Multiprotocol Universal Asynchronous Receiver Transmitter (UART) Module

HIGHLIGHTS

This section of the manual contains the following major topics:

Note: This family reference manual section is meant to serve as a complement to device data sheets. This document applies to all dsPIC33/PIC24 devices.

Please consult the note at the beginning of the **"UART"** chapter in the current device data sheet to check whether this document supports the device you are using.

Device data sheets and family reference manual sections are available for download from the Microchip Worldwide Web site at: http://www.microchip.com.

1.0 INTRODUCTION

The Universal Asynchronous Receiver Transmitter (UART) is a flexible serial communication peripheral used to interface PIC® microcontrollers with other equipment, including computers and peripherals. The UART is a full-duplex, asynchronous communication channel that can be used to implement protocols, such as RS-232 and RS-485. The UART also supports the following hardware extensions:

- LIN/J2602
- IrDA®
- Direct Matrix Architecture (DMX)
- Smart Card

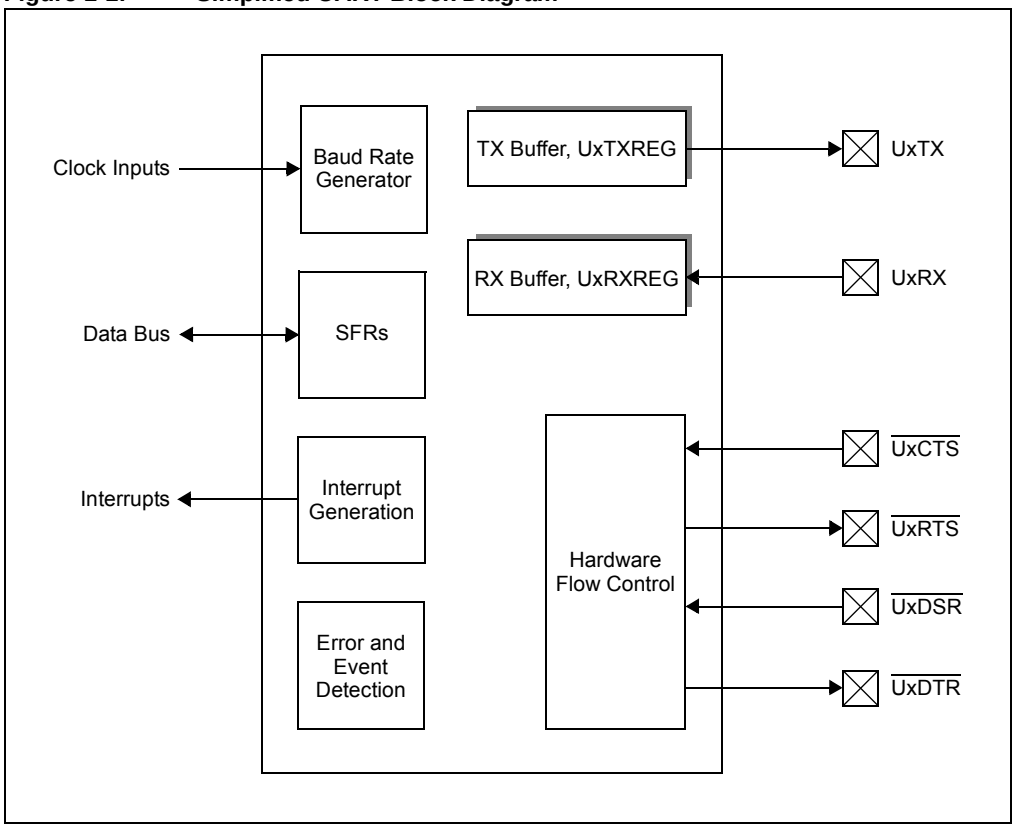
The primary features of the UART are:

- Full or Half-Duplex Operation
- Up to 8-Deep TX and RX First-In, First-Out (FIFO) Buffers
- 8-Bit Data Width
- Configurable Stop Bit Length
- Flow Control
- Auto-Baud Calibration
- Parity, Framing and Buffer Overrun Error Detection
- Address Detect
- Break Transmission
- Transmit and Receive Polarity Control
- Operation in Sleep mode
- Wake from Sleep on Sync Break Received Interrupt

1.1 Architectural Overview

The UART transfers bytes of data, to and from device pins, using First-In, First-Out (FIFO) buffers up to 8 bytes deep. The status of the buffers and data is made available to user software through Special Function Registers (SFRs). The UART implements multiple interrupt channels for handling transmit, receive and error events. A simplified block diagram of the UART is shown in Figure 1-1.

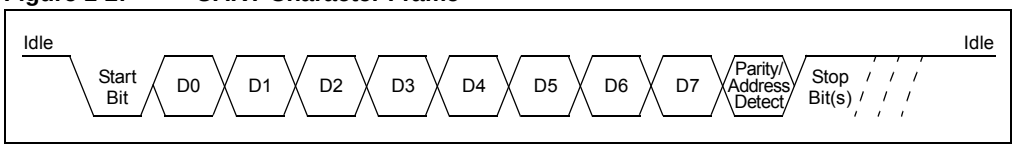
Figure 1-1: Simplified UART Block Diagram



1.2 Character Frame

A typical UART character frame is shown in Figure 1-2. The Idle state is high, with a 'Start' condition indicated by a falling edge. The Start bit is followed by the number of data, parity/address detect and Stop bits defined by the MOD<3:0> (UxMODE<3:0>) bits selected.

Figure 1-2: UART Character Frame



1.3 Data Buffers

Both transmit and receive functions use buffers to store data shifted to/from the pins. These buffers are FIFOs and are accessed by reading the SFRs, UxTXREG and UxRXREG, respectively. Each data buffer has multiple flags associated with its operation to allow software to read the status. Interrupts can also be configured based on the space available in the buffers (see **Section 9.0 "Interrupts"** for details). The Transmit and Receive FIFO Pointers and Counters can be reset using the associated UART TX/RX Buffer Empty Status bits, UTXBE (UxSTAH<5>) and URXBE (UxSTAH<1>).

1.4 Protocol Extensions

The UART provides hardware support for LIN/J2602, IrDA®, DMX and smart card protocol extensions to reduce software overhead. A protocol extension is enabled by writing a value to the MOD<3:0> (UxMODE<3:0>) selection bits and further configured using the UARTx Timing Parameter registers, UxP1 (Register 2-9), UxP2 (Register 2-10), UxP3 (Register 2-11) and UxP3H (Register 2-12). Details regarding operation and usage are discussed in their respective chapters. Not all protocols are available on all devices. Please refer to the specific device data sheet for availability.

2.0 CONTROL REGISTERS

Register 2-1: UxMODE: UARTx Configuration Register

R/W-0	U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W/HC-0(1)
UARTEN	—	USIDL	WAKE	RXBIMD	—	BRKOVR	UTXBRK
bit 15							bit 8

R/W-0	R/W/HC-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BRGH	ABAUD	UTXEN	URXEN	MOD3	MOD2	MOD1	MOD0
bit 7							bit 0

Legend:	HS = Hardware Settable bit	HC = Hardware Clearable bit	
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **UARTEN:** UART Enable bit
1 = UART is ready to transmit and receive
0 = UART state machine, FIFO Buffer Pointers and counters are reset; registers are readable and writable

bit 14 **Unimplemented:** Read as '0'

bit 13 **USIDL:** UART Stop in Idle Mode bit
1 = Discontinues module operation when device enters Idle mode
0 = Continues module operation in Idle mode

bit 12 **WAKE:** Wake-up Enable bit
1 = Module will continue to sample the UxRX pin – interrupt generated on falling edge, bit cleared in hardware on following rising edge; if ABAUD is set, Auto-Baud Detection (ABD) will begin immediately
0 = UxRX pin is not monitored nor rising edge detected

bit 11 **RXBIMD:** Receive Break Interrupt Mode bit
1 = RXBKIF flag when a minimum of 23 (DMX)/11 (asynchronous or LIN/J2602) low bit periods are detected
0 = RXBKIF flag when the Break makes a low-to-high transition after being low for at least 23/11 bit periods

bit 10 **Unimplemented:** Read as '0'

bit 9 **BRKOVR:** Send Break Software Override bit
Overrides the TX Data Line:
1 = Makes the TX line active (Output 0 when UTXINV = 0, Output 1 when UTXINV = 1)
0 = TX line is driven by the shifter

bit 8 **UTXBRK:** UART Transmit Break bit(1)
1 = Sends Sync Break on next transmission; cleared by hardware upon completion
0 = Sync Break transmission is disabled or has completed

bit 7 **BRGH:** High Baud Rate Select bit
1 = High Speed: Baud rate is baudclk/4
0 = Low Speed: Baud rate is baudclk/16

bit 6 **ABAUD:** Auto-Baud Detect Enable bit (read-only when MOD<3:0> = 1xxx)
1 = Enables baud rate measurement on the next character – requires reception of a Sync field (55h); cleared in hardware upon completion
0 = Baud rate measurement is disabled or has completed

Note 1: R/HS/HC in DMX and LIN mode.
2: These modes are not available on all devices. Refer to the specific device data sheet for availability.

Register 2-1: UxMODE: UARTx Configuration Register (Continued)

bit 5 **UTXEN:** UART Transmit Enable bit
1 = Transmit enabled – except during Auto-Baud Detection
0 = Transmit disabled – all transmit counters, pointers and state machines are reset; TX buffer is not flushed, status bits are not reset

bit 4 **URXEN:** UART Receive Enable bit
1 = Receive enabled – except during Auto-Baud Detection
0 = Receive disabled – all receive counters, pointers and state machines are reset; RX buffer is not flushed, status bits are not reset

bit 3-0 **MOD<3:0>:** UART Mode bits
Other = Reserved
1111 = Smart card(2)
1110 = IrDA®(2)
1101 = Reserved
1100 = LIN Master/Slave
1011 = LIN Slave only
1010 = DMX(2)
1001 = Reserved
1000 = Reserved
0111 = Reserved
0110 = Reserved
0101 = Reserved
0100 = Asynchronous 9-bit UART with address detect, ninth bit = 1 signals address
0011 = Asynchronous 8-bit UART without address detect, ninth bit is used as an even parity bit
0010 = Asynchronous 8-bit UART without address detect, ninth bit is used as an odd parity bit
0001 = Asynchronous 7-bit UART
0000 = Asynchronous 8-bit UART

Note 1: R/HS/HC in DMX and LIN mode.
2: These modes are not available on all devices. Refer to the specific device data sheet for availability.

Register 2-2: UxMODEH: UARTx Configuration Register High

R/W-0	R-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
SLPEN	ACTIVE	—	—	BCLKMOD	BCLKSEL1	BCLKSEL0	HALFDPLX
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RUNOVF	URXINV	STSEL1	STSEL0	C0EN	UTXINV	FLO1	FLO0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **SLPEN:** Run During Sleep Enable bit
1 = UART BRG clock runs during Sleep
0 = UART BRG clock is turned off during Sleep

bit 14 **ACTIVE:** UART Running Status bit
1 = UART clock request is active (user cannot update the UxMODE/UxMODEH registers)
0 = UART clock request is not active (user can update the UxMODE/UxMODEH registers)

bit 13-12 **Unimplemented:** Read as '0'

bit 11 **BCLKMOD:** Baud Clock Generation Mode Select bit
1 = Uses fractional Baud Rate Generation
0 = Uses legacy divide-by-x counter for baud clock generation (x = 4 or 16 depending on the BRGH bit)

bit 10-9 **BCLKSEL<1:0>:** Baud Clock Source Selection bits
Clock sources are device-dependent. Refer to the specific device data sheet for selections.

bit 8 **HALFDPLX:** UART Half-Duplex Selection Mode bit
1 = Half-Duplex mode: UxTX is driven high when transmitting and low when UxTX is Idle
0 = Full-Duplex mode: UxTX is driven high at all times when both UARTEN and UTXEN are set

bit 7 **RUNOVF:** Run During Overflow Condition Mode bit
1 = When an Overflow Error (OERR) condition is detected, the RX shifter continues to run so as to remain synchronized with incoming RX data; data is not transferred to UxRXREG when it is full (i.e., no UxRXREG data is overwritten)
0 = When an Overflow Error (OERR) condition is detected, the RX shifter stops accepting new data (Legacy mode)

bit 6 **URXINV:** UART Receive Polarity bit
1 = Inverts RX polarity; Idle state is low
0 = Input is not inverted; Idle state is high

bit 5-4 **STSEL<1:0>:** Number of Stop Bits Selection bits
11 = 2 Stop bits sent, 1 checked at receive
10 = 2 Stop bits sent, 2 checked at receive
01 = 1.5 Stop bits sent, 1.5 checked at receive
00 = 1 Stop bit sent, 1 checked at receive

bit 3 **C0EN:** Enable Legacy Checksum (C0) Transmit and Receive bit
1 = Checksum Mode 1 (enhanced LIN checksum in LIN mode; add all TX/RX words in all other modes)
0 = Checksum Mode 0 (legacy LIN checksum in LIN mode; not used in all other modes)

bit 2 **UTXINV:** UART Transmit Polarity bit
1 = Inverts TX polarity; TX is low in Idle state
0 = Output data is not inverted; TX output is high in Idle state

bit 1-0 **FLO<1:0>:** Flow Control Enable bits (only valid when MOD<3:0> = 0xxx)
11 = Reserved
10 = $\overline{\text{U}}$xRTS-$\overline{\text{U}}$xDSR (for TX side)/$\overline{\text{U}}$xCTS-$\overline{\text{U}}$xDTR (for RX side) hardware flow control
01 = XON/XOFF software flow control
00 = Flow control off

Register 2-3: UxSTA: UARTx Status Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TXMTIE	PERIE	ABDOVE	CERIE	FERIE	RXBKIE	OERIE	TXCIE
bit 15							bit 8

R-1	R-0	R/W/HS-0	R/W/HC-0	R-0	R/W/HC-0	R/W/HC-0	R/W/HC-0
TRMT	PERR	ABDOVF	CERIF	FERR	RXBKIF	OERR	TXCIF
bit 7							bit 0

Legend:	HS = Hardware Settable bit	HC = Hardware Clearable bit	
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15 **TXMTIE:** Transmit Shifter Empty Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 14 **PERIE:** Parity Error Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 13 **ABDOVE:** Auto-Baud Rate Acquisition Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 12 **CERIE:** Checksum Error Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 11 **FERIE:** Framing Error Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 10 **RXBKIE:** Receive Break Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 9 **OERIE:** Receive Buffer Overflow Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 8 **TXCIE:** Transmit Collision Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 7 **TRMT:** Transmit Shifter Empty Interrupt Flag bit
1 = Transmit Shift Register (TSR) is empty (end of last Stop bit when STPMD = 1 or middle of first Stop bit when STPMD = 0)
0 = Transmit Shift Register is not empty

bit 6 **PERR:** Parity Error/Address Received/Forward Frame Interrupt Flag bit
LIN and Parity Modes:
1 = Parity error detected
0 = No parity error detected
Address Mode:
1 = Address received
0 = No address detected
All Other Modes:
Not used.

Register 2-3: UxSTA: UARTx Status Register (Continued)

bit 5 **ABDOVF:** Auto-Baud Rate Acquisition Interrupt Flag bit (must be cleared by software)
1 = BRG rolled over during the auto-baud rate acquisition sequence (must be cleared in software)
0 = BRG has not rolled over during the auto-baud rate acquisition sequence

bit 4 **CERIF:** Checksum Error Interrupt Flag bit (must be cleared by software)
1 = Checksum error
0 = No checksum error

bit 3 **FERR:** Framing Error Interrupt Flag bit
1 = Framing Error: Inverted level of the Stop bit corresponding to the topmost character in the buffer; propagates through the buffer with the received character
0 = No framing error

bit 2 **RXBKIF:** Receive Break Interrupt Flag bit (must be cleared by software)
1 = A Break was received
0 = No Break was detected

bit 1 **OERR:** Receive Buffer Overflow Interrupt Flag bit (must be cleared by software)
1 = Receive buffer has overflowed
0 = Receive buffer has not overflowed

bit 0 **TXCIF:** Transmit Collision Interrupt Flag bit (must be cleared by software)
1 = Transmitted word is not equal to the received word
0 = Transmitted word is equal to the received word

Register 2-4: UxSTAH: UARTx Status High Register

U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	UTXISEL2	UTXISEL1	UTXISEL0	—	URXISEL2	URXISEL1	URXISEL0
bit 15							bit 8

R/W/HS-0	R/W-0	R/S-1	R-0	R-1	R-1	R/S-1	R-0
TXWRE	STPMD	UTXBE	UTXBF	RIDLE	XON	URXBE	URXBF
bit 7							bit 0

Legend:	S = Settable bit	HS = Hardware Settable bit	
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **Unimplemented:** Read as '0'

bit 14-12 **UTXISEL<2:0>:** UART Transmit Interrupt Select bits

111 = Sets transmit interrupt when there is 1 empty slot left in the buffer

•

•

•

010 = Sets transmit interrupt when there are 6 empty slots or more in the buffer

001 = Sets transmit interrupt when there are 7 empty slots or more in the buffer

000 = Sets transmit interrupt when there are 8 empty slots in the buffer; TX buffer is empty

bit 11 **Unimplemented:** Read as '0'

bit 10-8 **URXISEL<2:0>:** UART Receive Interrupt Select bits

111 = Triggers receive interrupt when there are 8 words in the buffer; RX buffer is full

•

•

•

001 = Triggers receive interrupt when there are 2 words or more in the buffer

000 = Triggers receive interrupt when there is 1 word or more in the buffer

bit 7 **TXWRE:** TX Write Transmit Error Status bit

LIN and Parity Modes:

1 = A new byte was written when the buffer was full or when P2<8:0> = 0 (must be cleared by software)

0 = No error

Address Detect Mode:

1 = A new byte was written when the buffer was full or to P1<8:0> when P1x was full (must be cleared by software)

0 = No error

Other Modes:

1 = A new byte was written when the buffer was full (must be cleared by software)

0 = No error

bit 6 **STPMD:** Stop Bit Detection Mode bit

1 = Triggers RXIF at the end of the last Stop bit

0 = Triggers RXIF in the middle of the first (or second, depending on the STSEL<1:0> setting) Stop bit

bit 5 **UTXBE:** UART TX Buffer Empty Status bit

1 = Transmit buffer is empty; writing '1' when UTXEN = 0 will reset the TX FIFO Pointers and counters

0 = Transmit buffer is not empty

bit 4 **UTXBF:** UART TX Buffer Full Status bit

1 = Transmit buffer is full

0 = Transmit buffer is not full

Register 2-4: UxSTAH: UARTx Status High Register (Continued)

bit 3 **RIDLE:** Receive Idle bit
1 = UART RX line is in the Idle state
0 = UART RX line is receiving something

bit 2 **XON:** UART in XON Mode bit
Only valid when FLO<1:0> control bits are set to XON/XOFF mode.
1 = UART has received XON
0 = UART has not received XON or XOFF was received

bit 1 **URXBE:** UART RX Buffer Empty Status bit
1 = Receive buffer is empty; writing '1' when URXEN = 0 will reset the RX FIFO Pointers and Counters
0 = Receive buffer is not empty

bit 0 **URXBF:** UART RX Buffer Full Status bit
1 = Receive buffer is full
0 = Receive buffer is not full

Register 2-5: UxBRG: UARTx Baud Rate Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BRG<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BRG<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15 **BRG<15:0>:** Baud Rate Divisor bits

Register 2-6: UxBRGH: UARTx Baud Rate High Register

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	BRG<19:16>			
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15-4 **Unimplemented:** Read as ‘0’

bit 3-0 **BRG<19:16>:** Baud Rate Divisor bits

Register 2-7: UxRXREG: UARTx Receive Buffer Register

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
RXREG<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15-8 **Unimplemented:** Read as ‘0’

bit 7-0 **RXREG<7:0>:** Received Character Data bits 7-0

Register 2-8: UxTXREG: UARTx Transmit Buffer Register

W-x	U-0	U-0	U-0	U-0	U-0	U-0	U-0
LAST	—	—	—	—	—	—	—
bit 15							bit 8

W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x
TXREG<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15 **LAST:** Last Byte Indicator for Smart Card Support bit

bit 14-8 **Unimplemented:** Read as ‘0’

bit 7-0 **TXREG<7:0>:** Transmitted Character Data bits 7-0
If the buffer is full, further writes to the buffer are ignored.

Register 2-9: UxP1: UARTx Timing Parameter 1 Register

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	P1<8>
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P1<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-9 **Unimplemented:** Read as '0'

bit 8-0 **P1<8:0>:** Parameter 1 bits

DMX TX:
Number of Bytes to Transmit – 1 (not including start code).

LIN Master TX:
PID to transmit (bits<5:0>).

Asynchronous TX with Address Detect:
ADDR to transmit. A '1' is automatically inserted into bit 9 (bits<7:0>).

Smart Card Mode:
Guard Time Counter (GTC) bits. This counter is operated on the bit clock whose period is always equal to one ETU (bits<8:0>).

Other Modes:
Not used.

Register 2-10: UxP2: UARTx Timing Parameter 2 Register

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	P2<8>
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P2<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-9 **Unimplemented:** Read as '0'

bit 8-0 **P2<8:0>:** Parameter 2 bits

DMX RX:
The First Byte Number to Receive – 1, not including start code (bits<8:0>).

LIN Slave TX:
Number of bytes to transmit (bits<7:0>).

Asynchronous RX with Address Detect:
ADDR to start matching (bits<7:0>).

Smart Card Mode:
Block Time Counter (BTC) bits. This counter is operated on the bit clock whose period is always equal to one ETU (bits<8:0>).

Other Modes:
Not used.

Register 2-11: UxP3: UARTx Timing Parameter 3 Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P3<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P3<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15-0 **P3<15:0>:** Parameter 3 bits

DMX RX:
The Last Byte Number to Receive – 1, not including start code (bits<8:0>).

LIN Slave RX:
Number of bytes to receive (bits<7:0>).

Asynchronous RX:
Used to mask the UxP2 address bits; 1 = P2 address bit is used, 0 = P2 address bit is masked off (bits<7:0>).

Smart Card Mode:
Waiting Time Counter (WTC) bits (bits<15:0>).

Other Modes:
Not used.

Register 2-12: UxP3H: UARTx Timing Parameter 3 Register High

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P3<23:16>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15-8 **Unimplemented:** Read as ‘0’

bit 7-0 **P3<23:16>:** Parameter 3 High bits

Smart Card Mode:
Waiting Time Counter (WTC) bits (bits<23:16>).

Other Modes:
Not used.

Register 2-13: UxTXCHK: UARTx Transmit Checksum Register

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TXCHK<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-8 **Unimplemented:** Read as '0'

bit 7-0 **TXCHK<7:0>:** Transmit Checksum bits (calculated from TX words)

LIN Modes:
C0EN = 1: Sum of all transmitted data + addition carries, including PID.
C0EN = 0: Sum of all transmitted data + addition carries, excluding PID.

LIN Slave:
Cleared when Break is detected.

LIN Master/Slave:
Cleared when Break is detected.

Other Modes:
C0EN = 1: Sum of every byte transmitted + addition carries.
C0EN = 0: Value remains unchanged.

Register 2-14: UxRXCHK: UARTx Receive Checksum Register

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RXCHK<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-8 **Unimplemented:** Read as '0'

bit 7-0 **RXCHK<7:0>:** Receive Checksum bits (calculated from RX words)

LIN Modes:
C0EN = 1: Sum of all received data + addition carries, including PID.
C0EN = 0: Sum of all received data + addition carries, excluding PID.

LIN Slave:
Cleared when Break is detected.

LIN Master/Slave:
Cleared when Break is detected.

Other Modes:
C0EN = 1: Sum of every byte received + addition carries.
C0EN = 0: Value remains unchanged.

Register 2-15: UxSCCON: UARTx Smart Card Configuration High Register

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
—	—	TXRPT1	TXRPT0	CONV	T0PD	PRTCL	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15-6 **Unimplemented:** Read as ‘0’

bit 5-4 **TXRPT<1:0>:** Transmit Repeat Selection bits
11 = Retransmits the error byte four times
10 = Retransmits the error byte three times
01 = Retransmits the error byte twice
00 = Retransmits the error byte once

bit 3 **CONV:** Logic Convention Selection bit
1 = Inverse logic convention
0 = Direct logic convention

bit 2 **T0PD:** Pull-Down Duration for T = 0 Error Handling bit
1 = 2 ETU
0 = 1 ETU

bit 1 **PRTCL:** Smart Card Protocol Selection bit
1 = T = 1
0 = T = 0

bit 0 **Unimplemented:** Read as ‘0’

Register 2-16: UxSCINT: UARTx Smart Card Interrupt Register

U-0	U-0	R/W/HS-0	R/W/HS-0	U-0	R/W/HS-0	R/W/HS-0	R/W/HS-0
—	—	RXRPTIF	TXRPTIF	—	BTCIF	WTCIF	GTCIF
bit 15							bit 8

U-0	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	—	RXRPTIE	TXRPTIE	—	BTCIE	WTCIE	GTCIE
bit 7							bit 0

Legend:	HS = Hardware Settable bit		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-14 **Unimplemented:** Read as '0'

bit 13 **RXRPTIF:** Receive Repeat Interrupt Flag bit
1 = Parity error has persisted after the same character has been received five times (four retransmits)
0 = Flag is cleared

bit 12 **TXRPTIF:** Transmit Repeat Interrupt Flag bit
1 = Line error has been detected after the last retransmit per TXRPT<1:0>
0 = Flag is cleared

bit 11 **Unimplemented:** Read as '0'

bit 10 **BTCIF:** Block Time Counter Interrupt Flag bit
1 = Block Time Counter has reached 0
0 = Block Time Counter has not reached 0

bit 9 **WTCIF:** Waiting Time Counter Interrupt Flag bit
1 = Waiting Time Counter has reached 0
0 = Waiting Time Counter has not reached 0

bit 8 **GTCIF:** Guard Time Counter Interrupt Flag bit
1 = Guard Time Counter has reached 0
0 = Guard Time Counter has not reached 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5 **RXRPTIE:** Receive Repeat Interrupt Enable bit
1 = An interrupt is invoked when a parity error has persisted after the same character has been received five times (four retransmits)
0 = Interrupt is disabled

bit 4 **TXRPTIE:** Transmit Repeat Interrupt Enable bit
1 = An interrupt is invoked when a line error is detected after the last retransmit per TXRPT<1:0> has been completed
0 = Interrupt is disabled

bit 3 **Unimplemented:** Read as '0'

bit 2 **BTCIE:** Block Time Counter Interrupt Enable bit
1 = Block Time Counter interrupt is enabled
0 = Block Time Counter interrupt is disabled

bit 1 **WTCIE:** Waiting Time Counter Interrupt Enable bit
1 = Waiting Time Counter interrupt is enabled
0 = Waiting Time Counter Interrupt is disabled

bit 0 **GTCIE:** Guard Time Counter interrupt enable bit
1 = Guard Time Counter interrupt is enabled
0 = Guard Time Counter interrupt is disabled

Register 2-17: UxINT: UARTx Interrupt Register

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

R/W/HS-0	R/W/HS-0	U-0	U-0	U-0	R/W-0	U-0	U-0
WUIF	ABDIF	—	—	—	ABDIE	—	—
bit 7							bit 0

Legend:	HS = Hardware Settable bit		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15-8 **Unimplemented:** Read as ‘0’

bit 7 **WUIF:** Wake-up Interrupt Flag bit
- 1 = Sets when WAKE = 1 and RX makes a 1-to-0 transition; triggers event interrupt (must be cleared by software)
- 0 = WAKE is not enabled or WAKE is enabled, but no wake-up event has occurred

bit 6 **ABDIF:** Auto-Baud Completed Interrupt Flag bit
- 1 = Sets when ABD sequence makes the final 1-to-0 transition; triggers event interrupt (must be cleared by software)
- 0 = ABAUD is not enabled or ABAUD is enabled but auto-baud has not completed

bit 5-3 **Unimplemented:** Read as ‘0’

bit 2 **ABDIE:** Auto-Baud Completed Interrupt Enable Flag bit
- 1 = Allows ABDIF to set an event interrupt
- 0 = ABDIF does not set an event interrupt

bit 1-0 **Unimplemented:** Read as ‘0’

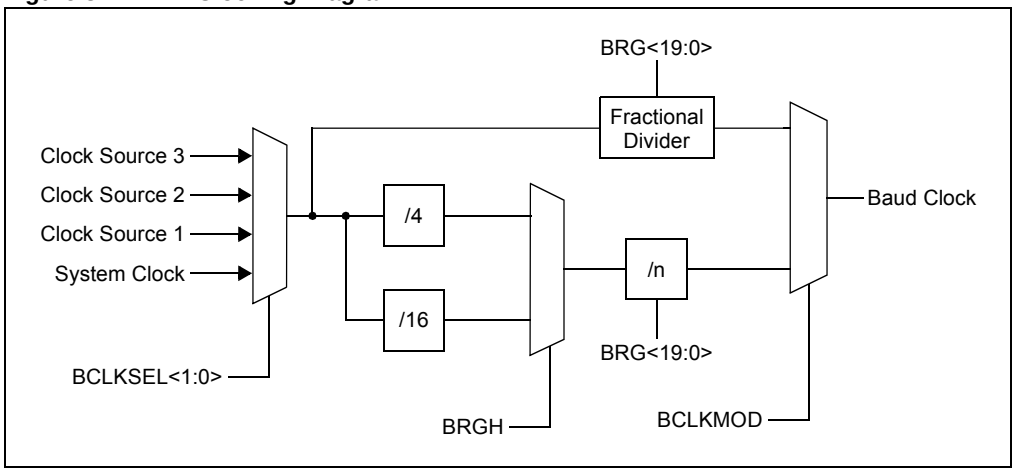
3.0 CLOCKING AND BAUD RATE CONFIGURATION

The UART supports multiple clock sources and two types of Baud Rate Generation (BRG). One of up to 4 clock sources is selected by the BCLKSEL<1:0> bits (UxMODEH<10:9>). Clock source selection and prescaler can only be changed when the UARTEN bit (UxMODE<15>) is cleared, effectively holding the peripheral in Reset. The baud clock can be generated with one of the following methods:

- Legacy mode, fixed division
- Fractional Division mode

To allow synchronized time of clock domains, do not make back-to-back writes to the UxBRG or UxBRGH registers. UxBRG should be written only when UARTEN = 0 to avoid corruption. A block diagram of the clocking is shown in Figure 3-1.

Figure 3-1: Clocking Diagram



3.1 Legacy Mode

In Legacy mode, the clock source is divided down to the desired baud clock using integer division. Legacy mode is selected when BCLKMOD = 0 (UxMODEH<11>). A selectable prescaler is present to support a wide range of baud rates and is controlled by the BRGH bit (UxMODE<7>). Up to a 20-bit value of BRG (UxBRG<15:0> and UxBRGH<3:0>) is used to further divide down the input clock to the final baud rate.

Equation 3-1 and Equation 3-2 show formulas for baud rate, and UxBRG value given by the BRGH for all protocol modes.

Equation 3-1: Baud Rate When BRGH = 0

$$Baud\ Rate = \frac{FP}{16 \bullet (UxBRG + 1)}$$

$$UxBRG = \frac{FP}{16 \bullet Baud\ Rate} - 1$$

Equation 3-2: Baud Rate When BRGH = 1

$$Baud\ Rate = \frac{FP}{4 \bullet (UxBRG + 1)}$$

$$UxBRG = \frac{FP}{4 \bullet Baud\ Rate} - 1$$

Note: UxBRG values should be 3 or more.

UART fixed division baud rate setup procedure:

1. Select the clock input source with the BCLKSEL<1:0> bits.
2. Select the clock prescaler by writing a value to BRGH.
3. Using Equation 3-1 or Equation 3-2, calculate the value for BRG and write to the UxBRG and UxBRGH registers (if upper 4 bits are needed).
4. Set the UARTEN bit.

3.2 Fractional Divider

To reduce baud rate error, a fractional division scheme can be used by setting BCLKMOD = 1. The fractional baud clock circuit works by occasionally extending clock pulses to achieve a baud clock closer to the ideal baud rate. This mode is most useful at higher speeds, where in Legacy mode, a small value of BRG results in unacceptable error. BRGH has no affect in this mode. Equation 3-3 shows the baud rate formulas.

Equation 3-3: Baud Rate Formulas

$$Baud\ Rate = \frac{FP}{BRG}$$

$$BRG = \frac{FP}{Baud\ Rate}$$

UART fractional baud rate setup procedure:

1. Select the clock input source with the BCLKSEL<1:0> bits.
2. Set the BCLKMOD bit.
3. Using Equation 3-3, calculate the value for BRG and write to the UxBRG and UxBRGH registers (if upper 4 bits are needed).
4. Set the UARTEN bit.

3.3 Auto-Baud Feature

The auto-baud feature allows the receiver to determine the baud rate of the transmitter and synchronize to it. The transmitter sends a byte value of 0x55 (Sync byte) to the receiver and the receiver calculates the average bit time from the falling edges. The UxBRG register is then written with the corresponding value. Auto-baud is supported in both Legacy and Fractional Baud Rate Generation modes (BCLKMOD = 1 or 0). The Sync byte may be preceded with a Break.

To enable auto-baud, the ABAUD bit (UxMODE<6>) is set and the UART will begin to look for a falling edge (Start bit of Sync byte). Once the auto-baud process is complete, the ABAUD bit will be cleared by hardware and the ABDIF flag (UxINT<6>) is set. If the ABDIE (UxINT<2>) interrupt enable bit is set, an event interrupt will be generated.

If the 5th and final falling edge is not detected before the BRG counter rolls over, the ABDOVF flag (UxSTA<5>) will set to indicate the condition. The flag cannot be cleared until ABAUD is cleared. If the ABDOVE bit (UxSTA<13>) is set, an event interrupt will be generated. For more information on interrupts, see **Section 9.0 “Interrupts”**.

Auto-baud setup procedure:

1. Configure the UART for receive operation as detailed in **Section 4.2 “Asynchronous Receive”**.
2. Set the ABAUD bit. If a Break precedes the Sync byte, also set the WAKE (UxMODE<12>) bit to configure the UART to perform the auto-baud procedure on the Sync and not the Break. The RXBKIF flag (UxSTA<2>) will not be set.
3. Poll the ABAUD or ABDIF bit to determine when auto-baud has finished.

Alternatively, if a Break precedes the Sync and it is desired to detect the Break, use the following sequence:

1. Configure the UART for receive operation as detailed in **Section 4.2 “Asynchronous Receive”**.
2. Wait for the RXBKIF flag to set (see **Section 4.4.2 “Break Character Reception”** for details).
3. Immediately set the ABAUD bit.
4. Poll the ABAUD or ABDIF bit to determine when auto-baud has finished.

4.0 ASYNCHRONOUS MODE

Asynchronous mode supports standard UART communication with the following configurable options:

- 7, 8-Bit Data Width
- 1, 1.5 or 2 Stop Bits
- Even, Odd or No Parity (9th data bit)
- Independently Selectable TX and RX Polarity
- Address Detect (9th data bit)
- Auto-Baud
- Break Transmission/Detection
- Flow Control (XON/XOFF and HW)
- Half/Full-Duplex TX Pin Control
- TX and RX Interrupt Configuration

The MOD<3:0> bits (UxMODE<3:0>) are used to select the high-level operational mode of the UART for both transmit and receive. The five Asynchronous mode selections configure data width, parity and address detect, with the rest of the configuration options as spate controls.

4.1 Asynchronous Transmit

The following procedure is used to transmit a byte of data:

1. Configure the clock input and baud rate as detailed in **Section 3.0 “Clocking and Baud Rate Configuration”**.
2. Configure the data width and parity by writing a selection to the MOD<3:0> bits.
3. Configure the polarity, Stop bit duration and flow control.
4. Configure the TX interrupt watermark using the UTXISEL<2:0> bits (UxSTAH<14:12>).
5. Configure the address detect if needed as detailed in **Section 4.5 “Address Detect”**.
6. Set the UARTEN bit (UxMODE<15>).
7. Set the UTXEN bit (UxMODE<5>).
8. Write the data byte value to the UxTXREG register.

A TX interrupt will be generated according to the UTXISEL<2:0> bits’ interrupt watermark setting. The UTXISELx bits can be configured to generate a TX interrupt when the buffer has 1 to 8 empty slots.

The UARTx Transmit Buffer (UxTXREG) has two associated flags to indicate its contents. The TX Buffer Empty Status bit, UTXBE (UxSTAH<5>), indicates that the buffer is empty, and the TX Buffer Full Status bit, UTXBF (UxSTAH<4>), indicates that there are no empty slots in the buffer and it should not be written.

4.1.1 TRANSMIT ERRORS AND EVENTS

The UART is capable of detecting bus collisions. The received byte is compared against the last byte transmitted to identify differences. The UxTX and UxRX pin functions need to be mapped to separate pins using Peripheral Pin Select (PPS). The UxRX pin has to be able to receive a byte in order for the comparison to happen. If the pin is stuck at VDD or ground, such that a valid Start and Stop bit are not detected, the comparison cannot take place. If a bus collision is detected, it is flagged by the TXCIF bit (UxSTA<0>). If the TXCIE bit (UxSTA<8>) is set, an error interrupt will be generated.

To be able to detect a stuck UxRX pin, the UART has both TX and RX timer Reset functions that may be mapped to Capture/Compare/PWM (CCP) timers. See the device-specific data sheet for availability of this feature. Software can then be used to detect a time-out after a byte has been transmitted without a reception.

If a write to UxTXREG is done when the buffer is already full, a transmit write error is indicated by the TXWRE bit (UxSTAH<7>).

The Transmit Shift Register (TSR) has a status flag, TRMT (UxSTA<7>), associated with it to indicate when a byte transmission is complete. The timing of the flag is configurable using the STPMD bit (UxSTAH<6>). By default, it will set the TRMT flag at the middle of the first Stop bit, and when STPMD = 1, the TRMT flag will set at the end of the last Stop bit. An interrupt can be generated by setting the TXMTIE bit (UxSTA<15>).

4.1.2 HALF-DUPLEX TRANSMIT

In a half-duplex application, the UxTX and UxRX lines are shorted together; this allows a reduction in wire count. However, control is needed to avoid both devices transmitting at the same time. Setting the HALFDPLX bit (UxMODEH<8>) configures the UxTX pin to only drive the line during a byte transmission and is tri-stated at all other times. The RIDLE bit (UxSTAH<3>) can be read to determine if the line is Idle and a byte can be sent. However, a collision can still occur during the transmission. The Transmit Collision Interrupt Flag bit, TXCIF (UxSTA<0>), can be read to determine if the byte was transmitted successfully.

4.2 Asynchronous Receive

The following procedure is used to receive a byte of data:

1. Configure the clock input and baud rate as detailed in **Section 3.0 “Clocking and Baud Rate Configuration”**.
2. Configure the data width and parity and by writing a selection to the MOD<3:0> bits.
3. Configure the polarity, Stop bit duration and flow control.
4. Configure the RX interrupt watermark using the URXISEL<2:0> bits (UxSTAH<10:8>).
5. Configure the address detect if needed as detailed in **Section 4.5 “Address Detect”**.
6. Set the UARTEN bit (UxMODE<15>).
7. Set the URXEN bit (UxMODE<4>).

An RX interrupt will be generated when a byte is received according to the UART Receive Interrupt Select bits setting, URXISEL<2:0> (UxSTAH<10:8>). The URXISELx bits can be configured to generate an RX interrupt when the RX buffer contains 1-8 bytes.

Software can then read the data from the UxRXREG register. The time, relative to the Stop bit when the RX interrupt is generated, is configurable using the STPMD bit (UxSTAH<6>). By default, an RX interrupt is generated in the middle of the Stop bit. Writing a ‘1’ will move the RX interrupt to the end of the Stop bit.

The URXBF status bit (UxSTAH<0>) can be read by software to determine if the receive buffer is full and a read operation of UxRXREG is required to allow reception of additional bytes. Similarly, the URXBE status bit (UxSTAH<1>) can be read with software to determine if the receive buffer is empty.

4.2.1 RECEIVE ERRORS AND EVENTS

The receive framing and parity errors are associated with each byte received. Frame and parity error flags, indicated by the FERR and PERR bits, will indicate only the error status of the last data byte received. As UxRXREG is read, the flags indicate the status of the current (top) byte in the buffer. This behavior differs from prior UART modules.

If a byte is received when the receive buffer is full, the OERR bit (UxSTA<1>) will set. Setting the corresponding error interrupt enable will generate an error interrupt. The receiver can handle overflow conditions in one of two options, defined by the RUNOVF (UxMODEH<7>) bit. By default, when RUNOVF = 0, the receiver will stop receiving data when the RX buffer is full. Alternatively, when RUNOVF = 1, the receiver will continue to receive data and overwrite the contents of the RX shifter.

A line Idle condition (line high) is indicated by the RIDLE bit (UxSTAH<3>). The flag will clear when a Start bit is detected and a reception is in progress.

4.3 Parity Support

Parity is a simple method of single bit error detection. The data bits are summed and compared to the parity value indicating a bit error. Parity selection can either be even or odd and is represented by the 9th data bit. To calculate parity, the number of data bits that are a '1' are counted.

- Even parity is defined as an odd number of data bits whose values are '1'
- Odd parity is defined as an even number of data bits whose values are '1'

The parity bit itself is then added to the count, hence the 'even' or 'odd' designation. Parity calculation and checking is enabled by selecting one of the two 8-Bit Asynchronous Parity modes (MOD<3:0> = 0b010x).

4.4 Break Character

A Break character is defined as several consecutive low bit times, usually longer than a whole byte. In Asynchronous mode, the UART will transmit a 13-bit long duration Break, and in receive, will flag 11 low bit times as a Break sequence.

4.4.1 TRANSMITTING A BREAK CHARACTER

A Break character is transmitted by setting the UTXBRK bit (UxMODE<8>) and then writing any value to UxTXREG. The contents of UxTXREG will follow the Break character.

Alternatively, the BRKOVR bit (UxMODE<9>) can be controlled by software to override and drive the UxTX line for any duration. When UTXINV (UxMODEH<2>) = 0, the UxTX line will be driven low and when UTXINV = 1, the UxTX line will be driven high.

4.4.2 BREAK CHARACTER RECEPTION

The receiver is always looking for a Break sequence and can detect one, even in the middle of a byte reception. A Break reception is indicated by the RXBKIF flag (UxSTA<2>). An interrupt can be optionally generated by setting the RXBKIE bit (UxSTA<10>). The Break detection criteria can be configured using the RXBIMD bit (UxMODE<11>). By default, the RXBKIF flag will set when the line makes a low-to-high transition after 11 low bit times, signaling the end of the Break sequence. Alternatively, when RXBIMD = 1, the flag will set when the 11th low bit time is detected.

4.5 Address Detect

Address Detect mode is used when multiple receivers are connected to a transmitter. It allows a receiver to determine if the message is intended for it and to ignore those that are not. If the 9th data bit is a '1', the data is recognized as an address to be processed by the receiver. An address mask is provided to allow multiple receivers to accept the same address.

If an address match is successful, the unmasked address is present in UxRXREG and a RX interrupt is generated. If the address match is not successful, all of the following data is ignored until a byte with the 9th bit set is received.

In 8-Bit Address Detect mode, the transmitted address IDs are written to Parameter 1. For receivers, the expected address is written to Parameter 2 (UxP2<7:0>) and the mask value to Parameter 3 (UxP3<7:0>). Mask bit values of '1' will include the respective bit position in the compare, whereas a '0' indicates a 'don't care'. A mask value of 0x00 will accept all address values, effectively disabling the address detect feature. A mask value of 0xFF will allow only one matching value.

4.5.1 ADDRESS DETECT TRANSMIT

The following procedure is used to transmit in Address Detect mode:

1. Configure the UART for asynchronous transmit as detailed in **Section 4.1 "Asynchronous Transmit"** with the MOD<3:0> bits set to '0b0100'.
2. If a Break is desired, write a '1' to UTXBRK (UxMODE<8>).
3. Write the address value to Parameter 1.
   a) If UTXBRK = 0, the contents of Parameter 1 will be transmitted with the 9th bit set.
   b) If UTXBRK = 1, a Break will be transmitted, followed by the contents of Parameter 1 with the 9th bit set.
4. Write data to be transmitted to the UxTXREG register.

4.5.2 ADDRESS DETECT RECEIVE

The following procedure is used for receive in Address Detect mode. A framing error will not prevent an address match.

1. Configure the UART for asynchronous transmit as detailed in **Section 4.1 "Asynchronous Transmit"** with the MOD<3:0> bits set to '0b0100'.
2. Write the address match value to Parameter 2.
3. Write the optional address mask value to Parameter 3.
4. Upon the reception of a valid address, the PERR bit will be set to indicate that the value stored in UxRXREG is an address. The subsequent data can be read from UxRXREG as it becomes available.

4.6 Flow Control

Flow control is used to prevent data loss between two devices. One device may be slower or have to process data. Flow control allows a device to tell the other to wait before sending additional bytes that may overrun its buffers. The UART supports two types of flow control:

- XON/OFF Messaging
- Hardware Flow Control ($\overline{\text{UxRTS}}$, $\overline{\text{UxCTS}}$, $\overline{\text{UxDTR}}$, $\overline{\text{UxDSR}}$)

4.6.1 XON/XOFF

XON/XOFF uses messages implemented as special byte values and does not require additional HW lines. An XON command is implemented by sending a byte value of 0x11 and an XOFF is implemented by a value of 0x13. There are two states of the control mechanism as indicated by the XON bit (UxSTAH<2>). By default, the UART is in the XON = 1 state and will transmit data as it become available in the TX buffer. If the device receives an XOFF command, the XON status bit is cleared and transmission stops until another XON command is received. XON/OFF commands are transmitted in the same manner as regular data.

A receive application would poll the RX Buffer Full Status (URXBF) flag, and if set, send an XOFF command to halt further data reception. When the receive buffer is no longer full (or empty), an XON command can be sent to resume reception.

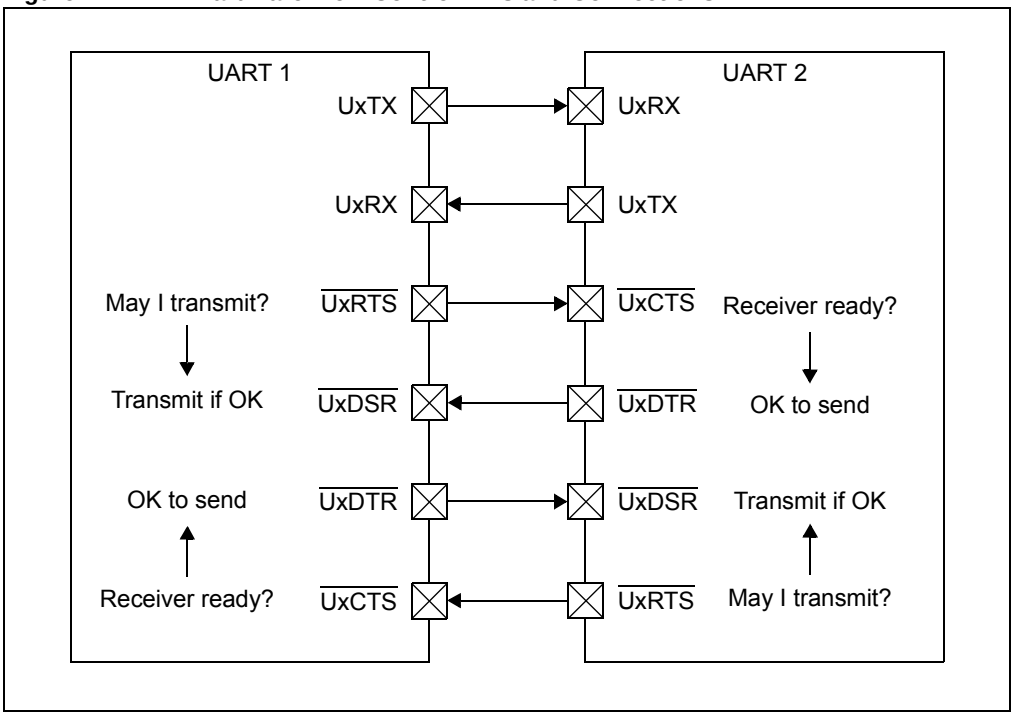
4.6.2 HARDWARE FLOW CONTROL

Hardware flow control uses up to 4 additional pins to indicate when a device is ready to receive additional data. The four active-low device pins are shown in Table 4-1. Figure 4-1 shows connections between 2 UARTs.

Table 4-1: Hardware Flow Control Pin Functions

Signal Name	Description	Used By	Direction
$\overline{\text{UxDSR}}$	Data-Set-Ready	Transmitter	Input
$\overline{\text{UxRTS}}$	Request-to-Send	Transmitter	Output
$\overline{\text{UxCTS}}$	Clear-to-Send	Receiver	Input
$\overline{\text{UxDTR}}$	Data-Terminal-Ready	Receiver	Output

Figure 4-1: Hardware Flow Control Pins and Connections



The transmitter asserts (drives low) the $\overline{\text{UxRTS}}$ output when it has one or more bytes in its TX buffer to indicate that it wants to send a byte. Then, the transmitter listens to the $\overline{\text{UxDSR}}$ to see if it is OK to do so. If $\overline{\text{UxDSR}}$ is active (low), the transmitter sends one byte. If $\overline{\text{UxDSR}}$ is inactive (high), it will wait.

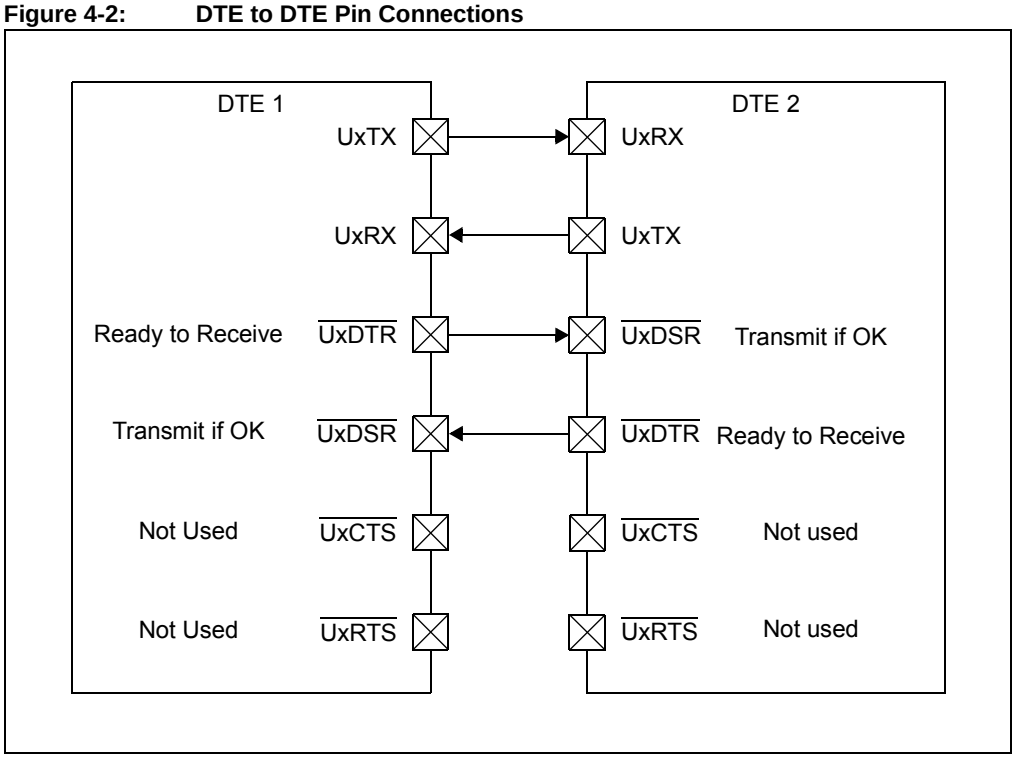
When the receiver detects the $\overline{\text{UxCTS}}$ signal going active (low), it checks to see if there are 2 empty slots in the receive buffer. If so, the receiver asserts (drives low) the $\overline{\text{UxDTR}}$ pin to indicate it is ready to receive data. No register setup is needed to enable the flow control pins. However, most devices will have the UART and associated flow control pins routed through the Peripheral Pin Select (PPS) feature.

When using flow control, devices can be categorized into two groups: DTE (Data Terminal Equipment) and DCE (Data Carrier Equipment). A typical DTE can be a computer or a microcontroller and a DCE is typically a modem. Not all hardware flow control pins are needed in all cases. The following sections show which flow control pins are used to interface different devices to one another.

4.6.2.1 DTE to DTE Configuration

When interfacing two DTE devices together, connect them as shown in Figure 4-2. The $\overline{\text{UxDTR}}$ output is connected to the $\overline{\text{UxDSR}}$ input terminal of the other. This allows the receiver to tell the transmitter that it is OK to transmit.

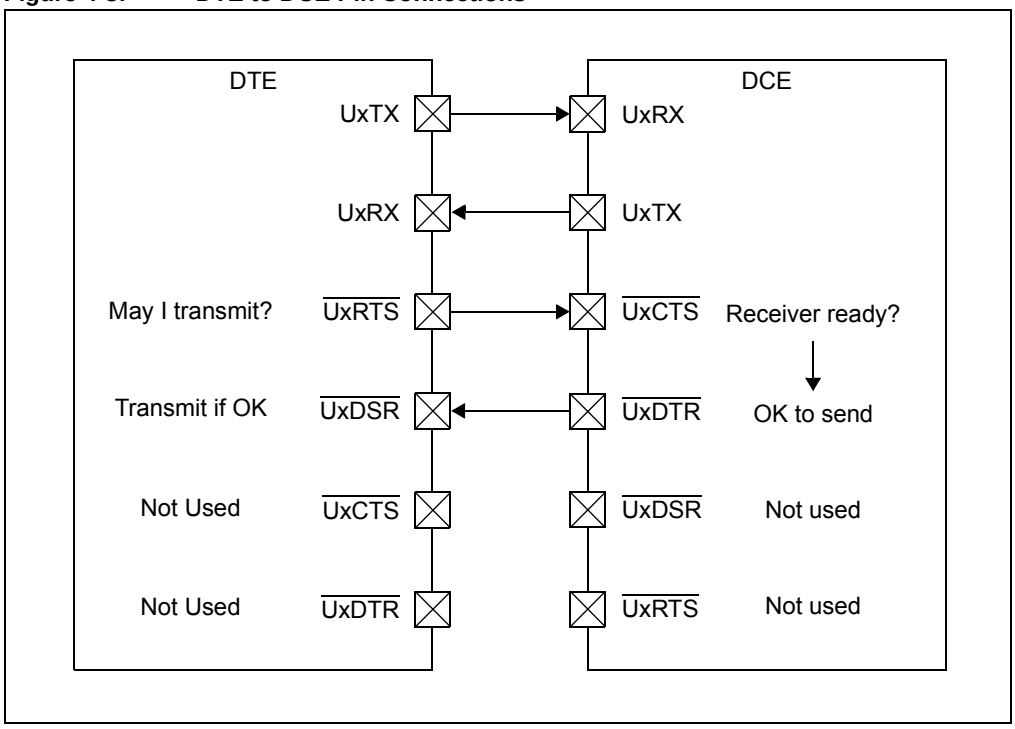
Figure 4-2: DTE to DTE Pin Connections



4.6.2.2 DTE to DCE Configuration

When interfacing a DTE device to a DCE device, connect them as shown in Figure 4-3. The $\overline{\text{UxRTS}}$ output of the DTE is connected to the $\overline{\text{UxCTS}}$ input terminal of the DCE, and the $\overline{\text{UxDTR}}$ output of the DCE is connected to the $\overline{\text{UxDSR}}$ input of the DTE. This allows the DCE to tell the DTE when it is ready to receive data.

Figure 4-3: DTE to DCE Pin Connections



5.0 LIN/J2602

The UART provides support for the Local Interconnect Network (LIN) protocol for both Master and Slave processes to reduce software overhead. The LIN protocol is typically used in automotive applications and packages bytes into message frames. The LIN protocol has 2 types of processes: Master and Slave. A network can have only one Master and multiple Slaves. The Master process transmits a header containing a command that the Slave(s) can respond to. The Master process part of a LIN message frame consists of the following:

1. Break character (11 bits minimum received, 13 transmitted).
2. Delimiter bit.
3. Sync byte (0x55).
4. Protected ID (PID) field.

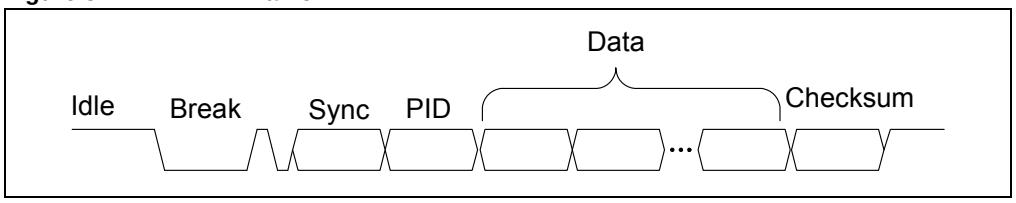
The Slave processes, then completes, the message frame by transmitting the requested data and checksum.

5. Data (up to 8 bytes).
6. Checksum.

The UART has two LIN modes, Master/Slave and Slave Only, selected by the MOD<3:0> bits (UxMODE<3:0>). The Master/Slave mode allows a single instance of a UART to handle both Master and Slave software processes.

A LIN frame starts with the Master process sending a Break, followed by a Sync to allow the receiver to synchronize the baud rate to the transmitter. The PID byte follows and is used by the Slave to determine if, or how, to respond. A Slave process then responds with up to 8 bytes of data and a checksum. A LIN frame is shown in Figure 5-1.

Figure 5-1: LIN Frame



The PID byte consists of 6 bits of data and 2 parity bits, P0 followed by P1. The PID value is written to Parameter 1 (UxP1<5:0>) and the parity bits are automatically calculated. Parameter 1 can only be written when the transmitter is Idle. The parity bits are calculated as follows:

$$P0 = PID[0] \text{ XOR } PID[1] \text{ XOR } PID[2] \text{ XOR } PID[4]$$

$$P1 = \text{NOT } (PID[1] \text{ XOR } PID[3] \text{ XOR } PID[4] \text{ XOR } PID[5])$$

The UART automatically calculates the checksum. Two types of LIN checksums are supported and selected by the C0EN bit (UxMODEH<3>). When C0EN = 0 (default), the legacy LIN checksum method is used, which uses only data bytes. When C0EN = 1, the checksum also includes the PID. The checksum is calculated by adding the number of data bytes, defined by Parameter 2, adding the carry result and finally inverting the sum.

Table 5-1 provides an example checksum calculation for a LIN frame of 4 data bytes in length, with data values of 0x4A, 0x55, 0x93 and 0xE5.

Table 5-1: LIN Checksum Example (C0EN = 1 or 0)

Action	Hex	Carry	D7	D6	D5	D4	D3	D2	D1	D0
0x4A	0x4A		0	1	0	0	1	0	1	0
+0x55 Add Carry	0x9F 0x9F	0	1 1	0 0	0 0	1 1	1 1	1 1	1 1	1 1
+0x93 Add Carry	0x132 0x33	1	0 0	0 0	1 1	1 1	0 0	0 0	1 1	0 1
+0xE5 Add Carry	0x118 0x19	1	0 0	0 0	0 0	1 1	1 1	0 0	0 0	0 1
Invert	0xE6(1)		1	1	1	0	0	1	1	0
Receiver Verification										
Check Local + Received	0x19(2) +0xE6(1)									

Note 1: This is the checksum value transmitted as the last byte.

2: This is the checksum value calculated by the receiver.

For a transmit operation, the calculated checksum is stored in UxTXCHK (Register 2-13). For a receive operation, the calculated checksum is stored in UxRXCHK (Register 2-14).

5.1 LIN Master/Slave Transmit

The following procedure is used for Master/Slave transmit:

1. Configure the clock input and baud rate as detailed in **Section 3.0 “Clocking and Baud Rate Configuration”**.
2. Configure LIN mode by writing ‘0b1100’ to the MOD<3:0> bits.
3. Configure the checksum type by writing the C0EN bit.
4. Set the UARTEN, URXEN and UTXEN bits.
5. Write the 6-bit PID value to Parameter 1 (UxP1<5:0>).

Writing to Parameter 1 will cause the Break, Sync and PID to be transmitted. If it is desired to complete the message frame in Master/Slave mode, the following steps are used:

6. Wait for the RXBKIF bit to set.
7. Write the number of bytes to transmit to Parameter 2 (UxP2<2:0>).
8. Write data to transmit to the UxTXREG register.

5.2 LIN Slave Only Receive

The Slave is typically listening for a PID that it should respond to. Reception of a Break resets the checksum, parity calculation and contents of Parameter 3. The baud rate is automatically calculated and written to UxBRG. The following procedure is used for Slave Only receive:

1. Configure LIN Slave Only mode by writing '0b1011' to the MOD<3:0> bits.
2. Set the UARTEN and URXEN bits.

Upon reception of the Break, the RXBKIF flag is set. Upon reception of the PID, an RX interrupt is generated regardless of the URXISEL<2:0> bits setting.

3. The PID can be read from UxRXREG. If a parity mismatch (P0 and P1) has occurred, the PERR flag will be set.
4. Write the number of bytes to receive to Parameter 3 (UxP3<2:0>).
5. Configure the RX watermark interrupt setting using the URXISEL<2:0> bits.
6. Read data from UxRXREG upon an interrupt event.

The UART automatically verifies the checksum. The checksum that the receiver calculates is stored in the RXCHK<7:0> bits (UxRXCHK<7:0>). If this value doesn't match that of the received checksum, the CERIF flag (UxSTA<4>) will be set, and if the CERIE (UxSTA<12>) bit was set, an interrupt will be generated.

5.3 LIN Slave Only Transmit

A LIN Slave Only transmit is a response to a header sent by the Master and is preceded by a receive event. The baud rate is already determined by the reception of the Sync field. The following procedure is used for Slave Only transmit:

1. Configure LIN Slave Only mode by writing '0b1011' to the MOD<3:0> bits.
2. Configure the checksum type by writing C0EN.
3. Set the UARTEN, URXEN and UTXEN bits.
4. Load UxTXREG with data to transmit.

Prior reception of a Sync is required to 'arm' the transmit event. After a Sync is received, writing to UxTXREG will cause the data and checksum to be transmitted. A TX interrupt will indicate transmission according to the UTXISEL<2:0> bits setting.

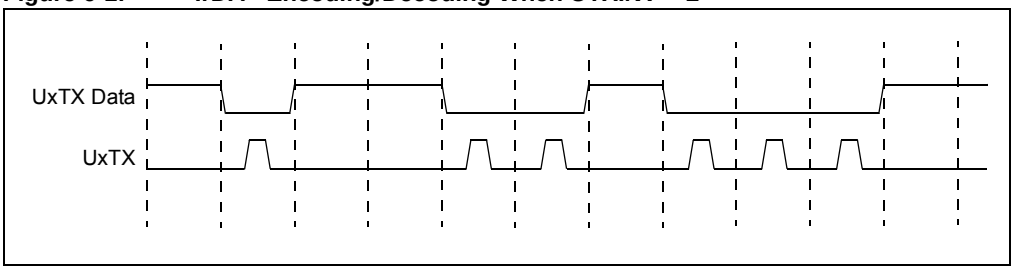
6.0 IrDA®

The UART supports the infrared IrDA protocol with a built-in encoder/decoder so an external codec is not required. Each bit time is divided into 16 baud clock cycles; therefore, the clock prescaler BRGH is forced to '0' and Equation 3-1 is used for the baud rate calculation. The IrDA encoding/decoding consists of the following translations:

- A value of '1' is translated to '0' for all of the 16 baud clocks
- A value of '0' is translated to '0' for the first 7 baud clocks, '1' for the next 3 and '0' for the remaining 6 clocks

To enable IrDA, set MOD<3:0> = 0b1110 (UxMODE<3:0>). Typical IrDA implementations require active-low Idle; therefore, it is recommended to operate the UART with both UTXINV and URXINV set to '1'. This allows the UxTX and UxRX pins to connect directly to an infrared transceiver. Figure 6-1 shows an example IrDA waveform.

Figure 6-1: IrDA® Encoding/Decoding When UTXINV = 1



To configure the UART for IrDA mode, the following sequence is used:

1. Configure the clock input and baud rate as detailed in **Section 3.0 "Clocking and Baud Rate Configuration"**.
2. Configure IrDA mode by writing '0b1110' to the MOD<3:0> bits.
3. Set the UTXINV and URXINV bits.
4. Configure the interrupt watermark using the UTXISEL<2:0> and URXISEL<2:0> bits.
5. Set the UARTEN bit.
6. Set the URXEN and UTXEN bits.

In IrDA mode, the UART functions similar to Asynchronous mode regarding errors, events and interrupts.

7.0 SMART CARD

The UART module supports communication with ISO 7816 smart cards. In a typical application, the UART module is intended to act as the Master or terminal that always initiates communication transactions. The smart card acts as a Slave, and always responds to commands and other stimulus from the terminal. Figure 7-1 shows a smart card subsystem using a microcontroller with a UART module for smart card data communication.

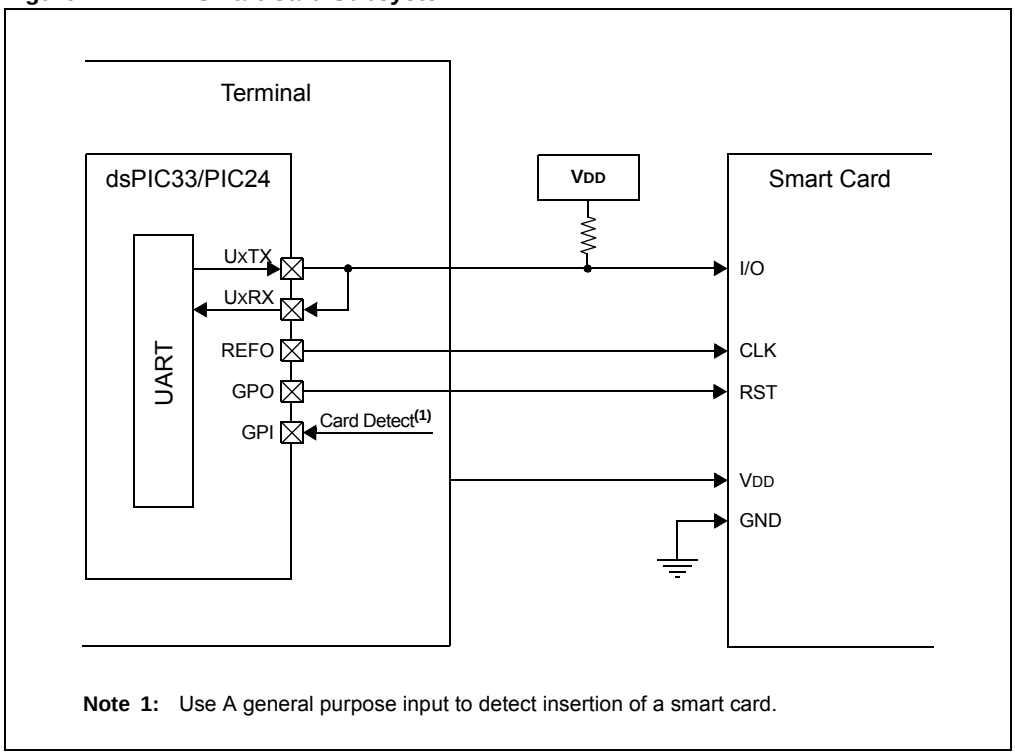
The terminal is also responsible for powering, clocking and resetting the smart card. The clock can be sourced by using the REFO output pin and the Reset signal can be implemented with a general purpose output. The system is based on a half-duplex single wire, requiring the UART UxTX and UART UxRX pins to be shorted externally, and pulled to VDD with a weak pull-up.

The module can be configured to support either block (T = 1) or byte (T = 0) protocol. Block mode is set up for a predetermined message block size, whereas Byte mode transmits one byte at a time.

Upon detection of the card insertion, the terminal pulls the Reset line low to initiate a Reset sequence. The smart card responds with an Answer-to-Reset (ATR), which contains parameters used for communication details. The ATR baud rate is predetermined at the REFO clk/372. The terminal will need to be configured for this baud rate at the time the Reset pulse is sent to the smart card. Typical REFO clock rates are 1 MHz to 5 MHz. See ISO 7816 for additional details on ATR.

Note: Protocol characteristics, electrical characteristics of the smart card, Answer-To-Reset (ATR), PPS (Protocol Parameter Selection), calculation of guard time and wait times are out of the scope of this Family Reference Manual section. Please refer to the licensed version of the ISO 7816-3 document for details about smart card communication.

Figure 7-1: Smart Card Subsystem



Note 1: Use A general purpose input to detect insertion of a smart card.

7.1 Protocol and Frame Details

The smart card communication scheme is based on an Elementary Time Unit (ETU) that is also the bit clock. The smart card will provide the ETU value in the ATR and the terminal is configured accordingly. A character frame consists of 10 bits; a Start bit, 8 data bits and a parity bit. Depending on the mode, guard and wait times are used to separate bytes and message transitions.

The ISO 7816 specification defines 2 communication logic conventions: direct and inverse. Direct convention is defined as LSB first and a high state as logic one. Inverse mode is defined as MSB first and a low on the line is interpreted as a logic low. The logic convention is set using the CONV bit (UxSCCON<3>).

7.1.1 GUARD TIME

Guard time is defined as the minimum delay between two consecutive character frames. The ISO 7816 specification defines both a Character Guard Time (CGT) and a Block Guard Time (BGT). In both T = 0 and T = 1 modes, CGT is defined as the minimum delay between the leading edges of the two consecutive characters in the same direction of transmission. Block Guard Time (BGT) for T = 1 mode only is defined as the minimum delay between the leading edges of the two consecutive characters in the opposite directions. The BGT has a standard fixed value of 22 ETUs.

7.1.2 WAIT TIME

Wait time is the maximum delay allowed between two consecutive characters transmitted by the card or an interfacing device. The Character Wait Time (CWT) is the maximum delay between the leading edges of the two consecutive characters in the block, as shown in Figure 7-2. The minimum delay is CGT. The Block Wait Time (BWT) is the maximum delay between the leading edge of the last character of the block received by the card, and the leading edge of the first character of the next block transmitted by the card, as shown in Figure 7-3. BWT helps the interfacing device in detecting the unresponsive smart cards. The minimum delay is BGT.

Note:	The LAST bit (UxTXREG<15>) set by user software is used to automatically start the guard or wait timers, depending on the state of the module.

Figure 7-2: Character Guard and Wait Time

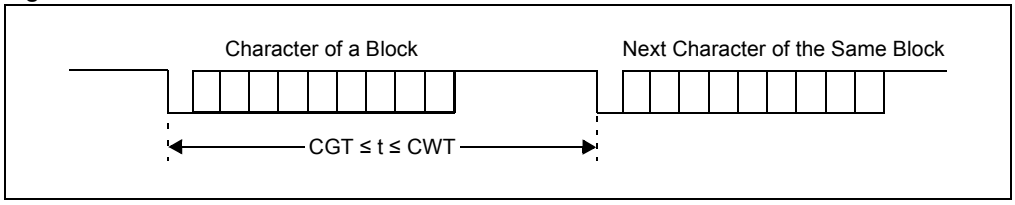
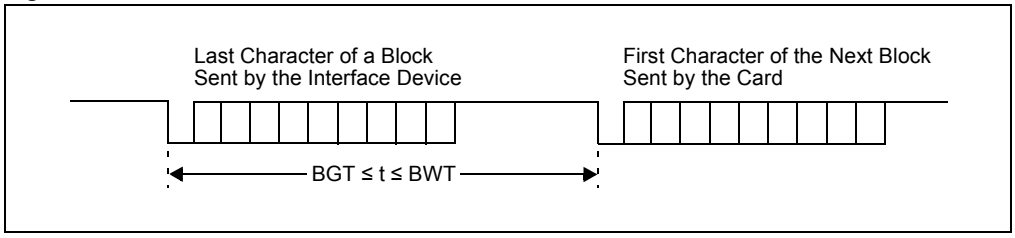


Figure 7-3: Block Guard and Wait Time

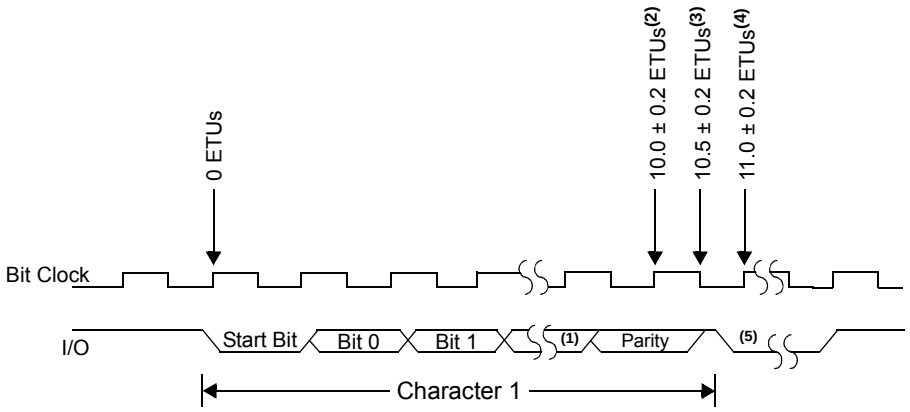


7.1.3 ERROR DETECTION

The transmitter is responsible for calculating the parity bit value. Parity is always even, defined as the number of logic ones and the parity bit always being an even count. The receiver also calculates the parity value and compares it to the received parity bit. If a discrepancy is found, the error is flagged by the receiver, pulling the line low for a duration defined by the T0PD bit (UxSCCON<2>).

The transmitter tests the I/O line at time 11 ± 0.2 ETUs after the leading edge of the Start bit of a character was sent. If the transmitter detects an error by detecting a low state on the I/O line, it will repeat the disputed character after a delay of at least 2 ETUs following the detection of the error. The number of repeats is configured with the TXRPT<1:0> bits (UxSCCON<5:4>). See Figure 7-4 for timing details in T = 0 mode.

Figure 7-4: T = 0 Character Repetition Timing



Note 1: 8-bit character.
2: At 10.0 ± 0.2 ETUs, the transmitter disables the driver.
3: At 10.5 ± 0.2 ETUs, the receiver sets the I/O line low if a parity error is detected.
4: At 11.0 ± 0.2 ETUs, the transmitter tests the I/O line.
5: See the T0PD bit (UxSCCON<2>) in Register 2-15.

7.2 Smart Card Operation

7.2.1 PRE-ATR INITIALIZATION

The module should be configured in Receive mode prior to the Reset line being pulled low to initiate the smart card's response as follows:

1. Write the BRG register with a value corresponding to REFO/372.
2. Configure Smart Card mode by writing '0b1111' to the MOD<3:0> bits.
3. Set the UARTEN, URXEN and UTXEN bits.
4. Configure the RX interrupt watermark using the URXISEL<2:0> bits (UxSTAH<10:8>).
5. Read data out of UxRXREG as it becomes available and save for ATR processing.

7.2.2 POST-ATR INITIALIZATION

After the terminal has done a Reset of the smart card and received the setup parameters contained in the ATR, the user software can configure the module for communication as follows:

1. Disable the UART for configuration changes by clearing the UARTEN bit.
2. Set the PRTCL (UxSCCON<1>), T0PD (UxSCCON<2>), CONV (UxSCCON<3>) and TXRPT<1:0> bits (UxSCCON<5:4>) according to ATR parameters.
3. Program the UxBRG register for the ETU defined in ATR.
4. Program guard time using Parameter 1 and set the GTCIE bit (UxSCINT<0>).
5. Program the wait time using Parameter 3/3H and set the WTCIE bit (UxSCINT<1>). If Parameter 3H needs to be written, it must be written before Parameter 3.
6. Configure the RX interrupt watermark using the URXISEL<2:0> bits (UxSTAH<10:8>).
7. Set the UARTEN, UTXEN and URXEN bits.

7.2.3 T = 0 PROTOCOL COMMUNICATION

Transmission with T = 0:

1. Write data into UxTXREG.
2. Take appropriate actions according to the ISO 7816 standard if the WTCIF, GTCIF and/or TXRPTIF bits are set.
3. Set LAST = 1 (UxTXREG<15>) for the last byte of the data.

Note:	Due to the UxTX and UxRX pins being shorted, a receive interrupt will be generated on transmission of a character if enabled. It is recommended to disable receive interrupts when transmitting.

Reception with T = 0:

1. Read the UxRXREG as the data becomes available upon a receive interrupt.
2. Take appropriate actions according to the ISO 7816 standard if the WTCIF, GTCIF and/or RXRPTIF bits are set.

Note:	For the last character, the user must ensure that the guard time is satisfied before transmitting the response. The GTC may be used for this purpose, whereas the WTC interrupt may be disabled or ignored.

7.2.4 T = 1 PROTOCOL COMMUNICATION

Transmission with T = 1:

1. Write data into UxTXREG.
2. Program the value of the BWT to Parameter 2 and set the WTCIE bit (UxSCINT<1>).
3. Take appropriate actions according to the ISO 7816 standard if the WTCIF, GTCIF and/or TXRPTIF bits are set.
4. Set LAST = 1 (UxTXREG<15>) for the last byte of the data.

Reception with T = 1:

1. Read the UxRXREG as the data becomes available upon a receive interrupt.
2. Program the value of the CWT to Parameter 3/3H and set the WTCIE bit (UxSCINT<1>) to '1'. If Parameter 3H needs to be written, it must be written before Parameter 3.
3. Take appropriate actions according to the ISO 7816 standard if the WTCIF, GTCIF and/or RXRPTIF bits are set.

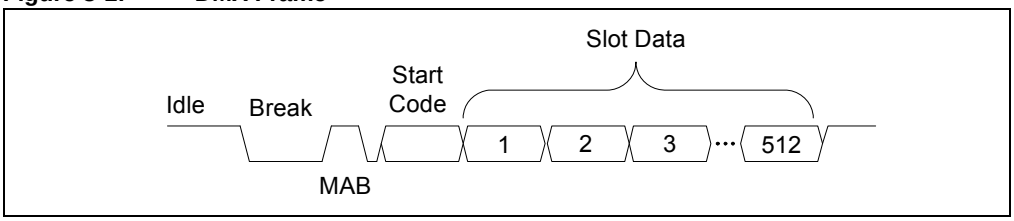
Note: For the last character, the user must ensure the guard time is satisfied before transmitting the response. The GTC may be used for this purpose, whereas the WTC interrupt may be disabled or ignored.

8.0 DMX

Digital Multiplex 512 (DMX) is a protocol used typically for stage lighting and effects. It supports a ‘universe’ of up to 512 channels and is typically implemented using EIA-485 at the physical layer. DMX communication is one way only, with the controller only sending messages and the Slave device only receiving. There is no error checking or confirmation that a command has been received. DMX operates at a baud rate of 250k with no parity and two Stop bits. A DMX message frame consists of a header and up to 512 ‘slots’ (data bytes). A Slave device can be configured to accept more than one slot, given start and stop assignment values.

A DMX message frame consists of a Break, a Mark After Break (MAB), a start code and finally the slot data bytes. The MAB is 3 bit times in length. The start code specifies the type of data and is typically at 0x00. Figure 8-1 shows a DMX frame.

Figure 8-1: DMX Frame



8.1 DMX Transmit

The following procedure is used for DMX transmit:

1. Configure the clock input and baud rate as detailed in **Section 3.0 “Clocking and Baud Rate Configuration”** for 250 kbaud.
2. Configure DMX mode by writing ‘0b1010’ to the MOD<3:0> bits.
3. Set STSEL to ‘0b10’.
4. Configure the TX interrupt watermark using the UTXISEL<2:0> bits.
5. Write to Parameter 1 equal to the number of bytes – 1 (not including start code).
6. Set the UARTEN and UTXEN bits.
7. Write the start code value to UxTXREG.

Writing the start code to UxTXREG will transmit a 25-bit Break, MAB and start code.

8. Write slot data bytes to UxTXREG.

If not all bytes defined by Parameter 1 are written, the line will return to the Idle state. Once all bytes are written, the frame is considered complete and the next write to UxTXREG will send a Break for the next frame.

8.2 DMX Receive

The following procedure is used for DMX receive:

1. Configure the clock input and baud rate as detailed in **Section 3.0 “Clocking and Baud Rate Configuration”** for 250 kbaud.
2. Configure DMX mode by writing ‘0b1010’ to the MOD<3:0> bits.
3. Configure the RX interrupt watermark using the URXISEL<2:0> bits.
4. Set the UARTEN bit
5. Set the URXEN bit.
6. Wait for the RXBKIF bit to set.
7. Write the byte start value – 1 to Parameter 2 (not including start code).
8. Write the byte end value – 1 to Parameter 3 (not including start code).

Once a Break is received, the UART will load the start code byte into UxRXREG and generate an RX interrupt, regardless of the RX watermark setting (URXISEL<2:0>). The range of bytes defined by Parameter 2 and Parameter 3 are then loaded into UxRXREG, and an RX interrupt is generated according to the URXISEL<2:0> bits.

9.0 INTERRUPTS

The UART has four separate interrupts. To determine which event has caused the interrupt, the associated flag needs to be read and evaluated. All interrupt sources are listed in Table 9-1.

Table 9-1: Interrupts

Interrupt Type	Condition	Flag
TX	Number of Empty Slots in UxTXREG Defined by UTXISEL<2:0> bits	TXIF(1)
RX	Number of Words in UxRXREG Defined by URXISEL<2:0> bits Address Match	RXIF(1) PERR
Event	Auto-Baud Complete RX Break Received Wake Event (line is high-to-low) Smart Card Guard Time Counter Match Smart Card Waiting Time Counter Match Smart Card Block Time Counter Match Smart Card Receive Repeat Smart Card Transmit Repeat	ABDIF RXBKIF WUIF GTCIF WTCIF BTCIF RXRPTIF TXRPTIF
Error	Parity Error Framing Error Transmit Collision Transmit Shift Register Empty RX Buffer Overflow Auto-Baud Rollover Checksum Error (LIN mode only)	PERR FERR TXCIF TXMTIE OERR ADBOVF CERIF

Note 1: Device-dependent, refer to the specific device data sheet for more information.

9.1 Interrupt Watermarks

Both TX and RX interrupt frequency can be configured using the watermark setting. For transmit, the UTXISEL<2:0> bits setting allows the TX interrupt frequency to be based on the number of empty slots left in the TX buffer (UxTXREG). By default, a TX interrupt will be generated when the TX buffer is empty. For receive, the URXISEL<2:0> bits setting allows the RX interrupt frequency to be based on how many bytes are in the RX buffer (UxRXREG). By default, an RX interrupt will be generated when the RX buffer has at least one byte in it. The receive watermark interrupt will not be set if PERIF or FERIF are set and the corresponding PERIE or FERIE bits are set.

10.0 POWER-SAVING MODES

The UART provides support in power-saving modes, including the capability to run in Sleep and Idle modes. If a transmission or reception is in progress, and a power save command is executed, the operation will abort. SFR data, including UxMODE, UxSTA, UxBRG and the RX and TX buffers, will retain their values upon a wake condition and do not need to be reinitialized.

10.1 Sleep

When a device enters Sleep mode, the system clock used by the core processor and peripherals is halted. To use run in Sleep mode, a clock source other than the system clock must be selected and the SLPEN bit (UxMODEH<15>) is set. Clock sources are device-specific (see the device data sheet and **Section 3.0 "Clocking and Baud Rate Configuration"** for details). This allows the UART to request the selected clock source and keep it active. The UART has the ability to continue transmitting the contents of UxTXREG, receiving data and storing it in UxRXREG.

The UART can also wake the processor from Sleep mode (when SPLEN = 0) on the detection of an incoming byte, including Break and Sync characters used for auto-baud. To enable the wake feature, set the WAKE bit (UxMODE<12>). When a wake from Sleep occurs, the WUIF (UxINT<7>) bit is set and an event interrupt is generated effectively waking the processor. If auto-baud is desired on wake, set the ABAUD bit before executing a SLEEP command.

10.2 Idle

In Idle mode, the core processor is halted. However, the peripherals, including the UART, continue to run. To also halt the UART in Idle, the UART Stop in Idle Mode bit, USIDL (UxMODE<13>), can be set.

11.0 RELATED APPLICATION NOTES

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the dsPIC33/PIC24 device families, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the Multiprotocol UART module are:

Title	Application Note #
No related application notes at this time.	N/A

Note: Please visit the Microchip web site (www.microchip.com) for additional application notes and code examples for the dsPIC33/PIC24 families of devices.

12.0 REVISION HISTORY

Revision A (September 2016)

This is the initial version of this document.

Revision B (December 2017)

Updated **Section 1.0 “Introduction”**, **Section 1.3 “Data Buffers”**, **Section 3.0 “Clocking and Baud Rate Configuration”**, **Section 3.1 “Legacy Mode”**, **Section 4.1 “Asynchronous Transmit”**, **Section 4.2 “Asynchronous Receive”**, **Section 4.2.1 “Receive Errors and Events”**, **Section 4.6.2 “Hardware Flow Control”**, **Section 5.1 “LIN Master/Slave Transmit”**, **Section 5.3 “LIN Slave Only Transmit”**, **Section 7.2.2 “Post-ATR Initialization”**, **Section 7.2.4 “T = 1 Protocol Communication”**, **Section 8.1 “DMX Transmit”**, **Section 9.1 “Interrupt Watermarks”** and **Section 10.1 “Sleep”**.

Removed **Section 4.7 “Manchester Encoding”** and **Section 9.0 “DALI”**.

Updated Table 4-1.

Updated Figure 1-1, Figure 4-1 and Figure 4-2.

Updated Register 2-1, Register 2-2, Register 2-3, Register 2-9 and Register 2-10.

Updated Equation 3-1 and Equation 3-2.

NOTES:

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

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

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

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

Trademarks

ISBN: 978-1-5224-2462-8

Worldwide Sales and Service

AMERICAS

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

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

Austin, TX
Tel: 512-257-3370

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

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

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

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

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

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

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

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

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

ASIA/PACIFIC

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

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

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

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

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

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

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

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

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

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

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

Singapore
Tel: 65-6334-8870

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

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

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

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

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

EUROPE

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

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

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

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

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

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

Germany - Karlsruhe
Tel: 49-721-625370

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

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

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

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

Italy - Padova
Tel: 39-049-7625286

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

Norway - Trondheim
Tel: 47-7289-7561

Poland - Warsaw
Tel: 48-22-3325737

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

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

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

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

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