Section 17. UART

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 sheet. Depending on the device variant, this manual section may not apply to all dsPIC33E/PIC24E 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

17.1 INTRODUCTION

The Universal Asynchronous Receiver Transmitter (UART) module is one of the serial I/O modules available in the dsPIC33E/PIC24E device family. The UART is a full-duplex, asynchronous communication channel that communicates with peripheral devices and personal computers using protocols, such as RS-232, RS-485, LIN and IrDA®. The module also supports the hardware flow control option with $\overline{\text{UxCTS}}$ and $\overline{\text{UxRTS}}$ pins and includes the IrDA encoder and decoder.

The primary features of the UART module are as follows:

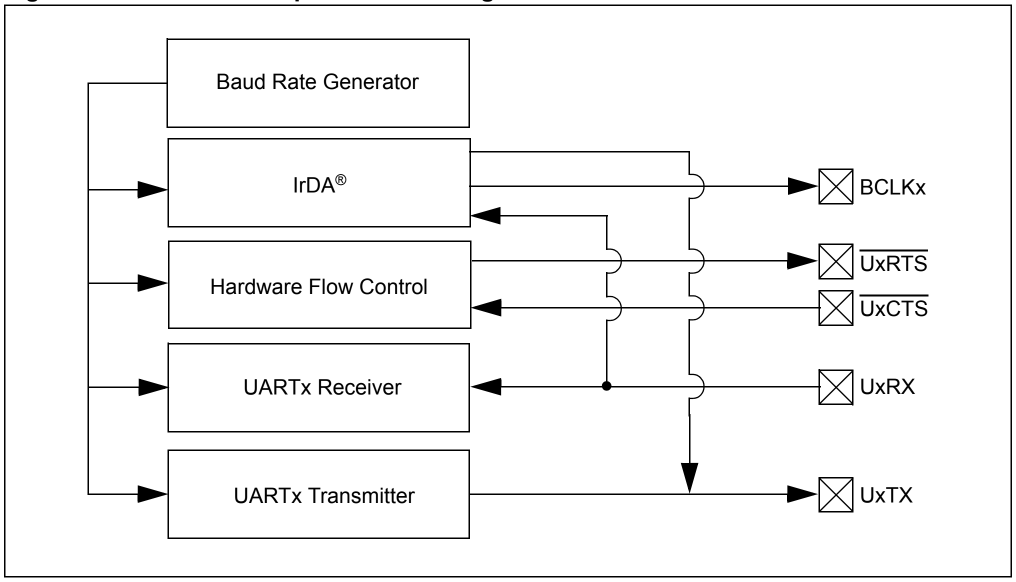
- Full-duplex, 8-bit or 9-bit data transmission through the UxTX and UxRX pins
- Even, odd or no parity options (for 8-bit data)
- One or two stop bits
- Hardware auto-baud feature
- Hardware flow control option with $\overline{\text{UxCTS}}$ and $\overline{\text{UxRTS}}$ pins
- Fully integrated Baud Rate Generator (BRG) with 16-bit prescaler
- Baud rates ranging from 10 Mbps to 38 bps at 40 MIPS
- Four-deep First In First Out (FIFO) transmit data buffer
- Four-deep FIFO receive data buffer
- Parity, framing and buffer overrun error detection
- Support for 9-bit mode with address detect (9th bit = `1`)
- Transmit and receive Interrupts
- Loopback mode for diagnostic support
- IrDA encoder and decoder logic
- LIN bus support
- 16x baud clock output for external IrDA encoder/decoder support

Note: Each dsPIC33E/PIC24E family device variant may have one or more UART modules. An ‘x’ used in the names of pins, control/status bits and registers denotes the particular UART module number. Refer to the specific device data sheet for more details.

A simplified block diagram of the UART is illustrated in Figure 17-1. The UART module consists of the following key hardware elements:

- Baud Rate Generator
- Asynchronous Transmitter
- Asynchronous Receiver

Figure 17-1: UART Simplified Block Diagram



17.2 CONTROL REGISTERS

This section outlines the specific functions of each register that controls the operation of the UART module:

- **UxMODE: UARTx Mode Register**
  - Enables or disables the UART module
  - Enables or disables the IrDA encoder and decoder
  - Enables or disables the WAKE, ABAUD and Loopback features
  - Enables or disables the $\overline{\text{UxRTS}}$ and $\overline{\text{UxCTS}}$ pins
  - Configures the $\overline{\text{UxRTS}}$ pin for the desired mode of operation
  - Configures the polarity of the UxRX pin
  - Selects the type of baud rate
  - Selects the number of data bits, parity and stop bits
- **UxSTA: UARTx Status and Control Register**
  - Selects the Transmission Interrupt mode
  - Selects the Receive Interrupt mode
  - Enables or disables the UART transmission
  - Controls the Address Detect mode
  - Indicates various status conditions, such as transmit and receive buffer state, parity error, framing error and overflow error
- **UxRXREG: UARTx Receive Register**
  - Stores the received data
- **UxTXREG: UARTx Transmit Register (Write-Only)**
  - Provides the data to be transmitted
- **UxBRG: UARTx Baud Rate Register**
  - Stores the baud rate value of the transmitted or received data

Register 17-1: UxMODE: UARTx Mode Register

R/W-0	U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
UARTEN	—	USIDL	IREN(1)	RTSMD	—	UEN<1:0>	
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
WAKE	LPBACK	ABAUD	URXINV	BRGH	PDSEL<1:0>		STSEL
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 **UARTEN:** UARTx Enable bit

1 = UARTx is enabled; UARTx pins are controlled by UARTx as defined by the UEN<1:0> and UTXEN control bits
0 = UARTx is disabled; UARTx pins are controlled by the corresponding PORT, LAT and TRIS bits

Note: Enable this bit before enabling the UTXEN bit (UxSTA<10>).

bit 14 **Unimplemented:** Read as ‘0’

bit 13 **USIDL:** Stop in Idle Mode bit

1 = Discontinue operation when the device enters Idle mode
0 = Continue operation in Idle mode

bit 12 **IREN:** IrDA Encoder and Decoder Enable bit(1)

1 = IrDA encoder and decoder are enabled
0 = IrDA encoder and decoder are disabled

bit 11 **RTSMD:** Mode Selection for $\overline{\text{UxRTS}}$ Pin bit

1 = $\overline{\text{UxRTS}}$ is in Simplex mode
0 = $\overline{\text{UxRTS}}$ is in Flow Control mode

bit 10 **Unimplemented:** Read as ‘0’

bit 9-8 **UEN<1:0>:** UARTx Enable bits

11 = UxTX, UxRX and BCLKx pins are enabled and used; $\overline{\text{UxCTS}}$ pin is controlled by port latches
10 = UxTX, UxRX, $\overline{\text{UxCTS}}$ and $\overline{\text{UxRTS}}$ pins are enabled and used
01 = UxTX, UxRX and $\overline{\text{UxRTS}}$ pins are enabled and used; $\overline{\text{UxCTS}}$ pin is controlled by port latches
00 = UxTX and UxRX pins are enabled and used; $\overline{\text{UxCTS}}$, $\overline{\text{UxRTS}}$ and BCLKx pins are controlled by port latches

bit 7 **WAKE:** Enable Wake-up on Start bit Detect During Sleep Mode bit

1 = Wake-up is enabled
0 = Wake-up is disabled

Note: The UART module does not recognize the first character received on a wake.

bit 6 **LPBACK:** UARTx Loopback Mode Select bit

1 = Enable Loopback mode
0 = Loopback mode is disabled

Note 1: This feature is only available for Low-Speed mode (BRGH = 0). For more details, refer to the specific device data sheet.

Register 17-1: UxMODE: UARTx Mode Register (Continued)

bit 5	**ABAUD:** Auto-Baud Enable bit 1 = Enable baud rate measurement on the next character. Requires reception of a Sync field (55h); cleared in hardware upon completion 0 = Baud rate measurement disabled or completed **Note:** The use of this feature consumes the corresponding ICx peripheral. Therefore, the user should not attempt to use the ICx peripheral in their applications.
bit 4	**URXINV:** Receive Polarity Inversion bit 1 = UxRX Idle state is ‘0’ 0 = UxRX Idle state is ‘1’
bit 3	**BRGH:** High Baud Rate Select bit 1 = High speed 0 = Low speed
bit 2-1	**PDSEL<1:0>:** Parity and Data Selection bits 11 = 9-bit data, no parity 10 = 8-bit data, odd parity 01 = 8-bit data, even parity 00 = 8-bit data, no parity
bit 0	**STSEL:** Stop Selection bit 1 = 2 Stop bits 0 = 1 Stop bit

Note 1: This feature is only available for Low-Speed mode (BRGH = 0). For more details, refer to the specific device data sheet.

Register 17-2: UxSTA: UARTx Status and Control Register

R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R-0	R-1
UTXISEL1	UTXINV	UTXISEL0	—	UTXBRK	UTXEN	UTXBF	TRMT
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R-1	R-0	R-0	R/C-0	R-0
URXISEL<1:0>		ADDEN	RIDLE	PERR	FERR	OERR	URXDA
bit 7							bit 0

Legend:	C = 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 15,13 **UTXISEL<1:0>:** Transmission Interrupt Mode Selection bits

`11` = Reserved
`10` = Interrupt generated when a character is transferred to the Transmit Shift register and the transmit buffer becomes empty
`01` = Interrupt generated when the last transmission is over, transmit buffer is empty (i.e., the last character has been shifted out of the Transmit Shift register), and all the transmit operations are completed
`00` = Interrupt generated when any character is transferred to the Transmit Shift register, and transmit buffer is empty (which implies at least one location is empty in the transmit buffer)

bit 14 **UTXINV:** Transmit Polarity Inversion bit

IREN = `0`:
`1` = UxTX Idle state is ‘`0`’
`0` = UxTX Idle state is ‘`1`’
IREN = `1`:
`1` = IrDA encoded, UxTX Idle state is ‘`1`’
`0` = IrDA encoded, UxTX Idle state is ‘`0`’

bit 12 **Unimplemented:** Read as ‘`0`’

bit 11 **UTXBRK:** Transmit Break bit

`1` = UxTX pin is driven low regardless of the transmitter state (Sync Break transmission – Start bit followed by twelve ‘`0`’s and a Stop bit)
`0` = Sync Break transmission is disabled or completed

bit 10 **UTXEN:** Transmit Enable bit

`1` = UARTx transmitter enabled; UxTX pin is controlled by UARTx (if UARTEN = `1`)
`0` = UARTx transmitter disabled; any pending transmission is aborted and the buffer is reset; UxTX pin is controlled by PORT

Note: Enable the UARTEN bit (UxMODE<15>) before enabling this bit.

bit 9 **UTXBF:** Transmit Buffer Full Status bit (read-only)

`1` = Transmit buffer is full
`0` = Transmit buffer is not full; at least one more data word can be written

bit 8 **TRMT:** Transmit Shift Register is Empty bit (read-only)

`1` = Transmit Shift register is empty and the transmit buffer is empty (i.e., the last transmission has completed)
`0` = Transmit Shift register is not empty; a transmission is in progress or queued in the transmit buffer

bit 7-6 **URXISEL<1:0>:** Receive Interrupt Mode Selection bits

`11` = Interrupt flag bit is set when the receive buffer is full (i.e., 4 data characters)
`10` = Interrupt flag bit is set when the receive buffer is 3/4 full (i.e., 3 data characters)
`0x` = Interrupt flag bit is set when a character is received

Register 17-2: UxSTA: UARTx Status and Control Register (Continued)

bit 5	**ADDEN:** Address Character Detect bit (bit 8 of received data = `1`) `1` = Address Detect mode enabled. If 9-bit mode is not selected, this control bit has no effect `0` = Address Detect mode disabled
bit 4	**RIDLE:** Receiver Idle bit (read-only) `1` = Receiver is Idle `0` = Data is being received
bit 3	**PERR:** Parity Error Status bit (read-only) `1` = Parity error has been detected for the current character `0` = Parity error has not been detected
bit 2	**FERR:** Framing Error Status bit (read-only) `1` = Framing error has been detected for the current character `0` = Framing error has not been detected
bit 1	**OERR:** Receive Buffer Overrun Error Status bit (clear/read-only) `1` = Receive buffer has overflowed `0` = Receive buffer has not overflowed. (Clearing a previously set OERR bit will reset the receive buffer and RSR to an empty state)
bit 0	**URXDA:** Receive Buffer Data Available bit (read-only) `1` = Receive buffer has data; at least one more character can be read `0` = Receive buffer is empty

Register 17-3: UxRXREG: UARTx Receive Register

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R-0
—	—	—	—	—	—	—	URX8
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
URX<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	**URX8:** Data bit 8 of the Received Character (in 9-bit mode)
bit 7-0	**URX<7:0>:** Data bits 7-0 of the Received Character

Register 17-4: UxTXREG: UARTx Transmit Register (Write-Only)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	W-x
—	—	—	—	—	—	—	UTX8
bit 15							bit 8

W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x
UTX<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	**UTX8:** Data bit 8 of the Transmitted Character (in 9-bit mode)
bit 7-0	**UTX<7:0>:** Data bits 7-0 of the Transmitted Character

Register 17-5: 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 15-0	**BRG<15:0>:** Baud Rate Divisor bits

17.3 UART BAUD RATE GENERATOR

The UART module consists of a dedicated 16-bit Baud Rate Generator. The UxBRG register controls the period of a free-running, 16-bit timer. Equation 17-1 shows the formula for computing the baud rate with BRGH = `0`.

Equation 17-1: UART Baud Rate (BRGH = `0`)

$$\text{Baud Rate} = \frac{F_{CY}}{16 \times (UxBRG + 1)} \text{ ...... (1)}$$

$$UxBRG = \frac{F_{CY}}{16 \times \text{Baud Rate}} - 1 \text{ ...... (2)}$$

Note: F_{CY} denotes the instruction cycle clock frequency (F_{OSC}/2).

Example 17-1 shows the calculation of baud rate error for the following conditions:

- F_{CY} = 4 MHz
- Desired Baud Rate = 9600

Example 17-1: Baud Rate Error Calculation (BRGH = `0`)

$$\text{Desired Baud Rate} = \frac{F_{CY}}{16 \times (UxBRG + 1)} \text{ ...... (1)}$$

Solving for UxBRG value:

$$UxBRG = \frac{F_{CY}/\text{Desired Baud Rate}}{16} - 1$$

$$= \left(\frac{4000000/9600}{16} - 1\right)$$

$$= 25$$

$$\text{Calculated Baud Rate} = \frac{4000000}{16 \times (25 + 1)}$$

$$= 9615$$

$$\text{Error} = \frac{\text{Calculated Baud Rate} - \text{Desired Baud Rate}}{\text{Desired Baud Rate}} \text{...... (2)}$$

$$= \frac{9615 - 9600}{9600}$$

$$= 0.16\%$$

The maximum baud rate (BRGH = `0`) possible is F_{CY}/16 (for UxBRG = `0`), and the minimum baud rate possible is F_{CY}/(16 * 65536).

Equation 17-2 shows the formula for computing the baud rate with BRGH = `1`.

Equation 17-2: UART Baud Rate (BRGH = `1`)

$$\text{Baud Rate} = \frac{F_{CY}}{4 \times (UxBRG + 1)} \text{ ...... (1)}$$

$$UxBRG = \frac{F_{CY}}{4 \times \text{Baud Rate}} - 1 \text{ ...... (2)}$$

Note: F_{CY} denotes the instruction cycle clock frequency.

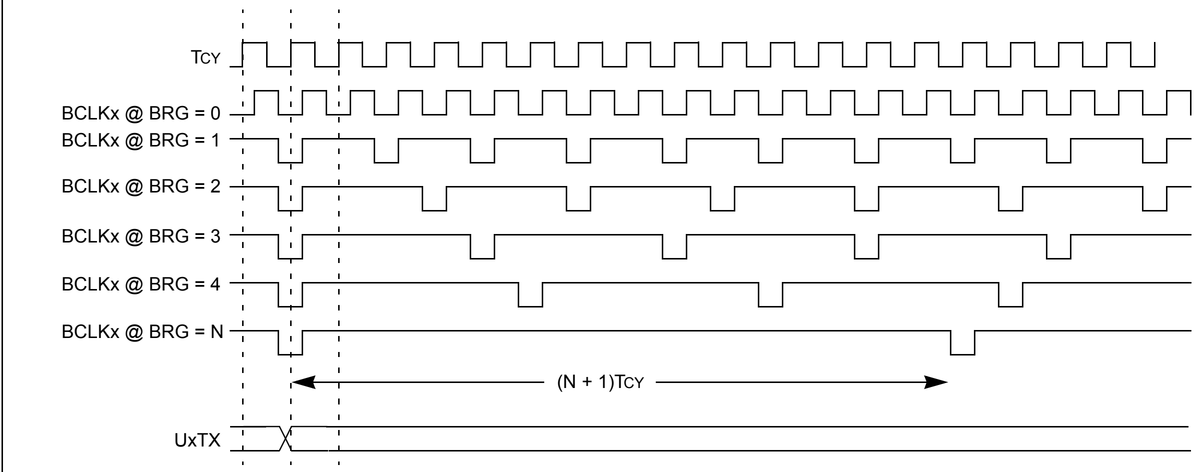
The maximum baud rate (BRGH = `1`) possible is F_{CY}/4 (for UxBRG = `0`), and the minimum baud rate possible is F_{CY}/(4 * 65536).

By writing a new value to the UxBRG register causes the BRG timer to reset (cleared). This ensures the BRG does not wait for a timer overflow before generating the new baud rate.

17.3.1 BCLKx Output

The BCLKx pin outputs the 16x baud clock if the UART and BCLKx output are enabled (UEN<1:0> = `11`). This feature is used for external IrDA encoder/decoder support (see Figure 17-2). The BCLKx output stays high during Sleep mode. BCLKx is forced as an output as long as the UART is kept in this mode (UEN<1:0> = `11`), regardless of the PORTx and TRISx latch bits.

Figure 17-2: BCLKx Output vs. UxBRG Programming



17.4 UART CONFIGURATION

The UART uses the standard Non-Return-to-Zero (NRZ) format (one Start bit, eight or nine Data bits, and one or two Stop bits). Parity is supported by the hardware and can be configured by the user application as even, odd or no parity. The most common data format is 8 bits, no parity, and one Stop bit (denoted as 8, N, 1), which is the default (POR) setting. The number of data bits, Stop bits and the parity are specified in the PDSEL<1:0> (UxMODE<2:1>) and STSEL (UxMODE<0>) bits. An on-chip, dedicated, 16-bit Baud Rate Generator can be used to derive standard baud rate frequencies from the oscillator. The UART transmits and receives the Least Significant bit (LSb) first. The transmitter and receiver of UART module are functionally independent, but use the same data format and baud rate.

17.4.1 Enabling the UART

The UART module is enabled by setting the UARTEN (UxMODE<15>) and UTXEN (UxSTA<10>) bits. Once enabled, the UxTX pin is configured as an output and UxRX pin as an input, overriding the TRIS and PORT register bit settings for the corresponding I/O port pins. The UxTX pin is at logic '1' when no transmission is taking place.

Note: The UTXEN bit is set after the UARTEN bit has been set; otherwise, UART transmissions will not be enabled.

17.4.2 Disabling the UART

The UART module is disabled by clearing the UARTEN bit (UxMODE<15>). This is the default state after any reset. If the UART is disabled, all UART pins operate as port pins under the control of their corresponding PORT and TRIS bits.

Disabling the UART module resets the buffers to empty states. Any data characters in the buffers are lost and the baud rate counter is reset.

All error and status flags associated with the UART module are reset when the module is disabled. The UTXBRK, UTXEN, UTXBF, PERR, FERR, OERR and URXDA bits are cleared, whereas the TRMT and RIDLE bits are set. Other control bits (including ADDEN, URXISEL<1:0> and UTXISEL<1:0>), and the UxMODE and UxBRG registers are not affected.

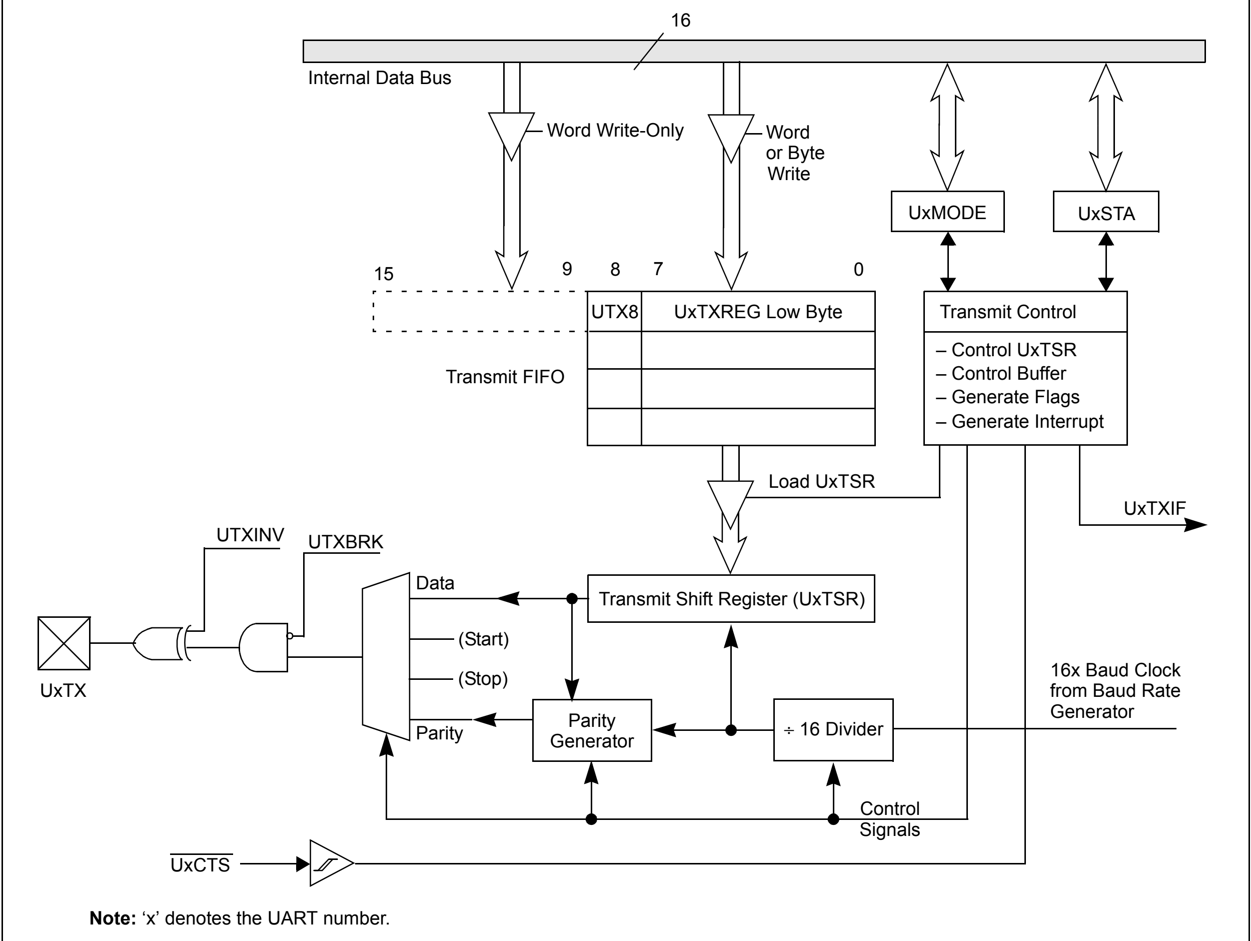
The clearing of the UARTEN bit while the UART is active will abort all pending transmissions and receptions, and resets the module as defined above. Re-enabling the UART will restart the UART in the same configuration.

17.5 UART TRANSMITTER

The transmitter block diagram of the UART module is illustrated in Figure 17-3. The important part of the transmitter is the Transmit Shift register (UxTSR). The Shift register obtains its data from the transmit FIFO buffer, UxTXREG. The UxTXREG register is loaded with data in software. The UxTSR register is not loaded until the Stop bit has been transmitted from the previous load. As soon as the Stop bit is transmitted, the UxTSR is loaded with new data from the UxTXREG register (if available).

Note: The UxTSR register is not mapped in data memory, so it is not available to the user application.

Figure 17-3: UART Transmitter Block Diagram



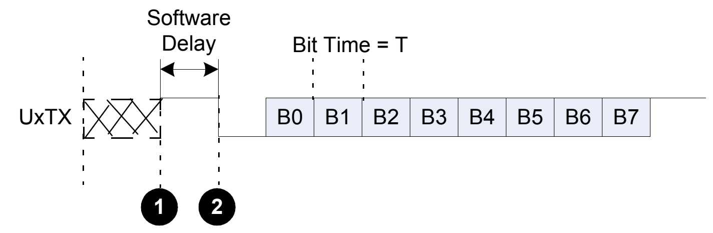
The transmission is enabled by setting the UTXEN enable bit (UxSTA<10>). The actual transmission will not occur until the UxTXREG register has been loaded with data and the Baud Rate Generator (UxBRG) has produced a shift clock (Figure 17-3). Normally, when the first transmission is started, the UxTSR register is empty, so a transfer to the UxTXREG register will result in an immediate transfer to UxTSR. Clearing the UTXEN bit during a transmission will cause the transmission to be aborted and will reset the transmitter. As a result, the UxTX pin will revert to a high-impedance state.

To select the 9-bit transmission, the PDSEL<1:0> bits (UxMODE<2:1>) should be set to '`11`' and the ninth bit should be written to the UTX8 bit (UxTXREG<8>). A word write should be performed to UxTXREG, so that all nine bits are written at the same time.

Note: There is no parity in case of a 9-bit data transmission.

On a device reset, the UxTX pin is configured as an input; therefore, the state of the UxTX pin is undefined. When the UART module is enabled, the transmit pin is driven high. It remains in this state until data is written to the transmit buffer (UxTXREG). The transmit pin is driven low as soon as the first data is written to the UxTXREG register. To ensure the Start bit detection, it is recommended to have a delay between enabling the UARTx (UARTEN = `1`) and initiating the first transmission. The delay is baud rate dependent and should be equal to or longer than the time it takes to transmit one data bit.

Figure 17-4: UART Transmission



1. The UART module is enabled (UARTEN = `1`).
2. Data is written to the transmit buffer (UxTXREG) to begin the transmission.

17.5.1 Transmit Buffer (UxTXREG)

The transmit buffer is 9 bits wide and 4 levels deep. Along with the Transmit Shift register (UxTSR), the user effectively has a 5-level deep buffer. It is organized as a First In First Out (FIFO). Once the UxTXREG contents are transferred to the UxTSR register, the current buffer location becomes available for new data to be written and the next buffer location is sourced to the UxTSR register. The UTXBF (UxSTA<9>) status bit is set whenever the buffer is full. If a user application attempts to write to a full buffer, the new data will not be accepted into the FIFO.

The FIFO is reset during any device Reset, but is not affected when the device enters a power-saving mode or wakes up from a power-saving mode.

17.5.2 Transmit Interrupt

The Transmit Interrupt Flag bit (UxTXIF) is located in the corresponding Interrupt Flag Status register (IFS). The UTXISEL<1:0> control bits (UxSTA<15:13>) determine when the UART will generate a transmit interrupt.

- If UTXISEL<1:0> = `00`, the UxTXIF bit is set, when a character is transferred from the transmit buffer to the Transmit Shift register (UxTSR), or the transmit buffer is empty. This implies at least one location is empty in the transmit buffer.
- If UTXISEL<1:0> = `01`, the UxTXIF bit is set, when the last character is shifted out of the UxTSR register, or the transmit buffer is empty. This implies that all the transmit operations are completed.
- If UTXISEL<1:0> = `10`, the UxTXIF bit is set when the character is transferred to the UxTSR register and the transmit buffer is empty.

The UxTXIF bit is set when the module is first enabled. The user application should clear the UxTXIF bit in the ISR.

Switching between the two interrupt modes during operation is possible.

Note:	When the UTXEN bit is set, the UxTXIF flag bit will also be set regardless of the UTXISEL bits (UxSTA<15,13>) settings.

While the UxTXIF flag bit indicates the status of the UxTXREG register, the TRMT bit (UxSTA<8>) indicates the status of the UxTSR. The TRMT status bit is a read-only bit, which is set when the UxTSR is empty. No interrupt logic is tied to this bit, so the user application has to poll this bit to determine if the UxTSR is empty.

17.5.3 Setup for UART Transmit

The following steps are used to set up a transmission:

1. Initialize the UxBRG register for the appropriate baud rate (see **17.3 "UART Baud Rate Generator"**).
2. Set the number of data bits, number of Stop bits, and parity selection by writing to the PDSEL<1:0> (UxMODE<2:1>) and STSEL (UxMODE<0>) bits.
3. If transmit interrupts are desired, set the UxTXIE control bit in the corresponding Interrupt Enable Control register (IEC).

   Specify the interrupt priority for the transmit interrupt using the UxTXIP<2:0> control bits in the corresponding Interrupt Priority Control register (IPC). Also, select the Transmit Interrupt mode by writing the UTXISEL<1:0> bits (UxSTA<15:13>).
4. Enable the UART module by setting the UARTEN bit (UxMODE<15>).
5. Enable the transmission by setting the UTXEN bit (UxSTA<10>), which also sets the UxTXIF bit.

   The UxTXIF bit should be cleared in the software routine that services the UART transmit interrupt. The operation of the UxTXIF bit is controlled by the UTXISEL<1:0> control bits.
6. Load data into the UxTXREG register (starts transmission).

   If 9-bit transmission has been selected, load a word. If 8-bit transmission is used, load a byte. Data can be loaded into the buffer until the UTXBF status bit (UxSTA<9>) is set.

Note: The UTXEN bit should not be set until the UARTEN bit has been set; otherwise, UART transmissions will not be enabled.

Example 17-2 provides the sample code that sets up the UART for transmission.

Figure 17-5: Transmission (8-bit or 9-bit Data)

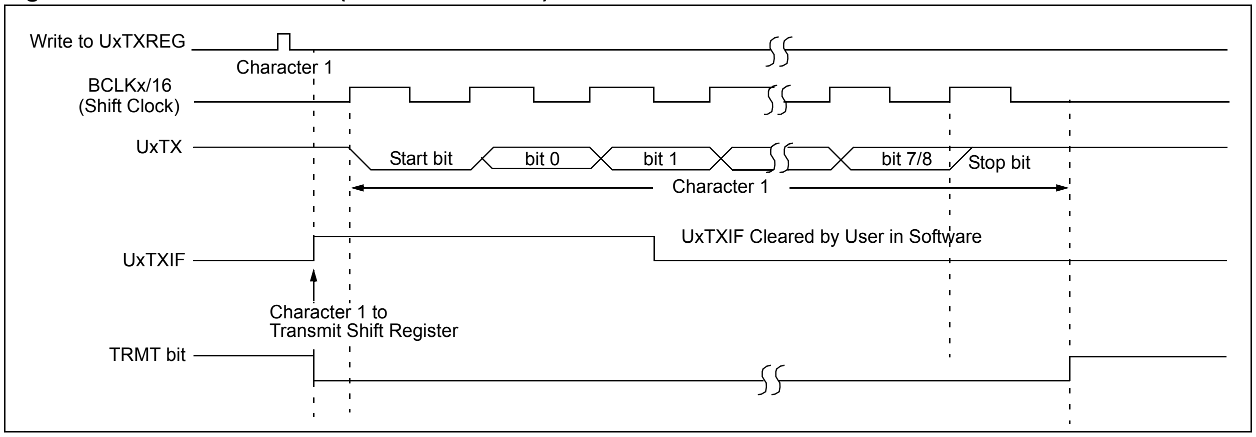
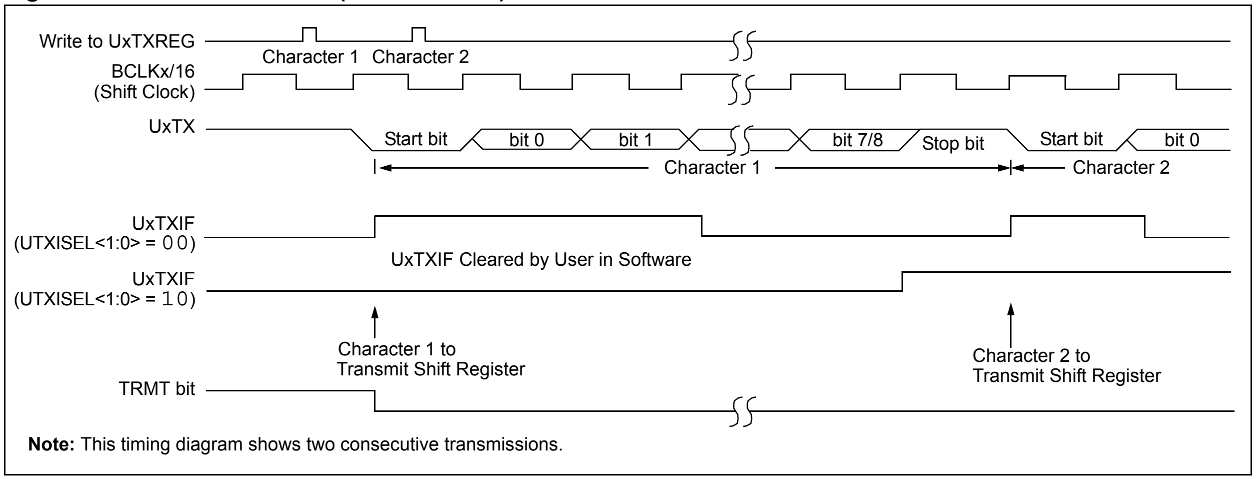


Figure 17-6: Transmission (Back-to-Back)



Example 17-2: UART Transmission with Interrupts

```
#define FCY 40000000
#define BAUDRATE 9600
#define BRGVAL ((FCY/BAUDRATE)/16)-1
unsigned int i;

int main(void)
{

    // Configure oscillator to operate the device at 40 MHz
    // Fosc = Fin * M/(N1 * N2), FCY = Fosc/2
    // Fosc = 8M*40(2*2) = 80 MHz for 8M input clock
    PLLFBD = 38;                            // M = 40
    CLKDIVbits.PLLPOST = 0;                 // N1 = 2
    CLKDIVbits.PLLPRE = 0;                  // N2 = 2
    OSCTUN = 0;                             // Tune FRC oscillator, if FRC is used
    RCONbits.SWDTEN = 0;                    // Disable Watchdog Timer

    while(OSCCONbits.LOCK! = 1) {};         // Wait for PLL to lock

    U1MODEbits.STSEL = 0;                   // 1-Stop bit
    U1MODEbits.PDSEL = 0;                   // No Parity, 8-Data bits
    U1MODEbits.ABAUD = 0;                   // Auto-Baud disabled
    U1MODEbits.BRGH = 0;                    // Low-Speed mode

    U1BRG = BRGVAL;                         // Baud Rate setting for 9600

    U1STAbits.UTXISEL0 = 0;                 // Interrupt after one TX character is transmitted
    U1STAbits.UTXISEL1 = 0;

    IEC0bits.U1TXIE = 1;                    // Enable UART TX interrupt

    U1MODEbits.UARTEN = 1;                  // Enable UART
    U1STAbits.UTXEN = 1;                    // Enable UART TX

    /* Wait at least 104 microseconds (1/9600) before sending first char */
    for(i = 0; i < 4160; i++)
    {
        Nop();
    }

    U1TXREG = 'a';                          // Transmit one character

    while(1)
    {
    }
}

void __attribute__((__interrupt__)) _U1TXInterrupt(void)
{
    IFS0bits.U1TXIF = 0;                    // Clear TX Interrupt flag
    U1TXREG = 'a';                          // Transmit one character
}
```

17.5.4 Transmission of Break Characters

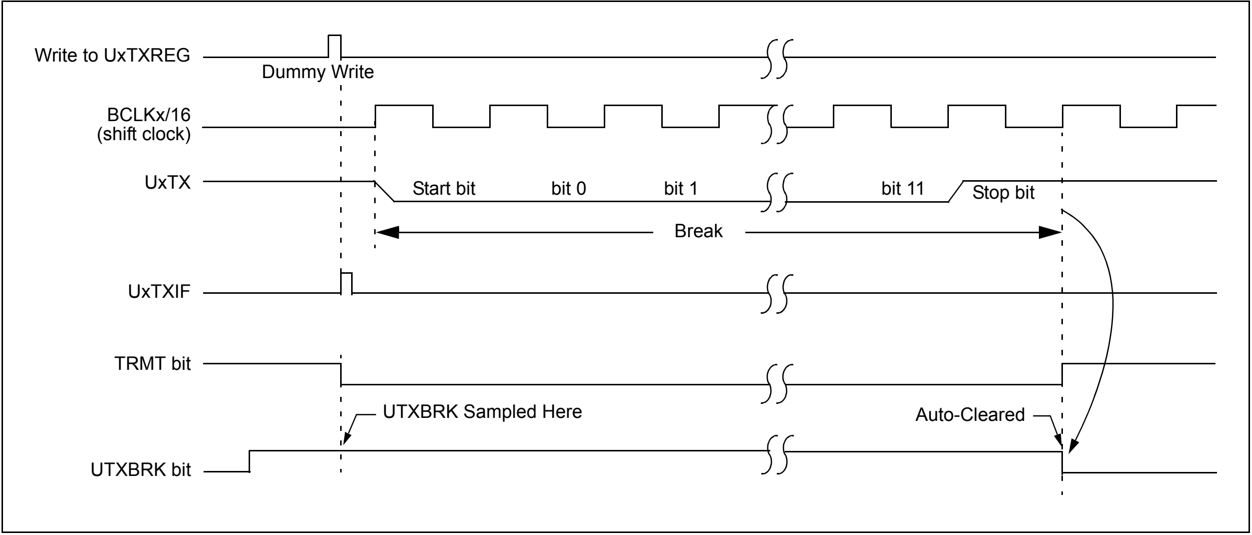
A Break character transmit consists of a Start bit, followed by twelve bits of '0' and a Stop bit. A Frame Break character is sent whenever the UTXBRK and UTXEN bits are set while the Transmit Shift register is loaded with data. A dummy write to the UxTXREG register is necessary to initiate the Break character transmission. Note that the data value written to the UxTXREG for the Break character is ignored. The write simply serves the purpose of initiating the proper sequence – all '0's will be transmitted.

The UTXBRK bit is automatically reset by hardware after the corresponding Stop bit is sent. This allows the user application to preload the transmit FIFO with the next transmit byte following the Break character (typically, the Sync character in the LIN specification).

Note: The user application should wait for the transmitter to be Idle (TRMT = 1) before setting the UTXBRK bit. The UTXBRK bit overrides any other transmitter activity. If the user application clears the TXBRK bit prior to sequence completion, unexpected module behavior can result. Sending a Break character does not generate a transmit interrupt.

The TRMT bit indicates when the Transmit Shift register is empty or full, just as it does during the normal transmission. See Figure 17-7 for the timing of the Break character sequence.

Figure 17-7: Send Break Character Sequence



17.5.4.1 BREAK AND SYNC TRANSMIT SEQUENCE

The following sequence sends a message frame header made up of a Break followed by an auto-baud Sync byte. This sequence is typical of a LIN bus master.

1. Configure the UART for the desired mode.
2. Set UTXEN and UTXBRK bits to transmit the Break character.
3. Load the UxTXREG with a dummy character to initiate transmission (value is ignored).
4. Write 0x55 to UxTXREG – loads Sync character into the transmit FIFO.

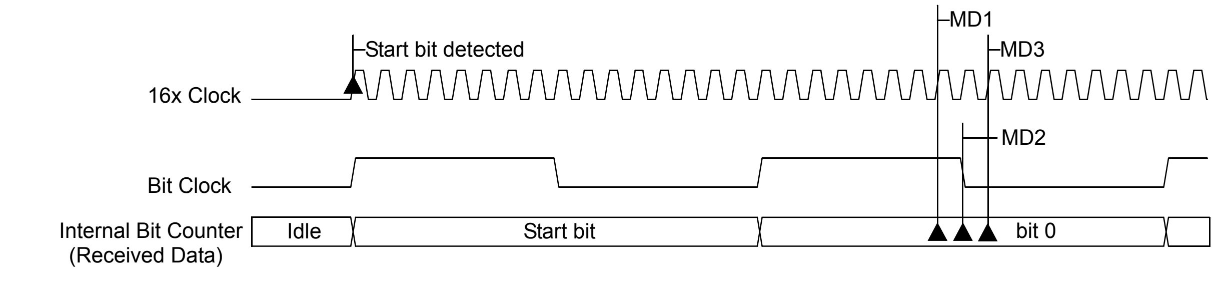
After the Break has been sent, the UTXBRK bit is reset by hardware. The Sync character is now transmitted.

17.6 DATA BIT DETECTION

17.6.1 16x Clock Mode (BRGH = 0)

In 16x Clock mode, each bit of the received data is 16 clock pulses wide. To detect the value of an incoming data bit, the bit is sampled at 7th, 8th and 9th rising edges of the clock. These rising edges are called Majority Detection Edges. This mode is more robust than 4x Clock mode.

Figure 17-8: 16x Clock Mode with Majority Detection

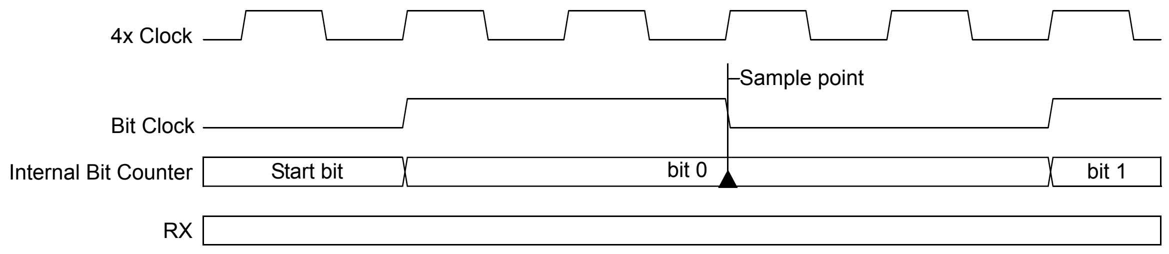


Note: In 16x Clock mode, each bit is sampled at 7th, 8th and 9th rising edges of the clock.

17.6.2 4x Clock Mode (BRGH = 1)

In 4x Clock mode, each bit of the received data is four clock pulses wide. The 4x Clock mode does not provide enough edges to support the Majority Detection Method. Therefore, the received data is sampled at the one-half bit width.

Figure 17-9: 4x Clock Mode without Majority Detection



Note: In 4x Clock mode, the sampling occurs only at the one-half bit width.

17.7 UART RECEIVER

The receiver block diagram of the UART module is illustrated in Figure 17-10. The important part of the receiver is the Receive (Serial) Shift register (UxRSR). The data is received on the UxRX pin and is sent to the data recovery block. After sampling the UxRX pin for the Stop bit, the received data in the UxRSR register is transferred to the receive FIFO (if it is empty). The data recovery block operates at 16 times the baud rate, whereas the main receive serial shifter operates at the baud rate.

Note: The UxRSR register is not mapped in the data memory, so it is not available to the user application.

The data on the UxRX pin is sampled multiple times by a majority detect circuit to determine if a high or a low level is present at the UxRX pin.

17.7.1 Receive Buffer (UxRXREG)

The UART receiver has a 4 deep, 9 bit wide FIFO receive data buffer. The UxRXREG is a memory mapped register that provides access to the output of the FIFO. It is possible for four words of data to be received and transferred to the FIFO and a fifth word to begin shifting to the UxRSR register before a buffer overrun occurs.

17.7.2 Receiver Error Handling

If the FIFO is full (four characters) and a fifth character is fully received into the UxRSR register, the OERR bit (UxSTA<1>) will be set. The word in the UxRSR register will be kept, but further transfers to the receive FIFO, and UART interrupts are inhibited as long as the OERR bit is set. The user application must clear the OERR bit in software to allow further data to be received.

If it is desired to keep the data received prior to the overrun, the user application should first read all five characters, then clear the OERR bit. If the five characters can be discarded, the user application can simply clear the OERR bit. This effectively resets the receive FIFO, and all prior received data is lost.

Note: The data in the receive FIFO should be read prior to clearing the OERR bit. The FIFO is reset when OERR is cleared, which causes all data in the buffer to be lost.

The Framing Error bit, FERR (UxSTA<2>), is set if a Stop bit is detected at a logic low level.

The Parity Error bit, PERR (UxSTA<3>), is set if a parity error has been detected in the data word at the top of the buffer (i.e., the current word). For example, a parity error would occur if the parity is set to be Even, but the total number of '`1`'s in the data has been detected to be odd. The PERR bit is irrelevant in the 9-bit mode. The FERR and PERR bits are buffered along with the corresponding word and should be read before reading the data word.

An interrupt is generated if any of these (OERR, FERR and PERR) errors occur. The user application needs to enable the corresponding Interrupt Enable Control bit (UxERIE) in the IEC4 register to go to the corresponding interrupt vector location.

17.7.3 Receive Interrupt

The UART Receive Interrupt Flag bit (UxRXIF) is located in the corresponding Interrupt Flag Status register (IFS). The URXISEL<1:0> control bits (UxSTA<7:6>) determine when the UART receiver generates an interrupt.

- If URXISEL<1:0> = `00` or `01`, an interrupt is generated each time a data word is transferred from the Receive Shift register (UxRSR) to the receive buffer. There may be one or more characters in the receive buffer.
- If URXISEL<1:0> = `10`, an interrupt is generated when a word is transferred from the UxRSR register to the receive buffer, and as a result, the receive buffer contains three or four characters.
- If URXISEL<1:0> = `11`, an interrupt is generated when a word is transferred from the UxRSR register to the receive buffer, and as a result, the receive buffer contains four characters (i.e., becomes full).

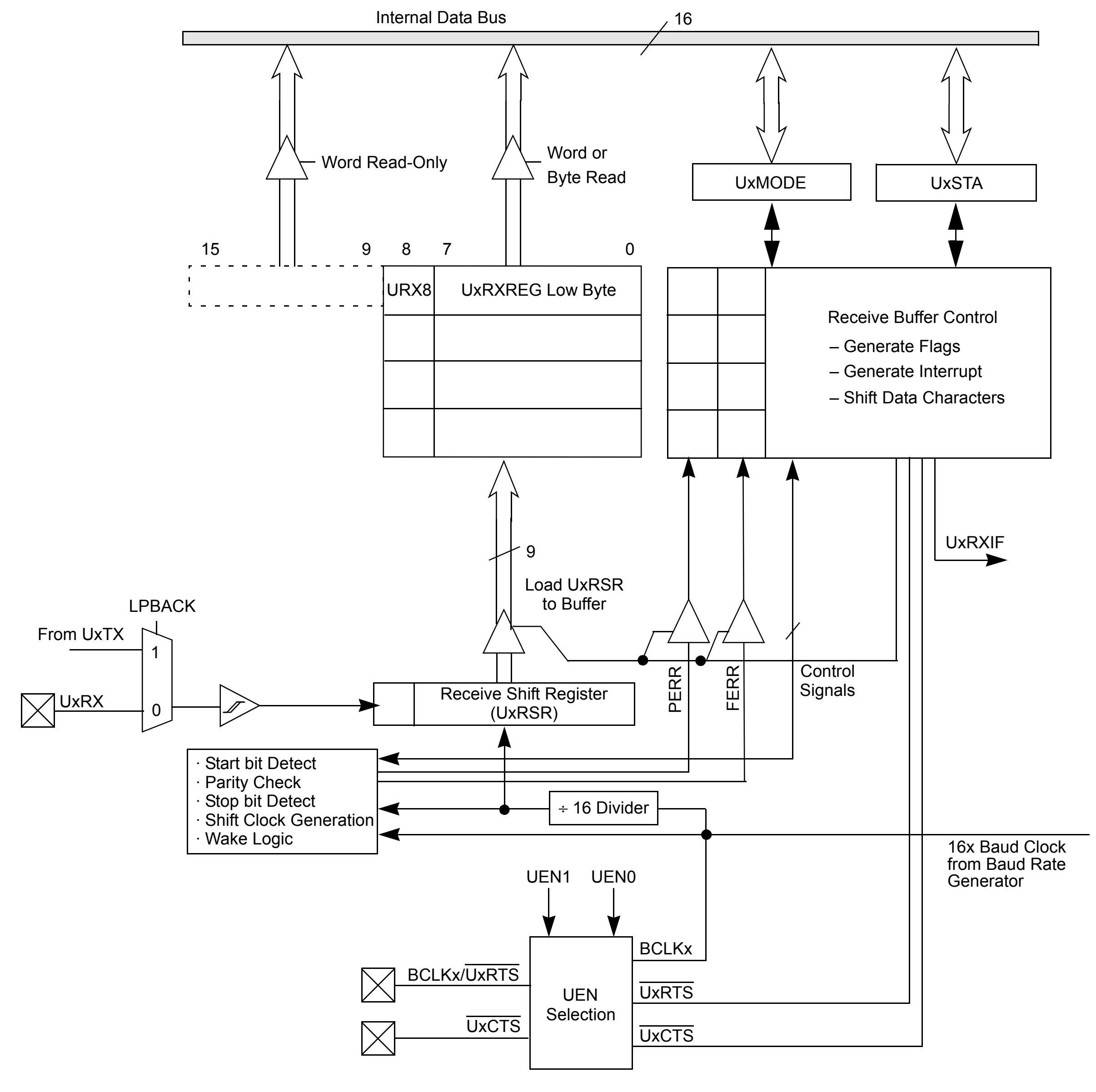
Switching between the three Interrupt modes during operation is possible.

While the URXDA and UxRXIF flag bits indicate the status of the UxRXREG register, the RIDLE bit (UxSTA<4>) shows the status of the UxRSR register. The RIDLE status bit is a read-only bit, and is set when the receiver is Idle (i.e., the UxRSR register is empty). No interrupt logic is tied to this bit, so the user application needs to poll this bit in order to determine if the UxRSR register is Idle.

The URXDA bit (UxSTA<0>) is a read-only bit, which indicates whether the receive buffer has data or is empty. This bit is set as long as there is at least one character to be read from the receive buffer.

Figure 17-10 illustrates a block diagram of the UART receiver.

Figure 17-10: UART Receiver Block Diagram



Note: The ‘x’ denotes the UART number.

17.7.4 Setup for UART Reception

The following steps are used to set up a UART reception:

1. Initialize the UxBRG register for the appropriate baud rate (see **17.3 "UART Baud Rate Generator"**).
2. Set the number of data bits, number of Stop bits, and parity selection by writing to the PDSEL<1:0> (UxMODE<2:1>) and STSEL (UxMODE<0>) bits.
3. If interrupts are desired, set the UxRXIE bit in the corresponding Interrupt Enable Control register (IECx).

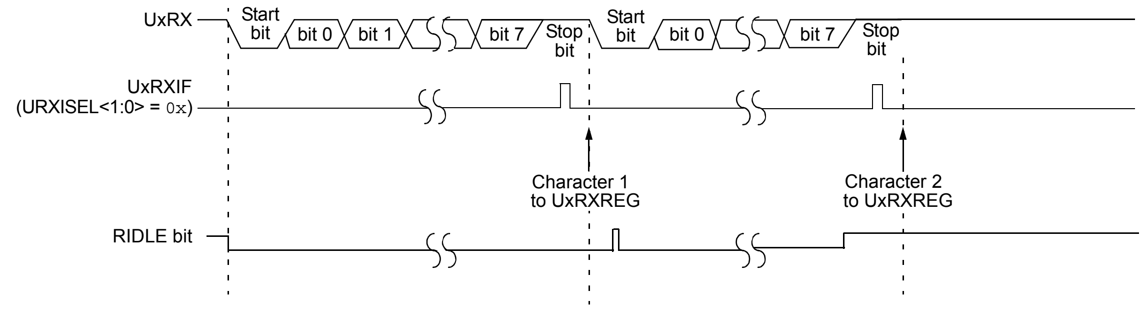
   Specify the interrupt priority for the interrupt using the UxRXIP<2:0> control bits in the corresponding Interrupt Priority Control register (IPC). Also, select the Receive Interrupt mode by writing to the URXISEL<1:0> bits (UxSTA<7:6>).
4. Enable the UART module by setting the UARTEN bit (UxMODE<15>).
5. Receive interrupts will depend on the URXISEL<1:0> control bit settings.

   If receive interrupts are not enabled, the user application can poll the URXDA bit. The UxRXIF bit should be cleared in the software routine that services the UART receive interrupt.
6. Read data from the receive buffer.

   If 9-bit transmission is selected, read a word; otherwise, read a byte. The URXDA status bit (UxSTA<0>) is set whenever the data is available in the buffer.

Example 17-3 provides the sample code that sets up the UART for reception.

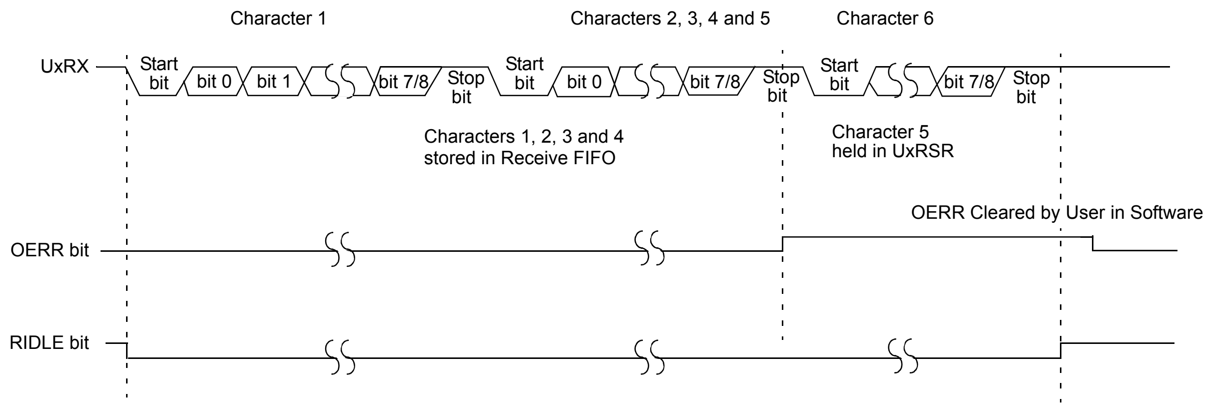
Figure 17-11: UART Reception



Note 1: This timing diagram shows two characters received on the UxRX input.

2: If the interrupt flag is used by the application software as a basis for disabling the UART transmission, the software should wait for 1-bit time before disabling the transmission.

Figure 17-12: UART Reception with Receive Overrun



Note: This diagram shows 6 characters received without the user reading the input buffer. The 5th character received is held in the Receive Shift register. An overrun error occurs at the start of the 6th character.

Example 17-3: UART Receive Only with Polling (Interrupts Disabled)

```
#define FCY 40000000
#define BAUDRATE 9600
#define BRGVAL ((FCY/BAUD RATE)/16) - 1

int main(void)
{

// Configure oscillator to operate the device at 40 MHz
// Fosc = Fin * M/(N1 * N2), FCY = Fosc/2
// Fosc = 8M*40(2*2) = 80 MHz for 8M input clock
PLLFBD = 38;                          // M = 40
CLKDIVbits.PLLPOST = 0;               // N1 = 2
CLKDIVbits.PLLPRE = 0;                // N2 = 2
OSCTUN = 0;                           // Tune FRC oscillator, if FRC is used

RCONbits.SWDTEN = 0;                  // Disable Watchdog Timer

while(OSCCONbits.LOCK! = 1) {};       // Wait for PLL to lock

U1MODEbits.STSEL = 0;                 // 1-stop bit
U1MODEbits.PDSEL = 0;                 // No Parity, 8-data bits
U1MODEbits.ABAUD = 0;                 // Auto-Baud disabled
U1MODEbits.BRGH = 0;                  // Low-Speed mode

U1BRG = BRGVAL;                       // Baud Rate setting for 9600

U1STAbits.URXISEL = 0;                // Interrupt after one RX character is received;

U1MODEbits.UARTEN = 1;                // Enable UART

while(1)
{
    char ReceivedChar;

    /* Check for receive errors */
    if(U1STAbits.FERR == 1)
    {
        continue;
    }

    /* Must clear the overrun error to keep UART receiving */
    if(U1STAbits.OERR == 1)
    {
        U1STAbits.OERR = 0;
        continue;
    }

    /* Get the data */
    if(U1STAbits.URXDA == 1)
    {
        ReceivedChar = U1RXREG;
    }
}
```

17.8 USING THE UART FOR 9-BIT COMMUNICATION

The UART receiver can be used in 9-bit Data mode for multiprocessor communication. With the ADDEN bit set in 9-bit Data mode, the receiver can ignore the data when the 9th bit of the data is '`0`'. This feature can be used in a multiprocessor environment.

17.8.1 Multiprocessor Communications

A typical multiprocessor communication protocol will differentiate between the data bytes and the address/control bytes. A common scheme is to use a 9th data bit to identify whether a data byte is address or data information. If the 9th bit is set, the data is processed as address or control information. If the 9th bit is cleared, the received data word is processed as data associated with the previous address/control byte.

The protocol operates as follows:

- The master device transmits a data word with the 9th bit set. The data word contains the address of a slave device.
- All slave devices in the communication chain receive the address word and check the slave address value.
- The slave device that was addressed will receive and process subsequent data bytes sent by the master device. All other slave devices will discard subsequent data bytes until a new address word (9th bit set) is received.

17.8.2 ADDEN Control Bit

The UART receiver has an Address Detect mode, which allows it to ignore data words with the 9th bit cleared. This reduces the interrupt overhead, because the data words with the 9th bit cleared are not buffered. This feature is enabled by setting the ADDEN bit (UxSTA<5>).

The UART must be configured for 9-bit Data mode to use the Address Detect mode. The ADDEN bit has no effect when the receiver is configured in 8-bit Data mode.

17.8.3 Setup for 9-bit Transmit

The setup procedure for 9-bit transmission is identical to the procedure for 8-bit Transmit modes, except that the PDSEL<1:0> bits (UxMODE<2:1>) should be set to '`11`' (see **17.5.3 "Setup for UART Transmit"**).

Word writes should be performed to the UxTXREG register (starts transmission).

17.8.4 Setup for 9-bit Reception Using Address Detect Mode

The setup procedure for 9-bit reception is similar to the procedure for 8-bit Receive modes, except that the PDSEL<1:0> bits (UxMODE<2:1>) should be set to '11' (see **17.7.4 "Setup for UART Reception"**).

The Receive Interrupt mode should be configured by writing to the URXISEL<1:0> and (UxSTA<7:6>) bits.

Note: If the Address Detect mode is enabled (ADDEN = 1), the URXISEL<1:0> control bits should be configured so that an interrupt will be generated after every received word. Each received data word must be checked in software for an address match immediately after reception.

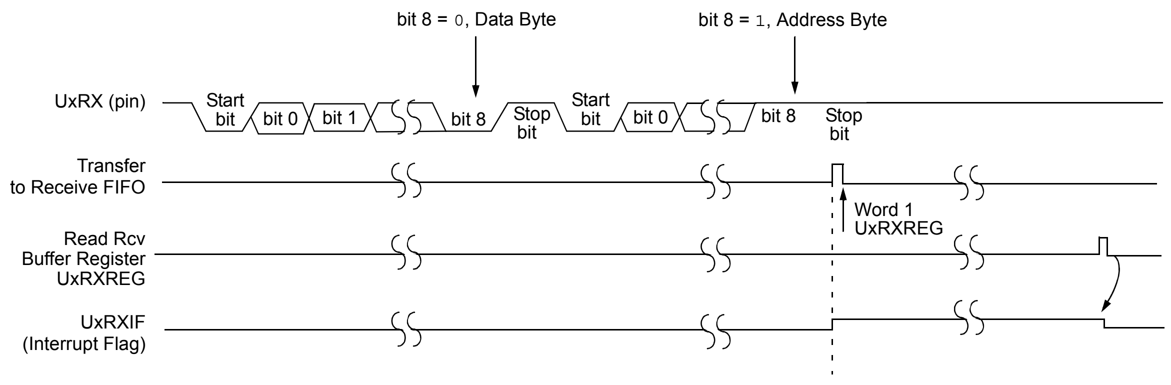
The procedure for using the Address Detect mode is as follows:

1. Set the ADDEN bit (UxSTA<5>) to enable address detect. Ensure that the URXISEL control bits are configured to generate an interrupt after each received word.
2. Check each 8-bit address by reading the UxRXREG register to determine if the device is being addressed.
3. If this device has not been addressed, discard the received word.
4. If this device has been addressed, clear the ADDEN bit to allow subsequent data bytes to be read into the receive buffer and interrupt the CPU.

   If a long data packet is expected, the Receive Interrupt mode can be changed to buffer more than one data byte between the interrupts.
5. After receiving the last data byte, set the ADDEN bit so that only address bytes are received.

   Also, ensure that the URXISEL control bits are configured to generate an interrupt after each received word.

Figure 17-13: Reception with Address Detect (ADDEN = 1)



Note: This timing diagram shows a data byte followed by an address byte. The data byte is not read into the UxRXREG (receive buffer) because ADDEN = 1 and bit 8 = 0.

17.9 OTHER FEATURES OF THE UART

17.9.1 UART in Loopback Mode

Setting the LPBACK bit (UxMODE<6>) enables the Loopback mode, in which the UxTX output is internally connected to the UxRX input. When configured for the Loopback mode, the UxRX pin is disconnected from the internal UART receive logic. However, the UxTX pin still functions normally.

To select Loopback mode, follow these steps:

1. Configure the UART module for the desired mode of operation.
2. Enable transmission as defined in **17.5 “UART Transmitter”**.
3. Set the LPBACK bit = `1` to enable Loopback mode.

The Loopback mode is dependent on the UEN<1:0> bits, as shown in Table 17-1.

Table 17-1: Loopback Mode Pin Function

UEN<1:0>	Pin Function, LPBACK = `1`(1)
`00`	UxRX input connected to UxTX; UxTX pin functions; UxRX pin ignored; $\overline{\text{UxCTS}}$/$\overline{\text{UxRTS}}$ unused
`01`	UxRX input connected to UxTX; UxTX pin functions; UxRX pin ignored; $\overline{\text{UxRTS}}$ pin functions; $\overline{\text{UxCTS}}$ unused
`10`	UxRX input connected to UxTX; UxTX pin functions; UxRX pin ignored; $\overline{\text{UxRTS}}$ pin functions; $\overline{\text{UxCTS}}$ input connected to $\overline{\text{UxRTS}}$; $\overline{\text{UxCTS}}$ pin ignored
`11`	UxRX input connected to UxTX; UxTX pin functions; UxRX pin ignored; BCLKx pin functions; $\overline{\text{UxCTS}}$/$\overline{\text{UxRTS}}$ unused

Note 1: The LPBACK bit should be set to ‘`1`’ only after enabling the other bits associated with the UART module.

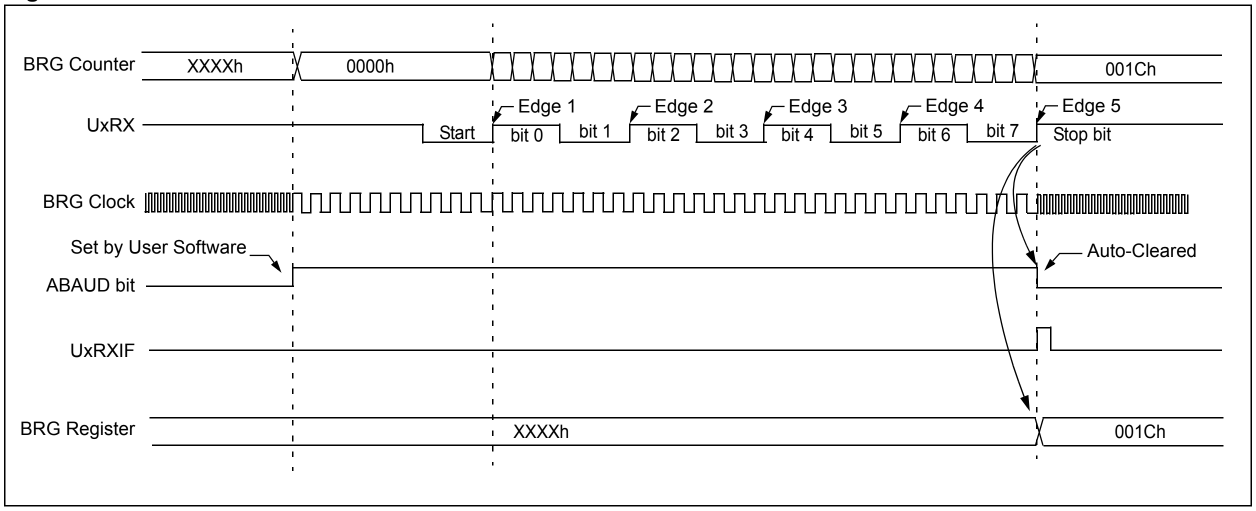
17.9.2 Auto-Baud Support

To allow the system to determine baud rates of the received characters, the ABAUD bit (UxMODE<5>) is enabled. The UART begins an automatic baud rate measurement sequence whenever a Start bit is received while the Auto-Baud Rate Detect is enabled (ABAUD = `1`). The calculation is self-averaging. Once the ABAUD bit is set, the BRG counter value will be cleared and will look for a Start bit, which in this case, is defined as a high-to-low transition followed by a low-to-high transition.

Following the Start bit, the auto-baud expects to receive an ASCII “U” (“55h”) in order to calculate the proper bit rate. The measurement is taken over both the low and the high bit time in order to minimize any effects caused by asymmetry of the incoming signal. On the 5th UxRX pin rising edge, an accumulated BRG counter value totalling the proper BRG period is transferred to the UxBRG register. The ABAUD bit is automatically cleared. If the user application clears the ABAUD bit prior to sequence completion, unexpected module behavior can result. See Figure 17-14 for the Auto-Baud Rate Detection sequence.

The ABAUD feature of a respective UART automatically uses the corresponding ICx peripheral for UARTx. The user application should not use the respective ICx peripheral if the ABAUD feature is used by the UARTx.

Figure 17-14: Automatic Baud Rate Calculation



While the auto-baud sequence is in progress, the UART state machine is held in an Idle state. The UxRXIF interrupt is set on the 5th UxRX rising edge, independent of the URXISEL<1:0> bits setting. The receiver FIFO is not updated.

17.10 UART OPERATION WITH DMA

On some dsPIC33E/PIC24E devices, the Direct Memory Access (DMA) module can be used to transfer data between the CPU and UART without CPU assistance. Refer to the specific device data sheet to see if DMA is present on your particular device. For more information on the DMA module, refer to **Section 22. “Direct Memory Access (DMA)”** (DS70348) in the “*dsPIC33E/PIC24E Family Reference Manual”*.

17.10.1 UART Receive with DMA

If the DMA channel is associated with the UART receiver, the UART should issue a DMA request every time there is a character ready to be moved from the UART module to RAM. The DMA transfers data from the UxRXREG register into RAM and issues a CPU interrupt after a predefined number of transfers. As the DMA channels are unidirectional, the UART receive operation would require one DMA channel. The DMA channel for UART reception should be configured as shown in Table 17-2.

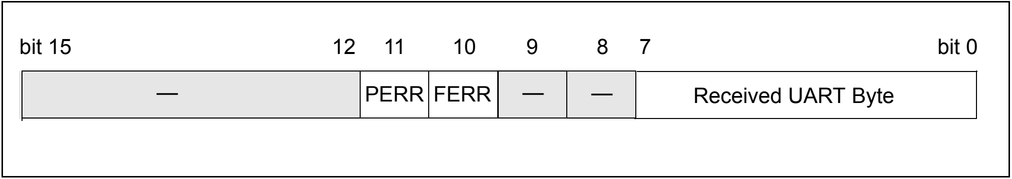
Table 17-2: DMA Channel Register Initialization for UART to DMA Association (UART Reception)

Peripheral to DMA Association	DMAxREQ Register IRQSEL<7:0> bits	DMAxPAD Register Values to Read from Peripheral
UART1RX – UART1 Receiver	00001011	0x0226 (U1RXREG)
UART2RX – UART2 Receiver	00011110	0x0236 (U2RXREG)
UART3RX – UART3 Receiver	01010010	0x0256 (U3RXREG)
UART4RX – UART4 Receiver	01011000	0x02B6 (U4RXREG)

The UART module must be configured to generate interrupts for every character received. For the UART receiver to generate an RX interrupt for each character received, the Receive Interrupt Mode Selection bits (URXISEL<1:0>) must be set to ‘00’ or ‘01’ in the Status and Control register (UxSTA). When the UART and DMA channel are properly configured, the UART receiver issues a DMA request as soon as data is received. No special steps need to be taken by the user application to initiate a DMA transfer.

While configuring for UART reception, the word size of the DMA channel should be set to 16-bit. This configures the DMA channel to read 16 bits from the UART module when data is available to be read. The lower byte of the data represents the actual data byte received by the UART module. The upper byte contains the UART status when the byte was received. Note that the reading of the UxSTA register when the UART reception is DMA enabled will not return the status of the FERR and PERR bits. This status is available in the upper byte of the 16-bit word that the DMA channel transfers from the UART module to DMA RAM. Figure 17-5 illustrates the organization of the 16-bit word transferred by DMA from UART module to the DMA RAM.

Figure 17-15: Format of 16-bit UART Receive Data Word Transferred by DMA to RAM



The DMAxPAD register should point to the UxRXREG register. The UART Error Interrupt Flag bit (UxEIF) gets set if the last UART reception caused a framing or a parity error. Setting the UxEIE bit causes the CPU to enter the UART error interrupt service routine. The user application can then inspect the upper byte of the last transferred word to check which error condition has caused the interrupt.

17.10.2 UART Transmit with DMA

If the DMA channel is associated with the UART transmitter, the UART issues a DMA request after each successful transmission. After each DMA request, the DMA transfers new data into the UxTXREG register and issues a CPU interrupt after a predefined number of transfers. As the DMA channels are unidirectional, one DMA channel is required for transmit. Each DMA channel must be initialized as shown in Table 17-3.

Table 17-3: DMA Channel Register Initialization for UART to DMA Association (UART Transmission)

Peripheral to DMA Association	DMAxREQ Register IRQSEL<7:0> bits	DMAxPAD Register Values to Write from Peripheral
UART1TX – UART1 Transmitter	`00001100`	0x0224 (U1TXREG)
UART2TX – UART2 Transmitter	`00011111`	0x0234 (U2TXREG)
UART3TX – UART3 Transmitter	`01010011`	0x0254 (U3TXREG)
UART4TX – UART4 Transmitter	`01011001`	0x02B4 (U4TXREG)

In addition, the UART must be configured to generate interrupts for every character transmitted. For the UART transmitter to generate a TX interrupt for each character transmitted, the Transmission Interrupt Mode Selection bits (UTXISEL0 and UTXISEL1) must be set to ‘`0`’ in the UxSTA register.

The UART transmitter issues a DMA request as soon as the UART and transmitter are enabled. This means that the DMA channel and buffers must be initialized and enabled before the UART and transmitter are enabled. Alternatively, the UART and UART transmitter can be enabled before the DMA channel is enabled. In this case, the UART transmitter DMA request will be lost, and the user application must issue a DMA request to start DMA transfers by setting the FORCE bit in the DMAxREQ register.

17.10.3 UART DMA Configuration Example

Example 17-4 provides a sample code for UART reception and transmission with the help of two DMA channels. The UART receives and buffers characters from the HyperTerminal at 9600 bps. After receiving eight characters, the UART transmits (echoes) them back to the HyperTerminal.

DMA Channel 0 is configured for UART transmission with the following configuration:

- Transfer data from RAM to UART
- One-Shot mode
- Register Indirect with Post-Increment
- Using single buffer
- Eight transfers per buffer
- Transfer words

DMA Channel 1 is configured for UART reception with the following configuration:

- Continuously transfer data from UART to RAM
- Register indirect with post-increment
- Using two ping-pong buffers
- Eight transfers per buffer
- Transfer words

Example 17-4: UART Reception and Transmission with DMA

Set Up the UART for RX and TX:

```
#include<p33EP512MU810.h>
#define FCY 40000000
#define BAUDRATE 9600
#define BRGVAL ((FCY/BAUDRATE)/16) - 1

// Protection disabled
_FGS(GWRP_OFF & GSS_OFF & GSSK_OFF);

// Start with FRC will switch to Primary Oscillator (XT, HS, EC) with PLL
_FOSCSEL (FNOSC_FRC);

// Clock Switching is enabled and Fail-Safe Clock Monitor is disabled

// Primary Oscillator Mode: XT Crystal
_FOSC (FCKSM_CSECMD & POSCMD_XT & OSCIOFNC_OFF & IOL1WAY_OFF);

// Turn off Watchdog Timer
_FWDT (FWDTEN_OFF);

// Alternate I2C Mapped to SDA1/SCL1

// FPOR power-on reset 128 ms
_FPOR (FPWRT_PWR128 & BOREN_ON & ALTI2C1_OFF & ALTI2C2_OFF);

// Use PGC1/PGD1 for programming and debugging
_FICD (ICS_PGD1 & RSTPRI_PF & JTAGEN_OFF);

unsigned int i = 0;

__eds__ unsigned int BufferA[8] __attribute__((space(eds),address(0xD000)));
__eds__ unsigned int BufferB[8] __attribute__((space(eds),address(0xD100)));
unsigned long address, addressb;

int main(void)
{
    // The settings below set up the oscillator and PLL for 40 MIPS:

    //            Crystal Frequency * (DIVISOR + 2)
    // FCY =       --------------------------------
    //              PLLPOST * (PRESCLR + 2) * 4

    // Crystal = 8 MHz
    // FOSC = 120 MHz
    // FCY = 60 MIPS

    PLLFBD = 38;                  /* M = 40 */

    CLKDIVbits.PLLPOST = 0;       /* N1 = 2 */

    CLKDIVbits.PLLPRE = 0;        /* N2 = 2 */
```

Example 17-4: UART Reception and Transmission with DMA (Continued)

```
    OSCTUN = 0;
    __builtin_write_OSCCONH(0x03);    // Initiate Clock Switch to Primary Oscillator with PLL
                                      // (NOSC = 0b011) */
    __builtin_write_OSCCONL(0x01);

    while (OSCCONbits.COSC! = 0b011);
    while(OSCCONbits.LOCK! = 1) {};           // Wait for PLL to lock

    _RP101R = 3;
    RPINR19 = 0;
    _U2RXR = 100;

    U2MODEbits.STSEL = 0;                     // 1-stop bit
    U2MODEbits.PDSEL = 0;                     // No Parity, 8-data bits
    U2MODEbits.ABAUD = 0;                     // Auto-Baud Disabled
    U2BRG = BRGVAL;                           // Baud Rate Setting for 9600
    U2STAbits.UTXISEL0 = 0;                   // Interrupt after one TX character is transmitted
    U2STAbits.UTXISEL1 = 0;
    U2STAbits.URXISEL = 0;                    // Interrupt after one RX character is received
    U2MODEbits.UARTEN = 1;                    // Enable UART
    IFS1bits.U2TXIF = 0;                      // Clear Interrupt Flag
    U2STAbits.UTXEN = 1;                      // Enable UART TX

    /* Wait at least 104 microseconds (1/9600) before sending first char */
    for(i = 0; i < 4160; i++)
    {
       Nop();
    }

    _U2EIF = 0;                               // Clear UART2 Error Interrupt Flag
    _U2EIE = 1;                               // Enable UART2 Error Interrupt Flag

    // Set Up DMA Channel 0 to Transmit in One-Shot, Single-Buffer Mode:

    DMA0CON = 0x2001;                         // One-Shot, Post-Increment, RAM-to-Peripheral
    DMA0CNT = 7;                              // 8 DMA Requests
    DMA0REQ = 0x001F;                         // Select UART2 Transmitter
    DMA0PAD = (volatile unsigned int) &U2TXREG;

    address = __builtin_edsoffset(&BufferA);
    address = address & 0x7FFF;
    address + = __builtin_edspage(&BufferA) << 15;   // Put the page in the upper portion of
                                                    // the address (starting at bit 15)
    DMA0STAL = address & 0xFFFF;
    DMA0STAH = address >> 16;

    IFS0bits.DMA0IF = 0;                      // Clear DMA Interrupt Flag
    IEC0bits.DMA0IE = 1;                      // Enable DMA Interrupt

    // Set Up DMA Channel 1 to Receive in Continuous Ping-Pong Mode:

    DMA1CON = 0x0002;                         // Continuous, Ping-Pong, Post-Inc., Periph-RAM
    DMA1CNT = 7;                              // 8 DMA Requests
    DMA1REQ = 0x001E;                         // Select UART2 Receiver
    DMA1PAD = (volatile unsigned int) &U2RXREG;

    // Put the page in the upper portion of the address (starting at bit 15)
    DMA1STAL = address & 0xFFFF;
    DMA1STAH = address >> 16;

    addressb = __builtin_edsoffset(&BufferB);
    addressb = addressb & 0x7FFF;
    addressb + = __builtin_edspage(&BufferB) << 15;  // Put the page in the upper portion of
                                                     // the address (starting at bit 15)
```

Example 17-4: UART Reception and Transmission with DMA (Continued)

```
    DMA1STBL = addressb & 0xFFFF;
    DMA1STBH = addressb >> 16;
    IFS0bits.DMA1IF = 0;                  // Clear DMA Interrupt
    IEC0bits.DMA1IE = 1;                  // Enable DMA Interrupt
    DMA1CONbits.CHEN = 1;                 // Enable DMA Channel

    while(1);
}

// Set Up DMA Interrupt Handlers:

void __attribute__((interrupt, no_auto_psv)) _DMA0Interrupt(void)
{
    IFS0bits.DMA0IF = 0;                  // Clear the DMA0 Interrupt Flag
}

void __attribute__((interrupt, no_auto_psv)) _DMA1Interrupt(void)
{
    static unsigned int BufferCount = 0;  // Keep record of which buffer contains RX Data
    if(BufferCount == 0)
    {
        DMA0STAL = address & 0xFFFF;      // Point DMA0 to data to be transmitted
    }
    else
    {
        DMA0STAL = addressb & 0xFFFF;     // Point DMA0 to data to be transmitted
    }

    DMA0CONbits.CHEN = 1;                 // Enable DMA0 Channel
    DMA0REQbits.FORCE = 1;                // Manual mode: Kick-start the 1st transfer
    BufferCount ^= 1;
    IFS0bits.DMA1IF = 0;                  // Clear the DMA1 Interrupt Flag
}

void __attribute__((__interrupt__,no_auto_psv)) _U2EInterrupt(void)
{
    // An error has occurred on the last reception. Check the last received word.

    _U2EIF = 0;
    int lastWord;

    // Check which DMA Ping-Pong channel was selected.

    if(DMAPPSbits.PPST1 == 0)
    {
        // Get the last word received from ping-pong buffer A.
        lastWord = *(unsigned int *)(DMA1STAL);
    }
    else
    {
        // Get the last word received from ping-pong buffer B.
        lastWord = *(unsigned int *)(DMA1STBL);
    }
    //Check for Parity Error
    if((lastWord & 0x800) != 0)
    {
        // There was a parity error.
        // Do something about it here.
    }
    // Check for framing error
    if ((lastWord & 0x400) != 0)
    {
        // There was a framing error.
        // Do some thing about it here.
    }
}
```

17.11 UART OPERATION DURING CPU SLEEP AND IDLE MODES

17.11.1 UART Operation in Sleep Mode

When the device enters Sleep mode, all clock sources supplied to the UART module are shut down and stay at logic '0'. If the device enters Sleep mode in the middle of a UART transmission or reception operation, the operation is aborted and the UART pins (BCLKx, $\overline{\text{UxRTS}}$ and UxTX) are driven to the default state.

A Start bit, when detected on the UART Receive pin (UxRX), can wake-up the device from Sleep mode if the WAKE bit (UxMODE<7>) is set just before the device enters Sleep mode. In this mode, if the UART receive interrupt (UxRXIE) is enabled, a falling edge on the UART Receive pin generates a UART receive interrupt (UxRXIF).

The receive interrupt wakes up the device from Sleep, and the following events occur:

- If the assigned priority for the interrupt is less than, or equal, to the current CPU priority, the device wakes up and continues code execution from the instruction following the PWRSAV instruction that initiated Sleep mode.
- If the assigned priority level for the interrupt source is greater than the current CPU priority, the device wakes up and the CPU execution process begins. Code execution continues from the first instruction of the captured ISR.
- The UART does not recognize the first character received after a wake from sleep, due to the delay in restoration of the clocks after the oscillator restart.

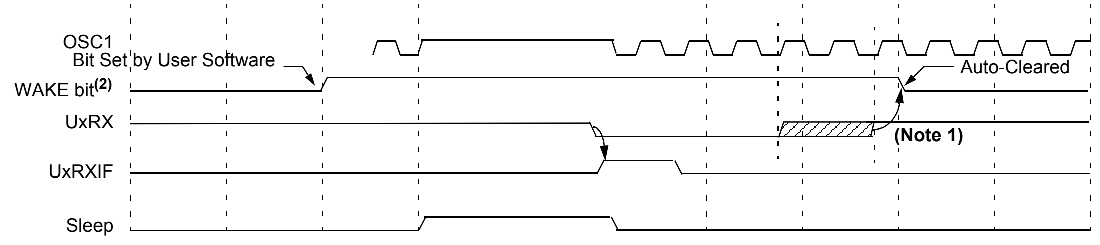
The WAKE bit is automatically cleared when a low-to-high transition is observed on the UxRX line following the wake-up event.

17.11.2 UART Operation in Idle Mode

- When the device enters Idle mode, the system clock sources remain functional, but the CPU stops code execution. The UART Stop in Idle bit (USIDL) in the UART Mode register (UxMODE<13>) determines whether the module stops in Idle mode or continues to operate in Idle mode.
- If USIDL = `0`, the module continues to operate in Idle mode and provides full functionality.
- If USIDL = `1`, the module stops in Idle mode. The module performs the same functions when stopped in Idle mode as in Sleep mode (refer to **17.11.1 "UART Operation in Sleep Mode"**).

Note 1:	The Sync Break (or Wake-up Signal) character must be of sufficient length to allow enough time for the selected oscillator to start and provide proper initialization to the UART. To ensure that the UART woke up in time, the user application should read the value of the WAKE bit (UxMODE<7>). If it is clear, it is possible that the UART was not ready in time to receive the next character and the module might need to be resynchronized to the bus.
2:	In Sleep mode, a start bit when detected, causes the device to wake-up only if the WAKE bit (UxMODE<7>) is set just before the device enters Sleep mode.
3:	In Sleep and Idle modes, as the falling edge on the UART receive pin generates a UART receive interrupt, a dummy byte is copied if the UART receive buffer is read in the first UART receive interrupt.

Figure 17-16: Auto-Wake-up Bit (WAKE) Timings During Sleep



Note 1: If the wake-up event requires long oscillator warm-up time, the auto-clear of the WAKE bit can occur while the system clocks are still active. This sequence should not depend on the presence of FCY.

2: The UART state machine is held in Idle while the WAKE bit is active.

17.12 OPERATION OF $\overline{\text{UxCTS}}$ AND $\overline{\text{UxRTS}}$ CONTROL PINS

The $\overline{\text{UxCTS}}$ (Clear to Send) and $\overline{\text{UxRTS}}$ (Request to Send) pins are the two hardware controlled pins associated with the UART module. These two pins allow the UART to operate in Flow Control and Simplex modes, which are explained in detail in **17.12.2 "UxRTS Function in Flow Control Mode"** and **17.12.3 "UxRTS Function in Simplex Mode"**, respectively. They are implemented to control transmission and reception between the UART and Data Terminal Equipment (DTE).

17.12.1 $\overline{\text{UxCTS}}$ Function

During UART operation, the $\overline{\text{UxCTS}}$ pin acts as an input pin, which can control the transmission. This pin is controlled by another device (typically a PC). The $\overline{\text{UxCTS}}$ pin is configured using the UEN<1:0> bits (UxMODE<9:8>).

- When UEN<1:0> = `10`, $\overline{\text{UxCTS}}$ is configured as an input
- If $\overline{\text{UxCTS}}$ = `1`, the transmitter will go as far as loading the data in the Transmit Shift register, but will not initiate a transmission. This will allow the DTE to control and receive data from the controller per its requirements.

The $\overline{\text{UxCTS}}$ pin is sampled while the transmit data changes (i.e., at the beginning of the 16 baud clocks). The transmission begins only when the $\overline{\text{UxCTS}}$ is sampled low. The $\overline{\text{UxCTS}}$ is sampled internally with a TCY, which means that there should be a minimum pulse width of 1 TCY on UxCTS. However, this cannot be a specification as the TCY can vary depending on the clock used.

The user application can also read the status of the $\overline{\text{UxCTS}}$ by reading the associated port pin.

17.12.2 $\overline{\text{UxRTS}}$ Function in Flow Control Mode

During Flow Control mode, the $\overline{\text{UxRTS}}$ pin of one DTE is connected to the $\overline{\text{UxCTS}}$ pin of the dsPIC33E/PIC24E device, and the $\overline{\text{UxCTS}}$ pin of the DTE is connected to the $\overline{\text{UxRTS}}$ pin of the device, as illustrated in Figure 17-17. The $\overline{\text{UxRTS}}$ signal indicates that the device is ready to receive the data. The $\overline{\text{UxRTS}}$ pin is driven as an output whenever UEN<1:0> = `01` or `10`. The $\overline{\text{UxRTS}}$ pin is asserted (driven low) whenever the receiver is ready to receive data. When the RTSMD bit = `0` (when the device is in Flow Control mode), the $\overline{\text{UxRTS}}$ pin is driven low whenever the receive buffer is not full or the OERR bit is not set. When the RTSMD bit = `0`, the $\overline{\text{UxRTS}}$ pin is driven high whenever the device is not ready to receive (i.e., when the receiver buffer is either full or in the process of shifting).

Since the $\overline{\text{UxRTS}}$ pin of the DTE is connected to the $\overline{\text{UxCTS}}$ pin of the dsPIC33E/PIC24E device, the $\overline{\text{UxRTS}}$ pin drives the $\overline{\text{UxCTS}}$ pin low whenever it is ready to receive the data. Transmission of the data begins when the $\overline{\text{UxCTS}}$ pin goes low, as explained in **17.12.1 "UxCTS Function"**.

17.12.3 $\overline{\text{UxRTS}}$ Function in Simplex Mode

During Simplex mode, the $\overline{\text{UxRTS}}$ pin of the DCE is connected to the $\overline{\text{UxRTS}}$ pin of the dsPIC33E/PIC24E device, and the $\overline{\text{UxCTS}}$ pin of the DCE is connected to the $\overline{\text{UxCTS}}$ pin of the device, as illustrated in Figure 17-18. In Simplex mode, the $\overline{\text{UxRTS}}$ signal indicates that the DTE is ready to transmit. The DCE replies to the $\overline{\text{UxRTS}}$ signal with the valid $\overline{\text{UxCTS}}$ whenever the DCE is ready to receive the transmission. When the DTE receives a valid $\overline{\text{UxCTS}}$ signal, it will begin transmission.

As illustrated in Figure 17-19, Simplex mode is also used in IEEE-485 systems to enable transmitters. When $\overline{\text{UxRTS}}$ indicates that the DTE is ready to transmit, the $\overline{\text{UxRTS}}$ signal enables the driver.

The $\overline{\text{UxRTS}}$ pin is configured as an output and is driven whenever UEN<1:0> = `01` or `10`. When RTSMD = `1`, the $\overline{\text{UxRTS}}$ is asserted (driven low) whenever data is available to transmit (TRMT = `0`). When RTSMD = `1`, $\overline{\text{UxRTS}}$ is deasserted (driven high) when the transmitter is empty (TRMT = `1`).

Figure 17-17: $\overline{\text{UxRTS}}$/$\overline{\text{UxCTS}}$ Flow Control for DTE-DTE (RTSMD = `0`, Flow Control Mode)

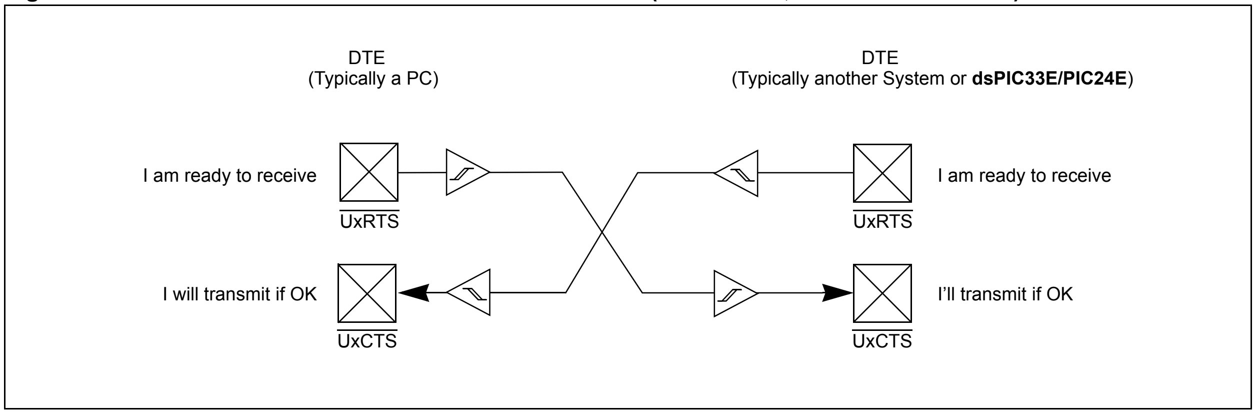


Figure 17-18: $\overline{\text{UxRTS}}$/$\overline{\text{UxCTS}}$ Handshake for DTE-DCE (RTSMD = `1`, Simplex Mode)

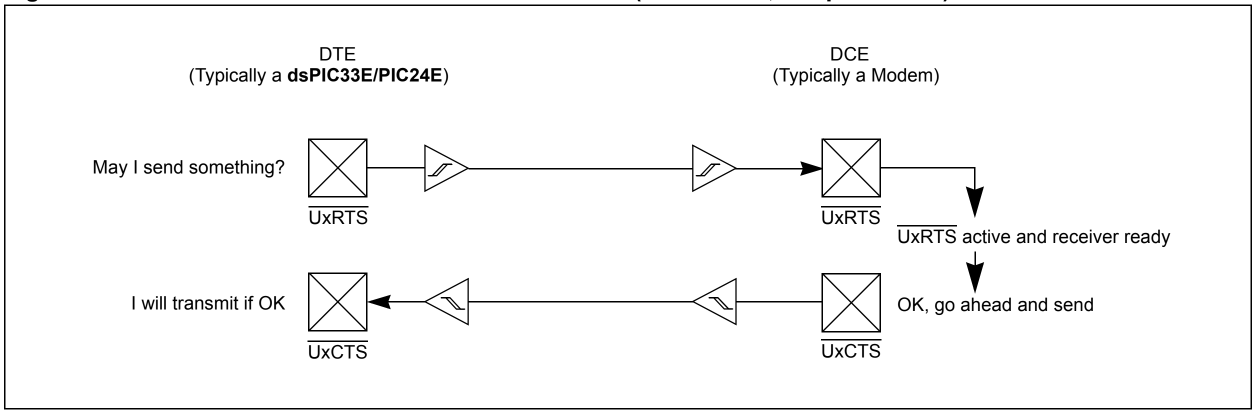
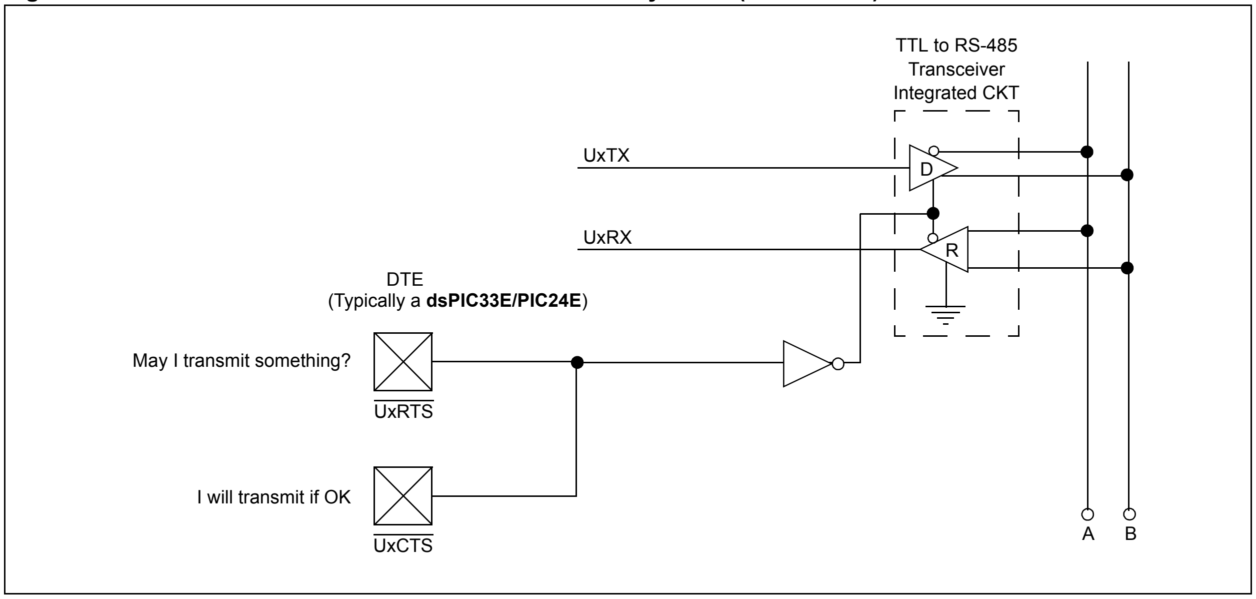


Figure 17-19: $\overline{\text{UxRTS}}$/$\overline{\text{UxCTS}}$ Bus Enable for IEEE-485 Systems (RTSMD = `1`)



17.13 INFRARED SUPPORT

The UART module provides two types of infrared UART support: one is the IrDA clock output to support the external IrDA encoder and decoder devices (legacy module support), and the other is the full implementation of the IrDA encoder and decoder.

Note: This feature is available only in Low-Speed mode (BRGH = `0`), with a baud rate above 1200 baud.

17.13.1 External IrDA Support – IrDA Clock Output

To support external IrDA encoder and decoder devices, the BCLKx pin can be configured to generate the 16x baud clock. When UEN<1:0> = `11`, the BCLKx pin outputs the 16x baud clock if the UART module is enabled; this pin can be used to support the IrDA codec chip.

17.13.2 Built-in IrDA Encoder and Decoder

The UART has full implementation of the IrDA encoder and decoder. The built-in IrDA encoder and decoder functionality is enabled using the IREN bit (UxMODE<12>). When enabled (IREN = `1`), the receive pin (UxRX) acts as an input from the infrared receiver. The transmit pin (UxTX) acts as an output to the infrared transmitter.

17.13.2.1 IrDA ENCODER FUNCTION

The encoder works by taking the serial data from the UART and replacing it. Transmit bit data of ‘`1`’ becomes encoded as ‘`0`’ for the entire 16 periods of the 16x baud clock. Transmit bit data of ‘`0`’ becomes encoded as ‘`0`’ for the first 7 periods of the 16x baud clock, as ‘`1`’ for the next 3 periods, and as ‘`0`’ for the remaining 6 periods. See Figure 17-20 and Figure 17-22 for details.

17.13.2.2 TRANSMIT POLARITY

The transmit polarity is selected using the UTXINV bit (UxSTA<14>). When UTXINV = `0`, the Idle state of the UxTX line is ‘`0`’ (see Figure 17-20). When UTXINV = `1`, the Idle state of the UxTX line is ‘`1`’ (see Figure 17-21).

Figure 17-20: IrDA® Encode Scheme

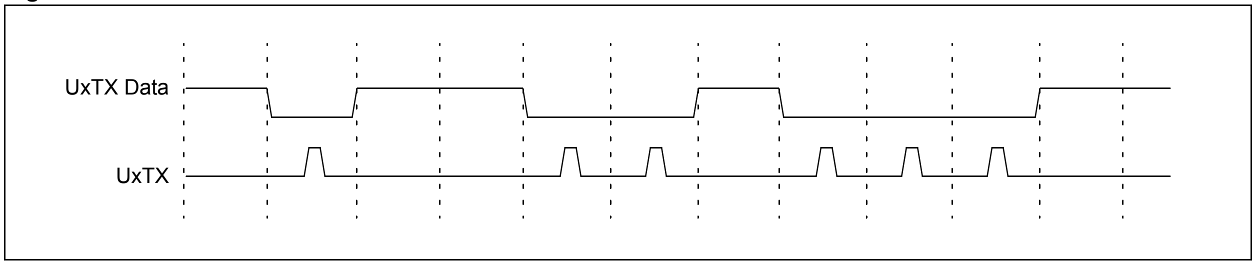


Figure 17-21: IrDA® Encode Scheme for ‘`0`’ Bit Data

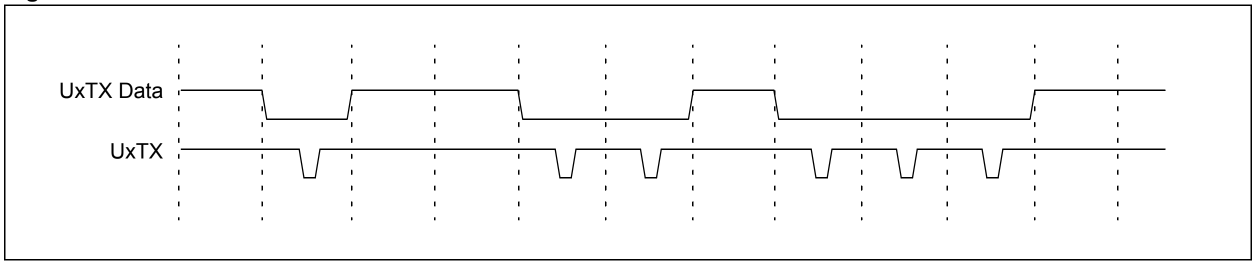
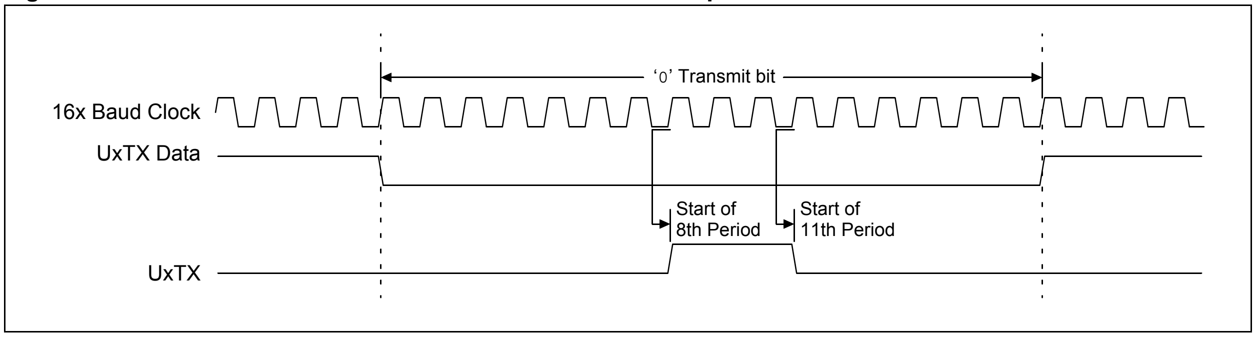


Figure 17-22: IrDA® Encode Scheme for '0' Bit Data with Respect to 16x Baud Clock



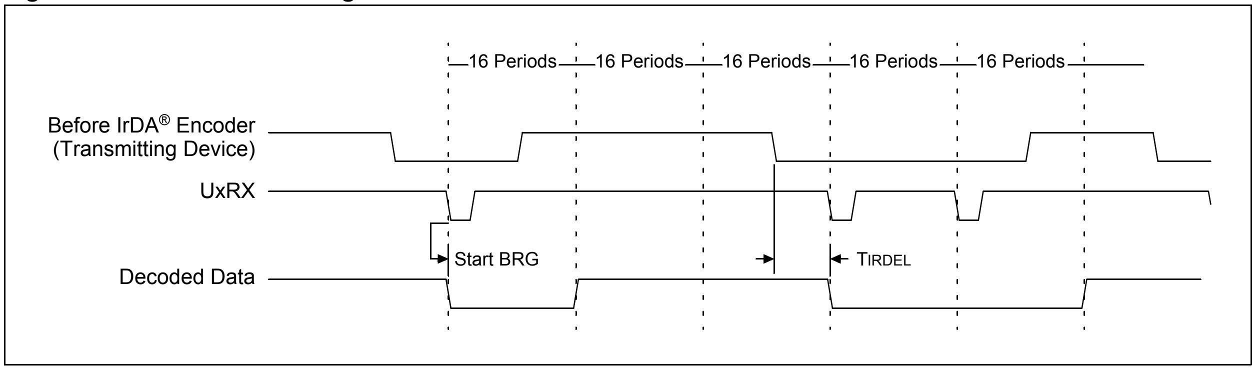
17.13.2.3 IrDA DECODER FUNCTION

The decoder works by taking the serial data from the UxRX pin and replacing it with the decoded data stream. The stream is decoded based on falling edge detection of the UxRX input.

Each falling edge of UxRX causes the decoded data to be driven low for 16 periods of the 16x baud clock. If, by the time the 16 periods expire, another falling edge is detected, the decoded data remains low for another 16 periods. If no falling edge was detected, the decoded data is driven high.

Note that the data stream into the device is shifted anywhere from 7 to 8 periods of the 16x baud clock from the actual message source. The one clock uncertainty is due to the clock edge resolution (see Figure 17-23 for details).

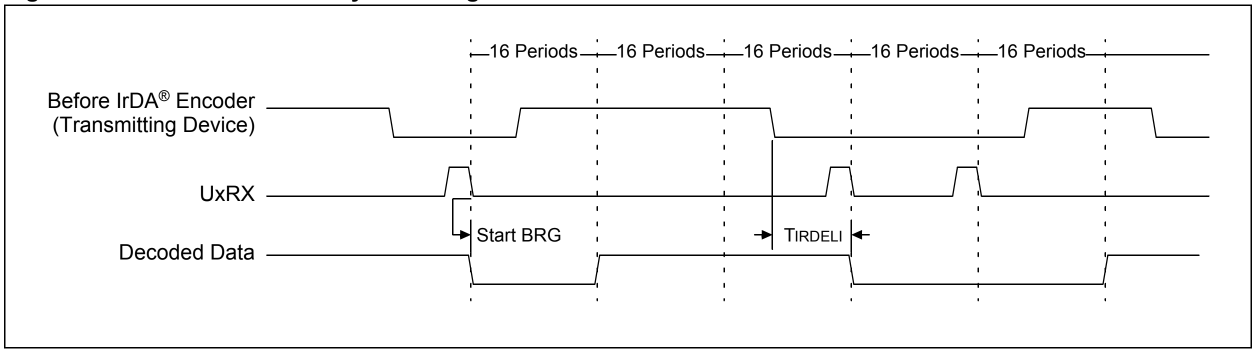
Figure 17-23: IrDA® Decoding Scheme



17.13.2.4 IrDA RECEIVE POLARITY

The input of the IrDA signal can have an inverted polarity. The same logic is able to decode the signal train, but in this case, the decoded data stream is shifted from 10 to 11 periods of the 16x baud clock from the original message source. Again, the one clock uncertainty is due to the clock edge resolution (see Figure 17-24 for details).

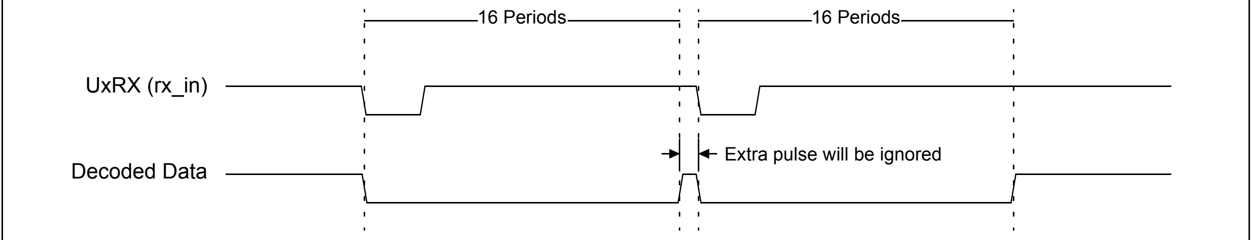
Figure 17-24: Inverted Polarity Decoding Results



17.13.2.5 CLOCK JITTER

Due to jitter or slight frequency differences between devices, it is possible for the next falling bit edge to be missed for one of the 16x periods. In that case, a one clock wide pulse appears on the decoded data stream. Since the UART performs a majority detect around the bit center, this does not cause erroneous data (see Figure 17-25 for details).

Figure 17-25: Clock Jitter Causing a Pulse Between Consecutive Zeros

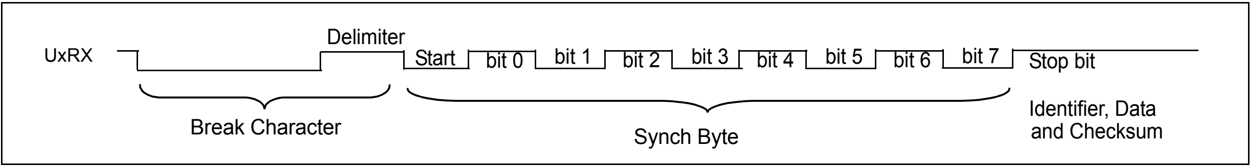


17.14 LIN SUPPORT

The LIN protocol transmits data in the form of small blocks, known as frames. Each frame consists of a Break character with a delimiter, a Synch byte, a protected identifier and the data to be transmitted (see Figure 17-26).

- Break Sequence: The break sequence indicates the beginning of a new frame. A break sequence generated by the master node consists of a Start bit followed by twelve bits of '0' and a break delimiter.
- Synch Byte: The Synch is a byte field loaded with the data value of 0x55. When the Auto-Baud feature is enabled, the UART module uses the Synch byte to compute the baud rate of the incoming data.
- Protected Identifier: The Protected Identifier contains the identifier and the identifier parity.

Figure 17-26: Frame Structure



17.14.1 Data Reception Using LIN Protocol

When the LIN protocol is used, the UART module receives data in the form of frames, and the incoming data is loaded into the receive buffer. For effective data reception, the BRG counter should be loaded with the baud rate of incoming data.

The bit rate of the incoming data can be detected, if the following occurs:

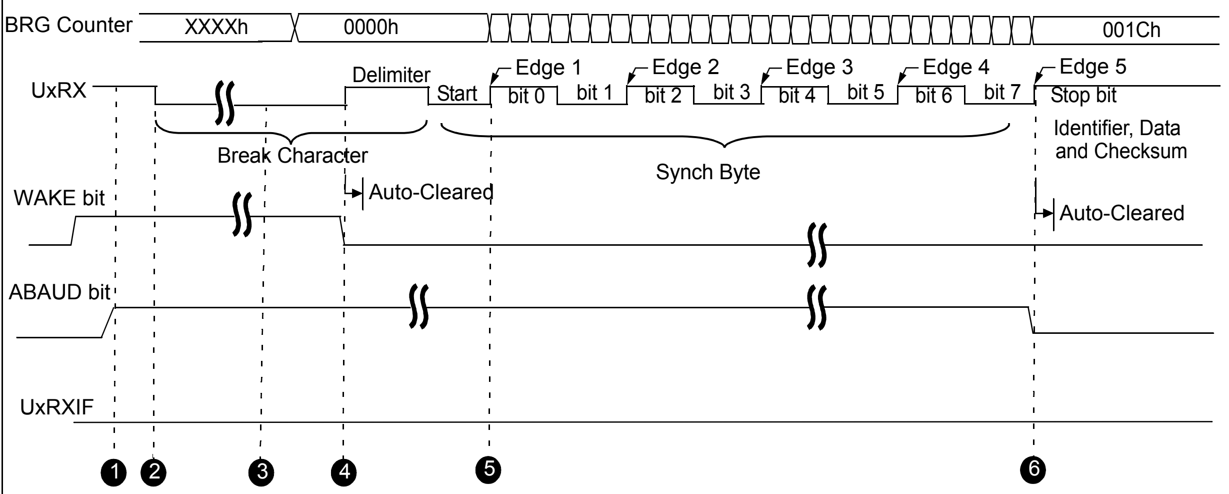
- The Auto-Baud feature is enabled
- The WAKE bit is set before setting the ABAUD bit

The UART module uses the Synch byte to compute the baud rate of the incoming data. If the WAKE bit is set before setting the ABAUD bit, the Auto-Baud Rate Detection occurs on the byte following the Break character. The module receives the Start bit of the Break character, the data and the invalid Stop bit (which sets FERR), but the receiver waits for a valid Stop bit before receiving the next Start bit. No further reception can occur until a Stop bit is received.The WAKE bit is automatically cleared once the Stop bit is received. After the fifth rising edge of the Synch character is detected, the baud rate of the incoming data is loaded into the BRG counter, and the ABAUD bit is automatically cleared.

If the Auto-Baud feature is enabled without setting the WAKE bit, the delimiter is assumed to be the first bit of the Synch byte instead of the Start bit. This results in erroneous baud rate calculation. This happens because the receiver expects a Synch byte at the start of the reception. The LIN protocol, however, initiates the transmission with the Break character and the Synch byte follows. Thus, the delimiter, which can range from one to four bits, is assumed to be the first low-to-high transition on the RX line. Therefore, the delimiter acts as a first bit of the Synch byte instead of the Start bit (see Figure 17-27).

Note:	Before data reception, the user application should load the BRG counter of the UART module with a value approximate to the bit rate of the incoming data.

Figure 17-27: Data Reception Using the LIN Protocol



1. The user application first sets the WAKE bit and then the ABAUD bit.
2. An RX interrupt is generated because the device wakes up on the reception of the Break character.
3. The FERR bit is set.
4. The WAKE bit is cleared.
5. The Auto-Baud feature detects the first rising edge of the Synch byte.
6. The fifth rising edge of the Synch byte is detected. The ABAUD bit is cleared and the BRG counter is loaded with the baud rate of the received data.

17.15 REGISTERS MAP

A summary of the registers associated with the UARTx module of the dsPIC33E/PIC24E device family is provided in Table 17-4.

Table 17-4: Registers Associated with UARTx

SFR Name	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
UxMODE	UARTEN	—	USIDL	IREN	RTSMD	—	UEN<1:0>		WAKE	LPBACK	ABAUD	URXINV	BRGH	PDSEL<1:0>		STSEL	0000
UxSTA	UTXISEL1	UTXINV	UTXISEL0	—	UTXBRK	UTXEN	UTXBF	TRMT	URXISEL<1:0>		ADDEN	RIDLE	PERR	FERR	OERR	URXDA	0110
UxTXREG	—	—	—	—	—	—	—	UTX8	Transmit Register								xxxx
UxRXREG	—	—	—	—	—	—	—	URX8	Receive Register								0000
UxBRG	Baud Rate Generator Prescaler																0000

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

Note: The registers associated with UARTx are shown for reference. See the device data sheet for the registers associated with other UART modules.

17.16 DESIGN TIPS

Question 1: ***The data I transmit with the UART does not get received correctly. What could cause this?***

Answer: The most common reason for reception errors is that an incorrect value has been calculated for the UART Baud Rate Generator. Ensure the value written to the UxBRG register is correct.

Question 2: ***I am getting framing errors even though the signal on the UART receive pin looks correct. What are the possible causes?***

Answer: Ensure the following control bits have been set up correctly:

- UxBRG: UART Baud Rate register bits
- PDSEL<1:0>: Parity and Data Size Selection bits
- STSEL: Stop bit Selection

17.17 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 dsPIC33E/PIC24E device family, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the UART module include the following:

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 dsPIC33E/PIC24E family of devices.

17.18 REVISION HISTORY

Revision A (January 2009)

This is the initial released version of this document.

Revision B (July 2010)

This revision includes the following changes:

- Changed the document name from dsPIC33E Family Reference Manual to dsPIC33E/PIC24E Family Reference Manual
- Deleted watermark from the code examples
- Updated the code in Example 17-4
- Added Note 3 in the note box in **17.11.2 “UART Operation in Idle Mode”**
- Updated and added a note to Figure 17-11
- Updated Figure 17-13
- Added Figure 17-15
- Changed the bit 8 characteristic from W-x to R/W-0 in Register 17-5
- Minor changes to the text and formatting have been incorporated throughout the document

ISBN: 978-1-60932-389-9

MICROCHIP®

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://support.microchip.com
Web Address:
www.microchip.com

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

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

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

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

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

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Kokomo
Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

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

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-6578-300
Fax: 886-3-6578-370

Taiwan - Kaohsiung
Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

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

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

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

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

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

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

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

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

01/05/10