TB3208

Basic Operation of UART with Protocol Support

Introduction

Author: Michael Wyman and Christopher Best, Microchip Technology Inc.

The Universal Asynchronous Receiver Transmitter (UART) module with built-in protocol support is a serial communications peripheral that allows users to interface other UART compatible devices. The UART module is designed for full-duplex communication, and supports the following protocols:

- LIN Master and Slave modes
- DMX mode
- DALI Control Gear and Control Device modes

This technical brief highlights the use of this UART module in Basic mode stand-alone, without the use of the other UART protocol support.

Info: For more technical detail regarding any of the supported protocol functions of this UART module, view the device data sheet, along with any other supporting documentation.

Table of Contents

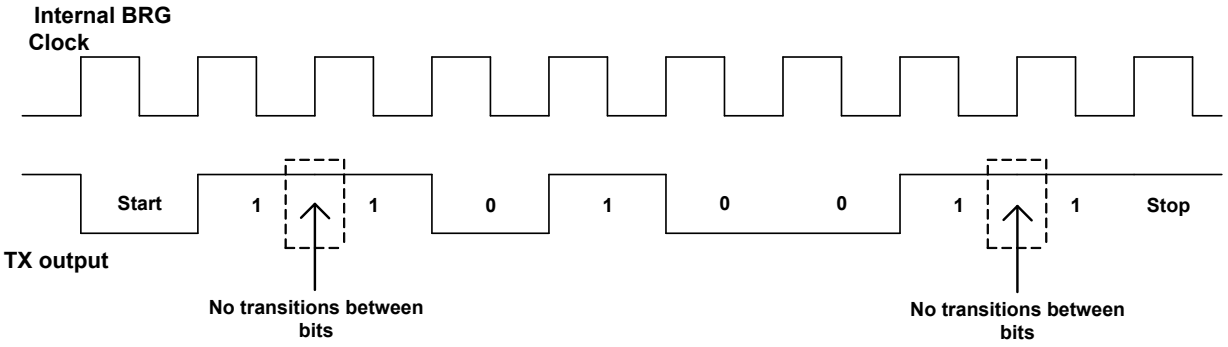
1. Overview

The UART module is a core-independent peripheral that performs serial data transfers without any CPU intervention. The module includes the following features:

- Full-Duplex Transmit and Receive
- Two-Character Input Buffer
- One-Character Output Buffer
- Programmable 7-Bit or 8-Bit Character Length
- 9th Bit Address Detection
- 9th Bit Parity
- Input Buffer Overrun Detection
- Framing Error Detection
- Hardware and Software Flow Control
- Automatic Checksums
- Programmable Stop Bit Count
- Programmable Data Polarity
- Automatic Detection and Calibration of the Baud Rate
- Operation in Sleep
- Wake-Up on Break Reception
- Automatic and User-Timed Break Period Generation
- Inactivity Time-Outs

The UART module transmits and receives data using the standard Non-Return-to-Zero (NRZ) format. The NRZ format uses two logic states, V_{OH} and V_{OL}. The V_{OH} Mark state represents a '1' data bit, while the V_{OL} Space state represents a '0' data bit. The NRZ format states that consecutively transmitted data bits of the same value stay at the output level of that bit without returning to the neutral level between bits (Figure 1-1). An NRZ transmission port idles in the Mark ('1') state. Each character transmission consists of a Start bit, followed by seven or eight data bits, one optional parity or address bit, and ends with one or more Stop bits. The Start bit is always a space ('0'), while Stop bits are always marks ('1'). The most common data format consists of eight data bits and no parity bit.

Figure 1-1. NRZ Format



UART Mode Select Bits (MODE<3:0> bits of the UxCON0 register):

- 0100 = Asynchronous 9-bit UART Address mode. 9th bit 1 = address; 0 = data
- 0011 = Asynchronous 8-bit UART mode with 9th bit even parity
- 0010 = Asynchronous 8-bit UART mode with 9th bit odd parity
- 0001 = Asynchronous 7-bit UART mode
- 0001 = Asynchronous 7-bit UART mode
- 0000 = Asynchronous 8-bit UART mode

The UART module transmits and receives the Least Significant bit (LSb) first, regardless of which mode is selected. Both transmitter and receiver blocks operate independently, but share the same baud rate and data format. A dedicated 16-bit Baud Rate Generator (BRG) derives the baud rate from the system oscillator. Each transmitted bit persists for a period of 1/Baud Rate.

1.1 Transmit and Receive Buffers

The UART module contains dedicated transmit and receive buffers. The Transmit Enable Control (TXEN) bit enables/disables the transmitter, while the Receive Enable Control (RXEN) bit enables/disables the receiver.

The transmitter consists of the Transmit Shift Register (TSR) and one buffer register, UxTXB. Writes to UxTXB are transferred to the TSR when the TSR is empty. Bytes written to UxTXB while the TSR is full will be held in UxTXB until the TSR has completed shifting out its data. The Transmit Shift Register Empty Interrupt Flag (TXMTIF) bit can be used to determine if the TSR is empty, while the Transmit Buffer Empty Status (TXBE) bit and the UART Transmit Interrupt Flag (UxTXIF) bit indicate if the UxTXB is empty. Additionally, the Transmit Buffer Full Status (TXBF) bit can be used to indicate if the transmit buffer is full. If a new byte was written to UxTXB while the TXBF bit is set (TXBF = 1), the Transmit Write Error Status (TXWRE) bit becomes set and must be cleared in software to continue transmission. The TXBE bit can also be set by software to clear both the transmit buffer and the transmit shift register.

The receiver consists of the Receive Shift Register (RSR) and a two-level First-In First-Out (FIFO) buffer area. The buffer at the top of the FIFO, UxRXB, holds the first received byte (earliest byte to enter the FIFO). When a byte is received, it is loaded into the RSR, and if the receive FIFO is empty, the byte is transferred to UxRXB. If a second byte is received while the first byte remains in UxRXB, the second byte is transferred to the bottom of the FIFO. If a third byte is received, it is stored in the RSR until UxRXB is read. If any further bytes are received while the FIFO and RSR are full, the Receive FIFO Overflow Interrupt Flag (RXFOIF) bit becomes set, indicating a receive overflow condition.

In the event of an overflow, the Run During Overflow Control (RUNOVF) bit can be used to determine whether the RSR continues to receive data (after the overflow condition is cleared) or stop receiving data (legacy mode). An overflow condition can be cleared by reading UxRXB and clearing the RXFOIF bit, setting the RXBE bit, which flushes the entire FIFO, or by clearing the ON bit, which clears both the transmit and receive buffers and shift registers. When RUNOVF is set, the module will continue to synchronize the RSR with the incoming Start bits after the overflow condition has been cleared. It is important to note that although the FIFO will continue to operate in an overflow condition, the data will not be valid until the overflow condition has been cleared.

Transmit and receive polarity is user selectable. Both transmit and receive lines default to a logic ‘high’ Idle state. The Transmit Polarity Control (TXPOL) bit controls the transmit line polarity, while the Receive Polarity Control (RXPOL) bit controls the receive line polarity.

1.2 Character Length

Character length is controlled by the MODE bits of UART Control Register 0 (UxCON0):

- In Asynchronous 7-bit UART mode, only the seven Least Significant bits (LSbs) of the transmit/receive byte are used, while the Most Significant bit (MSb) is ignored.
- In Asynchronous 8-bit UART mode, all eight bits of the transmitted or received byte are used.
- In Asynchronous 8-bit UART mode with 9th bit odd/even Parity modes, each byte transmitted or received consists of nine bits.

In Parity modes, the first eight bits of each byte are the data bits, and the ninth bit is the parity bit. Parity bits are used in error detection. Even Parity means that the expected number of '1' bits in a character is an even number, while Odd Parity means that the expected number of '1' bits in a character is an odd number. When in Even Parity mode, the parity bit will be set if the number of '1' bits in the 8-bit transmit data is an odd number; when the number of '1' bits are even, the parity bit is cleared. In Odd Parity mode, when the number of '1' bits in the 8-bit transmit data is odd, the parity bit is cleared; when the number of bits are even, the parity bit is set. Table 1-1 gives examples of how the parity bit value is determined.

Table 1-1. Even/Odd Parity Examples

Byte Value	Even Parity Bit Value	Odd Parity Bit Value
1010 1010	0	1
1010 1000	1	0

In Asynchronous 9-bit UART Address mode, each byte transmitted or received consists of nine bits, and the 9th bit determines whether the byte is an address or data. When the 9th bit is set, the other eight bytes are used as an address; when the 9th bit is clear, the other eight bytes are used as data (see 1.3 Address Mode for more details).

1.3 Address Mode

The Asynchronous 9-bit UART Address mode allows the UART to communicate with multiple receivers sharing the same transmission line.

When transmitting in Address mode, the address of the receiver of interest is loaded into the UART Parameter 1 Low (UxP1L) register. When the address is loaded into UxP1L, hardware sets the 9th bit, indicating to the receiver that the byte is an address. Data bytes are written to the UxTXB register, similarly to normal UART operation. It is important to note that writes to UxP1L take precedence over writes to UxTXB. If both registers are written while the TSR is busy, the next byte loaded into TSR will come from UxP1L.

When receiving in Address mode, no received data will be transferred into the FIFO until a valid address is received. When a character with the 9th bit set is received, the eight LSb's are compared to the UART Parameter 2 Low (UxP2L) register. In receive Address mode, UxP2L holds the receiver address, while the UART Parameter 3 Low (UxP3L) register holds the address mask. If an address match occurs, the eight LSb's are transferred into the FIFO, while the 9th bit is transferred into the Parity Error Interrupt Flag (PERIF) bit. In this case, when PERIF is set, the UART Receive Interrupt Flag (UxRXIF) is suppressed, and if the Direct Memory Access (DMA) module is using the UxRXIF as a trigger, DMA transfers will be suspended. This prevents the DMA from writing address information into memory.

1.4 Flow Control

Flow control allows the UART to suspend transactions in order to prevent input buffers from overflowing. The UART supports both hardware and software flow control methods. The flow control method is determined by the Handshake Flow Control (FLO) bits of the UxCON2 register.

1.4.1 Hardware Flow Control

For hardware flow control, the UART module utilizes a $\overline{RTS}$/$\overline{CTS}$ flow control scheme commonly found in RS-232 (Recommended Standard 232) networks. The RS-232 standard defines the signals connecting Data Terminal Equipment (DTE) to Data Communication Equipment (DCE). In addition to the standard transmit (TX) and receive (RX) lines, $\overline{RTS}$ flow control uses the Request-to-Send ($\overline{RTS}$) and Clear-to-Send ($\overline{CTS}$) lines. The UART module also provides a third line, the Transmit Drive Enable (TXDE) line, to control an RS-485 transceiver.

The UART module is configured as a DTE device; therefore, UART hardware configures the $\overline{RTS}$ signal as an output, while the $\overline{CTS}$ signal is an input. In a DCE system, the opposite is true; the $\overline{RTS}$ signal is an input, while the $\overline{CTS}$ signal is an output. It is important to note that connections between DTE and DTE devices (Figure 1-2) are different than connections between DTE and DCE devices (Figure 1-3).

The active-low $\overline{RTS}$ and $\overline{CTS}$ signals work together to control transmission flow. Hardware flow is typically controlled by the DTE device, which could be considered a ‘master’ device. In the case of a DTE-to-DTE configuration, either device can act as a master. When one DTE device wishes to transmit data, the DTE device pulls the $\overline{RTS}$ line low, which signals the slave device, through its $\overline{CTS}$ input, to begin to monitor its RX input. When the slave device is ready to accept the data, it pulls its $\overline{RTS}$ line low, informing the master, through its $\overline{CTS}$ line, to begin sending data. Once the transaction has completed, the master device pulls the $\overline{RTS}$ line high.

In a DTE-to-DCE configuration, the DTE is considered the master and the DCE is considered a slave. In this configuration, when the DTE device wishes to transmit data, the DTE device pulls the $\overline{RTS}$ line low, which signals the DCE device, through its $\overline{RTS}$ line, to begin to monitor its RX line. When the DCE device is ready to accept the data, it pulls its $\overline{CTS}$ line low, informing the DTE device, through its $\overline{CTS}$ connection, to begin sending data.

Figure 1-2. UART Connections Between Two DTE Devices

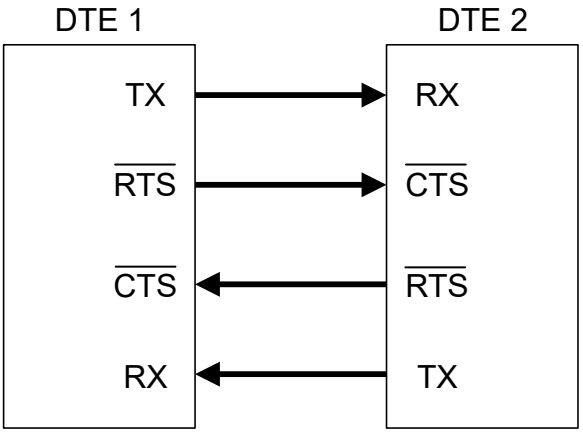
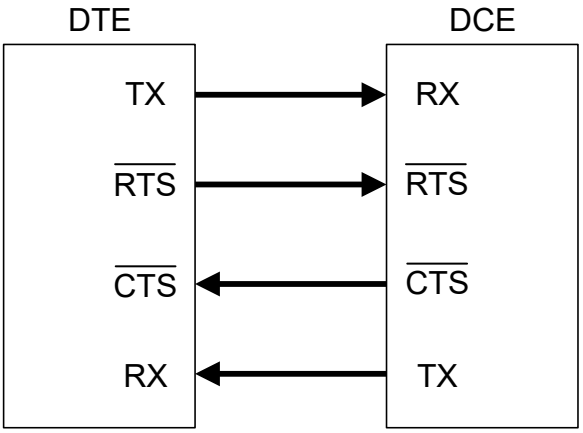


Figure 1-3. UART Connections Between a DTE Device and DCE Device



1.4.2 Software Flow Control

Software flow control uses the XON/XOFF (Transmit ON/Transmit OFF) method. This method has an advantage over hardware flow control methods since it does not require additional hardware lines, which significantly reduces wiring complexity.

In the XON/XOFF method, special characters are sent by the receiver to the transmitter that are used to suspend and resume transactions. These characters are not specifically defined by any standard, such as ASCII. However, the ASCII standard provides generic ‘device control’ characters, DC1 – DC4. The UART module specifically uses the ASCII control characters DC1 (0x11) and DC3 (0x13) as the XON and XOFF control characters, respectively. When one terminal is unable to accept data, it sends the XOFF character to the other terminal, which in turn suspends transmission. When the other terminal is ready to accept data again, it simply sends the XON character back to the first terminal, which resumes transmission. The Software Flow Control Transmit Enable Status (XON) bit of the UxFIFO STATUS register can be used to determine if the transmitter is enabled/disabled.

1.5 Stop Bits

The UART module offers a user-selectable number of Stop bits. The Stop bit selections are as follows:

- 1 Stop bit: UART transmits one Stop bit; the receiver verifies the first Stop bit received
- 1.5 Stop bits: UART transmits 1.5 Stop bits; the receiver verifies the first Stop bit only
- 2 Stop bits: UART transmits two Stop bits; the receiver verifies the first Stop bit only
- 2 Stop bits: UART transmits two Stop bits; the receiver verifies both

During normal operation, the transmitter returns to the Idle state for the number of Stop bit periods between each consecutively transmitted word. In all Stop bit configurations, the RX input is checked for an Idle condition during the middle of the first Stop bit period. In the case of the two Stop bit configuration where the receiver verifies both Stop bits, hardware checks the middle of both the first and second Stop bits for an Idle condition. If any Stop bit verification indicates a non-Idle condition, the Framing Error Interrupt Flag (FERIF) bit of the UART Error Interrupt Flag (UxERRIR) register becomes set, indicating a frame error for that particular byte.

1.6 Checksums

Checksum calculations are used to verify that the data contained in a packet was transmitted/received correctly. When the Checksum Mode Select (C0EN) bit of UxCON2 is set, transmit and receive checksum adders accumulate the value of each byte transmitted or received.

In Transmit mode, the checksum adders accumulate the value of each byte transmitted. Once all bytes have been transmitted, the sum of the byte values are loaded into the UART Transmit Checksum Result (UxTXCHK) register, which is then sent as the final byte of the transmission.

In Receive mode, the checksum adders accumulate the value of each byte received. Once all bytes have been received, including the checksum byte, the sum of the received byte values are loaded into the UART Receive Checksum Result (UxRXCHK) register.

It is important to note that although UART hardware calculates the checksums and loads the results into the appropriate checksum registers, software must clear UxTXCHK and UxRXCHK before beginning UART transactions, and also performs the checksum comparisons at the end of each transaction.

1.7 Baud Rate Generation

The UART Baud Rate Generator (BRG) is a 16-bit timer used to generate the clocking mechanism required for communication. The timer consists of the UART Baud Rate Generator High and Low register pair (UxBRGH:UxBRGL). The Baud Rate Generator Speed Select (BRGS) bit of UxCON0 determines the number of baud clocks used per bit.

When BRGS is clear (BRGS = 0), the BRG is configured in the normal baud rate range. In the normal baud rate range, the BRG generates 16 clock periods per data bit. Equation 1-1 shows the formula used to calculate the baud rate when BRGS is clear. The result of this formula should be loaded into the UxBRGH:UxBRGL register pair.

Equation 1-1. Baud Rate Formula (BRGS = 0)

$$Desired\ baud\ rate = \frac{Fosc}{16 * (UxBRGH{:}UxBRGL + 1)}$$

When BRGS is set, the module is configured in the high baud rate range. In high baud rate range, the BRG generates four clock periods per data bit. This range is intended for use when the normal baud range cannot produce the desired baud rate. Equation 1-2 shows the formula used to calculate the baud rate when BRGS is set.

Equation 1-2. Baud Rate Formula (BRGS = 1)

$$Desired\ baud\ rate = \frac{Fosc}{4 * (UxBRGH{:}UxBRGL + 1)}$$

Writing to the UxBRGH:UxBRGL register pair will immediately change the baud rate. If the register pair is written while the module is actively transmitting or receiving data, a receive error may occur. It is recommended that writes to the register pair occur when the Receive Pin Idle Status (RXIDL) bit of UxFIFO is set (RXIDL = 1), indicating an Idle condition. Additionally, if the system clock (F_{OSC}) is changed during active communication, a receive error may occur.

It is important to note that the BRG relies on the system clock (F_{OSC}) to generate the baud rate. If the system clock deviates from its frequency (due to noise, temperature, oscillator drift, etc.), the baud rate will also change, resulting in framing or overrun errors. The Auto-Baud Detection feature can be used to ensure that the bit rate is acceptable for error-free communication.

1.8 Auto-Baud Detection

The UART module offers an automatic baud rate detection system when operating in 8-bit modes only. The Auto-Baud Detection system prevents data loss or corruption by measuring the incoming signal on the RX pin and updating the UxBRGH:UxBRGL register pair to match the incoming rate. The Auto-baud Detection Enable (ABDEN) bit of UxCON0 controls the automatic baud rate detection system.

It is important to note that the BRG is used to measure the period of the received character ‘U’, or 0x55. This character is unique in the sense that it contains five alternating rising and falling edges. The auto-baud calibration sequence begins on the first falling edge of the character after the ABDEN bit is set. While the calibration sequence is active, the UART state machine is held in an Idle state, and the UxBRGH:UxBRGL register pair is clocked at 1/8th the base baud rate. The sequence ends on the 5th falling edge of the incoming character. At that point, the measured time value is loaded into the UxBRGH:UxBRGL register pair, the ABDEN bit is cleared, and the Auto-Baud Detect Interrupt Flag (ABDIF) is set.

1.8.1 Auto-Baud Overflow

If the baud rate counter overflows before the 5th falling edge is detected, the Auto-Baud Detect Overflow Interrupt Flag (ABDOVF) bit of UxERRIR is set, indicating an overflow condition. An overflow occurs when the 16-bit counter exceeds its maximum size. In this case, the state machine continues to wait for the 5th falling edge to occur on the RX pin. Once the 5th falling edge is detected, hardware sets the Auto-Baud Detect Interrupt Flag (ABDIF) bit of the UART General Interrupt Register (UxUIR) and clears the ABDOVF bit of UxERRIR, while the previous BRG values are retained. To terminate the auto-baud process once an overflow is detected, but before the 5th falling edge is detected, software must first clear the ABDEN bit and then clear the ABDOVF bit.

1.8.2 Wake-up on Break

The UART module is inactive in Sleep mode since the clock sources are disabled. This means that the UART cannot transmit or receive data while the device is in Sleep mode. The UART module offers an automatic wake-up feature that will wake the device when activity on the RX pin is detected.

The auto-wake-up feature is enabled by setting the Wake-Up Enable (WUE) bit of UxCON1. When WUE is set, the UART remains in an Idle state while waiting for a transition from the Idle state to the active

state on the RX pin. To prevent data loss, the RXIDL bit should be read to ensure that the receiver is idle before setting WUE and issuing a Sleep command.

If the WUE bit is set, but before the Sleep command is issued, and a transition from the Idle-to-Active state is detected, module hardware sets the WUIF at the beginning of the next instruction cycle, but does not enter Sleep. Once the transition from the Active-to-Idle state occurs, hardware clears the WUE bit.

When the wake-up event occurs, the Wake-Up Interrupt Flag (WUPIF) bit of the UxUIR register and the UART Interrupt Flag (UIF) of the Peripheral Interrupt (PIRx) register are set. At this point, the receiver will begin to monitor for a transition from the Active state back to the Idle state. When the Active state is detected, module hardware clears the WUE bit. It is important to note that the UIF is read-only; software must clear WUPIF to clear UIF.

1.8.2.1 Special Considerations

It is important to note that special considerations must be taken to prevent fragmented or lost data when using the auto-wake-up feature.

When using the auto-wake-up feature, a Break character must be received as the first character to wake the device. A Break character consists of all zeros and must persist for a minimum of 11 bit times before the next character is received. UART hardware uses a dedicated counter to detect when the RX input remains in the Space (0) state, and once the 11 bit period time has expired, the Break condition is detected. The Break Reception Interrupt Flag (RXBKIF) bit of the UxERRIR register can be used by software to detect that a Break condition has completed. The Receive Break Interrupt Mode Select (RXBIMD) bit of the UxCON1 register controls when the RXBKIF becomes set after a Break is detected. When RXBIMD is set, the RXBKIF is set immediately after a Break has been detected. When RXBIMD is clear, RXBKIF is set when the transition from the Active-to-Idle state on the RX input is detected following the Break character.

If WUE is set and a non-zero character is received to wake the device, the time between the Start bit and the first rising edge of the character will be interpreted as the wake-up event. Since the minimum 11 zero-bit times condition was not met, the remaining bits of the received character will be interpreted as a fragmented character, and all subsequent characters will result in farming or overrun errors.

A Break character can be sent by either using a fixed-length Break period or a software-timed Break period.

A fixed-length Break period consists of a Start bit, 12 zero-bits, and a Stop bit. To send a fixed-length Break, software must set the Send Break Control (SENDB) bit of the UxCON1 register. Once SENDB is set, a write to UxTXB will initiate the Break sequence. The receiver will receive the Break character first, followed by the character written into UxTXB.

A software-timed Break period does not contain any specific number of bit times. Instead, a Timer resource can be used to generate the Break condition. The software-timed Break is generated by setting and clearing the Send Break Software Override (BRKOVR) bit of the UxCON1 register. When BRKOVR is set, the TX output is forced into a non-Idle state; when clear, the TSR controls the TX output. In this case, the BRKOVR bit could be set when the Timer resource begins to count and cleared once the Timer has expired. This method allows the user to create a custom Break period.

Additionally, since the system clock is disabled during Sleep, the oscillator start-up time must be considered. In this case, additional time is needed for the oscillator to stabilize. To ensure that the Break period is long enough to meet the minimum eleven zero-bit periods and allow the oscillator to stabilize, the software-timed Break method may be used.

1.9 RX/TX Activity Time-Out

The Timer2/4/6 modules contain a Hardware Limit Timer (HLT) function that can be used to monitor activity on the RX and TX lines. The Timer2/4/6 module uses the Timer External Reset Signal Selection Register (TxRST) to reset the Timer when a user-selectable Reset event occurs. In this case, a value representing the desired time-out delay is loaded into the Timer Count Register (TxTMR). When a transition from an Idle state to an Active state on the TX output or RX input line will reset the Timer, indicating that the desired line (RX/TX) is active. If no transitions occur before the Timer expires, the HLT interrupt will occur, signaling a time-out event.

2. UART Transmit

The following steps outline the transmission process:

1. Configure the UxBRGH:UxBRGL register pair and BRGS bit to achieve the desired baud rate.
2. Set the MODE<3:0> bits of UxCON0 to the desired operation mode.
3. Set the TXPOL bit of UxCON2 if an inverted TX output is desired.
4. Load the RxyPPS register with the code associated with the TX output.
5. Clear the TRIS bit for the TX pin.
6. If UART interrupts are desired, set the appropriate interrupt enable bits, clear the associated interrupt flag bits, and enable the global interrupts.
7. Enable the UART module by setting the ON bit of UxCON1.
8. Set the TXEN bit of UxCON0 to enable the transmitter.
9. Check the UxTXIF bit to ensure the UxTXB is clear before writing.
10. When ready to begin transmission, write one byte of data to UxTXB.
11. Repeat steps 9 and 10 until all the data has been transmitted.

Example 2-1 shows the UART initialization sequence when operating in Transmit mode.

Example 2-1. UART Transmit Mode Initialization for PIC18(L)FXXK42

```
void UART1_Initialize(void)
{
    U1CON0 = 0x80;            // BRGS normal; 8-bit mode; ABDEN disabled;
    U1CON1 = 0x80;            // ON enabled;
    U1CON2 = 0x00;            // TXPOL not inverted; FLO off;
    U1BRGL = 0x19;            // BRGL 25 = 9600 BR @ 1 MHz FOSC
    U1BRGH = 0x00;
    U1FIFO = 0x00;            // STPMD in middle of first Stop bit;
    U1UIR = 0x00;             // Auto-baud not enabled
    U1ERRIR = 0x00;
    U1ERRIE = 0x00;
    U1CON0bits.TXEN = 1;      // Enable the transmitter
}
```

2.1 Printf In Transmit Mode

The UART transmitter supports the use of printf() statements. Printf() statements can be used instead of directly writing to the UTXB register. The printf() function is part of the STDIO library, which is included in the XC8 compiler tool suite. It is important to note that while the use of printf() statements simplify user software routines, the printf() function takes more RAM space and adds additional instruction cycles to complete.

The MPLAB® Code Configurator (MCC) plug-in tool can be used to add printf() functionality to the UART. Simply add a check to the “Redirect STDIO to UART” check box under the “Software Settings” drop-down menu in the UART **Easy Setup** tab. When the box is checked, MCC will generate the additional code needed by the STDIO library, as shown in Example 2-2. The example printf() statement, as well as the UART transmit routine, is also shown in the example below for reference. Refer to the data sheet for more details on interrupts.

Example 2-2. Printf() Additional Code for Transmit Mode for PIC18(L)FXXK42

```
void putch(char txData)
{
    UART1_Write(txData);
}

void UART1_Write(uint8_t txData)
{
    while(0 == PIR3bits.U1TXIF);
    U1TXB = txData;                         // Write the data byte to the UART.
}

printf ("Hello World!\r\n");
```

Info: MCC is an integral part of the code development for Microchip products. The MCC can be used to set up UART configuration code to allow the UART to function. MCC is a dynamic program that is used in conjunction with MPLAB® X as a plug in. Refer to http://www.microchip.com/mcc for more information.

3. UART Receive

The following steps outline the reception process:

1. Configure the UxBRGH:UxBRGL register pair and BRGS bit to achieve the desired baud rate.
2. Set the MODE<3:0> bits of UxCON0 to the desired operation mode.
3. Set the RXPOL bit of UxCON2 if an inverted RX input is desired.
4. Load the RXPPS register with the value associated with the desired RX pin.
5. Clear the associated ANSEL bit (if applicable).
6. Set the associated TRIS bit.
7. If UART interrupts are desired, set the appropriate interrupt enable bits, clear the associated interrupt flag bits, and set the global interrupts.
8. Enable the UART module by setting the ON bit of UxCON1.
9. Set the RXEN bit of UxCON0 to enable reception.
10. When a received byte is transferred from the RSR to UxRXB, the UxRIF bit becomes set, and an interrupt will be generated if the UxRXIE bit is set.
11. Read the UxERRIR register to check for any errors.
12. Read the UxRXB register to get the received byte and clear UxRXIF.
13. Repeat steps 10-12 until all data has been received.

Example 3-1 shows the initialization code used to configure the UART in receive mode.

Example 3-1. UART Receive Mode Initialization for PIC18(L)FXXK42

```
void UART1_Initialize(void)
{
    U1CON0 = 0x80;            // BRGS normal speed; 8-bit mode; ABDEN disabled;
    U1CON1 = 0x80;            // ON enabled;
    U1CON2 = 0x00;            // RXPOL not inverted; FLO off;
    U1BRGL = 0x19;            // BRGL 25 = 9600 BR @ 1 MHz FOSC
    U1BRGH = 0x00;
    U1FIFO = 0x00;            // STPMD in middle of first Stop bit;
    U1UIR = 0x00;             // Auto-baud not enabled
    U1ERRIR = 0x00;
    U1ERRIE = 0x00;
    U1CON0bits.RXEN = 1;         // Enable receiver
}
```

4. Conclusion

The Universal Asynchronous Receiver Transmitter (UART) with Protocol Support Module is a Serial Communications Interface that is used to communicate with other peripheral devices. This technical brief covers the basic functionality of the UART without the use of the protocol features, such as DMX, DALI and LIN. For more information, visit www.microchip.com.

5. Revision A (11/2018)

Initial release of this document.

The Microchip Web Site

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

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

Customer Change Notification Service

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

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

Customer Support

Users of Microchip products can receive assistance through several channels:

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

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

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

Microchip Devices Code Protection Feature

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

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

- 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.

Legal Notice

Trademarks

ISBN: 978-1-5224-3830-4

Quality Management System Certified by DNV

ISO/TS 16949

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

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
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	**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	**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	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4450-2828 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-67-3636 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820