU, indicating that the I2CDAO register contents have been transmitted.

5.4.3.5.3 EPROM [DATA]—31 Bytes

1. The data to be written to the EEPROM are written by the MCU into the I2CDAO register.
2. Bit 3 (TXE) in the I2CSTA register is cleared (indicates busy).
3. The contents of the I2CDAO register are transmitted to the device (EEPROM data).
4. Bit 3 (TXE) in the I2CSTA register is set and interrupts the MCU, indicating that the I2CDAO register contents have been transmitted.
5. This operation repeats 31 times.

5.4.3.5.4 EPROM [DATA]—Last Byte

1. The MCU sets bit 0 (SWR) in the I2CSTA register. This forces the I^2C controller to generate a stop condition after the contents of the I2CDAO register are transmitted.
2. The MCU writes the last date byte to be written to the EEPROM, into the I2CDAO register.
3. Bit 3 (TXE) in the I2CSTA register is cleared (indicates busy).
4. The contents of the I2CDAO register are transmitted to EEPROM (EEPROM data).
5. Bit 3 (TXE) in the I2CSTA register is set and interrupts the MCU, indicating that the I2CDAO register contents have been transmitted.
6. The I^2C controller generates a stop condition after the contents of the I2CDAO register are transmitted.

5.5 Memory

5.5.1 MCU Memory Map

Figure 5-8 illustrates the MCU memory map under boot and normal operation.

NOTE

The internal 256 bytes of RAM are not shown, because they are assumed to be in the standard 8052 location (0000h to 00FFh). The shaded areas represent the internal ROM/RAM.

- **When bit 0 (SDW) of the ROMS register is 0 (boot mode)**

  The 10K ROM is mapped to address (0x0000−0x27FF) and is duplicated in location (0x8000−0xA7FF) in code space. The internal 16K RAM is mapped to address range (0x0000−0x3FFF) in data space. Buffers, MMR, and I/O are mapped to address range (0xF800−0xFFFF) in data space.
- **When bit 0 (SDW) is 1 (normal mode)**

  The 10K ROM is mapped to (0x8000−0xA7FF) in code space. The internal 16K RAM is mapped to address range (0x0000−0x3FFF) in code space. Buffers, MMR, and I/O are mapped to address range (0xF800−0xFFFF) in data space.

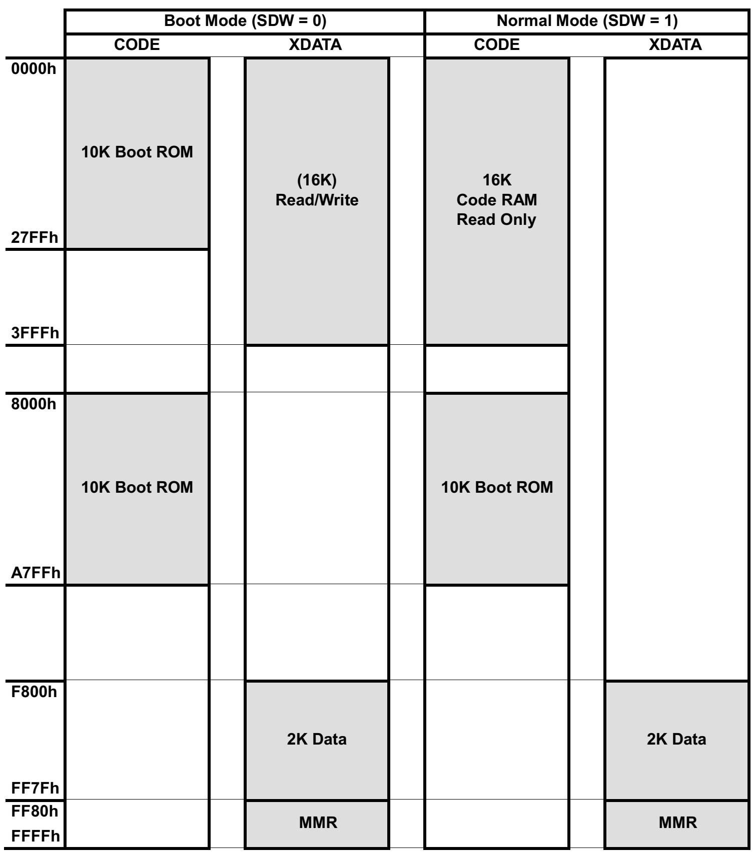


Figure 5-8. MCU Memory Map

5.5.2 Registers

5.5.2.1 Miscellaneous Registers

5.5.2.1.1 ROMS: ROM Shadow Configuration Register (Addr:FF90h)

This register is used by the MCU to switch from boot mode to normal operation mode (boot mode is set on power-on reset only). In addition, this register provides the device revision number and the ROM/RAM configuration.

7	6	5	4	3	2	1	0
ROA	S1	S0	RSVD	RSVD	RSVD	RSVD	SDW
R/O	R/O	R/O	R/O	R/O	R/O	R/O	R/W

BIT	NAME	RESET	FUNCTION
0	SDW	0	This bit enables/disables boot ROM. (Shadow the ROM). SDW = 0 When clear, the MCU executes from the 10K boot ROM space. The boot ROM appears in two locations: 0000h and 8000h. The 16K RAM is mapped to XDATA space; therefore, a read/write operation is possible. This bit is set by the MCU after the RAM load is completed. The MCU cannot clear this bit; it is cleared on power-up reset or watchdog time-out reset. SDW = 1 When set by the MCU, the 10K boot ROM maps to location 8000h, and the 16K RAM is mapped to code space, starting at location 0000h. At this point, the MCU executes from RAM, and the write operation is disabled (no write operation is possible in code space).
4−1	RSVD	No effect	These bits are always read as 0000b.
6−5	S[1:0]	No effect	Code space size. These bits define the ROM or RAM code-space size (bit 7 (ROA) defines ROM or RAM). These bits are permanently set to 10b, indicating 16K bytes of code space, and are not affected by reset (see Table 5-1). 00 = 4K bytes code space size 01 = 8K bytes code space size 10 = 16K bytes code space size 11 = 32K bytes code space size
7	ROA	No effect	ROM or RAM version. This bit indicates whether the code space is RAM or ROM based. This bit is permanently set to 1, indicating the code space is RAM, and is not affected by reset (see Table 5-1). ROA = 0 Code space is ROM ROA = 1 Code space is RAM

Table 5-1. ROM and RAM Size Definition Table

ROMS REGISTER			BOOT ROM	RAM CODE	ROM CODE
ROA	S1	S0			
0	0	0	None	None	4K
0	0	1	None	None	8K
0	1	0	None	None	16K (reserved)
1	1	1	None	None	32K (reserved)
1	0	0	10K	4K	None
1	0	1	10K	8K	None
1 (1)	1 (1)	0 (1)	10K (1)	16K (1)	None (1)
1	1	1	10K	32K (reserved)	None

(1) This is the hardwired setting.

5.5.2.1.2 Boot Operation (MCU Firmware Loading)

Because the code space is in RAM (with the exception of the boot ROM), the TUSB3410 firmware must be loaded from an external source. Two sources are available for booting: one from an external serial EEPROM connected to the I^2C bus and the other from the host through the USB. On device reset, bit 0 (SDW) in the ROMS register (see Section 5.5.2.1.1) and bit 7 (CONT) in the USBCTL register (see Section 5.5.5.4) are cleared. This configures the memory space to boot mode (see Table 5-3) and keeps the device disconnected from the host. The first instruction is fetched from location 0000h (which is in the 10K ROM). The 16K RAM is mapped to XDATA space (location 0000h). The MCU executes a read from an external EEPROM and tests whether it contains the code (by testing for boot signature). If it contains the code, then the MCU reads from EEPROM and writes to the 16K RAM in XDATA space. If it does not contain the code, then the MCU proceeds to boot from the USB.

When the code is loaded, the MCU sets the SDW bit to 1 in the ROMS register. This switches the memory map to normal mode; that is, the 16K RAM is mapped to code space, and the MCU starts executing from location 0000h. When the switch is done, the MCU sets the CONT bit to 1 in the USBCTL register. This connects the device to the USB and results in normal USB device enumeration.

5.5.2.1.3 WDCSR: Watchdog Timer, Control, and Status Register (Addr:FF93h)

A watchdog timer (WDT) with 1-ms clock is provided. If this register is not accessed for a period of 128 ms, then the WDT counter resets the MCU (see Figure 5-9). The watchdog timer is enabled by default and can be disabled by writing a pattern of 101010b into the WDD[5:0] bits. The 1-ms clock for the watchdog timer is generated from the SOF pulses. Therefore, for the watchdog timer to count, bit 7 (CONT) in the USBCTL register (see Section 5.5.5.4) must be set.

7	6	5	4	3	2	1	0
WDD0	WDR	WDD5	WDD4	WDD3	WDD2	WDD1	WDT
R/W	R/C	R/W	R/W	R/W	R/W	R/W	W/O

BIT	NAME	RESET	FUNCTION
0	WDT	0	MCU must write a 1 to this bit to prevent the watchdog timer from resetting the MCU. If the MCU does not write a 1 in a period of 128 ms, the watchdog timer resets the device. Writing a 0 has no effect on the watchdog timer. (The watchdog timer is a 7-bit counter using a 1-ms CLK.) This bit is read as 0.
5−1	WDD[5:1]	00000	These bits disable the watchdog timer. For the timer to be disabled these bits must be set to 10101b and bit 7 (WDD0) must also be set to 0. If any other pattern is present, then the watchdog timer is in operation.
6	WDR	0	Watchdog reset indication bit. This bit indicates if the reset occurred due to power-on reset or watchdog timer reset. WDR = 0 A power-up reset occurred WDR = 1 A watchdog time-out reset occurred. To clear this bit, the MCU must write a 1. Writing a 0 has no effect.
7	WDD0	1	This bit is one of the six disable bits for the watchdog timer. This bit must be cleared in order for the watchdog timer to be disabled.

5.5.3 Buffers + I/O RAM Map

The address range from F800h to FFFFh (2K bytes) is reserved for data buffers, setup packet, endpoint descriptors block (EDB), and all I/O. There are 128 locations reserved for memory-mapped registers (MMR). Table 5-2 represents the XDATA space allocation and access restriction for the DMA, USB buffer manager (UBM), and MCU.

Table 5-2. XDATA Space

DESCRIPTION	ADDRESS RANGE	UBM ACCESS	DMA ACCESS	MCU ACCESS
Internal MMRs (Memory-Mapped Registers)	FFFFh−FF80h	No (Only EDB-0)	No (only data register and EDB-0)	Yes
EDB (Endpoint Descriptors Block)	FF7Fh−FF08h	Only for EDB update	Only for EDB update	Yes
Setup Packet	FF07h−FF00h	Yes	No	Yes
Input Endpoint-0 Buffer	FEFFh−FEF8h	Yes	Yes	Yes
Output Endpoint-0 Buffer	FEF7h−FEF0h	Yes	Yes	Yes
Data Buffers	FEEFh−F800h	Yes	Yes	Yes

Table 5-3. Memory-Mapped Registers Summary (XDATA Range = FF80h → FFFFh)

ADDRESS	REGISTER	DESCRIPTION
FFFFh	FUNADR	Function address register
FFFEh	USBSTA	USB status register
FFFDh	USBMSK	USB interrupt mask register
FFFCh	USBCTL	USB control register
FFFBh	MODECNFG	Mode configuration register
FFFAh−FFF4h	—	Reserved
FFF3h	I2CADR	I^2C-port address register
FFF2h	I2CDATI	I^2C-port data input register
FFF1h	I2CDATO	I^2C-port data output register
FFF0h	I2CSTA	I^2C-port status register
FFEFh	SERNUM7	Serial number byte 7 register
FFEEh	SERNUM6	Serial number byte 6 register
FFEDh	SERNUM5	Serial number byte 5 register
FFECh	SERNUM4	Serial number byte 4 register
FFEBh	SERNUM3	Serial number byte 3 register
FFEAh	SERNUM2	Serial number byte 2 register
FFE9h	SERNUM1	Serial number byte 1 register
FFE8h	SERNUM0	Serial number byte 0 register
FFE7h−FFE6h	—	Reserved
FFE5h	DMACSR3	DMA-3: Control and status register
FFE4h	DMACDR3	DMA-3: Channel definition register
FFE3h−FFE2h	Reserved	
FFE1h	DMACSR1	DMA-1: Control and status register
FFE0h	DMACDR1	DMA-1: Channel definition register
FFDFh−FFACh	—	Reserved
FFABh	MASK	UART: Interrupt mask register
FFAAh	XOFF	UART: Xoff register
FFA9h	XON	UART: Xon register

Table 5-3. Memory-Mapped Registers Summary (XDATA Range = FF80h → FFFFh) (continued)

ADDRESS	REGISTER	DESCRIPTION
FFA8h	DLH	UART: Divisor high-byte register
FFA7h	DLL	UART: Divisor low-byte register
FFA6h	MSR	UART: Modem status register
FFA5h	LSR	UART: Line status register
FFA4h	MCR	UART: Modem control register
FFA3h	FCRL	UART: Flow control register
FFA2h	LCR	UART: Line control registers
FFA1h	TDR	UART: Transmitter data registers
FFA0h	RDR	UART: Receiver data registers
FF9Eh	PUR_3	GPIO: Pullup register for port 3
FF9Dh−FF94h	—	Reserved
FF93h	WDCSR	Watchdog timer control and status register
FF92h	VECINT	Vector interrupt register
FF91h	—	Reserved
FF90h	ROMS	ROM shadow configuration register
FF8Fh−FF84h	—	Reserved
FF83h	OEPBCNT_0	Output endpoint_0: Byte count register
FF82h	OEPCNFG_0	Output endpoint_0: Configuration register
FF81h	IEPBCNT_0	Input endpoint_0: Byte count register
FF80h	IEPCNFG_0	Input endpoint_0: Configuration register

Table 5-4. EDB Memory Locations

ADDRESS	REGISTER	DESCRIPTION
FF7Fh−FF60h	—	Reserved
FF5Fh	IEPSIZXY_3	Input endpoint_3: X-Y buffer size
FF5Eh	IEPBCTY_3	Input endpoint_3: Y-byte count
FF5Dh	IEPBBAY_3	Input endpoint_3: Y-buffer base address
FF5Ch	—	Reserved
FF5Bh	—	Reserved
FF5Ah	IEPBCTX_3	Input endpoint_3: X-byte count
FF59h	IEPBBAX	Input endpoint_3: X-buffer base address
FF58h	IEPCNF_3	Input endpoint_3: Configuration
FF57h	IEPSIZXY_2v	Input endpoint_2: X-Y buffer size
FF56h	IEPBCTY_2	Input endpoint_2: Y-byte count
FF55h	IEPBBAY_2	Input endpoint_2: Y-buffer base address
FF54h	—	Reserved
FF53h	—	Reserved
FF52h	IEPBCTX_2	Input endpoint_2: X-byte count
FF51h	IEPBBAX_2	Input endpoint_2: X-buffer base address
FF50h	IEPCNF_2	Input endpoint_2: Configuration
FF4Fh	IEPSIZXY_1	Input endpoint_1: X-Y buffer size
FF4Eh	IEPBCTY_1	Input endpoint_1: Y-byte count
FF4Dh	IEPBBAY_1	Input endpoint_1: Y-buffer base address
FF4Ch	—	Reserved
FF4Bh	—	Reserved

Table 5-4. EDB Memory Locations (continued)

ADDRESS	REGISTER	DESCRIPTION
FF4Ah	IEPBCTX_1	Input endpoint_1: X-byte count
FF49h	IEPBBAX_1	Input endpoint_1: X-buffer base address
FF48h	IEPCNF_1	Input endpoint_1: Configuration
FF47h ↑ FF20h	—	Reserved
FF1Fh	OEPSIZXY_3	Output endpoint_3: X-Y buffer size
FF1Eh	OEPBCTY_3	Output endpoint_3: Y-byte count
FF1Dh	OEPBBAY_3	Output endpoint_3: Y-buffer base address
FF1Bh−FF1Ch	—	Reserved
FF1Ah	OEPBCTX_3	Output endpoint_3: X-byte count
FF19h	OEPBBAX_3	Output endpoint_3: X-buffer base address
FF18h	OEPCNF_3	Output endpoint_3: Configuration
FF17h	OEPSIZXY_2	Output endpoint_2: X-Y buffer size
FF16h	OEPBCTY_2	Output endpoint_2: Y-byte count
FF15h	OEPBBAY_2	Output endpoint_2: Y-buffer base address
FF14h−FF13h	—	Reserved
FF12h	OEPBCTX_2	Output endpoint_2: X-byte count
FF11h	OEPBBAX_2	Output endpoint_2: X-buffer base address
FF10h	OEPCNF_2	Output endpoint_2: Configuration
FF0Fh	OEPSIZXY_1	Output endpoint_1: X-Y buffer size
FF0Eh	OEPBCTY_1	Output endpoint_1: Y-byte count
FF0Dh	OEPBBAY_1	Output endpoint_1: Y-buffer base address
FF0Ch−FF0Bh	—	Reserved
FF0Ah	OEPBCTX_1	Output endpoint_1: X-byte count
FF09h	OEPBBAX_1	Output endpoint_1: X-buffer base address
FF08h	OEPCNF_1	Output endpoint_1: Configuration
FF07h ↑ FF00h	(8 bytes)	Setup packet block
FEFFh ↑ FEF8h	(8 bytes)	Input endpoint_0 buffer
FEF7h ↑ FEF0h	(8 bytes)	Output endpoint_0 buffer
FEEFh	TOPBUFF	Top of buffer space
↑		Buffer space
F800h	STABUFF	Start of buffer space

5.5.4 Endpoint Descriptor Block (EDB−1 to EDB−3)

Data transfers between the USB, the MCU, and external devices that are defined by an endpoint descriptor block (EDB). Three input and three output EDBs are provided. With the exception of EDB-0 (I/O endpoint-0), all EDBs are located in SRAM as per Table 5-3. Each EDB contains information describing the X- and Y-buffers. In addition, each EDB provides general status information.

Table 5-5 describes the EDB entries for EDB−1 to EDB−3. EDB−0 registers are described in Table 5-6.

Table 5-5. Endpoint Registers and Offsets in RAM (n = 1 to 3)

OFFSET	ENTRY NAME	DESCRIPTION
07	EPSIZXY_n	I/O endpoint_n: X/Y-buffer size
06	EPBCTY_n	I/O endpoint_n: Y-byte count
05	EPBBAY_n	I/O endpoint_n: Y-buffer base address
04	SPARE	Not used
03	SPARE	Not used
02	EPBCTX_n	I/O endpoint_n: X-byte count
01	EPBBAX_n	I/O endpoint_n: X-buffer base address
00	EPCNF_n	I/O endpoint_n: Configuration

Table 5-6. Endpoint Registers Base Addresses

BASE ADDRESS	DESCRIPTION
FF08h	Output endpoint 1
FF10h	Output endpoint 2
FF18h	Output endpoint 3
FF48h	Input endpoint 1
FF50h	Input endpoint 2
FF58h	Input endpoint 3

5.5.4.1 OEPCNF_n: Output Endpoint Configuration (n = 1 to 3) (Base Addr: FF08h, FF10h, FF18h)

7	6	5	4	3	2	1	0
UBME	ISO=0	TOGLE	DBUF	STALL	USBIE	RSV	RSV
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
1−0	RSV	x	Reserved = 0
2	USBIE	x	USB interrupt enable on transaction completion. Set/cleared by the MCU. USBIE = 0 No interrupt on transaction completion USBIE = 1 Interrupt on transaction completion
3	STALL	0	USB stall condition indication. Set/cleared by the MCU. STALL = 0 No stall STALL = 1 USB stall condition. If set by the MCU, then a STALL handshake is initiated and the bit is cleared by the MCU.
4	DBUF	x	Double-buffer enable. Set/cleared by the MCU. DBUF = 0 Primary buffer only (X-buffer only) DBUF = 1 Toggle bit selects buffer
5	TOGLE	x	USB toggle bit. This bit reflects the toggle sequence bit of DATA0, DATA1.
6	ISO	x	ISO = 0 Nonisochronous transfer. This bit must be cleared by the MCU because only nonisochronous transfer is supported.
7	UBME	x	USB buffer manager (UBM) enable/disable bit. Set/cleared by the MCU. UBME = 0 UBM cannot use this endpoint UBME = 1 UBM can use this endpoint

5.5.4.2 OEPBBAX_n: Output Endpoint X-Buffer Base Address (n = 1 to 3) (Offset 1)

7	6	5	4	3	2	1	0
A10	A9	A8	A7	A6	A5	A4	A3
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
7–0	A[10:3]	x	A[10:3] of X-buffer base address (padded with 3 LSBs of zeros for a total of 11 bits). This value is set by the MCU. The UBM or DMA uses this value as the start-address of a given transaction. Note that the UBM or DMA does not change this value at the end of a transaction.

5.5.4.3 OEPBCTX_n: Output Endpoint X Byte Count (n = 1 to 3) (Offset 2)

7	6	5	4	3	2	1	0
NAK	C6	C5	C4	C3	C2	C1	C0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
6−0	C[6:0]	x	X-buffer byte count: X000.0000b Count = 0 X000.0001b Count = 1 byte : : X011.1111b Count = 63 bytes X100.0000b Count = 64 bytes Any value ≥ 100.0001b may result in unpredictable results.
7	NAK	x	NAK = 0 No valid data in buffer. Ready for host OUT NAK = 1 Buffer contains a valid packet from host (gives NAK response to Host OUT request)

5.5.4.4 OEPBBAY_n: Output Endpoint Y-Buffer Base Address (n = 1 to 3) (Offset 5)

7	6	5	4	3	2	1	0
A10	A9	A8	A7	A6	A5	A4	A3
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
7–0	A[10:3]	x	A[10:3] of Y-buffer base address (padded with 3 LSBs of zeros for a total of 11 bits). This value is set by the MCU. The UBM or DMA uses this value as the start-address of a given transaction. Furthermore, UBM or DMA does not change this value at the end of a transaction.

5.5.4.5 OEPBCTY_n: Output Endpoint Y-Byte Count (n = 1 to 3) (Offset 6)

7	6	5	4	3	2	1	0
NAK	C6	C5	C4	C3	C2	C1	C0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
6–0	C[6:0]	x	Y-byte count: X000.0000b Count = 0 X000.0001b Count = 1 byte : : X011.1111b Count = 63 bytes X100.0000b Count = 64 bytes Any value ≥ 100.0001b may result in unpredictable results.
7	NAK	x	NAK = 0 No valid data in buffer. Ready for host OUT NAK = 1 Buffer contains a valid packet from host (gives NAK response to Host OUT request)

5.5.4.6 OEPSIZXY_n: Output Endpoint X-/Y-Buffer Size (n = 1 to 3) (Offset 7)

7	6	5	4	3	2	1	0
RSV	S6	S5	S4	S3	S2	S1	S0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
6–0	S[6:0]	x	X- and Y-buffer size: 0000.0000b Size = 0 0000.0001b Size = 1 byte : : 0011.1111b Size = 63 bytes 0100.0000b Size = 64 bytes Any value ≥ 100.0001b may result in unpredictable results.
7	RSV	x	Reserved = 0

5.5.4.7 IEPCNF_n: Input Endpoint Configuration (n = 1 to 3) (Base Addr: FF48h, FF50h, FF58h)

7	6	5	4	3	2	1	0
UBME	ISO=0	TOGLE	DBUF	STALL	USBIE	RSV	RSV
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
1−0	RSV	x	Reserved = 0
2	USBIE	x	USB interrupt enable on transaction completion USBIE = 0 No interrupt on transaction completion USBIE = 1 Interrupt on transaction completion
3	STALL	0	USB stall condition indication. Set by the UBM but can be set/cleared by the MCU. STALL = 0 No stall STALL = 1 USB stall condition. If set by the MCU, then a STALL handshake is initiated and the bit is cleared automatically.
4	DBUF	x	Double buffer enable DBUF = 0 Primary buffer only (X-buffer only) DBUF = 1 Toggle bit selects buffer
5	TOGLE	x	USB toggle bit. This bit reflects the toggle sequence bit of DATA0, DATA1
6	ISO	x	ISO = 0 Nonisochronous transfer. This bit must be cleared by the MCU because only nonisochronous transfer is supported.
7	UBME	x	UBM enable/disable bit. Set/cleared by the MCU UBME = 0 UBM cannot use this endpoint UBME = 1 UBM can use this endpoint

5.5.4.8 IEPBBAX_n: Input Endpoint X-Buffer Base Address (n = 1 to 3) (Offset 1)

7	6	5	4	3	2	1	0
A10	A9	A8	A7	A6	A5	A4	A3
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
7–0	A[10:3]	x	A[10:3] of X-buffer base address (padded with 3 LSBs of zeros for a total of 11 bits). This value is set by the MCU. The UBM or DMA uses this value as the start-address of a given transaction, but note that the UBM or DMA does not change this value at the end of a transaction.

5.5.4.9 IEPBCTX_n: Input Endpoint X-Byte Count (n = 1 to 3) (Offset 2)

7	6	5	4	3	2	1	0
NAK	C6	C5	C4	C3	C2	C1	C0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
6−0	C[6:0]	x	X-Buffer byte count: X000.0000b Count = 0 X000.0001b Count = 1 byte : : X011.1111b Count = 63 bytes X100.0000b Count = 64 bytes Any value ≥ 100.0001b may result in unpredictable results.
7	NAK	x	NAK = 0 Buffer contains a valid packet for host-IN transaction NAK = 1 Buffer is empty (gives NAK response to host-IN request)

5.5.4.10 IEPBBAY_n: Input Endpoint Y-Buffer Base Address (n = 1 to 3) (Offset 5)

7	6	5	4	3	2	1	0
A10	A9	A8	A7	A6	A5	A4	A3
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
7–0	A[10:3]	x	A[10:3] of Y-buffer base address (padded with 3 LSBs of zeros for a total of 11 bits). This value is set by the MCU. The UBM or DMA uses this value as the start-address of a given transaction, but note that the UBM or DMA does not change this value at the end of a transaction.

5.5.4.11 IEPBCTY_n: Input Endpoint Y-Byte Count (n = 1 to 3) (Offset 6)

7	6	5	4	3	2	1	0
NAK	C6	C5	C4	C3	C2	C1	C0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
6−0	C[6:0]	x	Y-byte count: X000.0000b Count = 0 X000.0001b Count = 1 byte : : X011.1111b Count = 63 bytes X100.0000b Count = 64 bytes Any value ≥ 100.0001b may result in unpredictable results.
7	NAK	x	NAK = 0 Buffer contains a valid packet for host-IN transaction NAK = 1 Buffer is empty (gives NAK response to host-IN request)

5.5.4.12 IEPSIZXY_n: Input Endpoint X-/Y-Buffer Size (n = 1 to 3) (Offset 7)

7	6	5	4	3	2	1	0
RSV	S6	S5	S4	S3	S2	S1	S0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
6−0	S[6:0]	x	X- and Y-buffer size: 0000.0000b Size = 0 0000.0001b Size = 1 byte : : 0011.1111b Size = 63 bytes 0100.0000b Size = 64 bytes Any value ≥ 100.0001b may result in unpredictable results.
7	RSV	x	Reserved = 0

5.5.4.13 Endpoint-0 Descriptor Registers

Unlike registers EDB-1 to EDB-3, which are defined as memory entries in SRAM, endpoint-0 is described by a set of four registers (two for output and two for input). The registers and their respective addresses, used for EDB-0 description, are defined in Table 5-7. EDB-0 has no buffer base-address register, because these addresses are hardwired to FEF8h and FEF0h. Note that the bit positions have been preserved to provide consistency with EDB-n (n = 1 to 3).

Table 5-7. Input/Output EDB-0 Registers

ADDRESS	REGISTER NAME	DESCRIPTION	BUFFER BASE ADDRESS
FF83h	OEPBCNT_0	Output endpoint_0: Byte count register	FEF0h
FF82h	OEPCNFG_0	Output endpoint_0: Configuration register	
FF81h	IEPBCNT_0	Input endpoint_0: Byte count register	FEF8h
FF80h	IEPCNFG_0	Input endpoint_0: Configuration register	

5.5.4.13.1 IEPCNFG_0: Input Endpoint-0 Configuration Register (Addr:FF80h)

7	6	5	4	3	2	1	0
UBME	RSV	TOGLE	RSV	STALL	USBIE	RSV	RSV
R/W	R/O	R/O	R/O	R/W	R/W	R/O	R/O

BIT	NAME	RESET	FUNCTION
1−0	RSV	0	Reserved = 0
2	USBIE	0	USB interrupt enable on transaction completion. Set/cleared by the MCU. USBIE = 0 No interrupt USBIE = 1 Interrupt on transaction completion
3	STALL	0	USB stall condition indication. Set/cleared by the MCU STALL = 0 No stall STALL = 1 USB stall condition. If set by the MCU, then a STALL handshake is initiated and the bit is cleared automatically by the next setup transaction.
4	RSV	0	Reserved = 0
5	TOGLE	0	USB toggle bit. This bit reflects the toggle sequence bit of DATA0, DATA1.
6	RSV	0	Reserved = 0
7	UBME	0	UBM enable/disable bit. Set/cleared by the MCU UBME = 0 UBM cannot use this endpoint UBME = 1 UBM can use this endpoint

5.5.4.13.2 IEPBCNT_0: Input Endpoint-0 Byte Count Register (Addr:FF81h)

7	6	5	4	3	2	1	0
NAK	RSV	RSV	RSV	C3	C2	C1	C0
R/W	R/O	R/O	R/O	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
3−0	C[3:0]	0h	Byte count: 0000b Count = 0 : : 0111b Count = 7 1000b Count = 8 1001b to 1111b are reserved. (If used, they default to 8)
6−4	RSV	0	Reserved = 0
7	NAK	1	NAK = 0 Buffer contains a valid packet for host-IN transaction NAK = 1 Buffer is empty (gives NAK response to host-IN request)

5.5.4.13.3 OEPCNFG_0: Output Endpoint-0 Configuration Register (Addr:FF82h)

7	6	5	4	3	2	1	0
UBME	RSV	TOGLE	RSV	STALL	USBIE	RSV	RSV
R/W	R/O	R/O	R/O	R/W	R/W	R/O	R/O

BIT	NAME	RESET	FUNCTION
1−0	RSV	0	Reserved = 0
2	USBIE	0	USB interrupt enable on transaction completion. Set/cleared by the MCU. USBIE = 0 No interrupt on transaction completion USBIE = 1 Interrupt on transaction completion
3	STALL	0	USB stall condition indication. Set/cleared by the MCU STALL = 0 No stall STALL = 1 USB stall condition. If set by the MCU, then a STALL handshake is initiated and the bit is cleared automatically.
4	RSV	0	Reserved = 0
5	TOGLE	0	USB \toggle bit. This bit reflects the toggle sequence bit of DATA0, DATA1.
6	RSV	0	Reserved = 0
7	UBME	0	UBM enable/disable bit. Set/cleared by the MCU UBME = 0 UBM cannot use this endpoint UBME = 1 UBM can use this endpoint

5.5.4.13.4 OEPBCNT_0: Output Endpoint-0 Byte Count Register (Addr:FF83h)

7	6	5	4	3	2	1	0
NAK	RSV	RSV	RSV	C3	C2	C1	C0
R/W	R/O	R/O	R/O	R/O	R/O	R/O	R/O

BIT	NAME	RESET	FUNCTION
3−0	C[3:0]	0h	Byte count: 0000b Count = 0 : : 0111b Count = 7 1000b Count = 8 1001b to 1111b are reserved.
6−4	RSV	0	Reserved = 0
7	NAK	1	NAK = 0 No valid data in buffer. Ready for host OUT NAK = 1 Buffer contains a valid packet from host (gives NAK response to host-OUT request).

5.5.5 USB Registers

5.5.5.1 FUNADR: Function Address Register (Addr:FFFFh)

This register contains the device function address.

7	6	5	4	3	2	1	0
RSV	FA6	FA5	FA4	FA3	FA2	FA1	FA0
R/O	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
6−0	FA[6:0]	0	These bits define the current device address assigned to the function. The MCU writes a value to this register because of the SET-ADDRESS host command.
7	RSV	0	Reserved = 0

5.5.5.2 USBSTA: USB Status Register (Addr:FFFEh)

All bits in this register are set by the hardware and are cleared by the MCU when writing a 1 to the proper bit location (writing a 0 has no effect). In addition, each bit can generate an interrupt if its corresponding mask bit is set (R/C notation indicates read and clear only by the MCU).

7	6	5	4	3	2	1	0
RSTR	SUSR	RESR	RSV	URRI	SETUP	WAKEUP	STPOW
R/C	R/C	R/C	R/O	R/C	R/C	R/C	R/C

BIT	NAME	RESET	FUNCTION
0	STPOW	0	SETUP overwrite bit. Set by hardware when a setup packet is received while there is already a packet in the setup buffer. STPOW = 0 MCU can clear this bit by writing a 1 (writing 0 has no effect). STPOW = 1 SETUP overwrite
1	WAKEUP	0	Remote wakeup bit WAKEUP = 0 The MCU can clear this bit by writing a 1 (writing 0 has no effect). WAKEUP = 1 Remote wake-up request from WAKEUP terminal
2	SETUP	0	SETUP transaction received bit. As long as SETUP is 1, IN and OUT on endpoint-0 are NAKed, regardless of their real NAK bits value. SETUP = 0 MCU can clear this bit by writing a 1 (writing 0 has no effect). SETUP = 1 SETUP transaction received
3	URRI	0	UART RI (ring indicate) status bit – a rising edge causes this bit to be set. URRI = 0 The MCU can clear this bit by writing a 1 (writing 0 has no effect). URRI = 1 Ring detected, which is used to wake the chip up (bring it out of suspend).
4	RSV	0	Reserved
5	RESR	0	Function resume request bit RESR = 0 The MCU can clear this bit by writing a 1 (writing 0 has no effect). RESR = 1 Function resume is detected
6	SUSR	0	Function suspended request bit. This bit is set in response to a global or selective suspend condition. SUSR = 0 The MCU can clear this bit by writing a 1 (writing 0 has no effect). SUSR = 1 Function suspend is detected
7	RSTR	0	Function reset request bit. This bit is set in response to the USB host initiating a port reset. This bit is not affected by the USB function reset. RSTR = 0 The MCU can clear this bit by writing a 1 (writing 0 has no effect). RSTR = 1 Function reset is detected

5.5.5.3 USBMSK: USB Interrupt Mask Register (Addr:FFFDh)

7	6	5	4	3	2	1	0
RSTR	SUSR	RESR	RSV	URRI	SETUP	WAKEUP	STPOW
R/W	R/W	R/W	R/O	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
0	STPOW	0	SETUP overwrite interrupt-enable bit STPOW = 0 STPOW interrupt disabled STPOW = 1 STPOW interrupt enabled
1	WAKEUP	0	Remote wake-up interrupt enable bit WAKEUP = 0 WAKEUP interrupt disable WAKEUP = 1 WAKEUP interrupt enable
2	SETUP	0	SETUP interrupt enable bit SETUP = 0 SETUP interrupt disabled SETUP = 1 SETUP interrupt enabled
3	URRI	0	UART RI interrupt enable bit URRI = 0 UART RI interrupt disable URRI = 1 UART RI interrupt enable
4	RSV	0	Reserved
5	RESR	0	Function resume interrupt enable bit RESR = 0 Function resume interrupt disabled RESR = 1 Function resume interrupt enabled
6	SUSR	0	Function suspend interrupt enable SUSR = 0 Function suspend interrupt disabled SUSR = 1 Function suspend interrupt enabled
7	RSTR	0	Function reset interrupt bit. This bit is not affected by USB function reset. RSTR = 0 Function reset interrupt disabled RSTR = 1 Function reset interrupt enabled

5.5.5.4 USBCTL: USB Control Register (Addr:FFFCh)

Unlike the rest of the registers, this register is cleared by the power-up reset signal only. The USB reset cannot reset this register (see Figure 5-9).

7	6	5	4	3	2	1	0
CONT	IREN	RWUP	FRSTE	RSV	RSV	SIR	DIR
R/W	R/W	R/C	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
0	DIR	0	As a response to a setup packet, the MCU decodes the request and sets/clears this bit to reflect the data transfer direction. DIR = 0 USB data-OUT transaction (from host to TUSB3410) DIR = 1 USB data-IN transaction (from TUSB3410 to host)
1	SIR	0	SETUP interrupt-status bit. This bit is controlled by the MCU to indicate to the hardware when the SETUP interrupt is being serviced. SIR = 0 SETUP interrupt is not served. The MCU clears this bit before exiting the SETUP interrupt routine. SIR = 1 SETUP interrupt is in progress. The MCU sets this bit when servicing the SETUP interrupt.
2	RSV	0	Reserved = 0
3	RSV	0	This bit must always be written as 0.
4	FRSTE	1	Function reset-connection bit. This bit connects/disconnects the USB function reset to/from the MCU reset. FRSTE = 0 Function reset is not connected to MCU reset FRSTE = 1 Function reset is connected to MCU reset
5	RWUP	0	Device remote wake-up request. This bit is set by the MCU and is cleared automatically. RWUP = 0 Writing a 0 to this bit has no effect RWUP = 1 When MCU writes a 1, a remote-wakeup pulse is generated.
6	IREN	0	IR mode enable. This bit is set and cleared by firmware. IREN = 0 IR encoder/decoder is disabled, UART mode is selected IREN = 1 IR encoder/decoder is enabled, UART mode is deselected
7	CONT	0	Connect/disconnect bit CONT = 0 Upstream port is disconnected. Pullup disabled. CONT = 1 Upstream port is connected. Pullup enabled.

5.5.5.5 MODECNFG: Mode Configuration Register (Addr:FFFBh)

This register is cleared by the power-up reset signal only. The USB reset cannot reset this register.

7	6	5	4	3	2	1	0
RSV	RSV	RSV	RSV	CLKSLCT	CLKOUTEN	SOFTSW	TXCNTL
R/O	R/O	R/O	R/O	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
0	TSCNTL	0	Transmit output control: Hardware or firmware switching select for 3-state serial output buffer. TXCNTL = 0 Hardware automatic switching is selected TXCNTL = 1 Firmware toggle switching is selected
1	SOFTSW	0	Soft switch: Firmware controllable 3-state output buffer enable for serial output terminal. SOFTSW = 0 Serial output buffer is enabled SOFTSW = 1 Serial output buffer is disabled
2	CLKOUTEN	0	Clock output enable: Enables/disables the clock output at CLKOUT terminal. CLKOUTEN = 0 Clock output is disabled. Device drives low at CLKOUT terminal. CLKOUTEN = 1 Clock output is enabled
3	CLKSLCT	0	Clock output source select: Selects between 3.556-MHz fixed clock or UART baud out clock as output clock source. CLKSLCT = 0 UART baud out clock is selected as clock output CLKSLCT = 1 Fixed 3.556-MHz free running clock is selected as clock output
4–7	RSV	0	Reserved

5.5.5.6 Clock Output Control

Bit 2 (CLKOUTEN) in the MODECNFG register enables or disables the clock output at the CLKOUT terminal of the TUSB3410 device. The power-up default of CLKOUT is disabled. Firmware can write a 1 to enable the clock output if needed.

Bit 3 (CLKSLCT) in the MODECNFG register selects the output clock source from either a fixed 3.556-MHz free-running clock or the UART BaudOut clock.

5.5.5.7 Vendor ID/Product ID

USB−IF and Microsoft WHQL certification requires that end equipment makers use their own unique vendor ID and product ID for each product (model). OEMs cannot use silicon vendor's VID/PID (for instance, TI's default) in their end products. A unique VID/PID combination will avoid potential driver conflicts and enable logo certification. See www.usb.org for more information.

5.5.5.8 SERNUM7: Device Serial Number Register (Byte 7) (Addr:FFEFh)

Each TUSB3410 device has a unique 64-bit serial die id number, which is generated during manufacturing. The die id is incremented sequentially, however there is no assurance that numbers will not be skipped. The device serial number registers mirror this unique 64-bit serial die id value.

After power-up reset, this read-only register (SERNUM7) contains the most significant byte (byte 7) of the complete 64-bit device serial number. This register cannot be reset.

7	6	5	4	3	2	1	0
D63	D62	D61	D60	D59	D58	D57	D56
R/O	R/O	R/O	R/O	R/O	R/O	R/O	R/O

BIT	NAME	RESET	FUNCTION
7−0	D[63:56]	Device serial number byte 7 value	Device serial number byte 7 value

Procedure to load device serial number value in shared RAM:

- After power-up reset, the boot code copies the predefined USB descriptors to shared RAM. As a result, the default serial number hard-coded in the boot code (0x00 hex) is copied to the shared RAM data space.
- The boot code checks to see if an EEPROM is present on the I^2C port. If an EEPROM is present and contains a valid device serial number as part of the USB device descriptor information stored in EEPROM, then the boot code overwrites the serial number value stored in shared RAM with the one found in EEPROM. Otherwise, the device serial number value stored in shared RAM remains unchanged. If firmware is stored in the EEPROM, then it is executed. This firmware can read the SERNUM7 through SERNUM0 registers and overwrite the serial number stored in RAM or store a custom number in RAM.
- In summary, the serial number value in external EEPROM has the highest priority to be loaded into shared RAM data space. The serial number value stored in shared RAM is used as part of the valid device descriptor information during normal operation.

5.5.5.9 SERNUM6: Device Serial Number Register (Byte 6) (Addr:FFEEh)

The device serial number registers mirror the unique 64-bit die id value.

After power-up reset, this read-only register (SERNUM6) contains byte 6 of the complete 64-bit device serial number. This register cannot be reset.

7	6	5	4	3	2	1	0
D55	D54	D53	D52	D51	D50	D49	D48
R/O	R/O	R/O	R/O	R/O	R/O	R/O	R/O

BIT	NAME	RESET	FUNCTION
7−0	D[55:48]	Device serial number byte 6 value	Device serial number byte 6 value

NOTE

See the procedure described in the SERNUM7 register (see Section 5.5.5.8) to load the device serial number into shared RAM.

5.5.5.10 SERNUM5: Device Serial Number Register (Byte 5) (Addr:FFEDh)

The device serial number registers mirror the unique 64-bit die id value.

After power-up reset, this read-only register (SERNUM5) contains byte 5 of the complete 64-bit device serial number. This register cannot be reset.

7	6	5	4	3	2	1	0
D47	D46	D45	D44	D43	D42	D41	D40
R/O	R/O	R/O	R/O	R/O	R/O	R/O	R/O

BIT	NAME	RESET	FUNCTION
7−0	D[47:40]	Device serial number byte 5 value	Device serial number byte 5 value

NOTE

See the procedure described in the SERNUM7 register (see Section 5.5.5.8) to load the device serial number into shared RAM.

5.5.5.11 SERNUM4: Device Serial Number Register (Byte 4) (Addr:FFECh)

The device serial number registers mirror the unique 64-bit die id value.

After power-up reset, this read-only register (SERNUM4) contains byte 4 of the complete 64-bit device serial number. This register cannot be reset.

7	6	5	4	3	2	1	0
D39	D38	D37	D36	D35	D34	D33	D32
R/O	R/O	R/O	R/O	R/O	R/O	R/O	R/O

BIT	NAME	RESET	FUNCTION
7−0	D[39:32]	Device serial number byte 4 value	Device serial number byte 4 value

NOTE

See the procedure described in the SERNUM7 register (see Section 5.5.5.8) to load the device serial number into shared RAM.

5.5.5.12 SERNUM3: Device Serial Number Register (Byte 3) (Addr:FFEBh)

The device serial number registers mirror the unique 64-bit die id value.

After power-up reset, this read-only register (SERNUM3) contains byte 3 of the complete 64-bit device serial number. This register cannot be reset.

7	6	5	4	3	2	1	0
D31	D30	D29	D28	D27	D26	D25	D24
R/O	R/O	R/O	R/O	R/O	R/O	R/O	R/O

BIT	NAME	RESET	FUNCTION
7−0	D[31:24]	Device serial number byte 3 value	Device serial number byte 3 value

NOTE

See the procedure described in the SERNUM7 register (see Section 5.5.5.8) to load the device serial number into shared RAM.

5.5.5.13 SERNUM2: Device Serial Number Register (Byte 2) (Addr:FFEAh)

The device serial number registers mirror the unique 64-bit die id value.

After power-up reset, this read-only register (SERNUM2) contains byte 2 of the complete 64-bit device serial number. This register cannot be reset.

7	6	5	4	3	2	1	0
D23	D21	D20	D19	D18	D17	D16	D15
R/O	R/O	R/O	R/O	R/O	R/O	R/O	R/O

BIT	NAME	RESET	FUNCTION
7−0	D[23:16]	0	Device serial number byte 2 value

NOTE

See the procedure described in the SERNUM7 register (see Section 5.5.5.8) to load the device serial number into shared RAM.

5.5.5.14 SERNUM1: Device Serial Number Register (Byte 1) (Addr:FFE9h)

The device serial number registers mirror the unique 64-bit die id value.

After power-up reset, this read-only register (SERNUM1) contains byte 1 of the complete 64-bit device serial number. This register cannot be reset.

7	6	5	4	3	2	1	0
D15	D14	D13	D12	D11	D10	D9	D8
R/O	R/O	R/O	R/O	R/O	R/O	R/O	R/O

BIT	NAME	RESET	FUNCTION
7−0	D[15:8]	Device serial number byte 1 value	Device serial number byte 1 value

NOTE

See the procedure described in the SERNUM7 register (see Section 5.5.5.8) to load the device serial number into shared RAM.

5.5.5.15 SERNUM0: Device Serial Number Register (Byte 0) (Addr:FFE8h)

The device serial number registers mirror the unique 64-bit die id value.

After power-up reset, this read-only register (SERNUM0) contains byte 0 of the complete 64-bit device serial number. This register cannot be reset.

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0
R/O	R/O	R/O	R/O	R/O	R/O	R/O	R/O

BIT	NAME	RESET	FUNCTION
7−0	D[7:0]	Device serial number byte 0 value	Device serial number byte 0 value

NOTE

See the procedure described in the SERNUM7 register (see Section 5.5.5.8) to load the device serial number into shared RAM.

5.5.5.16 Function Reset and Power-Up Reset Interconnect

Figure 5-9 represents the logical connection of the USB-function reset ($\overline{\text{USBR}}$) signal and the power-up reset ($\overline{\text{RESET}}$) terminal. The internal $\overline{\text{RESET}}$ signal is generated from the $\overline{\text{RESET}}$ terminal ($\overline{\text{PURS}}$ signal) or from the USB reset ($\overline{\text{USBR}}$ signal). The $\overline{\text{USBR}}$ can be enabled or disabled by bit 4 (FRSTE) in the USBCTL register (see Section 5.5.5.4) (on power up, FRSTE = 0). The internal $\overline{\text{RESET}}$ is used to reset all registers and logic, with the exception of the USBCTL and MODECNFG registers, which are cleared by the $\overline{\text{PURS}}$ signal only.

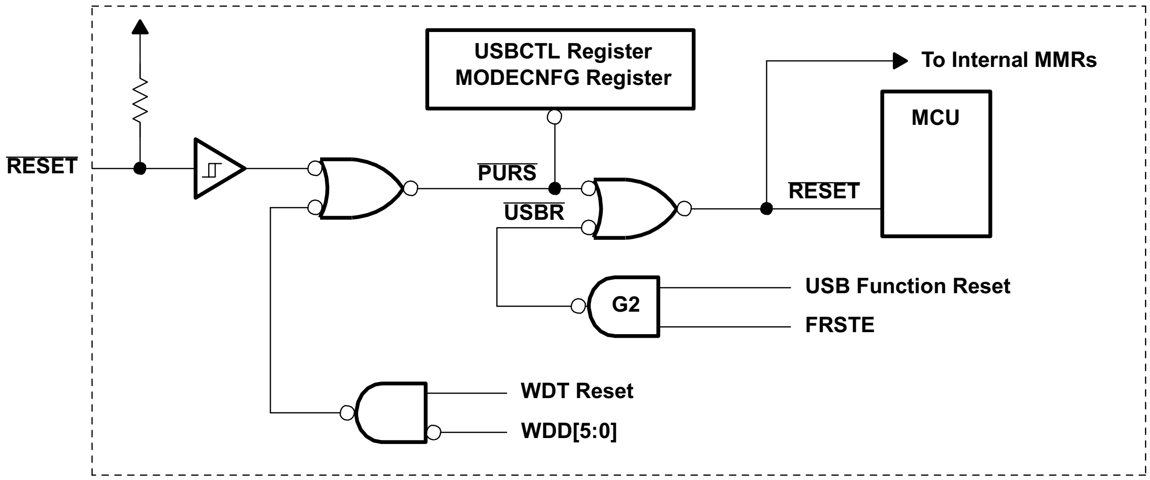


Figure 5-9. Reset Diagram

5.5.5.17 Pullup Resistor Connect and Disconnect

The TUSB3410 device enumeration can be activated by the MCU (there is no need to disconnect the cable physically). Figure 5-10 represents the implementation of the TUSB3410 device connect and disconnect from a USB up-stream port. When bit 7 (CONT) is 1 in the USBCTL register (see Section 5.5.5.4), the CMOS driver sources V_{DD} to the pullup resistor (PUR terminal) presenting a normal connect condition to the USB host. When CONT is 0, the PUR terminal is driven low. In this state, the 1.5-kΩ resistor is connected to GND, resulting in the device disconnection state. The PUR driver is a CMOS driver that can provide (V_{DD} − 0.1 V) minimum at 8-mA source current.

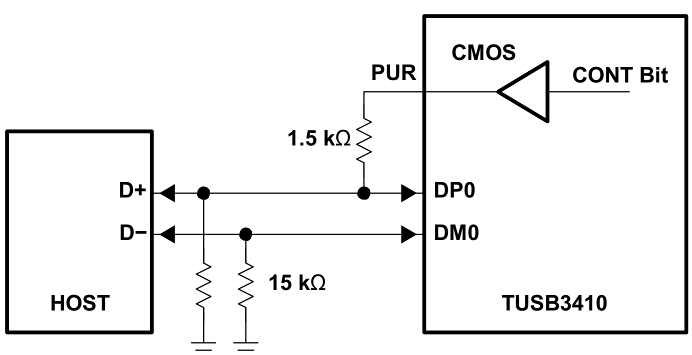


Figure 5-10. Pullup Resistor Connect and Disconnect Circuit

5.5.6 DMA Controller Registers

Table 5-8 outlines the DMA channels and their associated transfer directions. Two channels are provided for data transfer between the host and the UART.

Table 5-8. DMA Controller Registers

DMA CHANNEL	TRANSFER DIRECTION	COMMENTS
DMA−1	Host to UART	DMA writes to UART TDR register
DMA−3	UART to host	DMA reads from UART RDR register

Each DMA channel can point to one of three EDBs (EDB-1 to EDB-3) and transfer data to/from the UART channel. The DMA can move data from a given out-point buffer (defined by the EDB) to the destination port. Similarly, the DMA can move data from a port to a given input-endpoint buffer.

At the end of a block transfer, the DMA updates the byte count and bit 7 (NAK) in the EDB (see Section 5.5.4) when receiving. In addition, it uses bit 4 (XY) in the DMACDR register to switch automatically, without interrupting the MCU (the XY bit toggle is performed by the UBM). The DMA stops only when a time-out or error condition occurs. When the DMA is transmitting (from the X/Y buffer) it continues alternating between X/Y buffers until it detects a byte count smaller than the buffer size (buffer size is typically 64 bytes). At that point it completes the transfer and stops.

5.5.6.1 DMACDR1: DMA Channel Definition Register (UART Transmit Channel) (Addr:FFE0h)

These registers define the EDB number that the DMA uses for data transfer to the UARTS. In addition, these registers define the data transfer direction and selects X or Y as the transaction buffer.

7	6	5	4	3	2	1	0
EN	INE	CNT	XY	T/R	E2	E1	E0
R/W	R/W	R/W	R/W	R/O	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
2–0	E[2:0]	0	Endpoint descriptor pointer. This field points to a set of EDB registers that is to be used for a given transfer.
3	T/R	0	This bit is always 1, indicating that the DMA data transfer is from SRAM to the UART TDR register (see Section 5.5.7.2). (The MCU cannot change this bit.)
4	XY	0	X/Y buffer select bit. XY = 0 Next buffer to transmit/receive is the X buffer XY = 1 Next buffer to transmit/receive is the Y buffer
5	CNT	0	DMA continuous transfer control bit. This bit defines the mode of the DMA transfer. This bit must always be written as 1. In this mode, the DMA and UBM alternate between the X- and Y-buffers. The DMA sets bit 4 (XY) and the UBM uses it for the transfer. The DMA alternates between the X-/Y-buffers and continues transmitting (from X-/Y-buffer) without MCU intervention. The DMA terminates, and interrupts the MCU, under the following conditions: 1. When the UBM byte count < buffer size (in EDB), the DMA transfers the partial packet and interrupt the MCU on completion. 2. Transaction timer expires. The DMA interrupts the MCU.
6	INE	0	DMA Interrupt enable/disable bit. This bit enables/disables the interrupt on transfer completion. INE = 0 Interrupt is disabled. In addition, bit 0 (PPKT) in the DMACSR1 register (see Section 5.5.6.2) does not clear bit 7 (EN) and the DMAC is not disabled. INE = 1 Enables the EN interrupt. When this bit is set, the DMA interrupts the MCU on a 1 to 0 transition of the bit 7 (EN). (When transfer is completed, EN = 0.)
7	EN	0	DMA channel enable bit. The MCU sets this bit to start the DMA transfer. When the transfer completes, or when it is terminated due to error, this bit is cleared. The 1 to 0 transition of this bit generates an interrupt (if the interrupt is enabled). EN = 0 DMA is halted. The DMA is halted when the byte count reaches zero or transaction time-out occurs. When halted, the DMA updates the byte count, sets NAK = 0 in the output endpoint byte count register, and interrupts the MCU (if bit 6 (INE) = 1). EN = 1 Setting this bit starts the DMA transfer.

5.5.6.2 DMACSR1: DMA Control And Status Register (UART Transmit Channel) (Addr:FFE1h)

This register defines the transaction time-out value. In addition, it contains a completion code that reports any errors or a time-out condition.

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	PPKT
R	R	R	R	R	R	R	R/C

BIT	NAME	RESET	FUNCTION
0	PPKT	0	Partial packet condition bit. This bit is set by the DMA and cleared by the MCU. PPKT = 0 No partial-packet condition PPKT = 1 Partial-packet condition detected. When INE = 0, this bit does not clear bit 7 (EN) in the DMACDR1 register; therefore, the DMAC stays enabled, ready for the next transaction. Clears when MCU writes a 1. Writing a 0 has no effect.
7–1		0	These bits are read-only and return 0s when read.

5.5.6.3 DMACDR3: DMA Channel Definition Register (UART Receive Channel) (Addr:FFE4h)

These registers define the EDB number that the DMA uses for data transfer from the UARTS. In addition, these registers define the data transfer direction and selects X or Y as the transaction buffer.

7	6	5	4	3	2	1	0
EN	INE	CNT	XY	T/R	E2	E1	E0
R/W	R/W	R/W	R/W	R/O	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
2−0	E[2:0]	0	Endpoint descriptor pointer. This field points to a set of EDB registers that is to be used for a given transfer.
3	T/R	1	This bit is always read as 1. This bit must be written as 0 to update the X/Y buffer bit (bit 4 in this register), which must only be performed in burst mode.
4	XY	0	X/Y buffer select bit. XY = 0 Next buffer to transmit/receive is X XY = 1 Next buffer to transmit/receive is Y
5	CNT	0	DMA continuous transfer control bit. This bit defines the mode of the DMA transfer. This bit must always be written as 1. In this mode, the DMA and UBM alternate between the X- and Y-buffers. The UBM sets bit 4 (XY) and the DMA uses it for the transfer. The DMA alternates between the X-/Y-buffers and continues receiving (to X-/Y-buffer) without MCU intervention. The DMA terminates the transfer and interrupts the MCU, under the following conditions: 1. Transaction time-out expired: DMA updates EDB and interrupts the MCU. UBM transfers the partial packet to the host. 2. UART receiver error condition: DMA updates EDB and does not interrupt the MCU. UBM transfers the partial packet to the host.
6	INE	0	DMA Interrupt enable/disable bit. This bit enables/disables the interrupt on transfer completion. INE = 0 Interrupt is disabled. In addition, bit 0 (OVRUN) and bit 1 (TXFT) in the DMACSR3 register (see Section 5.5.6.4) do not clear bit 7 (EN) and the DMAC is not disabled. INE = 1 Enables the EN interrupt. When this bit is set, the DMA interrupts the MCU on a 1-to-0 transition of bit 7 (EN). (When transfer is completed, EN = 0).
7	EN	0	DMA channel enable bit. The MCU sets this bit to start the DMA transfer. When transfer completes, or when terminated due to error, this bit is cleared. The 1-to-0 transition of this bit generates an interrupt (if the interrupt is enabled). EN = 0 DMA is halted. The DMA is halted when transaction time-out occurs, or under a UART receiver-error condition. When halted, the DMA updates the byte count and sets NAK = 0 in the input endpoint byte count register. If the termination is due to transaction time-out, then the DMA generates an interrupt. However, if the termination is due to a UART error condition, then the DMA does not generate an interrupt. (The UART generates the interrupt.) EN = 1 Setting this bit starts the DMA transfer.

5.5.6.4 DMACSR3: DMA Control And Status Register (UART Receive Channel) (Addr:FFE5h)

This register defines the transaction time-out value. In addition, it contains a completion code that reports any errors or a time-out condition.

7	6	5	4	3	2	1	0
TEN	C4	C3	C2	C1	C0	TXFT	OVRUN
R/W	R/W	R/W	R/W	R/W	R/W	R/C	R/C

BIT	NAME	RESET	FUNCTION
0	OVRUN	0	Overrun condition bit. This bit is set by DMA and cleared by the MCU (see Table 5-9) OVRUN = 0 No overrun condition OVRUN = 1 Overrun condition detected. When IEN = 0, this bit does not clear bit 7 (EN) in the DMACDR register; therefore, the DMAC stays enabled, ready for the next transaction. Clears when the MCU writes a 1. Writing a 0 has no effect.
1	TXFT	0	Transfer time-out condition bit (see Table 5-9) TXFT = 0 DMA stopped transfer without time-out TXFT = 1 DMA stopped due to transaction time-out. When IEN = 0, this bit does not clear bit 7 (EN) in the DMACDR3 register (see Section 5.5.6.3); therefore, the DMAC stays enabled, ready for the next transaction. Clears when the MCU writes a 1. Writing a 0 has no effect.
6−2	C[4:0]	00000b	This field defines the transaction time-out value in 1-ms increments. This value is loaded to a down counter every time a byte transfer occurs. The down counter is decremented every SOF pulse (1 ms). If the counter decrements to zero, then it sets bit 1 (TXFT) = 1 and halts the DMA transfer. The counter starts counting only when bit 7 (TEN) = 1 and bit 7 (EN) = 1 in the DMACDR3 register and the first byte has been received. 00000 = 0-ms time-out : : 11111 = 31-ms time-out
7	TEN	0	Transaction time-out counter enable/disable bit TEN = 0 Counter is disabled (does not time-out) TEN = 1 Counter is enabled

Table 5-9. DMA IN-Termination Condition

IN TERMINATION	TXFT	OVRUN	COMMENTS
UART error	0	0	UART error condition detected
UART partial packet	1	0	This condition occurs when UART receiver has no more data for the host (data starvation).
UART overrun	1	1	This condition occurs when X- and Y-input buffers are full and the UART FIFO is full (host is busy).

5.5.7 UART Registers

Table 5-10 summarizes the UART registers. These registers are used for data I/O, control, and status information. UART setup is done by the MCU. Data transfer is typically performed by the DMAC. However, the MCU can perform data transfer without a DMA; this is useful when debugging the firmware.

Table 5-10. UART Registers Summary

REGISTER ADDRESS	REGISTER NAME	ACCESS	FUNCTION	COMMENTS
FFA0h	RDR	R/O	UART receiver data register	Can be accessed by MCU or DMA
FFA1h	TDR	W/O	UART transmitter data register	Can be accessed by MCU or DMA
FFA2h	LCR	R/W	UART line control register	
FFA3h	FCRL	R/W	UART flow control register	
FFA4h	MCR	R/W	UART modem control register	
FFA5h	LSR	R/O	UART line status register	Can generate an interrupt
FFA6h	MSR	R/O	UART modem status register	Can generate an interrupt
FFA7h	DLL	R/W	UART divisor register (low byte)	
FFA8h	DLH	R/W	UART divisor register (high byte)	
FFA9h	XON	R/W	UART Xon register	
FFAAh	XOFF	R/W	UART Xoff register	
FFABh	MASK	R/W	UART interrupt mask register	Can control three interrupt sources

5.5.7.1 RDR: Receiver Data Register (Addr:FFA0h)

The receiver data register consists of a 32-byte FIFO. Data received through the SIN terminal is converted from serial-to-parallel format and stored in this FIFO. Data transfer from this register to the RAM buffer is the responsibility of the DMA controller.

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0
R/O	R/O	R/O	R/O	R/O	R/O	R/O	R/O

BIT	NAME	RESET	FUNCTION
7−0	D[7:0]	0	Receiver byte

5.5.7.2 TDR: Transmitter Data Register (Addr:FFA1h)

The transmitter data register is double buffered. Data written to this register is loaded into the shift register, and shifted out on SOUT. Data transfer from the RAM buffer to this register is the responsibility of the DMA controller.

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0
W/O	W/O	W/O	W/O	W/O	W/O	W/O	W/O

BIT	NAME	RESET	FUNCTION
7−0	D[7:0]	0	Transmit byte

5.5.7.3 LCR: Line Control Register (Addr:FFA2h)

This register controls the data communication format. The word length, number of stop bits, and parity type are selected by writing the appropriate bits to the LCR.

7	6	5	4	3	2	1	0
FEN	BRK	FPTY	EPRTY	PRTY	STP	WL1	WL0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
1–0	WL[1:0]	0	Specifies the word length for transmit and receive 00b = 5 bits 01b = 6 bits 10b = 7 bits 11b = 8 bits
2	STP	0	Specifies the number of stop bits for transmit and receive STP = 0 1 stop bit (word length = 5, 6, 7, 8) STP = 1 1.5 stop bits (word length = 5) STP = 1 2 stop bits (word length = 6, 7, 8)
3	PRTY	0	Specifies whether parity is used PRTY = 0 No parity PRTY = 1 Parity is generated
4	EPRTY	0	Specifies whether even or odd parity is generated EPRTY = 0 Odd parity is generated (if bit 3 (PRTY) = 1) EPRTY = 1 Even parity is generated (if PRTY = 1)
5	FPTY	0	Selects the forced parity bit FPTY = 0 Parity is not forced FPTY = 1 Parity bit is forced. If bit 4 (EPRTY) = 0, the parity bit is forced to 1
6	BRK	0	This bit is the break-control bit BRK = 0 Normal operation BRK = 1 Forces SOUT into break condition (logic 0)
7	FEN	0	FIFO enable. This bit disables/enables the FIFO. To reset the FIFO, the MCU clears and then sets this bit. FEN = 0 The FIFO is cleared and disabled. When disabled, the selected receiver flow control is activated. FEN = 1 The FIFO is enabled and it can receive data.

5.5.7.4 FCRL: UART Flow Control Register (Addr:FFA3h)

This register provides the flow-control modes of operation (see Table 5-12 for more details).

7	6	5	4	3	2	1	0
485E	DTR	RTS	RXOF	DSR	CTS	TXOA	TXOF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
0	TXOF	0	This bit controls the transmitter Xon/Xoff flow control. TXOF = 0 Disable transmitter Xon/Xoff flow control TXOF = 1 Enable transmitter Xon/Xoff flow control
1	TXOA	0	This bit controls the transmitter Xon-on-any/Xoff flow control TXOA = 0 Disable the transmitter Xon-on-any/Xoff flow control TXOA = 1 Enable the transmitter Xon-on-any/Xoff flow control
2	CTS	0	Transmitter $\overline{CTS}$ flow-control enable bit CTS = 0 Disables transmitter $\overline{CTS}$ flow control CTS = 1 $\overline{CTS}$ flow control is enabled, that is, when $\overline{CTS}$ input terminal is high, transmission is halted; when the $\overline{CTS}$ terminal is low, transmission resumes. When loopback mode is enabled, this bit must be set if flow control is also required.
3	DSR	0	Transmitter $\overline{DSR}$ flow-control enable bit DSR = 0 Disables transmitter $\overline{DSR}$ flow control DSR = 1 $\overline{DSR}$ flow control is enabled, that is, when $\overline{DSR}$ input terminal is high, transmission is halted; when the $\overline{DSR}$ terminal is low, transmission resumes. When loopback mode is enabled, this bit must be set if flow control is also required.
4	RXOF	0	This bit controls the receiver Xon/Xoff flow control. RXOF = 0 Receiver does not attempt to match Xon/Xoff characters RXOF = 1 Receiver searches for Xon/Xoff characters
5	RTS	0	Receiver $\overline{RTS}$ flow control enable bit RTS = 0 Disables receiver $\overline{RTS}$ flow control RTS = 1 Receiver $\overline{RTS}$ flow control is enabled. $\overline{RTS}$ output terminal goes high when the receiver FIFO HALT trigger level is reached; it goes low, when the receiver FIFO RESUME receiving trigger level is reached.
6	DTR	0	Receiver $\overline{DTR}$ flow-control enable bit DTR = 0 Disables receiver $\overline{DTR}$ flow control DTR = 1 Receiver $\overline{DTR}$ flow control is enabled. $\overline{DTR}$ output terminal goes high when the receiver FIFO HALT trigger level is reached; it goes low, when the receiver FIFO RESUME receiving trigger level is reached.
7	485E	0	RS-485 enable bit. This bit configures the UART to control external RS-485 transceivers. When configured in half-duplex mode (485E = 1), $\overline{RTS}$ or $\overline{DTR}$ can be used to enable the RS-485 driver or receiver (see Figure 5-5). 485E = 0 UART is in normal operation mode (full duplex) 485E = 1 The UART is in half duplex RS-485 mode. In this mode, $\overline{RTS}$ and $\overline{DTR}$ are active with opposite polarity (when $\overline{RTS}$ = 0, $\overline{DTR}$ = 1). When the DMA is ready to transmit, it drives $\overline{RTS}$ = 1 (and DTR = 0) 2-bit times before the transmission starts. When the DMA terminates the transmission, it drives $\overline{RTS}$ = 0 (and $\overline{DTR}$ = 1) after the transmission stops. When 485E is set to 1, bit 4 (DTR) and bit 5 (RTS) in the MCR register (see Section 5.5.7.6) have no effect. Also, see bit 1 (RCVE) in the MCR register (see Section 5.5.7.6).

5.5.7.5 Transmitter Flow Control

On reset (power up, USB, or soft reset) the transmitter defaults to the Xon state and the flow control is set to mode-0 (flow control is disabled).

Table 5-11. Transmitter Flow-Control Modes

	BIT 3	BIT 2	BIT 1	BIT 0
	DSR	**CTS**	**TXOA**	**TXOF**
All flow control is disabled	0	0	0	0
Xon/Xoff flow control is enabled	0	0	0	1
Xon on any/ Xoff flow control	0	0	1	0
Not permissible(1)	X	X	1	1
$\overline{\text{CTS}}$ flow control	0	1	0	0
Combination flow control(2)	0	1	0	1
Combination flow control	0	1	1	0
$\overline{\text{DSR}}$ flow control	1	0	0	0
Combination flow control	1	0	0	1
	1	0	1	0
	1	1	0	0
	1	1	0	1
	1	1	1	0

(1) This is a no permissible combination. If used, TXOA and TXOF are cleared.
(2) Combination example: Transmitter stops when either CTS or Xoff is detected. Transmitter resumes when both CTS is negated and Xon is detected.

Table 5-12. Receiver Flow-Control Possibilities

MODE		BIT 6	BIT 5	BIT 4
		DTR	**RTS**	**RXOF**
0	All flow control is disabled	0	0	0
1	Xon/Xoff flow control is enabled	0	0	1
2	$\overline{\text{RTS}}$ flow control	0	1	0
3	Combination flow control(1)	0	1	1
4	$\overline{\text{DTR}}$ flow control	1	0	0
5	Combination flow control	1	0	1
6	Combination flow control(2)	1	1	0
7	Combination flow control	1	1	1

(1) Combination example: Both RTS is asserted and Xoff transmitted when the FIFO is full. Both RTS is deasserted and Xon is transmitted when the FIFO is empty.
(2) Combination example: Both DTR and RTS are asserted when the FIFO is full. Both DTR and RTS are deasserted when the FIFO is empty.

5.5.7.6 MCR: Modem-Control Register (Addr:FFA4h)

This register provides control for modem interface I/O and definition of the flow control mode.

7	6	5	4	3	2	1	0
LCD	LRI	RTS	DTR	RSV	LOOP	RCVE	URST
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
0	URST	0	UART soft reset. This bit can be used by the MCU to reset the UART. URST = 0 Normal operation. Writing a 0 by MCU has no effect. URST = 1 When the MCU writes a 1 to this bit, a UART reset is generated (ORed with hard reset). When the UART exits the reset state, URST is cleared. The MCU can monitor this bit to determine if the UART completed the reset cycle.
1	RCVE	0	Receiver enable bit. This bit is valid only when bit 7 (485E) in the FCRL register (see Section 5.5.7.4) is 1 (RS-485 mode). When 485E = 0, this bit has no effect on the receiver. RCVE = 0 When 485E = 1, the UART receiver is disabled when $\overline{\text{RTS}}$ = 1, that is, when data is being transmitted, the UART receiver is disabled. RCVE = 1 When 485E = 1, the UART receiver is enabled regardless of the RTS state, that is, UART receiver is enabled all the time. This mode can detect collisions on the RS-485 bus when received data does not match transmitted data.
2	LOOP	0	This bit controls the normal-/loop-back mode of operation (see Figure 5-11). LOOP = 0 Normal operation LOOP = 1 Enable loop-back mode of operation. In this mode the following occur: • SOUT is set high • SIN is disconnected from the receiver input. • The transmitter serial output is looped back into the receiver serial input. • The four modem-control inputs: $\overline{\text{CTS}}$, $\overline{\text{DSR}}$, $\overline{\text{DCD}}$, and $\overline{\text{RI}}$/CP are disconnected. • DTR, RTS, LRI and LCD are internally connected to the four modem-control inputs, and read in the MSR register (see Section 5.5.7.8) as described below. Note: the FCRL register (see Section 5.5.7.4) must be configured to enable bits 2 (CTS) and 3 (DSR) to maintain proper operation with flow control and loop back. – DTR is reflected in MSR register bit 4 (LCTS) – RTS is reflected in MSR register bit 5 (LDSR) – LRI is reflected in MSR register bit 6 (LRI) – LCD is reflected in MSR register bit 7 (LCD)
3	RSV	0	Reserved
4	DTR	0	This bit controls the state of the $\overline{\text{DTR}}$ output terminal (see Figure 5-11). This bit has no effect when auto-flow control is used or when bit 7 (485E) = 1 (in the FCRL register, see Section 5.5.7.4). DTR = 0 Forces the $\overline{\text{DTR}}$ output terminal to inactive (high) DTR = 1 Forces the $\overline{\text{DTR}}$ output terminal to active (low)
5	RTS	0	This bit controls the state of the $\overline{\text{RTS}}$ output terminal (see Figure 5-11). This bit has no effect when auto-flow control is used or when bit 7 (485E) = 1 (in the FCRL register, see Section 5.5.7.4). RTS = 0 Forces the $\overline{\text{RTS}}$ output terminal to inactive (high) RTS = 1 Forces the $\overline{\text{RTS}}$ output terminal to active (low)
6	LRI	0	This bit is used for loop-back mode only. When in loop-back mode, this bit is reflected in bit 6 (LRI) in the MSR register, (see Section 5.5.7.8 and Figure 5-11). LRI = 0 Clears the MSR register bit 6 to 0 LRI = 1 Sets the MSR register bit 6 to 1
7	LCD	0	This bit is used for loop-back mode only. When in loop-back mode, this bit is reflected in bit 7 (LCD) in the MSR register, (see Section 5.5.7.8 and Figure 5-11). LCD = 0 Clears the MSR register bit 7 to 0 LCD = 1 Sets the MSR register bit 7 to 1

5.5.7.7 LSR: Line-Status Register (Addr:FFA5h)

This register provides the status of the data transfer. DMA transfer is halted when any of bit 0 (OVR), bit 1 (PTE), bit 2 (FRE), or bit 3 (BRK) is 1.

7	6	5	4	3	2	1	0
RSV	TEMT	TxE	RxF	BRK	FRE	PTE	OVR
R/O	R/O	R/O	R/O	R/C	R/C	R/C	R/C

BIT	NAME	RESET	FUNCTION
0	OVR	0	This bit indicates the overrun condition of the receiver. If set, it halts the DMA transfer and generates a status interrupt (if enabled). OVR = 0 No overrun error OVR = 1 Overrun error has occurred. Clears when the MCU writes a 1. Writing a 0 has no effect.
1	PTE	0	This bit indicates the parity condition of the received byte. If set, it halts the DMA transfer and generates a status interrupt (if enabled). PTE = 0 No parity error in data received PTE = 1 Parity error in data received. Clears when the MCU writes a 1. Writing a 0 has no effect.
2	FRE	0	This bit indicates the framing condition of the received byte. If set, it halts the DMA transfer and generates a status interrupt (if enabled). FRE = 0 No framing error in data received FRE = 1 Framing error in data received. Clears when MCU writes a 1. Writing a 0 has no effect.
3	BRK	0	This bit indicates the break condition of the received byte. If set, it halts the DMA transfer and generates a status interrupt (if enabled). BRK = 0 No break condition BRK = 1 A break condition in data received was detected. Clears when the MCU writes a 1. Writing a 0 has no effect.
4	RxF	0	This bit indicates the condition of the receiver data register. Typically, the MCU does not monitor this bit because data transfer is done by the DMA controller. RxF = 0 No data in the RDR RxF = 1 RDR contains data. Generates RX interrupt (if enabled).
5	TxE	1	This bit indicates the condition of the transmitter data register. Typically, the MCU does not monitor this bit because data transfer is done by the DMA controller. TxE = 0 TDR is not empty TxE = 1 TDR is empty. Generates TX interrupt (if enabled).
6	TEMT	1	This bit indicates the condition of both transmitter data register and shift register is empty. TEMT = 0 Either TDR or TSR is not empty TEMT = 1 Both TDR and TSR are empty
7	RSV	0	Reserved = 0

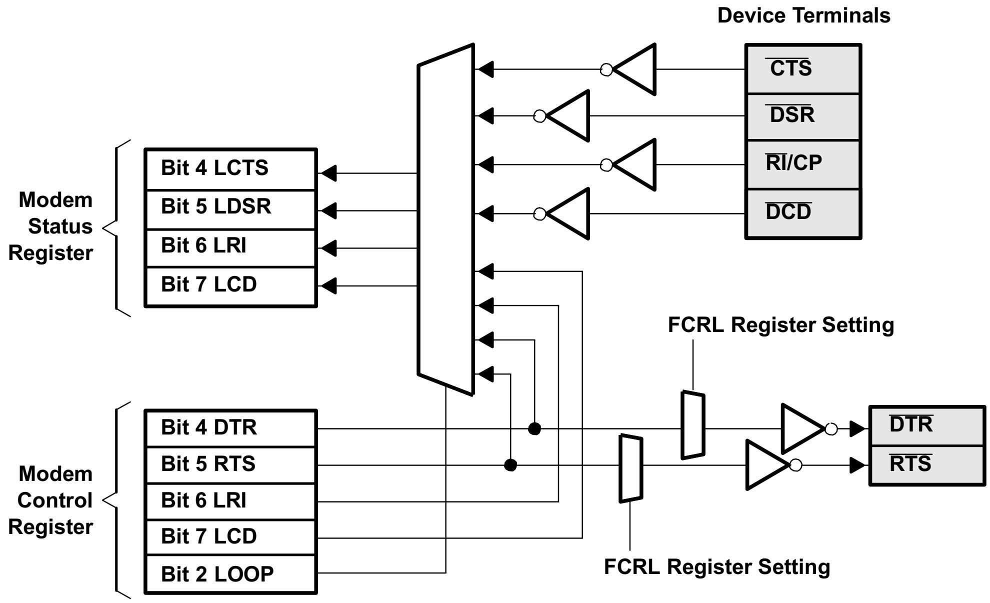


Figure 5-11. MSR and MCR Registers in Loop-Back Mode

5.5.7.8 MSR: Modem-Status Register (Addr:FFA6h)

This register provides information about the current state of the control lines from the modem.

7	6	5	4	3	2	1	0
LCD	LRI	LDSR	LCTS	ΔCD	TRI	ΔDSR	ΔCTS
R/O	R/O	R/O	R/O	R/C	R/C	R/C	R/C

BIT	NAME	RESET	FUNCTION
0	ΔCTS	0	This bit indicates that the $\overline{\text{CTS}}$ input has changed state. Cleared when the MCU writes a 1 to this bit. Writing a 0 has no effect.
1	ΔDSR	0	This bit indicates that the $\overline{\text{DSR}}$ input has changed state. Cleared when the MCU writes a 1 to this bit. Writing a 0 has no effect. ΔDSR = 0 Indicates no change in the $\overline{\text{DSR}}$ input ΔDSR = 1 Indicates that the $\overline{\text{DSR}}$ input has changed state since the last time it was read. Clears when the MCU writes a 1. Writing a 0 has no effect.
2	TRI	0	Trailing edge of the ring indicator. This bit indicates that the $\overline{\text{RI}}$/CP input has changed from low to high. This bit is cleared when the MCU writes a 1 to this bit. Writing a 0 has no effect. TRI = 0 Indicates no applicable transition on the $\overline{\text{RI}}$/CP input TRI = 1 Indicates that an applicable transition has occurred on the $\overline{\text{RI}}$/CP input.
3	ΔCD	0	This bit indicates that the $\overline{\text{CD}}$ input has changed state. Cleared when the MCU writes a 1 to this bit. Writing a 0 has no effect. ΔCD = 0 Indicates no change in the $\overline{\text{CD}}$ input ΔCD = 1 Indicates that the $\overline{\text{CD}}$ input has changed state since the last time it was read.
4	LCTS	0	During loopback, this bit reflects the status of bit 4 (DTR) in the MCR register (see Section 5.5.7.6 and Figure 5-11). LCTS = 0 $\overline{\text{CTS}}$ input is high LCTS = 1 $\overline{\text{CTS}}$ input is low
5	LDSR	0	During loop back, this bit reflects the status of bit 5 (RTS) in the MCR register (see Section 5.5.7.6 and Figure 5-11). LDSR = 0 $\overline{\text{DSR}}$ input is high LDSR= 1 $\overline{\text{DSR}}$ input is low
6	LRI	0	During loop back, this bit reflects the status of bit 6 (LRI) in the MCR register (see Section 5.5.7.6 and Figure 5-11). LRI = 0 $\overline{\text{RI}}$/CP input is high LRI = 1 $\overline{\text{RI}}$/CP input is low
7	LCD	0	During loopback, this bit reflects the status of bit 7 (LCD) in the MCR register (see Section 5.5.7.6 and Figure 5-11). LCD = 0 $\overline{\text{CD}}$ input is high LCD = 1 $\overline{\text{CD}}$ input is low

5.5.7.9 DLL: Divisor Register Low Byte (Addr:FFA7h)

This register contains the low byte of the baud-rate divisor.

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
7−0	D[7:0]	08h	Low-byte value of the 16-bit divisor for generation of the baud clock in the baud-rate generator.

5.5.7.10 DLH: Divisor Register High Byte (Addr:FFA8h)

This register contains the high byte of the baud-rate divisor.

7	6	5	4	3	2	1	0
D15	D14	D13	D12	D11	D10	D9	D8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
7–0	D[15:8]	00h	High-byte value of the 16-bit divisor for generation of the baud clock in the baud-rate generator.

5.5.7.11 Baud-Rate Calculation

Equation 1 and Equation 2 calculate the baud-rate clock and the divisors. The baud-rate clock is derived from the 96-MHz master clock (dividing by 6.5). Table 5-13 presents the divisors used to achieve the desired baud rates, together with the associate rounding errors.

$$\text{Baud CLK} = \frac{96\text{ MHz}}{6.5} = 14.76923077\text{ MHz} \tag{1}$$

$$\text{Divisor} = \frac{14.76923077 \times 10^6}{\text{Desired Baud Rate} \times 16} \tag{2}$$

Table 5-13. DLL/DLH Values and Resulted Baud Rates(1)

DESIRED BAUD RATE	DLL/DLH VALUE		BAUD RATE (bps)	ERROR %
	DECIMAL	HEXADECIMAL		
1 200	769	301	1 200.36	0.03
2 400	385	181	2 397.60	0.01
4 800	192	00C0	4 807.69	0.16
7 200	128	80	7 211.54	0.16
9 600	96	60	9 615.38	0.16
14 400	64	40	14 423.08	0.16
19 200	48	30	19 230.77	0.16
38 400	24	18	38 461.54	0.16
57 600	16	10	57 692.31	0.16
115 200	8	8	115 384.62	0.16
230 400	4	4	230 769.23	0.16
460 800	2	2	461 538.46	0.16
921 600	1	1	923 076.92	0.16

(1) The TUSB3410 device does support baud rates lower than 1200 bps, which are not listed due to less interest.

5.5.7.12 XON: Xon Register (Addr:FFA9h)

This register contains a value that is compared to the received data stream. Detection of a match interrupts the MCU (only if the interrupt enable bit is set). This value is also used for Xon transmission.

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
7−0	D[7:0]	0000	Xon value to be compared to the incoming data stream

5.5.7.13 XOFF: Xoff Register (Addr:FFAAh)

This register contains a value that is compared to the received data stream. Detection of a match halts the DMA transfer, and interrupts the MCU (only if the interrupt enable bit is set). This value is also used for Xoff transmission.

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
7−0	D[7:0]	0000	Xoff value to be compared to the incoming data stream

5.5.7.14 MASK: UART Interrupt-Mask Register (Addr:FFABh)

This register controls the UART interrupt sources.

7	6	5	4	3	2	1	0
RSV	RSV	RSV	RSV	RSV	TRI	SIE	MIE
R/O	R/O	R/O	R/O	R/O	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
0	MIE	0	This bit controls the UART-modem interrupt. MIE = 0 Modem interrupt is disabled MIE = 1 Modem interrupt is enabled
1	SIE	0	This bit controls the UART-status interrupt. SIE = 0 Status interrupt is disabled SIE = 1 Status interrupt is enabled
2	TRI	0	This bit controls the UART-TxE/RxF interrupts TRI = 0 TxE/RxF interrupts are disabled TRI = 1 TxE/RxF interrupts are enabled
7−3	RSV	0	Reserved = 0

5.5.8 Expanded GPIO Port

5.5.8.1 Input/Output and Control Registers

The TUSB3410 device has four general-purpose I/O terminals (P3.0, P3.1, P3.3, and P3.4) that are controlled by firmware running on the MCU. Each terminal can be controlled individually and each is implemented with a 12-mA push/pull CMOS output with 3-state control plus input. The MCU treats the outputs as open drain types in that the output can be driven low continuously, but a high output is driven for two clock cycles and then the output is high impedance.

An input terminal can be read using the MOV instruction. For example, MOV C, P3.3 reads the input on P3.3. As a precaution, be certain the associated output is high impedance before reading the input.

An output can be set high (and then high impedance) using the SETB instruction. For example, SETB P3.1 sets P3.1 high. An output can be set low using the CLR instruction, as in CLR P3.4, which sets P3.4 low (driven continuously until changed).

Each GPIO terminal has an associated internal pullup resistor. It is strongly recommended that the pullup resistor remain connected to the terminal to prevent oscillations in the input buffer. The only exception is if an external source always drives the input.

5.5.8.1.1 PUR_3: GPIO Pullup Register for Port 3 (Addr:FF9Eh)

7	6	5	4	3	2	1	0
RSV	RSV	RSV	Pin4	Pin3	RSV	Pin1	Pin0
R/O	R/O	R/O	R/W	R/W	R/O	R/W	R/W

BIT	NAME	RESET	FUNCTION
0 1 3 4	Pin0 Pin1 Pin3 Pin4	0 0 0 0	The MCU may write to this register. If the MCU sets any of these bits to 1, then the pullup resistor is disconnected from the associated terminal. If the MCU clears any of these bits to 0, then the pullup resistor is connected from the terminal. The pullup resistor is connected to the V_{CC} power supply.
2, 5, 6, 7	RSV	Reserved	This bit controls the UART-status interrupt.

5.5.9 Interrupts

5.5.9.1 8052 Interrupt and Status Registers

All 8052 standard, five interrupt sources are preserved. SIE is the standard interrupt-enable register that controls the five interrupt sources. This is also known as IE0 located at S:A8h in the special function register area. All the additional interrupt sources are ORed together to generate EX0.

Table 5-14. 8052 Interrupt Location Map

INTERRUPT SOURCE	DESCRIPTION	START ADDRESS	COMMENTS
ES	UART interrupt	0023h	
ET1	Timer-1 interrupt	001Bh	
EX1	External interrupt-1	0013h	
ET0	Timer-0 interrupt	000Bh	
EX0	External interrupt-0	0003h	Used for all internal peripherals
Reset		0000h	

5.5.9.1.1 8052 Standard Interrupt Enable (SIE) Register

7	6	5	4	3	2	1	0
EA	RSV	RSV	ES	ET1	EX1	ET0	EX0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
0	EX0	0	Enable or disable external interrupt-0 EX0 = 0 External interrupt-0 is disabled EX0 = 1 External interrupt-0 is enabled
1	ET0	0	Enable or disable timer-0 interrupt ET0 = 0 Timer-0 interrupt is disabled ET0 = 1 Timer-0 interrupt is enabled
2	EX1	0	Enable or disable external interrupt-1 EX1 = 0 External interrupt-1 is disabled EX1 = 1 External interrupt-1 is enabled
3	ET1	0	Enable or disable timer-1 interrupt ET1 = 0 Timer-1 interrupt is disabled EX1 = 1 Timer-1 interrupt is enabled
4	ES	0	Enable or disable serial port interrupts ES = 0 Serial-port interrupt is disabled ES = 1 Serial-port interrupt is enabled
5, 6	RSV	0	Reserved
7	EA	0	Enable or disable all interrupts (global disable) EA = 0 Disable all interrupts EA = 1 Each interrupt source is individually controlled

5.5.9.1.2 Additional Interrupt Sources

All nonstandard 8052 interrupts (DMA, I^2C, and so on) are ORed to generate an internal INT0. Furthermore, the INT0 must be programmed as an active low-level interrupt (not edge-triggered). After reset, if INT0 is not changed, then it is an edge-triggered interrupt. A vector interrupt register is provided to identify all interrupt sources (see Section 5.5.9.1.3. Up to 64 interrupt vectors are provided. It is the responsibility of the MCU to read the vector and dispatch to the proper interrupt routine.

5.5.9.1.3 VECINT: Vector Interrupt Register (Addr:FF92h)

This register contains a vector value, which identifies the internal interrupt source that is trapped to location 0003h. Writing (any value) to this register removes the vector and updates the next vector value (if another interrupt is pending).

NOTE

The vector value is offset; therefore, its value is in increments of two (bit 0 is set to 0).

When no interrupt is pending, the vector is set to 00h (see Table 5-15). As shown, the interrupt vector is divided to two fields: I[2:0] and G[3:0]. The I field defines the interrupt source within a group (on a first-come-first-served basis). In the G field, which defines the group number, group G0 is the lowest and G15 is the highest priority.

7	6	5	4	3	2	1	0
G3	G2	G1	G0	I2	I1	I0	0
R/O	R/O	R/O	R/O	R/O	R/O	R/O	R/O

BIT	NAME	RESET	FUNCTION
3−1	I[2:0]	0H	This field defines the interrupt source in a given group (see Table 5-15). Bit 0 = 0 always; therefore, vector values are offset by two.
7−4	G[3:0]	0H	This field defines the interrupt group. I[2:0] and G[3:0] combine to produce the actual interrupt vector.

Table 5-15. Vector Interrupt Values

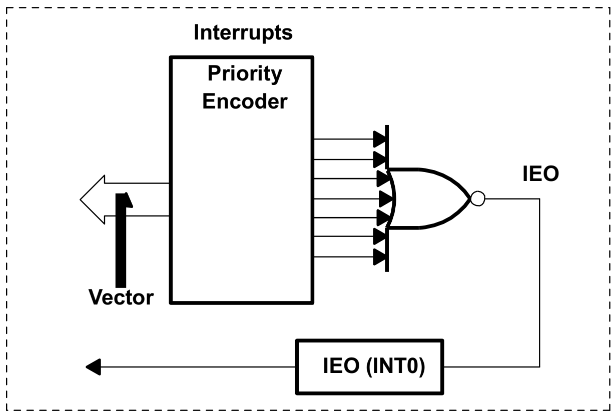
G[3:0] (Hex)	I[2:0] (Hex)	VECTOR (Hex)	INTERRUPT SOURCE
0	0	00	No interrupt
1	0	10	Not used
1	1	12	Output endpoint-1
1	2	14	Output endpoint-2
1	3	16	Output endpoint-3
1	4−7	18−1E	Reserved
2	0	20	Reserved
2	1	22	Input endpoint-1
2	2	24	Input endpoint-2
2	3	26	Input endpoint-3
2	4−7	28−2E	Reserved
3	0	30	STPOW packet received
3	1	32	SETUP packet received
3	2	34	Reserved
3	3	36	Reserved
3	4	38	RESR interrupt
3	5	3A	SUSR interrupt
3	6	3C	RSTR interrupt
3	7	3E	Wakeup
4	0	40	I^2C TXE interrupt
4	1	42	I^2C RXF interrupt
4	2	44	Input endpoint-0
4	3	46	Output endpoint-0
4	4−7	48 → 4E	Reserved
5	0	50	UART status interrupt
5	1	52	UART modem interrupt
5	2−7	54 → 5E	Reserved
6	0	60	UART RXF interrupt
6	1	62	UART TXE interrupt
6	2−7	64 → 6E	Reserved
7	0−7	70 → 7E	Reserved

Table 5-15. Vector Interrupt Values (continued)

G[3:0] (Hex)	I[2:0] (Hex)	VECTOR (Hex)	INTERRUPT SOURCE
8 8 8	0 2 3−7	80 84 86−8E	DMA1 interrupt DMA3 interrupt Reserved
9−15	X	90 → FE	Not used

5.5.9.1.4 Logical Interrupt Connection Diagram (Internal/External)

Figure 5-12 shows the logical connection of the interrupt sources and its relationship to INT0. The priority encoder generates an 8-bit vector, corresponding to 64 interrupt sources (not all are used). The interrupt priorities are hardwired. Vector 0x88 is the highest and 0x12 is the lowest.



Copyright © 2017, Texas Instruments Incorporated

Figure 5-12. Internal Vector Interrupt

5.5.10 I²C Registers

5.5.10.1 I2CSTA: I²C Status and Control Register (Addr:FFF0h)

This register controls the stop condition for read and write operations. In addition, it provides transmitter and receiver handshake signals with their respective interrupt enable bits.

7	6	5	4	3	2	1	0
RXF	RIE	ERR	1/4	TXE	TIE	SRD	SWR
R/O	R/W	R/C	R/W	R/O	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
0	SWR	0	Stop write condition. This bit determines if the I²C controller generates a stop condition when data from the I2CDAO register is transmitted to an external device. SWR = 0 Stop condition is not generated when data from the I2CDAO register is shifted out to an external device. SWR = 1 Stop condition is generated when data from the I2CDAO register is shifted out to an external device.
1	SRD	0	Stop read condition. This bit determines if the I²C controller generates a stop condition when data is received and loaded into the I2CDAI register. SRD = 0 Stop condition is not generated when data from the SDA line is shifted into the I2CDAI register. SRD = 1 Stop condition is generated when data from the SDA line are shifted into the I2CDAI register.
2	TIE	0	I²C transmitter empty interrupt enable TIE = 0 Interrupt disable TIE = 1 Interrupt enable
3	TXE	1	I²C transmitter empty. This bit indicates that data can be written to the transmitter. It can be used for polling or it can generate an interrupt. TXE = 0 Transmitter is full. This bit is cleared when the MCU writes a byte to the I2CDAO register. TXE = 1 Transmitter is empty. The I²C controller sets this bit when the contents of the I2CDAO register are copied to the SDA shift register.
4	1/4	0	Bus speed selection(1) 1/4 = 0 100-kHz bus speed 1/4 = 1 400-kHz bus speed
5	ERR	0	Bus error condition. This bit is set by the hardware when the device does not respond. It is cleared by the MCU. ERR = 0 No bus error ERR = 1 Bus error condition has been detected. Clears when the MCU writes a 1. Writing a 0 has no effect.
6	RIE	0	I²C receiver ready interrupt enable RIE = 0 Interrupt disable RIE = 1 Interrupt enable
7	RXF	0	I²C receiver full. This bit indicates that the receiver contains new data. It can be used for polling or it can generate an interrupt. RXF = 0 Receiver is empty. This bit is cleared when the MCU reads the I2CDAI register. RXF = 1 Receiver contains new data. This bit is set by the I²C controller when the received serial data has been loaded into the I2CDAI register.

(1) The bootcode automatically sets the I²C bus speed to 400 kHz. Only 400-kHz I²C EEPROMs can be used.

5.5.10.2 I2CADR: I²C Address Register (Addr:FFF3h)

This register holds the device address and the read/write command bit.

7	6	5	4	3	2	1	0
A6	A5	A4	A3	A2	A1	A0	R/W
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
0	R/W	0	Read/write command bit R/W = 0 Write operation R/W = 1 Read operation
7−1	A[6:0]	0h	Seven address bits for device addressing

5.5.10.3 I2CDAI: I²C Data-Input Register (Addr:FFF2h)

This register holds the received data from an external device.

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0
R/O	R/O	R/O	R/O	R/O	R/O	R/O	R/O

BIT	NAME	RESET	FUNCTION
7-0	D[7:0]	0	8-bit input data from an I²C device

5.5.10.4 I2CDAO: I²C Data-Output Register (Addr:FFF1h)

This register holds the data to be transmitted to an external device. Writing to this register starts the transfer on the SDA line.

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0
W/O	W/O	W/O	W/O	W/O	W/O	W/O	W/O

BIT	NAME	RESET	FUNCTION
7-0	D[7:0]	0	8-bit input data from an I²C device

5.6 Boot Modes

5.6.1 Introduction

The TUSB3410 device bootcode is a program embedded in the 10k-byte boot ROM within the TUSB3410 device. This program is designed to load application firmware from either an external I²C memory device or USB host bootloader device driver. After the TUSB3410 device finishes downloading, the bootcode releases its control to the application firmware.

This section describes how the bootcode initializes the TUSB3410 device in detail. In addition, the default USB descriptor, I²C device header format, USB host driver firmware downloading format, and supported built-in USB vendor specific requests are listed for reference. Users should carefully follow the appropriate format to interface with the bootcode. Unsupported formats may cause unexpected results.

The bootcode source code is also provided for programming reference.

5.6.2 Bootcode Programming Flow

After power-on reset, the bootcode initializes the I^2C and USB registers along with internal variables. The bootcode then checks to see if an I^2C device is present and contains a valid signature. If an I^2C device is present and contains a valid signature, the bootcode continues searching for descriptor blocks and then processes them if the checksum is correct. If application firmware was found, then the bootcode downloads it and releases the control to the application firmware. Otherwise, the bootcode connects to the USB and waits for host driver to download application firmware. Once firmware downloading is complete, the bootcode releases the control to the firmware.

The following is the bootcode step-by-step operation.

- Check if bootcode is in the application mode. This is the mode that is entered after application code is downloaded through either an I^2C device or the USB. If the bootcode is in the application mode, then the bootcode releases the control to the application firmware. Otherwise, the bootcode continues.
- Initialize all the default settings.
  - Call CopyDefaultSettings() routine.

    Set I^2C to 400-kHz speed.
  - Call UsbDataInitialization() routine.

    Set bFUNADR = 0
    Disconnect from USB (bUSBCTL = 0x00)
    Bootcode handles USB reset
    Copy predefined device, configuration, and string descriptors to RAM
    Disable all endpoints and enable USB interrupts (SETUP, RSTR, SUSR, and RESR)
- Search for product signature
  - Check if valid signature is in I^2C. If not, skip the I^2C process.

    Read 2 bytes from address 0x0000 with type III and device address 0. Stop searching if valid signature is found.
    Read 2 bytes from address 0x0000 with type II and device address 4. Stop searching if valid signature is found.
- If a valid I^2C signature is found, then load the customized device, configuration and string descriptors from I^2C EEPROM.
  - Process each descriptor block from I^2C until *end of header* is found

    If the descriptor block contains device, configuration, or string descriptors, then the bootcode overwrites the default descriptors.

    If the descriptor block contains binary firmware, then the bootcode sets the header pointer to the beginning of the binary firmware in the I^2C EEPROM.

    If the descriptor block is *end of header*, then the bootcode stops searching.
- Enable global and USB interrupts and set the connection bit to 1.
  - Enable global interrupts by setting bit 7 (EA) within the SIE register (see Section 5.5.9.1.1) to 1.
  - Enable all internal peripheral interrupts by setting the EX0 bit within the SIE register to 1.
  - Connect to the USB by setting bit 7 (CONT) within the USBCNTL register (see Section 5.5.5.4) to 1.

- Wait for any interrupt events until Get DEVICE DESCIPTOR setup packet arrives.
  - Suspend interrupt

    The idle bit in the MCU PCON register is set and suspend mode is entered. USB reset wakes up the microcontroller.
  - Resume interrupt

    Bootcode wakes up and waits for new USB requests.
  - Reset interrupt

    Call UsbReset() routine.
  - Setup interrupt

    Bootcode processes the request.
  - USB reboot request

    Disconnect from the USB by clearing bit 7 (CONT) in the USBCTL register and restart at address 0x0000.
- Download firmware from I^2C EEPROM
  - Disable global interrupts by clearing bit 7 (EA) within the SIE register
  - Load firmware to XDATA space if available.
- Download firmware from the USB.
  - If no firmware is found in an I^2C EEPROM, the USB host downloads firmware through output endpoint 1.
  - In the first data packet to output endpoint 1, the USB host driver adds 3 bytes before the application firmware in binary format. These three bytes are the LSB and MSB indicating the firmware size and followed by the arithmetic checksum of the binary firmware.
- Release control to the application firmware.
  - Update the USB configuration and interface number.
  - Release control to application firmware.
- Application firmware
  - Either disconnect from the USB or continue responding to USB requests.

5.6.3 Default Bootcode Settings

The bootcode has its own predefined device, configuration, and string descriptors. These default descriptors should be used in evaluation only. They must not be used in the end-user product.

5.6.3.1 Device Descriptor

The device descriptor provides the USB version that the device supports, device class, protocol, vendor and product identifications, strings, and number of possible configurations. The operation system (Windows, MAC, or Linux) reads this descriptor to decide which device driver should be used to communicate with this device.

The bootcode uses 0x0451 (Texas Instruments) as the vendor ID and 0x3410 (TUSB3410) as the product ID. It also supports three different strings and one configuration. Table 5-16 lists the device descriptor.

Table 5-16. Device Descriptor

OFFSET (decimal)	FIELD	SIZE	VALUE	DESCRIPTION
0	bLength	1	0x12	Size of this descriptor in bytes
1	bDescriptorType	1	1	Device descriptor type
2	bcdUSB	2	0x0110	USB spec 1.1
4	bDeviceClass	1	0xFF	Device class is vendor−specific
5	bDeviceSubClass	1	0	We have no subclasses.
6	bDeviceProtocol	1	0	We use no protocols.
7	bMaxPacketSize0	1	8	Max. packet size for endpoint zero
8	idVendor	2	0x0451	USB−assigned vendor ID = TI
10	idProduct	2	0x3410	TI part number = TUSB3410
12	bcdDevice	2	0x100	Device release number = 1.0
14	iManufacturer	1	1	Index of string descriptor describing manufacturer
15	iProducct	1	2	Index of string descriptor describing product
16	iSerialNumber	1	3	Index of string descriptor describing the serial number of the device
17	bNumConfigurations	1	1	Number of possible configurations

5.6.3.2 Configuration Descriptor

The configuration descriptor provides the number of interfaces supported by this configuration, power configuration, and current consumption.

The bootcode declares only one interface running in bus-powered mode. It consumes up to 100 mA at boot time. Table 5-17 lists the configuration descriptor.

Table 5-17. Configuration Descriptor

OFFSET (decimal)	FIELD	SIZE	VALUE	DESCRIPTION
0	bLength	1	9	Size of this descriptor in bytes.
1	bDescriptor Type	1	2	Configuration descriptor type
2	wTotalLength	2	25 = 9 + 9 + 7	Total length of data returned for this configuration. Includes the combined length of all descriptors (configuration, interface, endpoint, and class- or vendor-specific) returned for this configuration.
4	bNumInterfaces	1	1	Number of interfaces supported by this configuration
5	bConfigurationValue	1	1	Value to use as an argument to the SetConfiguration() request to select this configuration.
6	iConfiguration	1	0	Index of string descriptor describing this configuration.
7	bmAttributes	1	0x80	Configuration characteristics: D7: Reserved (set to one) D6: Self-powered D5: Remote wake up is supported D4−0: Reserved (reset to zero)
8	bMaxPower	1	0x32	This device consumes 100 mA.

5.6.3.3 Interface Descriptor

The interface descriptor provides the number of endpoints supported by this interface as well as interface class, subclass, and protocol.

The bootcode supports only one endpoint and use its own class. Table 5-18 lists the interface descriptor.

Table 5-18. Interface Descriptor

OFFSET (decimal)	FIELD	SIZE	VALUE	DESCRIPTION
0	bLength	1	9	Size of this descriptor in bytes
1	bDescriptorType	1	4	Interface descriptor type
2	bInterfaceNumber	1	0	Number of interface. Zero-based value identifying the index in the array of concurrent interfaces supported by this configuration.
3	bAlternateSetting	1	0	Value used to select alternate setting for the interface identified in the prior field
4	bNumEndpoints	1	1	Number of endpoints used by this interface (excluding endpoint zero). If this value is zero, this interface only uses the default control pipe.
5	bInterfaceClass	1	0xFF	The interface class is vendor specific.
6	bInterfaceSubClass	1	0	
7	bInterfaceProtocol	1	0	
8	iInterface	1	0	Index of string descriptor describing this interface

5.6.3.4 Endpoint Descriptor

The endpoint descriptor provides the type and size of communication pipe supported by this endpoint.

The bootcode supports only one output endpoint with the size of 64 bytes in addition to control endpoint 0 (required by all USB devices). Table 5-19 lists the endpoint descriptor.

Table 5-19. Output Endpoint1 Descriptor

OFFSET (decimal)	FIELD	SIZE	VALUE	DESCRIPTION
0	bLength	1	7	Size of this descriptor in bytes
1	bDescriptorType	1	5	Endpoint descriptor type
2	bEndpointAddress	1	0x01	Bit 3…0: The endpoint number Bit 7: Direction 0 = OUT endpoint 1 = IN endpoint
3	bmAttributes	1	2	Bit 1…0: Transfer type 10 = Bulk 11 = Interrupt
4	wMaxPacketSize	2	64	Maximum packet size this endpoint is capable of sending or receiving when this configuration is selected.
6	bInterval	1	0	Interval for polling endpoint for data transfers. Expressed in milliseconds.

5.6.3.5 String Descriptor

The string descriptor contains data in the Unicode format. It is used to show the manufacturers name, product model, and serial number in human readable format.

The bootcode supports three strings. The first string is the manufacturers name. The second string is the product name. The third string is the serial number. Table 5-20 lists the string descriptor.

Table 5-20. String Descriptor

OFFSET (decimal)	FIELD	SIZE	VALUE	DESCRIPTION
0	bLength	1	4	Size of string 0 descriptor in bytes
1	bDescriptorType	1	0x03	String descriptor type
2	wLANGID[0]	2	0x0409	English
4	bLength	1	36 (decimal)	Size of string 1 descriptor in bytes
5	bDescriptorType	1	0x03	String descriptor type
6	bString	2	T,0x00	Unicode, T is the first byte
8		2	e,0x00	Texas Instruments
10		2	x,0x00	
12		2	a,0x00	
14		2	s,0x00	
16		2	‘ ’,0x00	
18		2	I,0x00	
20		2	n,0x00	
22		2	s,0x00	
24		2	t,0x00	
26		2	r,0x00	
28		2	u,0x00	
30		2	m,0x00	
32		2	e,0x00	
34		2	n,0x00	
36		2	t,0x00	
38		2	s,0x00	
40	bLength	1	42 (decimal)	Size of string 2 descriptor in bytes
41	bDescriptorType	1	0x03	STRING descriptor type
42	bString	2	T,0x00	UNICODE, T is first byte
44		2	U,0x00	TUSB3410 boot device
46		2	S,0x00	
48		2	B,0x00	
50		2	3,0x00	
52		2	4,0x00	
54		2	1,0x00	
56		2	0,0x00	
58		2	‘ ‘,0x00	
60		2	B,0x00	
62		2	o,0x00	
64		2	o,0x00	
66		2	t,0x00	
68		2	‘ ’,0x00	
70		2	D,0x00	

Table 5-20. String Descriptor (continued)

OFFSET (decimal)	FIELD	SIZE	VALUE	DESCRIPTION
72		2	e,0x00	
74		2	v,0x00	
76		2	I,0x00	
78		2	c,0x00	
80		2	e,0x00	
82	bLength	1	34 (decimal)	Size of string 3 descriptor in bytes
84	bDescriptorType	1	0x03	STRING descriptor type
86	bString	2	r0,0x00	UNICODE
88		2	r1,0x00	R0 to rF are BCD of SERNUM0 to
90		2	r2,0x00	SERNUM7 registers. 16 digit hex
92		2	r3,0x00	16 digit hex numbers are created from
94		2	r4,0x00	SERNUM0 to SERNUM7 registers
96		2	r5,0x00	
98		2	r6,0x00	
100		2	r7,0x00	
102		2	r8,0x00	
104		2	r9,0x00	
106		2	rA,0x00	
108		2	rB,0x00	
110		2	rC,0x00	
112		2	rD,0x00	
114		2	rE,0x00	
116		2	rF,0x00	

5.6.4 *External I²C Device Header Format*

A valid header should contain a product signature and one or more descriptor blocks. The descriptor block contains the descriptor prefix and content. In the descriptor prefix, the data type, size, and checksum are specified to describe the content. The descriptor content contains the necessary information for the bootcode to process.

The header processing routine always counts from the first descriptor block until the desired block number is reached. The header reads in the descriptor prefix with a size of 4 bytes. This prefix contains the type of block, size, and checksum. For example, if the bootcode would like to find the position of the third descriptor block, then it reads in the first descriptor prefix, calculates the position on the second descriptor prefix based on the size specified in the prefix. bootcode, then repeats the same calculation to find out the position of the third descriptor block.

5.6.4.1 Product Signature

The product signature must be stored at the first 2 bytes within the I²C storage device. These 2 bytes must match the product number. The order of these 2 bytes must be the LSB first followed by the MSB. For example, the TUSB3410 device is 0x3410. Therefore, the first byte must be 0x10 and the second byte must be 0x34.

The TUSB3410 device bootcode searches the first 2 bytes of the I²C device. If the first 2 bytes are not 0x10 and 0x34, then the bootcode skips the header processing.

5.6.4.2 Descriptor Block

Each descriptor block contains a prefix and content. The size of the prefix is always 4 bytes. It contains the data type, size, and checksum for data integrity. The descriptor content contains the corresponding information specified in the prefix. It could be as small as 1 byte or as large as 65535 bytes. The next descriptor immediately follows the previous descriptor. If there are no more descriptors, then an extra byte with a value of zero should be added to indicate the end of header.

5.6.4.2.1 Descriptor Prefix

The first byte of the descriptor prefix is the data type. This tells the bootcode how to process the data in the descriptor content. The second and third bytes are the size of descriptor content. The second byte is the low byte of the size and the third byte is the high byte. The last byte is the 8-bit arithmetic checksum of descriptor content.

5.6.4.2.2 Descriptor Content

Information stored in the descriptor content can be the USB information, firmware, or other type of data. The size of the content should be from 1 byte to 65535 bytes.

5.6.5 Checksum in Descriptor Block

Each descriptor prefix contains one checksum of the descriptor content. If the checksum is wrong, the bootcode simply ignores the descriptor block.

5.6.6 Header Examples

The header can be specified in different ways. The following descriptors show examples of the header format and the supported descriptor block.

5.6.6.1 TUSB3410 Bootcode Supported Descriptor Block

The TUSB3410 device bootcode supports the following descriptor blocks.

- USB Device Descriptor
- USB Configuration Descriptor
- USB String Descriptor
- Binary Firmware (1)
- Autoexec Binary Firmware (2)

5.6.6.2 USB Descriptor Header

Table 5-21 contains the USB device, configuration, and string descriptors for the bootcode. The last byte is zero to indicate the end of header.

Table 5-21. USB Descriptors Header

OFFSET	TYPE	SIZE	VALUE	DESCRIPTION
0	Signature0	1	0x10	FUNCTION_PID_L
1	Signature1	1	0x34	FUNCTION_PID_H
2	Data Type	1	0x03	USB device descriptor
3	Data Size (low byte)	1	0x12	The device descriptor is 18 (decimal) bytes.
4	Data Size (high byte)	1	0x00	
5	Check Sum	1	0xCC	Checksum of data below
6	bLength	1	0x12	Size of device descriptor in bytes

(1) Binary firmware is loaded when the bootcode receives the first get device descriptor request from host. Downloading the firmware should either continue that request in the data stage or disconnect from the USB and then reconnect to the USB as a new device.
(2) The bootcode loads this autoexec binary firmware before it connects to the USB. The firmware should connect to the USB once it is loaded.

Table 5-21. USB Descriptors Header (continued)

OFFSET	TYPE	SIZE	VALUE	DESCRIPTION
7	bDescriptorType	1	0x01	Device descriptor type
8	bcdUSB	2	0x0110	USB spec 1.1
10	bDeviceClass	1	0xFF	Device class is vendor-specific
11	bDeviceSubClass	1	0x00	We have no subclasses.
12	bDeviceProtocol	1	0x00	We use no protocols
13	bMaxPacketSize0	1	0x08	Maximum packet size for endpoint zero
14	idVendor	2	0x0451	USB−assigned vendor ID = TI
16	idProduct	2	0x3410	TI part number = TUSB3410
18	bcdDevice	2	0x0100	Device release number = 1.0
20	iManufacturer	1	0x01	Index of string descriptor describing manufacturer
21	iProducct	1	0x02	Index of string descriptor describing product
22	iSerialNumber	1	0x03	Index of string descriptor describing device's serial number
23	bNumConfigurations	1	0x01	Number of possible configurations:
24	Data Type	1	0x04	USB configuration descriptor
25	Data Size (low byte)	1	0x19	25 bytes
26	Data Size (high byte)	1	0x00	
27	Check Sum	1	0xC6	Checksum of data below
28	bLength	1	0x09	Size of this descriptor in bytes
29	bDescriptorType	1	0x02	CONFIGURATION descriptor type
30	wTotalLength	2	25(0x19) = 9 + 9 + 7	Total length of data returned for this configuration. Includes the combined length of all descriptors (configuration, interface, endpoint, and class- or vendor-specific) returned for this configuration.
32	bNumInterfaces	1	0x01	Number of interfaces supported by this configuration
33	bConfigurationValue	1	0x01	Value to use as an argument to the SetConfiguration() request to select this configuration
34	iConfiguration	1	0x00	Index of string descriptor describing this configuration.
35	bmAttributes	1	0xE0	Configuration characteristics: D7: Reserved (set to one) D6: Self-powered D5: Remote wakeup is supported D4−0: Reserved (reset to zero)
36	bMaxPower	1	0x64	This device consumes 100 mA.
37	bLength	1	0x09	Size of this descriptor in bytes
38	bDescriptorType	1	0x04	INTERFACE descriptor type
39	bInterfaceNumber	1	0x00	Number of interface. Zero-based value identifying the index in the array of concurrent interfaces supported by this configuration.
40	bAlternateSetting	1	0x00	Value used to select alternate setting for the interface identified in the prior field
41	bNumEndpoints	1	0x01	Number of endpoints used by this interface (excluding endpoint zero). If this value is zero, this interface only uses the default control pipe.
42	bInterfaceClass	1	0xFF	The interface class is vendor specific.
43	bInterfaceSubClass	1	0x00	
44	bInterfaceProtocol	1	0x00	
45	iInterface	1	0x00	Index of string descriptor describing this interface
46	bLength	1	0x07	Size of this descriptor in bytes
47	bDescriptorType	1	0x05	ENDPOINT descriptor type:

Table 5-21. USB Descriptors Header (continued)

OFFSET	TYPE	SIZE	VALUE	DESCRIPTION
48	bEndpointAddress	1	0x01	Bit 3…0: The endpoint number Bit 7: Direction 0 = OUT endpoint 1 = IN endpoint
49	bmAttributes	1	0x02	Bit 1…0: Transfer Type 10 = Bulk 11 = Interrupt
50	wMaxPacketSize	2	0x0040	Maximum packet size this endpoint is capable of sending or receiving when this configuration is selected.
52	bInterval	1	0x00	Interval for polling endpoint for data transfers. Expressed in milliseconds.
53	Data Type	1	0x05	USB String descriptor
54	Data Size (low byte)	1	0x1A	26(0x1A) = 4 + 6 + 6 + 10
55	Data Size (high byte)	1	0x00	
56	Check Sum	1	0x50	Checksum of data below
57	bLength	1	0x04	Size of string 0 descriptor in bytes
58	bDescriptorType	1	0x03	STRING descriptor type
59	wLANGID[0]	2	0x0409	English
61	bLength	1	0x06	Size of string 1 descriptor in bytes
62	bDescriptorType	1	0x03	STRING descriptor type
63	bString	2	T,0x00	UNICODE, T is the first byte.
65		2	I,0x00	TI = 0x54, 0x49
67	bLength	1	0x06	Size of string 2 descriptor in bytes
68	bDescriptorType	1	0x03	STRING descriptor type
69	bString	2	u,0x00	UNICODE, u is the first byte.
71		2	C,0x00	µC = 0x75, 0x43
73	bLength	1	0x0A	Size of string 3 descriptor in bytes
74	bDescriptorType	1	0x03	STRING descriptor type
75	bString	2	3,0x00	UNICODE, T is the first byte.
77		2	4,0x00	3410 = 0x33, 0x34, 0x31, 0x30
79		2	1,0x00	
81		2	0,0x00	
83	Data Type	1	0x00	End of header

5.6.6.3 Autoexec Binary Firmware

If the application requires firmware loaded prior to establishing a USB connection, then the following header can be used. The bootcode loads the firmware and releases control to the firmware directly without connecting to the USB. However, per the USB specification requirement, any USB device should connect to the bus and respond to the host within the first 100 ms. Therefore, if downloading time is more than 100 ms, the USB and header speed descriptor blocks should be added before the autoexec binary firmware. Table 5-22 shows an example of autoexec binary firmware header.

Table 5-22. Autoexec Binary Firmware

OFFSET	TYPE	SIZE	VALUE	DESCRIPTION
0x0000	Signature0	1	0x10	FUNCTION_PID_L
0x0001	Signature1	1	0x34	FUNCTION_PID_H
0x0002	Data Type	1	0x07	Autoexec binary firmware
0x0003	Data Size (low byte)	1	0x67	0x4567 bytes of application code
0x0004	Data Size (high byte)	1	0x45	
0x0005	Check Sum	1	0xNN	Checksum of the following firmware
0x0006	Program	0x4567		Binary application code
0x456d	Data Type	1	0x00	End of header

5.6.7 *USB Host Driver Downloading Header Format*

If firmware downloading from the USB host driver is desired, then the USB host driver must follow the format in Table 5-23. The Texas Instruments bootloader driver generates the proper format. Therefore, users only need to provide the binary image of the application firmware for the Bootloader. If the checksum is wrong, then the bootcode disconnects from the USB and waits before it reconnects to the USB.

Table 5-23. Host Driver Downloading Format

OFFSET	TYPE	SIZE	VALUE	DESCRIPTION
0x0000	Firmware size (low byte)	1	0xXX	Application firmware size
0x0001	Firmware size (low byte)	1	0xYY	
0x0002	Checksum	1	0xZZ	Checksum of binary application code
0x0003	Program	0xYYXX		Binary application code

5.6.8 *Built-In Vendor Specific USB Requests*

The bootcode supports several vendor specific USB requests. These requests are primarily for internal testing only. These functions should not be used in normal operation.

5.6.8.1 Reboot

The reboot command forces the bootcode to execute.

VARIABLE	CONSTANT NAME	VALUE
bmRequestType	USB_REQ_TYPE_DEVICE | USB_REQ_TYPE_VENDOR | USB_REQ_TYPE_OUT	01000000b
bRequest	BTC_REBOOT	0x85
wValue	None	0x0000
wIndex	None	0x0000
wLength	None	0x0000
Data	None	

5.6.8.2 Force Execute Firmware

The force execute firmware command requests the bootcode to execute the downloaded firmware unconditionally.

VARIABLE	CONSTANT NAME	VALUE
bmRequestType	USB_REQ_TYPE_DEVICE | USB_REQ_TYPE_VENDOR | USB_REQ_TYPE_OUT	01000000b
bRequest	BTC_FORCE_EXECUTE_FIRMWARE	0x8F
wValue	None	0x0000
wIndex	None	0x0000
wLength	None	0x0000
Data	None	

5.6.8.3 External Memory Read

The bootcode returns the content of the specified address.

VARIABLE	CONSTANT NAME	VALUE
bmRequestType	USB_REQ_TYPE_DEVICE | USB_REQ_TYPE_VENDOR | USB_REQ_TYPE_OUT	11000000b
bRequest	BTC_EXETERNAL_MEMORY_WRITE	0x90
wValue	None	0x0000
wIndex	Data address	0xNNNN (From 0x0000 to 0xFFFF)
wLength	None	0x0000
Data	None	

5.6.8.4 External Memory Write

The external memory write command tells the bootcode to write data to the specified address.

VARIABLE	CONSTANT NAME	VALUE
bmRequestType	USB_REQ_TYPE_DEVICE | USB_REQ_TYPE_VENDOR | USB_REQ_TYPE_OUT	01000000b
bRequest	BTC_EXETERNAL_MEMORY_WRITE	0x91
wValue	HI: 0x00 LO: Data	0x00NN
wIndex	Data address	0xNNNN (From 0x0000 to 0xFFFF)
wLength	None	0x0000
Data	None	

5.6.8.5 I^{2}C Memory Read

The bootcode returns the content of the specified address in I^{2}C EEPROM.

In the wValue field, the I^{2}C device number is from 0x00 to 0x07 in the high byte. The memory type is from 0x01 to 0x03 for CAT I to CAT III devices. If bit 7 of bValueL is set, then the bus speed is 400 kHz. This request is also used to set the device number and speed before the I^{2}C write request.

VARIABLE	CONSTANT NAME	VALUE
bmRequestType	USB_REQ_TYPE_DEVICE | USB_REQ_TYPE_VENDOR | USB_REQ_TYPE_OUT	11000000b
bRequest	BTC_I2C_MEMORY_READ	0x92
wValue	HI: I^{2}C device number LO: Memory type bit[1:0] Speed bit[7]	0xXXYY
wIndex	Data address	0xNNNN (From 0x0000 to 0xFFFF)
wLength	1 byte	0x0001
Data	Byte in the specified address	0xNN

5.6.8.6 I^{2}C Memory Write

The I^{2}C memory write command tells the bootcode to write data to the specified address.

VARIABLE	CONSTANT NAME	VALUE
bmRequestType	USB_REQ_TYPE_DEVICE | USB_REQ_TYPE_VENDOR | USB_REQ_TYPE_OUT	01000000b
bRequest	BTC_I2C_MEMORY_WRITE	0x93
wValue	HI: should be zero LO: Data	0x00NN
wIndex	Data address	0xNNNN (From 0x0000 to 0xFFFF)
wLength	None	0x0000
Data	None	

5.6.8.7 Internal ROM Memory Read

The bootcode returns the byte of the specified address within the boot ROM. That is, the binary code of the bootcode.

VARIABLE	CONSTANT NAME	VALUE
bmRequestType	USB_REQ_TYPE_DEVICE | USB_REQ_TYPE_VENDOR | USB_REQ_TYPE_OUT	01000000b
bRequest	BTC_INTERNAL_ROM_MEMORY_READ	0x94
wValue	None	0x0000
wIndex	Data address	0xNNNN (From 0x0000 to 0xFFFF)
wLength	1 byte	0x0001
Data	Byte in the specified address	0xNN

5.6.9 Bootcode Programming Consideration

5.6.9.1 USB Requests

For each USB request, the bootcode follows these steps to ensure proper operation of the hardware:

1. Determine the direction of the request by checking the MSB of the bmRequestType field and set the DIR bit within the USBCTL register accordingly.
2. Decode the command
3. If another setup is pending, then return. Otherwise, serve the request.
4. Check again, if another setup is pending then go to step 2.
5. Clear the interrupt source and then the VECINT register.
6. Exit the interrupt routine.

5.6.9.1.1 USB Request Transfers

The USB request consist of three types of transfers. They are control-read-with-data-stage, control-write-without-data-stage, and control-write-with-data-stage transfer. In each transfer, arrows indicate interrupts generated after receiving the setup packet, in or out token.

Figure 5-13 and Figure 5-14 show the USB data flow and how the hardware and firmware respond to the USB requests. Table 5-24 and Table 5-25 lists the bootcode reposes to the standard USB requests.

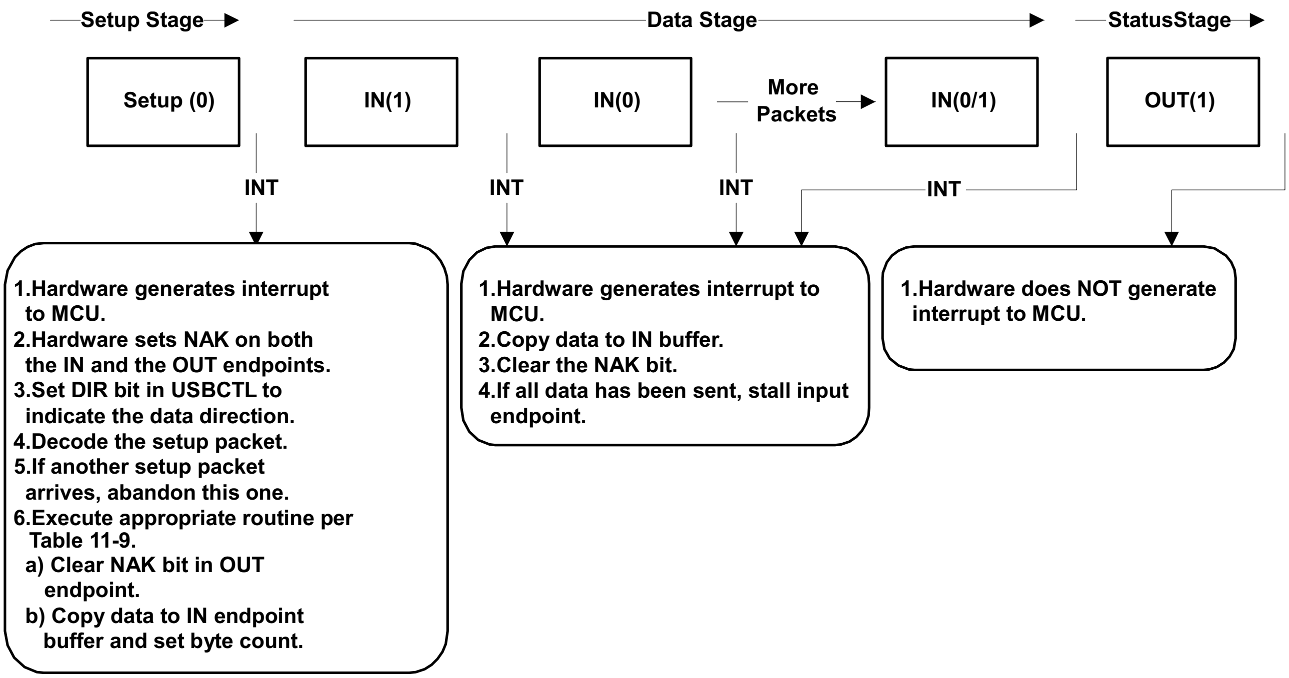


Figure 5-13. Control Read Transfer

Table 5-24. Bootcode Response to Control Read Transfer

CONTROL READ	ACTION IN BOOTCODE
Get status of device	Return power and remote wake-up settings
Get status of interface	Return 2 bytes of zeros
Get status of endpoint	Return endpoint status
Get descriptor of device	Return device descriptor
Get descriptor of configuration	Return configuration descriptor
Get descriptor of string	Return string descriptor
Get descriptor of interface	Stall
Get descriptor of endpoint	Stall
Get configuration	Return bConfiguredNumber value
Get interface	Return bInterfaceNumber value

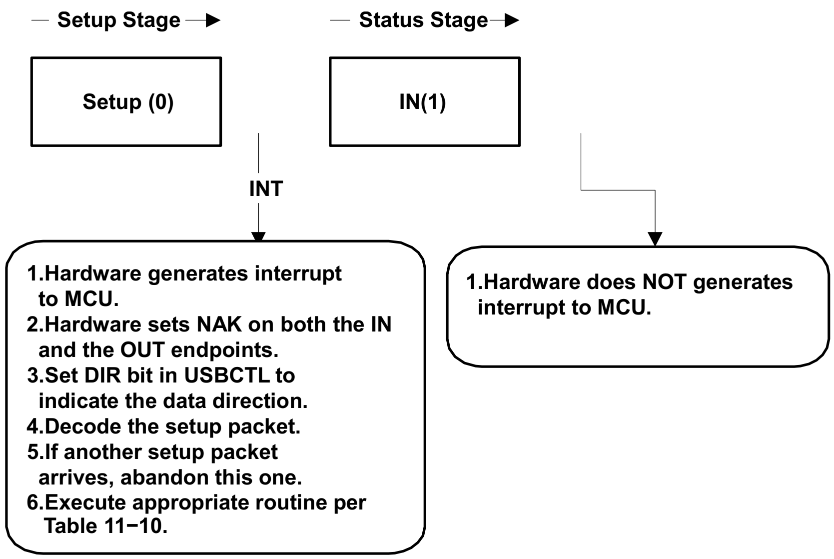


Figure 5-14. Control Write Transfer Without Data Stage

Table 5-25. Bootcode Response to Control Write Without Data Stage

CONTROL WRITE WITHOUT DATA STAGE	ACTION IN BOOTCODE
Clear feature of device	Stall
Clear feature of interface	Stall
Clear feature of endpoint	Clear endpoint stall
Set feature of device	Stall
Set feature of interface	Stall
Set feature of endpoint	Stall endpoint
Set address	Set device address
Set descriptor	Stall
Set configuration	Set bConfiguredNumber
Set interface	SetbInterfaceNumber
Sync. frame	Stall

5.6.9.1.2 Interrupt Handling Routine

The higher-vector number has a higher priority than the lower-vector number. Table 5-26 lists all the interrupts and source of interrupts.

Table 5-26. Vector Interrupt Values and Sources

G[3:0] (Hex)	I[2:0] (Hex)	VECTOR (Hex)	INTERRUPT SOURCE	INTERRUPT SOURCE MUST BE CLEARED
0	0	0	No Interrupt	No Source
1 1 1	1 2 3	12 14 16	Output−endpoint−1 Output−endpoint−2 Output−endpoint−3	VECINT register VECINT register VECINT register
1	4−7	18→1E	Reserved	
2 2 2	1 2 3	22 24 26	Input−endpoint−1 Input−endpoint−2 Input−endpoint−3	VECINT register VECINT register VECINT register
2	4−7	28→2E	Reserved	

Table 5-26. Vector Interrupt Values and Sources (continued)

G[3:0] (Hex)	I[2:0] (Hex)	VECTOR (Hex)	INTERRUPT SOURCE	INTERRUPT SOURCE MUST BE CLEARED
3	0	30	STPOW packet received	USBSTA / VECINT registers
3	1	32	SETUP packet received	USBSTA / VECINT registers
3	2	34	Reserved	—
3	3	36	Reserved	—
3	4	38	RESR interrupt	USBSTA / VECINT registers
3	5	3A	SUSR interrupt	USBSTA / VECINT registers
3	6	3C	RSTR interrupt	USBSTA / VECINT registers
3	7	3E	Wake-up interrupt	USBSTA / VECINT registers
4	0	40	I^2C TXE interrupt	VECINT register
4	1	42	I^2C TXE interrupt	VECINT register
4	2	44	Input−endpoint−0	VECINT register
4	3	46	Output−endpoint−0	VECINT register
4	4−7	48→4E	Reserved	
5	0	50	UART1 status interrupt	LSR / VECNT register
5	1	52	UART1 modern interrupt	LSR / VECINT register
5	2−7	54→5E	Reserved	
6	0	60	UART1 RXF interrupt	LSR / VECNT register
6	1	62	UART1 TXE interrupt	LSR / VECINT register
6	2−7	64→6E	Reserved	
7	0−7	70→7E	Reserved	
8	0	80	DMA1 interrupt	DMACSR/VECINT register
8	1	82	Reserved	—
8	2	84	DMA3 interrupt	DMACSR/VECINT register
8	3−7	86→7E	Reserved	—
9−15	0−7	90→FE	Reserved	—

5.6.9.2 Hardware Reset Introduced by the Firmware

This feature can be used during a firmware upgrade. Once the upgrade is complete, the application firmware disconnects from the USB for at least 200 ms to ensure the operating system has unloaded the device driver. The firmware then enables the watchdog timer (enabled by default after power-on reset) and enters an endless loop without resetting the watchdog timer. Once the watchdog timer times out, it resets the TUSB3410 device similar to a power on reset. The bootcode takes control and executes the power-on boot sequence.

5.6.10 File Listings

The TUSB3410 Bootcode Source Listing (SLLC139) is available on the *Tools & Software* tab of the TUSB3410 device product page on the TI website. The following files are included in the zip file.

- Types.h
- USB.h
- TUSB3410.h
- Bootcode.h
- Watchdog.h
- Bootcode.c
- Bootlsr.c
- BootUSB.c
- Header.h
- Header.c
- I2c.h
- I2c.c

6 Application, Implementation, and Layout

NOTE

Information in the following applications sections is not part of the TI component specification, and TI does not warrant its accuracy or completeness. TI's customers are responsible for determining suitability of components for their purposes. Customers should validate and test their design implementation to confirm system functionality.

6.1 Application Information

The implementation in Section 6.2 describes the minimum requirements to set up the TUSB3410 device for use as a basic USB to UART bridge to link the communication of a PC to any serial device through a USB port (see Figure 6-1).

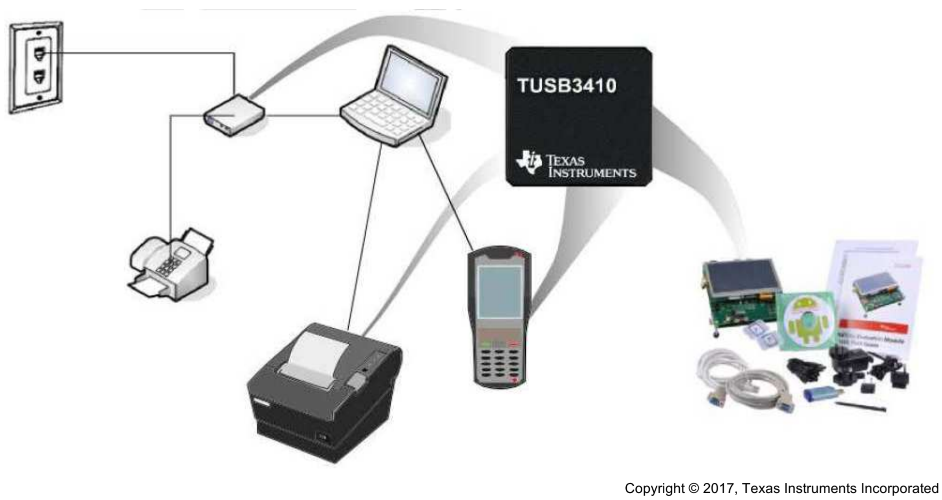


Figure 6-1. Typical Example for TUSB3410 as USB to UART Bridge

6.2 Typical Application

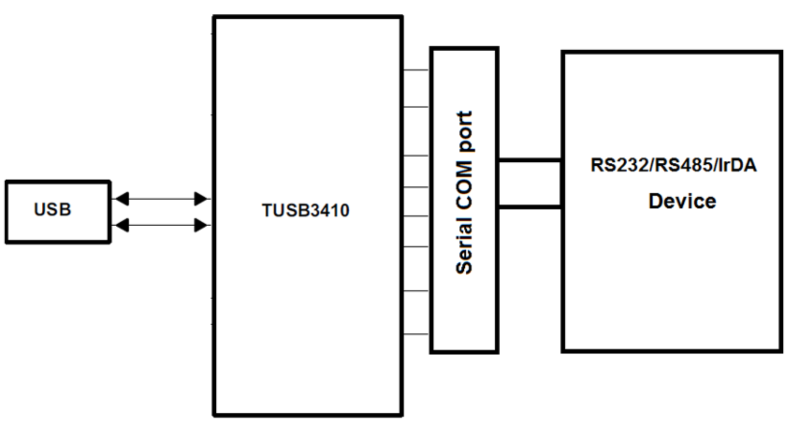


Figure 6-2. USB to UART Implementation

6.2.1 Design Requirements

Table 6-1 lists the design parameters for the typical application shown in Section 6.2.

Table 6-1. Design Parameters

DESIGN PARAMETER	VALUE
VCC Supply	3.3 V
VDD1/8	1.8 V
Upstream port USB (HS, FS)	HS, FS
RS-232 Transceivers	RS-232
XTAL	12 MHz

6.2.2 Detailed Design Procedure

6.2.2.1 Upstream Port Implementation

Figure 6-3 shows how the upstream of the TUSB3410 device is connected to a USB-2.0 Type B connector. The VBUS of the USB-2.0 connector is connected to a 3.3-V voltage regulator, which generates the 3.3 V required for VCC. The 3.3 V generated by this voltage regulator will pass through a voltage divider to generate the 1.8 V that is required for VDD.

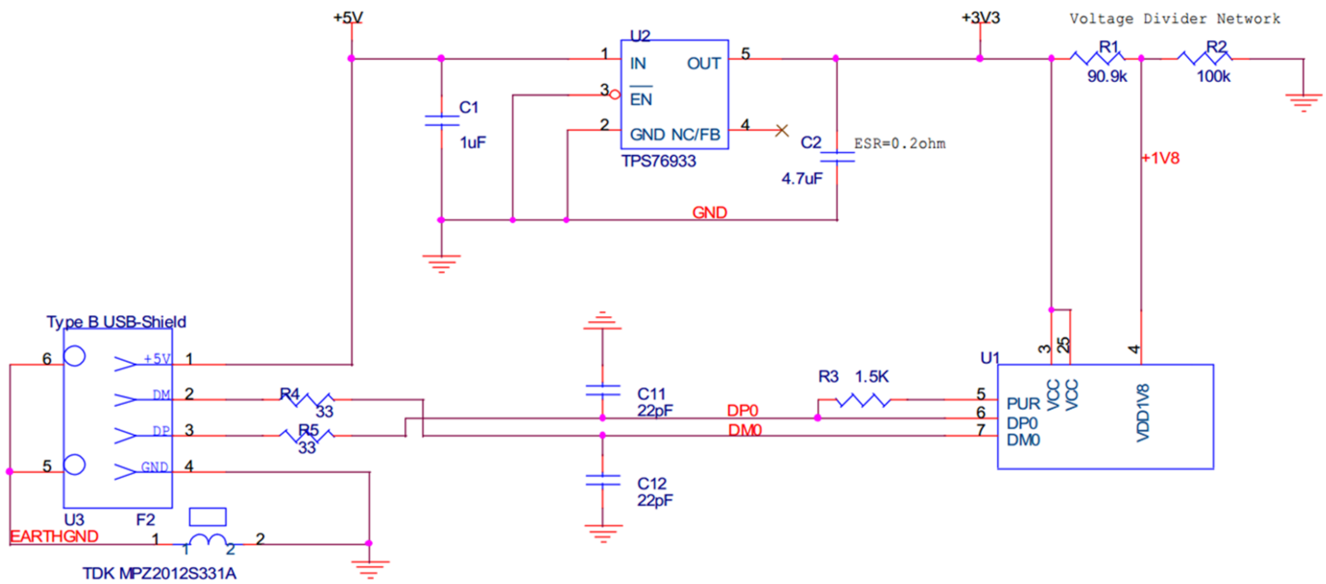


Figure 6-3. Upstream Port Implementation Schematic

6.2.2.2 Crystal Implementation

The TUSB3410 device requires a 12-MHz clock source to work properly, which is placed across the X1 and X2 terminals as shown in Figure 6-4.

TI recommends using a parallel-resonant crystal. Most parallel-resonant crystals are specified at a frequency with a load capacitance of 18 pF. This load can be realized by placing 33-pF capacitors from each end of the crystal to ground. Together with the input capacitance of the TUSB3410 device and stray board capacitance, this setup provides close to two 36-pF capacitors in series to emulate the 18-pF load requirement.

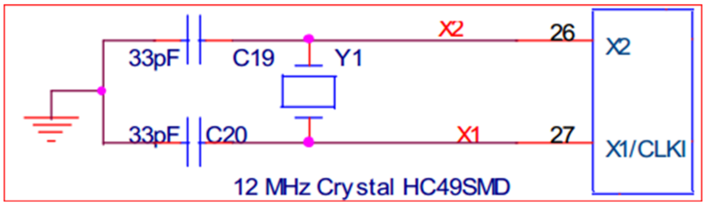


Figure 6-4. Crystal Implementation Schematic

6.2.2.3 RS-232 Implementation

All the serial data lines and serial control signals (DTR, RTS, SOUT/IR_SOUT, SIN/IR_SIN, RI/CP, DCD, DSR, and CTS) must go through an RS-232 driver (see Figure 6-5). For this example, the SN75LV4737A device is used (see SLLS178 for more details about the RS-232 driver). After the RS-232 driver is placed, the serial data lines and serial control signals are connected to a DB9 connector.

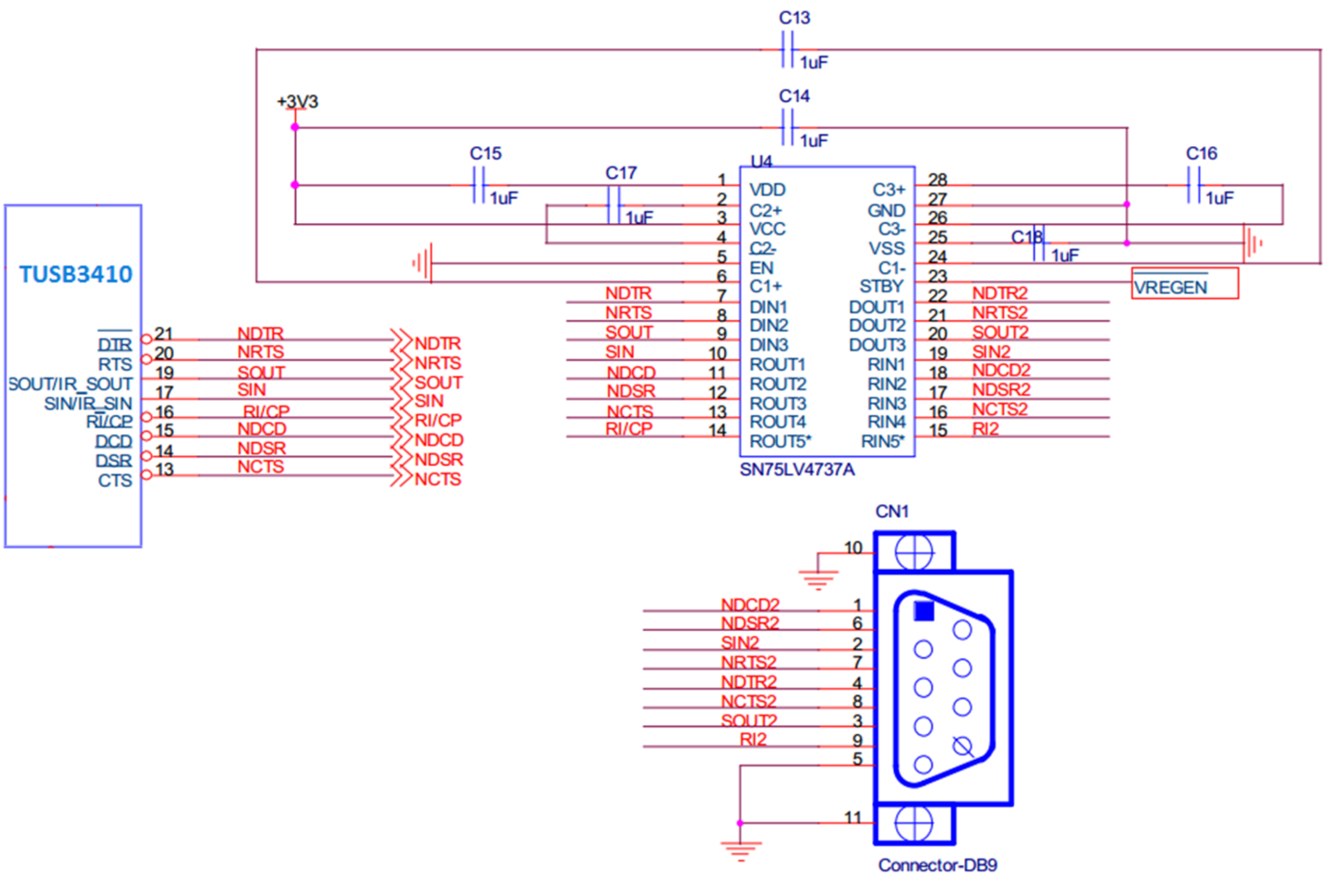


Figure 6-5. RS-232 Implementation Schematic

6.2.2.4 TUSB3410 Power Implementation

Figure 6-6 shows the power implementation for the TUSB3410 device.

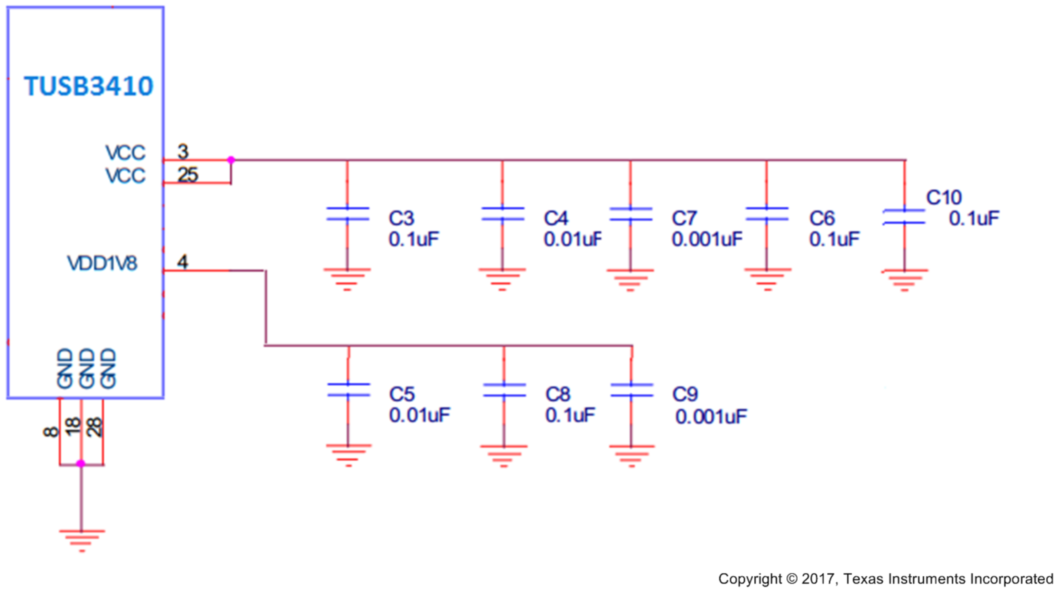


Figure 6-6. Power Implementation

6.2.3 Application Performance Plot

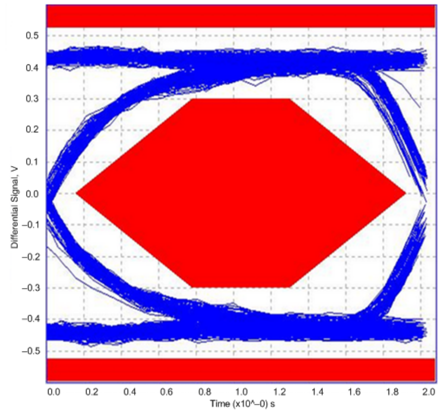


Figure 6-7. High-Speed Upstream Port

6.3 Layout

6.3.1 Layout Guidelines

A primary concern when designing a system is accommodating and isolating high-speed signals. As highspeed signals are most likely to impact or be impacted by other signals, they must be laid out early (preferably first) in the PCB design process to ensure that prescribed routing rules can be followed. Table 6-2 outlines the signals requiring the most attention in a USB layout.

Table 6-2. Critical Signals

SIGNAL NAME	DESCRIPTION
DP	USB 2.0 differential pair, positive
DM	USB 2.0 differential pair, negative
SSTXP	SuperSpeed differential pair, TX, positive
SSTXN	SuperSpeed differential pair, TX, negative
SSRXP	SuperSpeed differential pair, RX, positive
SSRXN	SuperSpeed differential pair, RX, negative

Use the following routing and placement guidelines when laying out a new design for the USB physical layer (PHY). These guidelines help minimize signal quality and electromagnetic interference (EMI) problems on a four-or-more layer evaluation module (EVM).

- Place the USB PHY and major components on the un-routed board first.
- Route the high-speed clock and high-speed USB differential signals with minimum trace lengths.
- Route the high-speed USB signals on the plane closest to the ground plane, whenever possible.
- Route the high-speed USB signals using a minimum of vias and corners. This reduces signal reflections and impedance changes.
- When it becomes necessary to turn 90°, use two 45° turns or an arc instead of making a single 90° turn. This reduces reflections on the signal traces by minimizing impedance discontinuities.
- Do not route USB traces under or near crystals, oscillators, clock signal generators, switching regulators, mounting holes, magnetic devices or ICs that use or duplicate clock signals.
- Avoid stubs on the high-speed USB signals because they cause signal reflections. If a stub is unavoidable, then the stub should be less than 200 mils.
- Route all high-speed USB signal traces over continuous planes (VCC or GND), with no interruptions. Avoid crossing over anti-etch, commonly found with plane splits.

6.3.2 Differential Signal Spacing

To minimize crosstalk in USB implementations, the spacing between the signal pairs must be a minimum of 5 times the width of the trace. This spacing is the 5W rule. Also, maintain a minimum keep-out area of 30 mils to any other signal throughout the length of the trace. Where the USB differential pair abuts a clock or a periodic signal, increase this keep-out to a minimum of 50 mils to ensure proper isolation. Figure 6-8 shows an example of USB2 differential signal spacing.

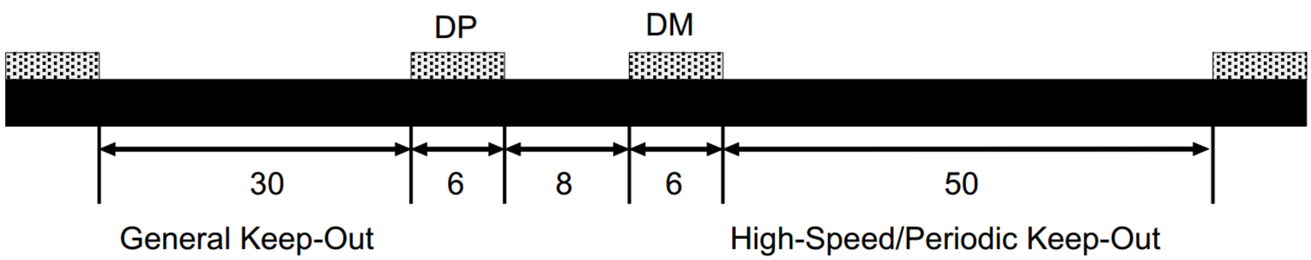


Figure 6-8. USB2 Differential Signal Spacing (mils)

6.3.3 Differential Signal Rules

- Do not place probe or test points on any USB differential signal.
- Do not route USB traces under or near crystals, oscillators, clock signal generators, switching power regulators, mounting holes, magnetic devices, or ICs that use or duplicate clock signals.
- After BGA breakout, keep USB differential signals clear of the SoC because high current transients produced during internal state transitions can be difficult to filter out.
- When possible, route the USB differential pair signals on the top or bottom layer of the PCB with an adjacent GND layer. TI does not recommend stripline routing of the USB differential signals.
- Ensure that USB differential signals are routed ≥ 90 mils from the edge of the reference plane.
- Ensure that USB differential signals are routed at least 1.5 W (calculated trace-width × 1.5) away from voids in the reference plane. This rule does not apply where SMD pads on the USB differential signals are voided.
- Maintain constant trace width after the SoC BGA escape to avoid impedance mismatches in the transmission lines.
- Maximize differential pair-to-pair spacing when possible.

For specific USB-2.0 layout guidelines, refer to *USB Layout Guidelines* (SPRAAR7).

6.3.4 Layout Example

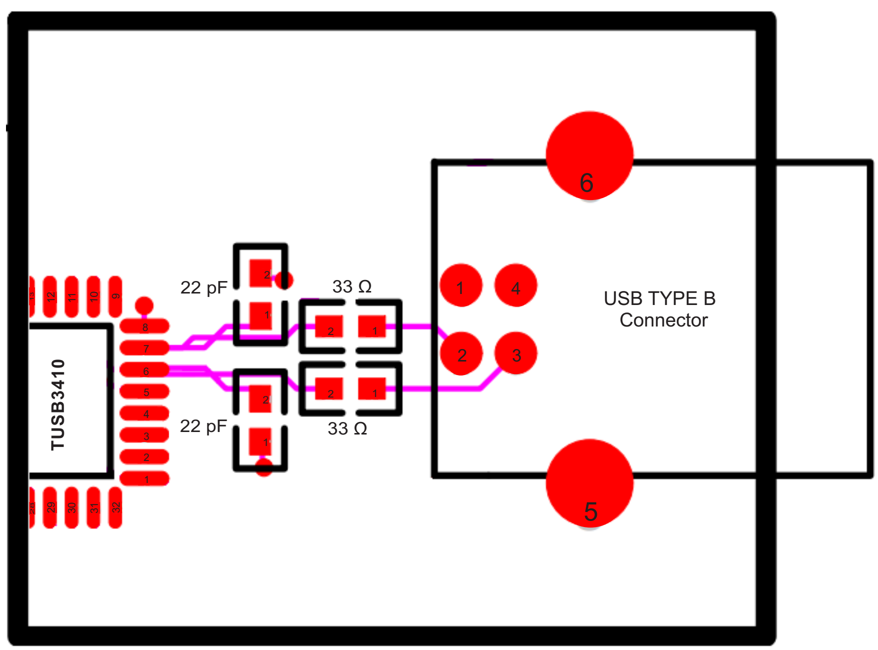


Figure 6-9. Layout Example for TUSB3410

6.4 Power Supply Recommendations

6.4.1 Digital Supplies 3.3 V

The TUSB3410 requires a 3.3-V digital power source.

The 3.3-V terminals are named VCC and supply power to most of the input and output cells. VCC supplies must have 0.1-µF bypass capacitors to VSS (ground) to ensure proper operation. One capacitor per power terminal is sufficient and should be placed as close to the terminal as possible to minimize trace length. TI also recommends smaller value capacitors like 0.01-µF on the digital supply terminals.

When placing and connecting all bypass capacitors, follow high-speed board design rules.

6.4.2 Digital Supplies 1.8 V

The TUSB3410 requires a 1.8-V digital power source.

The 3.3-V terminals are named VDD18 and supply power to most of the input and output cells. VDD18 supplies must have 0.1-µF bypass capacitors to VSS (ground) to ensure proper operation. One capacitor per power terminal is sufficient and should be placed as close to the terminal as possible to minimize trace length. TI also recommends smaller value capacitors like 0.01-µF on the digital supply terminals.

When placing and connecting all bypass capacitors, follow high-speed board design rules.

An internal voltage regulator generates this supply voltage when terminal VREGEN is low. When VREGEN is high, 1.8 V must be supplied externally.

6.5 Crystal Selection

The TUSB3410 device requires a 12-MHz clock source to work properly (see Figure 6-10). This clock source can be a crystal placed across the X1 and X2 terminals. A parallel resonant crystal is recommended. Most parallel resonant crystals are specified at a frequency with a load capacitance of 18 pF. This load can be realized by placing 33-pF capacitors from each end of the crystal to ground. Together with the input capacitance of the TUSB3410 device and stray board capacitance, this provides close to two 36-pF capacitors in series to emulate the 18-pF load requirement.

NOTE

When using a crystal, it takes about 2 ms after power up for a stable clock to be produced.

When using a clock oscillator, the signal applied to the X1/CLKI terminal must not exceed 1.8 V. In this configuration, the X2 terminal is unconnected.

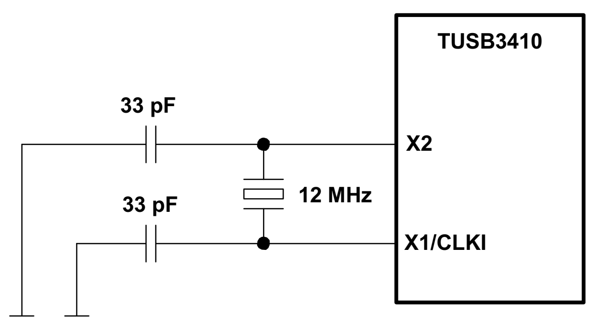


Figure 6-10. Crystal Selection

6.6 External Circuit Required for Reliable Bus Powered Suspend Operation

TI has found a potential problem with the action of the SUSPEND output terminal immediately after power on. In some cases the SUSPEND terminal can power up asserted high. When used in a bus powered application this can cause a problem because the $\overline{\text{VREGEN}}$ input is usually connected to the SUSPEND output. This in turn causes the internal 1.8-V voltage regulator to shut down, which means an external crystal may not have time to begin oscillating, thus the device will not initialize itself correctly.

TI has determined that using components R2 and D1 (rated to 25 mA) in the circuit shown in Figure 6-11 can be used as a workaround.

NOTE

R1 and C1 are required components for proper reset operation, unless the reset signal is provided by another means.

Use of an external oscillator (1.8-V output) versus a crystal would avoid this situation. Self-powered applications would probably not see this problem because the $\overline{\text{VREGEN}}$ input would likely be tied low, enabling the internal 1.8-V regulator at all times.

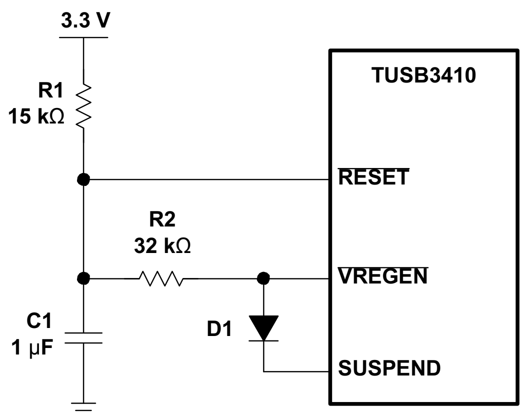


Figure 6-11. External Circuit

7 Device and Documentation Support

7.1 Documentation Support

7.1.1 Related Documentation

For related documentation, see the following:

SLLS178 *SN75LV4737A 3.3-V/5-V Multichannel RS-232 Line Driver/Receiver*

SPRAAR7 *USB Layout Guidelines*

7.2 Related Links

The table below lists quick access links. Categories include technical documents, support and community resources, tools and software, and quick access to sample or buy.

Table 7-1. Related Links

PARTS	PRODUCT FOLDER	ORDER NOW	TECHNICAL DOCUMENTS	TOOLS & SOFTWARE	SUPPORT & COMMUNITY
TUSB3410	Click here	Click here	Click here	Click here	Click here
TUSB3410I	Click here	Click here	Click here	Click here	Click here

7.3 Receiving Notification of Documentation Updates

To receive notification of documentation updates, navigate to the device product folder on ti.com. In the upper right corner, click on *Alert me* to register and receive a weekly digest of any product information that has changed. For change details, review the revision history included in any revised document.

7.4 Community Resources

The following links connect to TI community resources. Linked contents are provided "AS IS" by the respective contributors. They do not constitute TI specifications and do not necessarily reflect TI's views; see TI's Terms of Use.

TI E2E™ Online Community The TI engineer-to-engineer (E2E) community was created to foster collaboration among engineers. At e2e.ti.com, you can ask questions, share knowledge, explore ideas and help solve problems with fellow engineers.

Design Support ***TI's Design Support*** Quickly find helpful E2E forums along with design support tools and contact information for technical support.

7.5 Trademarks

E2E is a trademark of Texas Instruments.
All other trademarks are the property of their respective owners.

7.6 Electrostatic Discharge Caution

This integrated circuit can be damaged by ESD. Texas Instruments recommends that all integrated circuits be handled with appropriate precautions. Failure to observe proper handling and installation procedures can cause damage.

ESD damage can range from subtle performance degradation to complete device failure. Precision integrated circuits may be more susceptible to damage because very small parametric changes could cause the device not to meet its published specifications.

7.7 Glossary

TI Glossary This glossary lists and explains terms, acronyms, and definitions.

8 Mechanical Packaging and Orderable Information

8.1 Packaging Information

The following pages include mechanical packaging and orderable information. This information is the most current data available for the designated devices. This data is subject to change without notice and revision of this document. For browser-based versions of this data sheet, refer to the left-hand navigation.

PACKAGING INFORMATION

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/Ball Finish (6)	MSL Peak Temp (3)	Op Temp (°C)	Device Marking (4/5)	Samples
TUSB3410IRHB	ACTIVE	VQFN	RHB	32	73	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	-40 to 85	3410I	Samples
TUSB3410IRHBR	ACTIVE	VQFN	RHB	32	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	-40 to 85	3410I	Samples
TUSB3410IRHBRG4	ACTIVE	VQFN	RHB	32	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	-40 to 85	3410I	Samples
TUSB3410IRHBT	ACTIVE	VQFN	RHB	32	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	-40 to 85	3410I	Samples
TUSB3410IVF	ACTIVE	LQFP	VF	32	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-3-260C-168 HR	-40 to 85	TUSB3410I	Samples
TUSB3410IVFG4	ACTIVE	LQFP	VF	32	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-3-260C-168 HR	-40 to 85	TUSB3410I	Samples
TUSB3410RHB	ACTIVE	VQFN	RHB	32	73	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	0 to 70	3410	Samples
TUSB3410RHBG4	ACTIVE	VQFN	RHB	32	73	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	0 to 70	3410	Samples
TUSB3410RHBR	ACTIVE	VQFN	RHB	32	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	0 to 70	3410	Samples
TUSB3410RHBT	ACTIVE	VQFN	RHB	32	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	0 to 70	3410	Samples
TUSB3410VF	ACTIVE	LQFP	VF	32	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-3-260C-168 HR	0 to 70	TUSB3410	Samples
TUSB3410VFG4	ACTIVE	LQFP	VF	32	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-3-260C-168 HR	0 to 70	TUSB3410	Samples

(1) The marketing status values are defined as follows:
ACTIVE: Product device recommended for new designs.
LIFEBUY: TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.
NRND: Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.
PREVIEW: Device has been announced but is not in production. Samples may or may not be available.
OBSOLETE: TI has discontinued the production of the device.

(2) **RoHS:** TI defines "RoHS" to mean semiconductor products that are compliant with the current EU RoHS requirements for all 10 RoHS substances, including the requirement that RoHS substance do not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, "RoHS" products are suitable for use in specified lead-free processes. TI may reference these types of products as "Pb-Free".
RoHS Exempt: TI defines "RoHS Exempt" to mean products that contain lead but are compliant with EU RoHS pursuant to a specific EU RoHS exemption.

Green: TI defines "Green" to mean the content of Chlorine (Cl) and Bromine (Br) based flame retardants meet JS709B low halogen requirements of <=1000ppm threshold. Antimony trioxide based flame retardants must also meet the <=1000ppm threshold requirement.

(3) MSL, Peak Temp. - The Moisture Sensitivity Level rating according to the JEDEC industry standard classifications, and peak solder temperature.

(4) There may be additional marking, which relates to the logo, the lot trace code information, or the environmental category on the device.

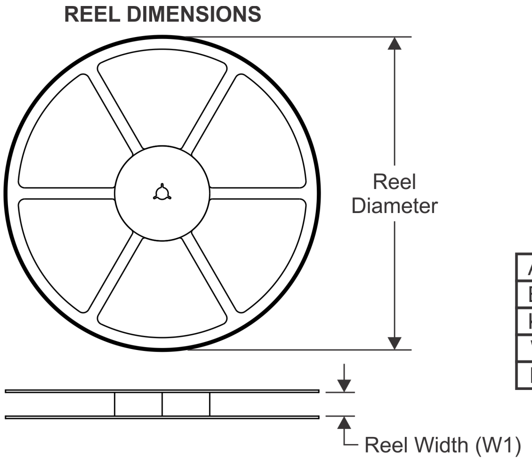
(5) Multiple Device Markings will be inside parentheses. Only one Device Marking contained in parentheses and separated by a "~" will appear on a device. If a line is indented then it is a continuation of the previous line and the two combined represent the entire Device Marking for that device.

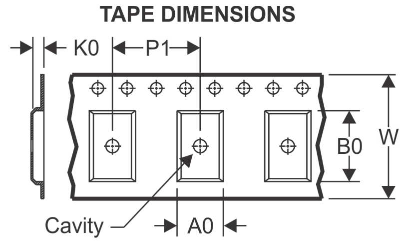
(6) Lead/Ball Finish - Orderable Devices may have multiple material finish options. Finish options are separated by a vertical ruled line. Lead/Ball Finish values may wrap to two lines if the finish value exceeds the maximum column width.

Important Information and Disclaimer: The information provided on this page represents TI's knowledge and belief as of the date that it is provided. TI bases its knowledge and belief on information provided by third parties, and makes no representation or warranty as to the accuracy of such information. Efforts are underway to better integrate information from third parties. TI has taken and continues to take reasonable steps to provide representative and accurate information but may not have conducted destructive testing or chemical analysis on incoming materials and chemicals. TI and TI suppliers consider certain information to be proprietary, and thus CAS numbers and other limited information may not be available for release.

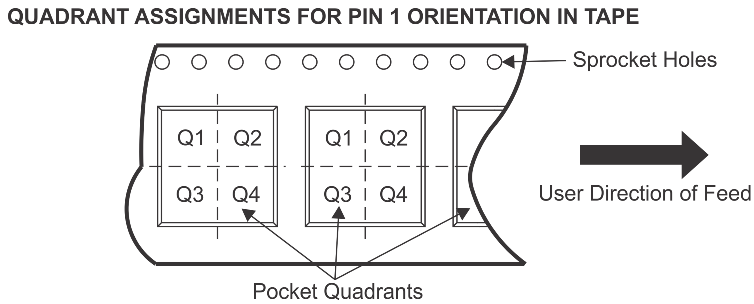
In no event shall TI's liability arising out of such information exceed the total purchase price of the TI part(s) at issue in this document sold by TI to Customer on an annual basis.

TAPE AND REEL INFORMATION





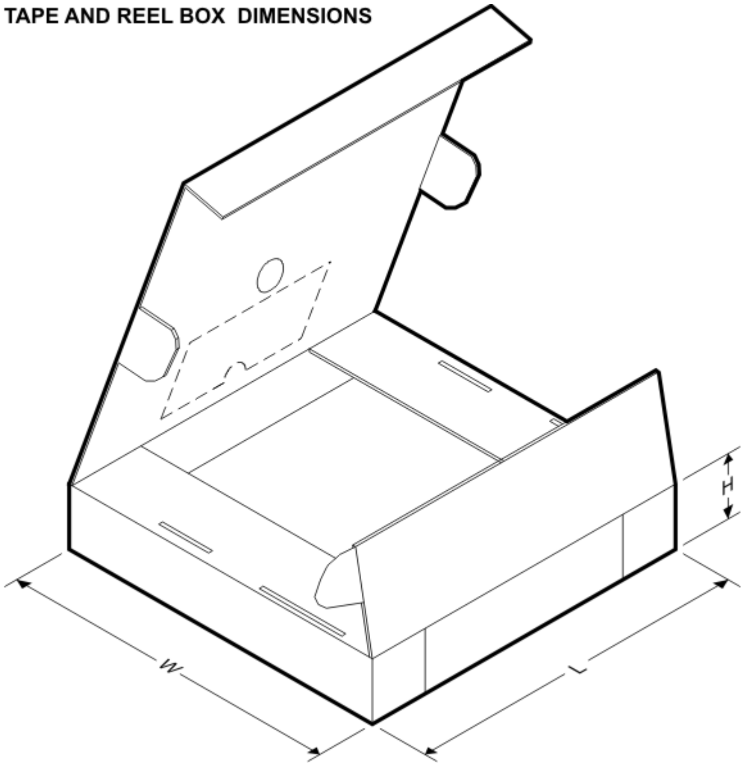
A0	Dimension designed to accommodate the component width
B0	Dimension designed to accommodate the component length
K0	Dimension designed to accommodate the component thickness
W	Overall width of the carrier tape
P1	Pitch between successive cavity centers



*All dimensions are nominal

Device	Package Type	Package Drawing	Pins	SPQ	Reel Diameter (mm)	Reel Width W1 (mm)	A0 (mm)	B0 (mm)	K0 (mm)	P1 (mm)	W (mm)	Pin1 Quadrant
TUSB3410IRHBR	VQFN	RHB	32	3000	330.0	12.4	5.3	5.3	1.5	8.0	12.0	Q2
TUSB3410IRHBT	VQFN	RHB	32	250	180.0	12.4	5.3	5.3	1.5	8.0	12.0	Q2
TUSB3410RHBR	VQFN	RHB	32	3000	330.0	12.4	5.3	5.3	1.5	8.0	12.0	Q2
TUSB3410RHBT	VQFN	RHB	32	250	180.0	12.4	5.3	5.3	1.5	8.0	12.0	Q2

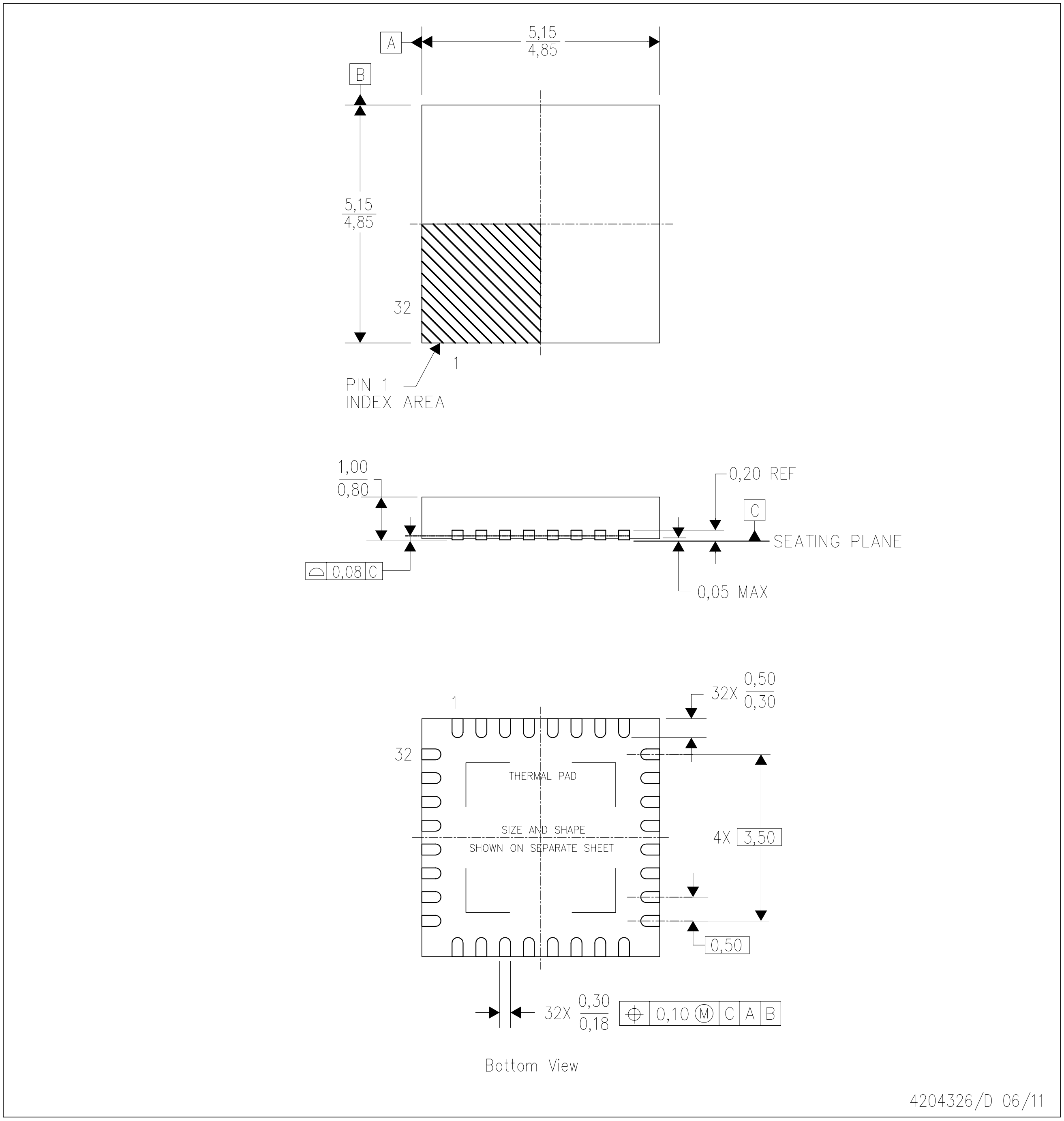
TAPE AND REEL BOX DIMENSIONS



*All dimensions are nominal

Device	Package Type	Package Drawing	Pins	SPQ	Length (mm)	Width (mm)	Height (mm)
TUSB3410IRHBR	VQFN	RHB	32	3000	336.6	336.6	28.6
TUSB3410IRHBT	VQFN	RHB	32	250	210.0	185.0	35.0
TUSB3410RHBR	VQFN	RHB	32	3000	336.6	336.6	28.6
TUSB3410RHBT	VQFN	RHB	32	250	210.0	185.0	35.0

RHB (S-PVQFN-N32) PLASTIC QUAD FLATPACK NO-LEAD



NOTES:
A. All linear dimensions are in millimeters. Dimensioning and tolerancing per ASME Y14.5M—1994.
B. This drawing is subject to change without notice.
C. QFN (Quad Flatpack No-Lead) Package configuration.
D. The package thermal pad must be soldered to the board for thermal and mechanical performance.
E. See the additional figure in the Product Data Sheet for details regarding the exposed thermal pad features and dimensions.
F. Falls within JEDEC MO-220.

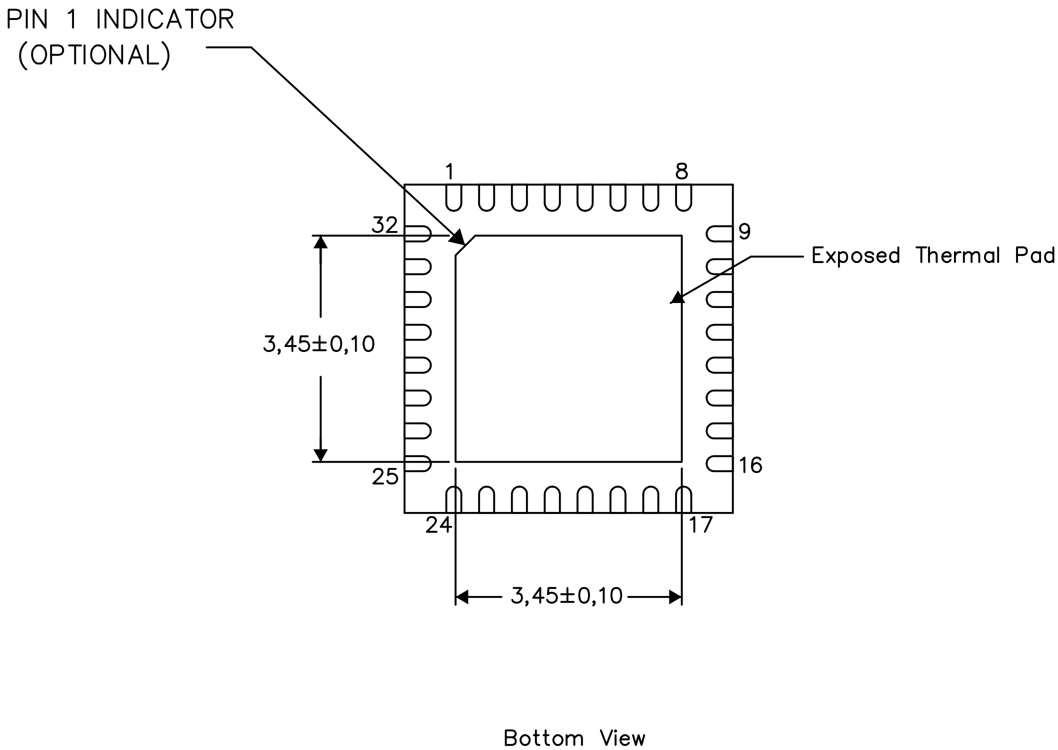
RHB (S-PVQFN-N32) PLASTIC QUAD FLATPACK NO-LEAD

THERMAL INFORMATION

This package incorporates an exposed thermal pad that is designed to be attached directly to an external heatsink. The thermal pad must be soldered directly to the printed circuit board (PCB). After soldering, the PCB can be used as a heatsink. In addition, through the use of thermal vias, the thermal pad can be attached directly to the appropriate copper plane shown in the electrical schematic for the device, or alternatively, can be attached to a special heatsink structure designed into the PCB. This design optimizes the heat transfer from the integrated circuit (IC).

For information on the Quad Flatpack No-Lead (QFN) package and its advantages, refer to Application Report, QFN/SON PCB Attachment, Texas Instruments Literature No. SLUA271. This document is available at www.ti.com.

The exposed thermal pad dimensions for this package are shown in the following illustration.

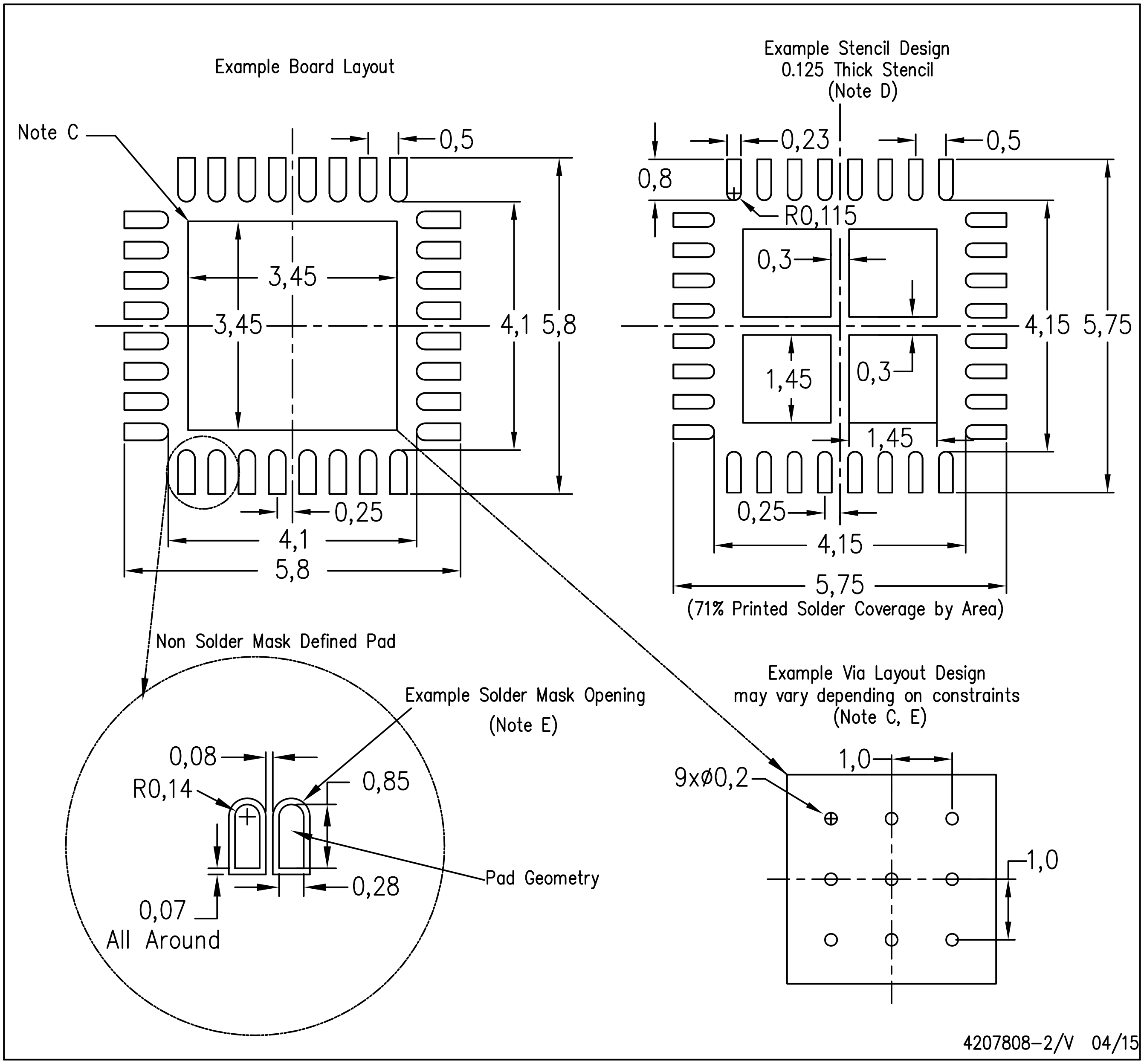


Exposed Thermal Pad Dimensions

4206356-2/AC 05/15

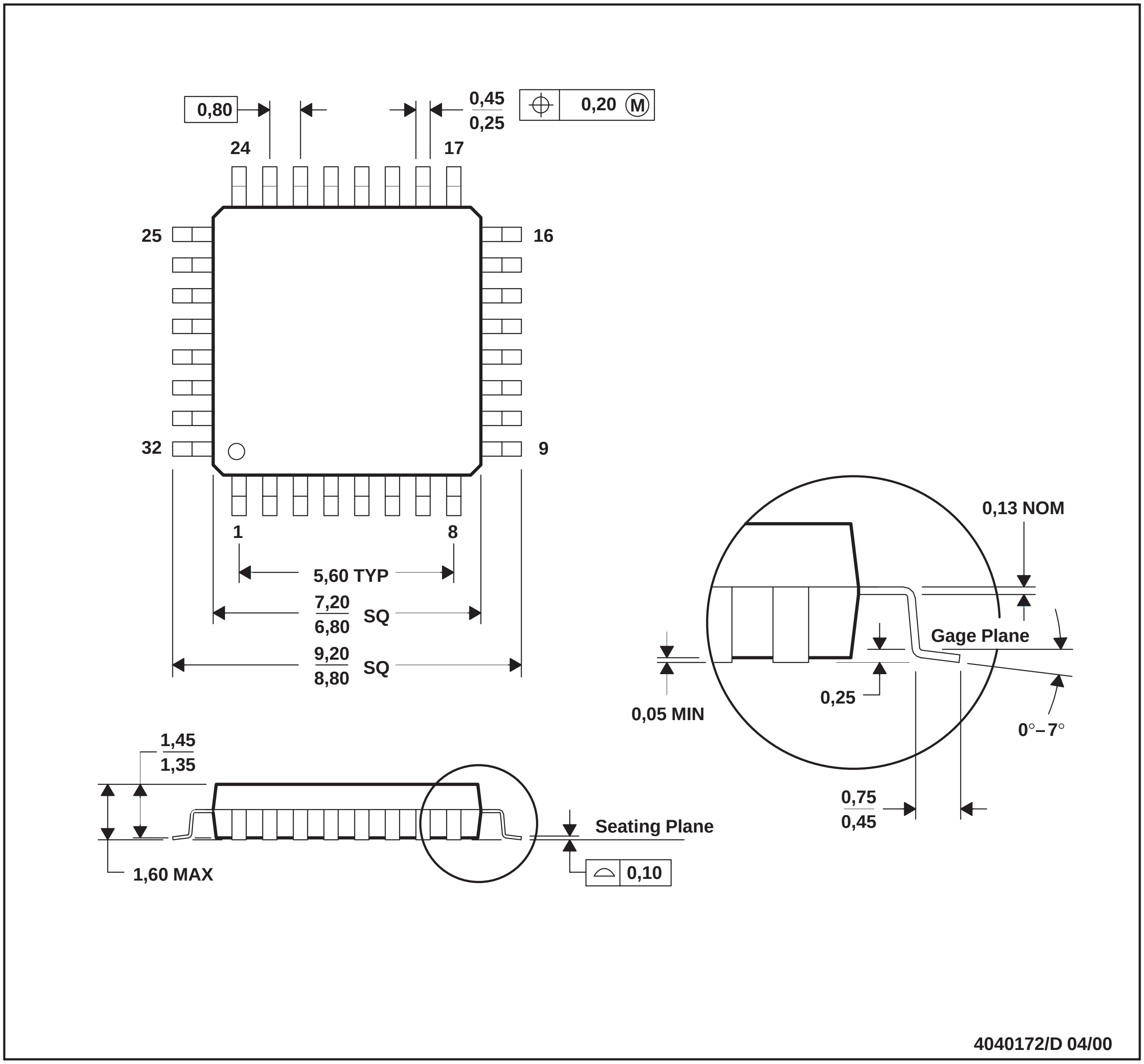
NOTE: A. All linear dimensions are in millimeters

RHB (S-PVQFN-N32) PLASTIC QUAD FLATPACK NO-LEAD



NOTES:
A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. This package is designed to be soldered to a thermal pad on the board. Refer to Application Note, Quad Flat-Pack Packages, Texas Instruments Literature No. SLUA271, and also the Product Data Sheets for specific thermal information, via requirements, and recommended board layout. These documents are available at www.ti.com <http://www.ti.com>.
D. Laser cutting apertures with trapezoidal walls and also rounding corners will offer better paste release. Customers should contact their board assembly site for stencil design recommendations. Refer to IPC 7525 for stencil design considerations.
E. Customers should contact their board fabrication site for recommended solder mask tolerances and via tenting recommendations for any larger diameter vias placed in the thermal pad.

VF (S-PQFP-G32) **PLASTIC QUAD FLATPACK**



NOTES: A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.

IMPORTANT NOTICE

Texas Instruments Incorporated (TI) reserves the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

TI’s published terms of sale for semiconductor products (http://www.ti.com/sc/docs/stdterms.htm) apply to the sale of packaged integrated circuit products that TI has qualified and released to market. Additional terms may apply to the use or sale of other types of TI products and services.

Reproduction of significant portions of TI information in TI data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such reproduced documentation. Information of third parties may be subject to additional restrictions. Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyers and others who are developing systems that incorporate TI products (collectively, “Designers”) understand and agree that Designers remain responsible for using their independent analysis, evaluation and judgment in designing their applications and that Designers have full and exclusive responsibility to assure the safety of Designers' applications and compliance of their applications (and of all TI products used in or for Designers’ applications) with all applicable regulations, laws and other applicable requirements. Designer represents that, with respect to their applications, Designer has all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. Designer agrees that prior to using or distributing any applications that include TI products, Designer will thoroughly test such applications and the functionality of such TI products as used in such applications.

TI’s provision of technical, application or other design advice, quality characterization, reliability data or other services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, “TI Resources”) are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using TI Resources in any way, Designer (individually or, if Designer is acting on behalf of a company, Designer’s company) agrees to use any particular TI Resource solely for this purpose and subject to the terms of this Notice.

TI’s provision of TI Resources does not expand or otherwise alter TI’s applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

Designer is authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED “AS IS” AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS. TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY DESIGNER AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Unless TI has explicitly designated an individual product as meeting the requirements of a particular industry standard (e.g., ISO/TS 16949 and ISO 26262), TI is not responsible for any failure to meet such industry standard requirements.

Where TI specifically promotes products as facilitating functional safety or as compliant with industry functional safety standards, such products are intended to help enable customers to design and create their own applications that meet applicable functional safety standards and requirements. Using products in an application does not by itself establish any safety features in the application. Designers must ensure compliance with safety-related requirements and standards applicable to their applications. Designer may not use any TI products in life-critical medical equipment unless authorized officers of the parties have executed a special contract specifically governing such use. Life-critical medical equipment is medical equipment where failure of such equipment would cause serious bodily injury or death (e.g., life support, pacemakers, defibrillators, heart pumps, neurostimulators, and implantables). Such equipment includes, without limitation, all medical devices identified by the U.S. Food and Drug Administration as Class III devices and equivalent classifications outside the U.S.

TI may expressly designate certain products as completing a particular qualification (e.g., Q100, Military Grade, or Enhanced Product). Designers agree that it has the necessary expertise to select the product with the appropriate qualification designation for their applications and that proper product selection is at Designers’ own risk. Designers are solely responsible for compliance with all legal and regulatory requirements in connection with such selection.

Designer will fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of Designer’s non-compliance with the terms and provisions of this Notice.

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