INTRODUCTION

Throughput is based on many factors, some of which are determined by the Secondary Device (host controller + MCP215X) and others that are determined by the Primary Device.

This application note discusses techniques that will improve the data transfer throughput between an IrDA® standard Primary Device and embedded system and focus on what techniques can be used on the interface between the host controller and the MCP215X. The embedded system (Secondary Device) uses a MCP215X device for the IrDA® standard communication (IrCOMM 9-wire “cooked” service class). Figure 1 shows a typical IrDA standard system.

All timing measurements and screen captures in this application note are performed using a Palm™ Tungsten™ T2 PDA as the Primary Device. Using a different Primary Device may yield different results.

Data Transfers

Data Transfers can be broken down into three classes:

1. Data downloading
2. Data uploading
3. Bidirectional

In many typical systems, the majority of data will travel either in one direction or the other. That is, the majority of data will be transmitted (downloaded) from the Primary Device (PDA) to the Secondary Device (embedded system), or will be transmitted (uploaded) from the Secondary Device (embedded system) to the Primary Device (PDA).

Typical downloading application examples include:

• Embedded system configuration programming
• Embedded system firmware update

The data throughput during downloading cannot be controlled by the embedded system’s host controller. In this case, the data throughput is primarily determined by the characteristics of the Primary Device and the hardware characteristics of the MCP215X. The hardware characteristics of the MCP215X requires the CRC generation timing and parsing of the incoming IR Frame.

Typical uploading application examples include:

• Data loggers
• Diagnostic ports

During upload, the host controller firmware can have a significant impact on the data throughput.

In cases where small amounts of data are passed between devices and require user interface input, IR throughput requirements are minimized due to the human interface. An example of this would be when the PDA sends some codes to the embedded system, which then responds to the codes. The PDA operator then needs to determine what to do next based on this response. So the actual data throughput is not really an issue.

FIGURE 1: PALM™ PDA - EMBEDDED SYSTEM BLOCK DIAGRAM
MCP215X Host UART Interface and Receive Buffer

The MCP215X receive buffer is 64 bytes. The host UART also has a two-byte buffer. The limited size of the receive buffer requires that the host controller firmware takes care not to overflow the receive buffer. The MCP215X host UART interface includes signals to indicate to the host controller when data can be sent.

The MCP215X UART interface includes eight signals (TX, RX, CTS, RTS, CD, RI, DTR and DSR). The number of signals implemented in the embedded system is dependent on the requirements of the application. For more information, please refer to Application Note 858, “Interfacing the MCP215X to a Host Controller”, DS00858. In this application note, we will focus on three signals: the TX pin, the RX pin and the CTS pin. The CTS pin is an output signal from the MCP215X that indicates when the host controller can transfer data to the MCP215X. The MCP215X implements a windowed technique for receiving data. Once the CTS pin has a high-to-low transition, a 22 ms timer starts. The CTS pin will then be forced high when:
- the 22 ms timer ends, and
- the MCP215X receive buffer has received 60 bytes.

When the CTS pin is driven high due to the host receive buffer having received 60 bytes, the Receive Data window is still open (see Figure 3). Remember that the receive buffer is 64 bytes, so the Receive Data window will not be closed until either:
- the 22 ms timer has completed, or
- the MCP215X receive buffer has received 64 bytes.

Once the Receive Data window has been closed, the data in the host UART receive buffer will be processed for transmission on the TXIR pin.

Table 1 shows the minimum time to transmit 64 bytes at a given baud rate. This requires that there be no byte-to-byte spacing.

Table 2 shows how many bytes may be transferred on the host controller’s TX pin in 22 ms for a given baud rate. For baud rates 38400 and above, it is very easy to transmit the 64 bytes within the 22 ms Receive Data window. For baud rates 19200 and below, it is not possible to transmit 64 bytes within the 22 ms Receive Data window.

### Table 1: Time to Transfer 64 Bytes (Back to Back)

<table>
<thead>
<tr>
<th>Baud Rate</th>
<th>Time (ms)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>9600</td>
<td>66.7 (1)</td>
<td>&gt; 22 ms Receive Data window</td>
</tr>
<tr>
<td>19200</td>
<td>33.4 (1)</td>
<td>&gt; 22 ms Receive Data window</td>
</tr>
<tr>
<td>38400</td>
<td>16.7</td>
<td></td>
</tr>
<tr>
<td>57600</td>
<td>11.2</td>
<td></td>
</tr>
<tr>
<td>115200</td>
<td>5.6</td>
<td></td>
</tr>
</tbody>
</table>

**Note 1:** The Receive Data window time-out causes the CTS pin to be forced high and the MCP215X to process the data packet that has been loaded into the receive buffer.

### Table 2: Bytes Transferred at Given Baud Rates

<table>
<thead>
<tr>
<th>Baud Rate</th>
<th>Bytes Transferred in 22 ms</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>9600</td>
<td>22 (1, 2)</td>
<td>Host UART receive buffer NOT filled</td>
</tr>
<tr>
<td>19200</td>
<td>43 (1, 2)</td>
<td>Host UART receive buffer NOT filled</td>
</tr>
<tr>
<td>38400</td>
<td>85 (3)</td>
<td>&gt; Host UART receive buffer</td>
</tr>
<tr>
<td>57600</td>
<td>127 (3)</td>
<td>&gt; Host UART receive buffer</td>
</tr>
<tr>
<td>115200</td>
<td>254 (3)</td>
<td>&gt; Host UART receive buffer</td>
</tr>
</tbody>
</table>

**Note 1:** The Receive Data window time-out causes the CTS pin to be forced high and the MCP215X to process the data packet that has been loaded into the receive buffer.

2: Any byte that is being transmitted when the MCP215X CTS signal goes high will be received.

3: The CTS signal will go high when the receive buffer receives 60 bytes or the Receive Data window time-out occurs. The MCP215X will start processing the data packet when 64 bytes have been received. The host controller MUST NOT send more than 64 bytes for each CTS window (or data may be lost).
CTS STATE AND THE RECEIVE BUFFER

Once a link has been established, the level (high, low) of the CTS pin is influenced by the state of the MCP215X host UART receive buffer.

**Figure 2** shows the operation of the CTS pin with respect to the 22 ms Receive Data window and the number of bytes that have been received into the host UART receive buffer. This illustrates a situation when the host UART baud rate is either 9600 baud or 19200 baud. This can also occur at higher baud rates, if the host controller is not "streaming" the data quickly. **Figure 5** shows the CTS-to-CTS timing that occurs in this case.

**Figure 3** shows when the host controller transfers at least 60 bytes in this CTS window, but does not reach the 64-byte receive buffer limit. So the data is only processed when the Receive Data window closes. CTS-to-CTS timing will be slightly longer than what is shown in **Figure 5** due to the additional data that is processed and transmitted on TXIR.

**Figure 4** illustrates a situation where the host controller is "streaming" the data quickly and the baud rate is 38400, 57600 or 115200. Once the 64 data bytes have been received by the MCP215X, the data is processed and then transmitted on the TXIR pin. Therefore, for the greatest data throughput, a host UART baud rate of 115200 is recommended. **Figure 6** also shows the CTS-to-CTS timing (30.567 ms) that occurs in this case (64 bytes after falling edge of CTS). **Figure 5** shows the CTS-to-CTS timing (40.417 ms) that occurred when the receive buffer was not filled (40 bytes while CTS is low). So the transfer time of 40 bytes per CTS low takes about 10 ms longer than does 64 bytes transferred after CTS falling edge.

The CTS signal was designed to go high once 60 bytes were received so that a host controller with a transmit FIFO of 4 bytes (or less) could interface to the MCP215X.

To demonstrate the effects of the host controller firmware on throughput from a Secondary Device to a Primary Device (S → P), two programs have been implemented that use different techniques for transmitting data with respect to the CTS signal.

The techniques used in these programs are:

1. Transmit data while the CTS pin is low (see **Figure 3**).
2. Transmits 64 bytes with minimal byte-to-byte spacing (see **Figure 4** and **Figure 6**) after the falling edge of the CTS signal.
FIGURE 2: CTS WAVEFORM FOR < 60 BYTES INTO UART RECEIVE FIFO

- Receive Data Window (22 ms)
- CTS
- Receive Buffer Empty
- MCP215x can receive data
- Receive buffer has < 60 bytes.
- Receive buffer will accept a byte that is being currently transmitted.
- Data in receive buffer is processed and then the frame will be transmitted.

FIGURE 3: CTS WAVEFORM FOR BETWEEN 60 AND 64 BYTES INTO UART RECEIVE FIFO

- Receive Data Window (22 ms)
- CTS
- Receive Buffer Empty
- MCP215x can receive data
- Receive buffer has < 64 bytes.
- Receive buffer will accept a byte that is being currently transmitted.
- Data in receive buffer is processed and then the frame will be transmitted.
- Receive buffer has 60 bytes, CTS pin driven high.

FIGURE 4: CTS WAVEFORM FOR 64 BYTES INTO UART RECEIVE FIFO

- Receive Data Window Closed Early (<22 ms)
- CTS
- Receive Buffer Empty
- MCP215x can receive data
- Receive buffer has 60 bytes, CTS pin driven high.
- Receive Buffer Full (64 bytes)
- The Receive Data window is closed (early), the data in the receive buffer is processed and then the frame will be transmitted.
FIGURE 5: CTS TO CTS WHEN MCP215X RECEIVE BUFFER IS NOT FILLED (40 BYTES)

40.417 ms
FIGURE 6: CTS TO CTS WHEN MCP215X RECEIVE BUFFER IS FILLED (64 BYTES)

30.567 ms
MCP215X Data Processing

After the Receive Data window has closed, the MCP215X will process the data in the receive buffer to generate the CRC and construct the transmit frame.

When the host controller fills the 64-byte receive buffer, this processing time is from TX to TXIR (see Figure 7).

When the host controller does not fill the 64-byte receive buffer, the processing time starts at the end of the 22 ms timer and ends at TXIR (see Figure 8). This extends the time measured from TX to TXIR.

Table 3 shows the delta time of the Receive Data window when 64 bytes are transmitted by the host controller (which forces the Receive Data window closed). Table 3 also illustrates when 63 bytes are transmitted by the host controller (so the Receive Data window is closed after the 22 ms time-out).

When the host UART is at 115200 baud, the host controller could transmit 64 bytes in approximately 5.6 ms. So, for a transmission of 64 bytes, the data processing begins at about 5.6 ms after the CTS falling edge. For a transmission of 63 bytes, the data processing begins at approximately 22 ms after the CTS falling edge. Since the processing time will be approximately the same between 63 and 64 bytes, the transmission of that one extra byte can save about 16.4 ms from the time the activity on TX ends to the activity on TXIR begins.

Table 4 gives the times, between the end of activity on TX to the beginning of activity on TXIR, that were measured in Figure 7 and Figure 8.

Figure 7 shows the timings when 64 bytes are transmitted after the falling edge of the CTS signal.

Figure 8 shows the timings when data is transmitted while the CTS signal is low. The MCP215X will force the CTS signal high after the receive buffer has 60 bytes. So, depending on how quickly the host controller is testing the CTS signal before transmitting the data, the MCP215X receive buffer will have either 60 or 61 data bytes in the transmitted packet.

### Table 3: RECEIVE DATA WINDOW TIMES

<table>
<thead>
<tr>
<th>Baud Rate</th>
<th>64 Byte TX Receive Data Window (ms)</th>
<th>63 Byte TX Receive Data Window (ms)</th>
<th>Δ Receive Data Window Time (22 ms - TX Time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>38400</td>
<td>16.7</td>
<td>22</td>
<td>5.3 ms</td>
</tr>
<tr>
<td>57600</td>
<td>11.2</td>
<td>22</td>
<td>10.8 ms</td>
</tr>
<tr>
<td>115200</td>
<td>5.6</td>
<td>22</td>
<td>16.4 ms</td>
</tr>
</tbody>
</table>

### Table 4: MCP215X TO PALM TUNGSTEN T2 PDA TIMINGS: TX TO TXIR (MEASURED)

<table>
<thead>
<tr>
<th>Timing</th>
<th>Condition</th>
<th>Figure #</th>
<th>Time (ms)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX to TXIR</td>
<td>Transmit 64 Bytes after CTS Falling Edge</td>
<td>Figure 7</td>
<td>6.15</td>
<td>Figure at 5 ms/div.</td>
</tr>
<tr>
<td></td>
<td>Transmit while CTS Low (Receive buffer loaded with 60 or 61 bytes)</td>
<td>Figure 8</td>
<td>24.57</td>
<td>Figure at 10 ms/div.</td>
</tr>
</tbody>
</table>

**Note 1:** Data packet processing starts once the 64th byte has been received (in MCP215X receive buffer).

**Note 2:** Data packet processing starts once the MCP215X Receive Data window timer times-out (22 ms).
FIGURE 7: TX TO TXIR TIME FOR 64 BYTE DATA TRANSMISSION PER CTS

6.15 ms

Receive Buffer Window

Frame Processing Time
FIGURE 8: TX TO TXIR TIME FOR DATA TRANSMISSION WHILE CTS IS LOW

24.5667 ms

22 ms

(Receive Buffer Window)

Frame Processing Time

24.5667 ms

524 PM
Primary Device Response Time

Once the last bit has been transmitted on TXIR, the MCP215X waits for a response from the Primary Device. Since this response time also affects the data throughput, whatever can be done to minimize this response time from the Primary Device will increase the data throughput. Figure 9 and Figure 10 show response times from the end of TXIR to the beginning of RXIR for the Palm™ Tungsten™ T2 running the 215xDemo program from Application Note 888, “Programming the Palm OS® for Embedded IR Applications”, DS00888. These times will vary from capture-to-capture due to the asynchronous nature of the communication and PDA operation.

Table 5 shows the times that were measured in Figure 9 and Figure 10.

<table>
<thead>
<tr>
<th>Timing</th>
<th>Condition</th>
<th>Figure #</th>
<th>Time (ms)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXIR to RXIR</td>
<td>Transmit 64 Bytes after CTS Falling Edge</td>
<td>Figure 9</td>
<td>9.32</td>
<td>Figure at 5 ms/div (Note 1)</td>
</tr>
<tr>
<td>TXIR to RXIR</td>
<td>Transmit while CTS low (Receive buffer loaded with 60 or 61 bytes)</td>
<td>Figure 10</td>
<td>11.1</td>
<td>Figure at 10 ms/div (Note 1)</td>
</tr>
</tbody>
</table>

Note 1: The TXIR to RXIR delay time is determined by the characteristics of the Primary Device.

FIGURE 9: TXI TO RXIR TIME FOR 64 BYTE DATA TRANSMISSION PER CTS

![Graph showing TXIR to RXIR time for 64 byte data transmission per CTS](image-url)
FIGURE 10: TXIR TO RXIR TIME FOR DATA TRANSMISSION WHILE CTS IS LOW

<table>
<thead>
<tr>
<th>File</th>
<th>Control</th>
<th>Setup</th>
<th>Measure</th>
<th>Analyze</th>
<th>Utilities</th>
<th>Help</th>
</tr>
</thead>
</table>

![Graph showing TXIR to RXIR time for data transmission while CTS is low.](image)

- TXIR to RXIR time: 11.0667 ms
Measured Data Throughput

An Agilent® 54825A oscilloscope was used to capture the data transfer of 1000 bytes from the Secondary Device to the Primary Device (S → P). This 4-channel scope allowed the CTS, TX, TXIR and RXIR signals to be monitored.

The data throughput time was measured at the host controller TX pin. The time measurement started at the first transmitted bit on the TX pin (a Start bit) to the last data bit on the TX pin (actually, the last ‘0’ bit in the data). Given the data that was transmitted, there are no more than 2 bits of error (a data ‘1’ bit plus the Stop bit). The Primary Device used was a Palm Tungsten T2 PDA, while the embedded system (Secondary Device) was an internal Microchip demo board.

Appendix C: “Oscilloscope Screen Captures - MCP215X Rev C” and Appendix D: “Oscilloscope Screen Captures - MCP215X Rev B” show the captured waveform for the transfer of 1000 bytes from the embedded system to the Primary Device.

Figure C-1 shows an oscilloscope screen capture of a 1000-byte transfer from a host controller to the Primary Device, where the host controller transmits 64 bytes of data when it detects the falling edge of CTS. Notice the time from the falling edge of CTS pin to the activity on the TXIR pin.

Figure C-2 shows an oscilloscope screen capture for a 1000 byte transfer from a host controller to the Primary Device, where the host controller only transmits data when the CTS signal is low. The time from the falling edge of the CTS pin to the activity on the TXIR pin is much longer in this case, since the data is not processed until the 22 ms Receive Data window is closed. This, then, adds a delay from the time the host controller stops sending data until the MCP2150 starts to process the data.

Table 6 shows the transfer times of 1000 bytes from the embedded system to the Primary Device (Palm Tungsten T2). These times were measured from the oscilloscope screen captures (see Appendix C: “Oscilloscope Screen Captures - MCP215X Rev C” and Appendix D: “Oscilloscope Screen Captures - MCP215X Rev B”).

Table 7 takes the times shown in Table 6 and converts them to the approximate data throughput. The host controller program is shown in Appendix B: “PIC16F877 Source Code”.

### TABLE 6: 1000 BYTES TRANSFER TIME (NOTE 1)

<table>
<thead>
<tr>
<th>Host Controller to MCP215X Technique</th>
<th>MCP2150</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rev C</td>
<td>Rev B</td>
</tr>
<tr>
<td>Transfer 64 Bytes after CTS falling edge (S → P)</td>
<td>454 (2)</td>
<td>755 (4)</td>
</tr>
<tr>
<td>Transfer Bytes while CTS is low (S → P)</td>
<td>803 (3)</td>
<td>802 (5)</td>
</tr>
</tbody>
</table>

**Note 1:** Transfer time is dependent on many factors, including the characteristics of the Primary Device. These measurements were taken using a Palm Tungsten T2 PDA. Due to the asynchronous nature of communication, these times can vary from transfer to transfer. Use of a different Primary Device may also lead to different transfer times.

1: See Figure C-1.
2: See Figure C-2.
3: See Figure D-1.
4: See Figure D-2.
5: See Figure D-2.
6: Appendix B of the MCP215X Device Errata (DS80139) describes how to determine the revision of the device.

### TABLE 7: THROUGHPUT

<table>
<thead>
<tr>
<th>Host Controller to MCP215X Technique</th>
<th>Theoretical(1) (115200 bit/sec)</th>
<th>MCP2150</th>
<th>Units(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rev C</td>
<td>Rev B</td>
<td></td>
</tr>
<tr>
<td>Constant Streaming (2) (No IrDA standard IrCOMM protocol overhead)</td>
<td>11,520</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Transfer 64 Bytes after CTS falling edge (S → P)</td>
<td>—</td>
<td>2200</td>
<td>1324</td>
</tr>
<tr>
<td>Transfer Bytes while CTS is low (S → P)</td>
<td>—</td>
<td>1245</td>
<td>1246</td>
</tr>
</tbody>
</table>

**Note 1:** Each 8-bit data byte also includes a Start bit and a Stop bit. So 8 bits of data requires a transfer of 10 bits.
2: This would be the transfer rate for a constant stream of data using an encoder/decoder (such as the MCP2120).
Test System Overview

EMBEDDED SYSTEM HARDWARE
The embedded system is a board that contains a PIC16F877 as the host controller, a MCP215X device and an integrated transceiver. The board is an internal Microchip demo board.

EMBEDDED SYSTEM (HOST CONTROLLER) FIRMWARE
To demonstrate the effects of the host controller firmware on throughput from a Secondary Device to a Primary Device, two programs have been implemented. The PIC16F877 firmware program (in Appendix B: “PIC16F877 Source Code”) uses conditional assembly to select between these two programs. The value of the TXMode constant determines which program to assemble. A variable called "TableCNTR" determines how many times to transmit the 250-byte table.

These programs are:
1. Transmit data while the CTS pin is low (TXMode = H’0’).
2. Transmits 64 bytes with minimal byte-to-byte spacing (as shown in Figure 4) after the falling edge of the CTS signal (TXMode = H’1’).

In Program #1, the Receive Data window only closes once the 22 ms timer has completed (as shown in Figure 3).

In Program #2, the Receive Data window closes once the 64th byte has been received by the MCP215X (as shown in Figure 4 and Figure 6).

PRIMARY DEVICE
Table 8 shows the characteristics of the Primary Device used, as well as the application program on the Primary Device.

<table>
<thead>
<tr>
<th>TABLE 8: PALM SYSTEM SETUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
</tr>
<tr>
<td>Manufacturer:</td>
</tr>
<tr>
<td>Model:</td>
</tr>
<tr>
<td>O.S. Version:</td>
</tr>
<tr>
<td>Software Application</td>
</tr>
</tbody>
</table>
APPENDIX A: PIC16F877
FIRMWARE CODE

DESCRIPTION

The “215x Tput V001.ASM” application firmware has two main conditional assembly switch constants. These are:

1. "MCP215X", and
2. "TXMode".

The "MCP215X" constant selects which MCP21XX device to assemble for, while the "TXMode" constant selects the in which the host controller will transmit data to the MCP215X. This allows one file to be used for four different configurations.

The host controller timings (such as the UART) are based on a 20 MHz device frequency.
APPENDIX B: PIC16F877 SOURCE CODE

FIGURE B-1: “215X TPUT V001.ASM” - PAGE 1

LIST    C=132
include P16F877.inc
ERRORLEVEL -J02
;
;******************************************************************************************

; Software License Agreement
;
; The software supplied herewith by Microchip Technology Incorporated
; (the "Company") is intended and supplied to you, the Company's
; customer, for use solely and exclusively with products manufactured
; by the Company.
;
; The software is owned by the Company and/or its supplier, and is
; protected under applicable copyright laws. All rights are reserved.
; Any use in violation of the foregoing restrictions may subject the
; user to criminal sanctions under applicable laws, as well as to
; civil liability for the breach of the terms and conditions of this
; license.
;
; THIS SOFTWARE IS PROVIDED IN AN "AS IS" CONDITION. NO WARRANTIES,
; WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED
; TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
; PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. THE COMPANY SHALL NOT,
; IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR
; CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
;
;******************************************************************************************

; SELECT THE MCP215x Device to interface (Host UART Signals) to.
; This code supports the MCP2155 and MCP2150. The conditional assembly is
; defined here. The allowable choices in this version are 50h, 55h or 40h
;
MCP215X   equ     H'40'   ; assemble for MCP2140 (same as Flow Control as MCP2155)
MCP215X   equ     H'55'   ; assemble for MCP2155
MCP215X   equ     H'50'   ; assemble for MCP2150
;
TXMode    equ     H'1'    ; assemble for Transfer 64 Bytes per CTS falling Edge
TXMode    equ     H'0'    ; assemble for Transfer while CTS is Low
;
;
FIGURE B-2: “215X TPUT V001.ASM” - PAGE 2

```asm
; The use of these Assembler Directives is to verify that a valid target product was selected for the Firmware generation. If not, an ERROR MESSAGE will be generated.
;
if ( MCP215X != H'55' && MCP215X != H'50' && MCP215X != H'40' )
  error "MCP21xx Device Selected NOT VALID"
endif

if ( TXMode != H'1' && TXMode != H'0' )
  error "Data Transfer Mode Selected NOT VALID"
endif
if MCP215X==H'50'
  messg "MCP2150 has been Selected"
endif

if MCP215X==H'55'
  messg "MCP2155 has been Selected"
endif

if MCP215X==H'40'
  messg "MCP2140 has been Selected"
endif

if TXMode==H'0'
  messg "Data Transfer Mode is while CTS is low"
endif

if TXMode==H'1'
  messg "Data Transfer Mode is 64 Bytes after CTS Falling Edge"
endif

******************************************************************************
Revision History
1.00 03/15/04  S -> P configured to do multiple of 250 byte transmissions.
   This allows data throughput to be analysed and to ensure that no bytes are "lost" when sending a continuous stream of data.
   This program can be assembled for one of 2 modes of host controller to MCP215x communication:
   1. Send 64 bytes on each CTS falling edge
   2. Send data bytes while CTS is low
******************************************************************************
MCP2150 Developer's Board with PICDEM-2 Demo Board Demo
PIC16F877 code to interface to MCP215x Controller
Program resets MCP215x and waits for "IR connection"
Once a connection is established, the host controller monitors the CTS signal (for a Low) to sends a stream of bytes.
Once the table has completed being transmitted, the program "stops" (that is the program loops forever)

NOTE: The MCP2150 Developer's Board requires that the RESET pin of the MCP215x device be disconnected from the MCP2150 Developer Board circuitry and connected to the specified I/O pin of the PIC16F877 device on the PICDEM(tm)-2 Demo Board

PICDEM-2 Requirements
Device: PIC16F877
Clock Frequency: 20.00 MHz
UART: User-Defined Baud

MCP215x Requirements
Clock Frequency: 11.0952 MHz
```
### FIGURE B-3: “215X TPUT V001.ASM” - PAGE 3

#### PIC16F877 PORT Functions

<table>
<thead>
<tr>
<th>PORTA</th>
<th>Function</th>
<th>---</th>
<th>---</th>
<th>SW2</th>
<th>SW3</th>
<th>LCDRS</th>
<th>LCDRW</th>
<th>LCDE</th>
<th>XferStr</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRIS Direction</td>
<td>---</td>
<td>---</td>
<td>I</td>
<td>I</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td></td>
<td>Initial value</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
</tbody>
</table>

#### PORTB (For MCP2150)

<table>
<thead>
<tr>
<th>PORTB</th>
<th>Function</th>
<th>PktStr</th>
<th>ByteStr</th>
<th>RI</th>
<th>CD</th>
<th>LCDD7</th>
<th>LCDD6</th>
<th>LCDD5</th>
<th>LCDD4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRIS Direction</td>
<td>O</td>
<td>O</td>
<td>I</td>
<td>I</td>
<td>I/O</td>
<td>I/O</td>
<td>I/O</td>
<td>I/O</td>
</tr>
<tr>
<td></td>
<td>Initial value</td>
<td>H</td>
<td>L</td>
<td>---</td>
<td>---</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
</tbody>
</table>

#### PORTB (For MCP2155/MCP2140)

<table>
<thead>
<tr>
<th>PORTB</th>
<th>Function</th>
<th>PktStr</th>
<th>ByteStr</th>
<th>RI</th>
<th>CD</th>
<th>LCDD7</th>
<th>LCDD6</th>
<th>LCDD5</th>
<th>LCDD4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRIS Direction</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>I</td>
<td>I/O</td>
<td>I/O</td>
<td>I/O</td>
<td>I/O</td>
</tr>
<tr>
<td></td>
<td>Initial value</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
</tbody>
</table>

#### PORTC

<table>
<thead>
<tr>
<th>PORTC</th>
<th>Function</th>
<th>RX</th>
<th>TX</th>
<th>NA</th>
<th>NA</th>
<th>NA</th>
<th>DTR</th>
<th>DSR</th>
<th>RST215X</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRIS Direction</td>
<td>I</td>
<td>I</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>I</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Initial value</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
</tbody>
</table>

#### PORTD

<table>
<thead>
<tr>
<th>PORTD</th>
<th>Function</th>
<th>LED7</th>
<th>LED6</th>
<th>LED5</th>
<th>LED4</th>
<th>LED3</th>
<th>LED2</th>
<th>LED1</th>
<th>LED0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRIS Direction</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td></td>
<td>Initial value</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
</tbody>
</table>

#### PORTE

<table>
<thead>
<tr>
<th>PORTE</th>
<th>Function</th>
<th>---</th>
<th>---</th>
<th>---</th>
<th>---</th>
<th>---</th>
<th>NA</th>
<th>CTS</th>
<th>RTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRIS Direction</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td></td>
<td>Initial value</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
</tbody>
</table>

---

```c
#define rxd PORTC, 7    ; input, serial data from MCP215x
#define txd PORTC, 6    ; output UART overrides TRIS bit,
                     ; serial data to MCP215x
#define cts PORTE, 1    ; input, MCP215x is ready to receive data
#define rts PORTE, 0    ; output, PIC16F877 (host controller) is ready
                     ; to receive data. At RESET,
                     ; Low for pgm mode, High for normal
#define dtr PORTC, 2    ; output, force high or low (LOW). At RESET,
                     ; High for pgm mode, Low for normal
#define dsr PORTC, 1    ; input, Indicates MCP2150 has completed
                     ; Reset, or
                     ; Indicates MCP2155 has established
                     ; a valid link,
                     ; high for no link, low for link
```

---

### Configuration Bits

```c
#maybe CONFIG _CP_OFF & _PWRTE_ON & _HS_OSC & _WDT_OFF & _LVP_OFF
#maybe _IDLOCS H'0010'
```

---

```c
#define reset H'00'    ; Reset vector
```

---

```c
/* Configuration Bits */
```
#define cd PORTB, 4  ; input, Indicates MCP2150 has established
                  ; a valid link,
                  ; high for link, low for no link; or
                  ; output, The MCP2155 communicates this value
                  ; is to the Primary Device.
                  ; For this application, this signal
                  ; (CD) can be static
#define ri PORTB, 5   ; input, MCP2150 - Driven high
                  ; output, MCP2155 - This value is communicated
                  ; to the Primary Device.
                  ; For this application, this signal
                  ; (RI) can be static
#define rst215x PORTC,0 ; output, used to reset the MCP2155
                  ; high for normal operation, low to
                  ; RESET device
#define en PORTE, 2   ; output, used to enable MCP215x
                  ; High for enable, Low for disable
#define baud1 PORTD, 7 ; output
#define baud0 PORTD, 6 ; output, 11 = 115200
                  ; 10 =  57600
                  ;  1 =  19200
                  ;  0 =   9600
#define XferStrobe PORTA, 0 ; Output,
                  ; Used in Program 2 to indicate Start
                  ; and Stop of Receiving Data
                  ; Used in Program 3 to indicate Start
                  ; and Stop of Transmitting Data
#define ByteStrobe PORTB, 6 ; Output, Used in Program 2 and 3 to
                  ; indicate Byte Count
#definePacketStrobe PORTB, 7 ; Output, Packet Strobe signal

#define sw2 PORTA, 5 
SW2 EQU 5
#define sw3 PORTA, 4 
SW3 EQU 4

; PORTA, PORTC, PORTD, and PORTE Data Direction values
ddra equ B'00110000' ; Data Direction for PORTA
ddrc equ B'11000010' ; Data Direction for PORTC
ddrd equ B'00000000' ; Data Direction for PORTD
ddre equ B'00000010' ; Data Direction for PORTE

;***
;*** Conditional assembly on PORTB, PORTC, and PORTE Data Direction values
;*** for Host UART
;***
  if MCP215X==H'50'
    ddbb equ B'00110000'
    ; CD and RI are INPUTs
  endif
  if (MCP215X==H'55' || MCP215X==H'40')
    ddbb equ B'00000000'
    ; CD and RI are OUTPUTs
  endif
  cffo3t equ B'11001000' ; option reg setup
ICD   EQU     1        ; When ICD is TRUE, Address 0x00
          ; must be a NOP and RB7:RB6 are used
          ; by the ICD module (override TRIS
          ; settings).
TermEmulator  EQU  1     ; When the PDA uses a Terminal Emulator program

Constants

Host UART Data Rate/BRG Value (BRGH = 1)
; SPBRG Value
; Baud Rate @ 20MHz
9600   D'129'
19200  D'64'
57600  D'21'
115200 D'10'

SPBRG Value
B9600at20MHz  EQU    D'129'
B19200at20MHz  EQU    D'64'
B57600at20MHz EQU    D'21'
B115200at20MHz EQU    D'10'

Registers

cblock    H'20'
delreg          ; register for timing delays & scratchpad
MENUCNTR        ; Pointer to the Menu character to send
MENUBYTES       ; This is the # of bytes in the MENU
hostdata        ; Host Data to Transmit
BYTERX          ; Received Byte on UART
RXBufferSize    ; The size of the MCP215x Host UART Receive Buffer
; LCD variables
RXTableTXCount  ; The counter for the number of times the 256-byte
                ; table is transmitted
                   ; (250-byte table + 6 extra characters)
B256CNTR        ; Counter to display on PORTD how many times the 256-byte
                ; table is transmitted
ERRCNT
COUNTER
delay
ptr_pos
ptr_count
SodaCNTR
CandyCNTR
hundreds        ; 8-bit to 3-Digit conversion, Hundreds place
tens            ; Tens place
REMAINDER       ; Ones place
temp3
temp2
temp1
temp
temp_wr
temp_rd
endcode
AN927

FIGURE B-6: “215X TPUT V001.ASM” - PAGE 6

******************************************************************************
or\n    H'00'         ; use 00h as reset vector
if ICD
    NOP                   ; Use of the ICD requires the first
endif                     ;   instruction to be a NOP
        goto   START
;
;
******************************************************************************

;       Start Routine
;       Initialization is done here
;       (Ports, Option Register, and UART).
;       Option Register has:
;            RBPU disabled,
;            RB0 on Rising Edge,
;            TMRO Clock Source internal
;            TOCKI inc on L-to-H
;            Prescaler assigned to WDT
;            WDT = 1:1
;******************************************************************************

STARTclrf    STATUS          ; Bank 0
          movlw   0xFF            ; Force PORTs to display High when configured
          movwf   PORTA           ;    as Output
          movwf   PORTB           ;
          movwf   PORTC           ;
          movwf   PORTD           ;
          movwf   PORTE           ;
          bsf     STATUS, RP0     ; Bank 1
          movlw   ddra
          movwf   TRISA           ; Configure PORTA
          movlw   ddb\n
          movwf   TRISB           ; Configure PORTB
          movlw   ddc\n
          movwf   TRISC           ; Configure PORTC
          movlw   ddd\n
          movwf   TRISD           ; Configure PORTD
          movlw   dde\n
          movwf   TRISE           ; Configure PORTE
/
          movlw   cfgopt          ; setup option reg
          movwf   OPTION_REG

MOVlw    0x0E            ; Configure A/D Module so that RA0 is Analog,
MOVwF    ADCON1           ; ALL other pins Digital; Result is Left-Justified
; Initialize UART
;    BRGH = 1
;    8-bit
;    TX Enabled
;    Async. Operation
;    Continuous receive
;    Enable UART
; Write value (0xFF) to PORTB
;    (ICD uses RB7:RB6, so with ICD 0x3F will be displayed)
;
movlw 0x24            ; BRGH = 1, 8-bit, TX Enabled, Async.
movwf TXSTA           ;
if (MCP215X==H'55' || MCP215X==H'50')          ;
movlw B115200at20MHz ;
endif
if (MCP215X==H'40')         ;
movlw B9600at20MHz ;
endif
movwf SPBRG           ;
clrf STATUS          ; Bank 0
movlw 0x90            ; Enable serial port, continuous receive
movwf RCSTA           ;
clrif PORTD           ; clear outputs (Display on LEDs)
MOVLW B'10000001'     ; Configure A/D Module so that RA0 is Selected,
MOVWF ADCON1          ; A/D clock is /32, and A/D is ON.
;
; Initialize MCP215x Flow Control signals,
; Reset MCP215x,
; BAUD1:BAUD0 Baud Rate
; 0 0  9600
; 0 1  19200
; 1 0  57600
; 1 1  115200
;
; Delay for 1us,
; then release Reset
;
RESET215X
  bsf en              ; enable MCP215x
  bsf baud1
  bsf baud0          ; 11 = 115200
  bcf dtr          ; dtr low is the normal mode for the MCP215x
  bcf rst215x       ; Reset the MCP215x
  nop                ; Delay to ensure MCP215x RESET pin is
  nop                ; detected (driven) low
  nop
  nop
  bsf rst215x       ; Release the MCP215x from RESET

; MCP215x requires 2000 Tosc (at 11.0592MHz = 180 us)
; delay before the device initialization should be
; complete
;
movlw H'FF'            
call DELAY
BSF XferStrobe       ; Output, Default High
BCF ByteStrobe       ; Output, Default Low
BSF PacketStrobe     ; Output, Default High
The following delay is done only on the MCP2150, since the MCP2150 has a signal (DSR) which is used to indicate if the MCP2150 has completed RESET. There is no corresponding signal on the MCP2155.

```assembly
if MCP215X==H'50' ; Conditional Assemble for MCP2150
    ; Has MCP2150 completed initialization?,
    ; if not continue to wait
    WAIT2150 ; Now test the state of the DSR pin
    btfss dsr
        goto WAIT2150 ; NO, wait more time
    goto MAIN ; YES, continue
    endif ; End of Conditional Assemble for MCP2150

;*****************************************************************************
; Main Routine - MCP215x Has completed initialization
;
; Wait for MCP215x to establish a link. Indicate to MCP215x to Send byte that established link
; Wait for byte to be received by PIC16F877 (while waiting, test to ensure link is still present)
; Read byte and display on PORTB
; Call Subroutine which Transmits entire Table of Data
; Then Loop forever.
;
;*****************************************************************************

MAIN
WaitForSW2SW3
    CLRF STATUS ; Bank 0
    ;
;****************************************************************************
;*****************************************************************************
;  Program3 Routine - Transmit a Table of 256 bytes a LoopCounter number
;  of times. The transmissions starts after receiving a
;  character from an IrDA(r) standard Primary device. Data starts
;  transmitting when the CTS signal goes low and stops
;  transmitting that data packet when either:
;  1. CTS goes high
;  2. 64 bytes have been transmitted
;  The transmit mode is determined by user input (SW2/SW3)
;
; Wait for MCP215x to establish a link.
; Indicate to MCP215x to Send byte that established link
; Wait for byte to be received by PIC16F877
; (while waiting, test to ensure link is still present)
; Read Byte and display on PORTB
; Clear Timer to measure the time to send data.
; Start Timer
; Call Subroutine which Transmits entire Table of Data
; Stop Timer and display bytes/time on LCD
; Then Loop forever.
;
;*****************************************************************************
;
#define      TableCNTR   D'4'   ; Send 250 Byte Table this # of times
Program3

        BSF     XferStrobe      ; Set the Stobe High, can trigger Oscilliscope when
        ;    XferStrobe goes Low.
        BCF     ByteStrobe      ; Set the Stobe Low, can trigger Oscilliscope when
        ;    ByteStrobe goes High.
if (TXMode==H'1')            ; assemble for Transfer 64 Bytes per CTS falling Edge
;
;***************************************************************
;
; This code sends the data as 64 byte blocks after the a falling edge
; has been detected on the CTS signal
;***************************************************************
;
Xfer64Byte:

if (MCP215X==H'50')           ; Conditional Assemble for MCP2150
   P3X64WAITCD
   btfsc    cd              ; Has the MCP2150 made a link?
   goto    P3X64WAITCD     ; NO, wait for a link to be established
endif                       ; End of Conditional Assemble for MCP2150
;
if (MCP215X==H'55' || MCP215X==H'40')    ; Conditional Assemble for MCP2155/MCP2140
   P3X64WAITDSR
   btfsc    dsr             ; Has the MCP2155/MCP2140 made a link?
   goto    P3X64WAITDSR    ; NO, wait for a link to be established
   bcf     cd              ;*** Light the CD LED to show that DSR was low
endif                       ; End of Conditional Assemble for MCP2155/MCP2140
;
bcf     rts             ; YES, Host can receive the "Dummy" byte
;
P3X64RXWAIT1                    ; The program will wait here before sending
;   the "data" out. This is because to
;   create a link, many devices (PDAs)
;   must transmit at least one data byte.

btfsc    PIR1, RCIF      ; Has a byte been received yet?
   goto    P3X64GOTBYTE1   ; YES
   if (MCP215X==H'50')           ; Conditional Assemble for MCP2150
      btfsc   cd              ; NO, so test if MCP2150 link still active?
   endif                       ; End of Conditional Assemble for MCP2150
;
if (MCP215X==H'55' || MCP215X==H'40')    ; Conditional Assemble for MCP2155/MCP2140
   btfsc    dsr             ; NO, so test if MCP2155/MCP2140 link still active?
   endif                       ; End of Conditional Assemble for MCP2155/MCP2140
;
goto    MAIN              ; NO, Link was lost, so start over
   goto    P3X64GOTBYTE1     ; YES, Have not received a byte yet
   movf    RCREG, W        ; Get byte into W register and this clears
   movwf   PORTD           ; the RCIF flag. Link is established,
   call    SENDX64DATA    ; display on PORTB then send bytes
   call    SENDX64DATA    ; Send the MENU character string
GOTO    Program3        ;
; Send String (MENU) routine
;
; This routine Transmits the String (MENU) Data to the MCP215x
; The First byte of the String (Menu) is the length of the Data
; MENUCNTR is pointer into Table MENU to get the Table lookup data
; MENUBYTES contains the number of bytes of the String (MENU) still to
; be transmitted. hostdata contains the value returned from MENU, to
; be transmitted
;
; Determine if PIC16F877 can transmit UART data (monitor CTS signal)
; After Calling Serial Send Routine, decrement the number of bytes to send
; Test to see if still more bytes to send.
;
; CTS Window 22ms.
; Baud Rate     Max Bytes Transferred
; 9600          23
; 19200         46
; 57600         134 Exceeds MCP215x Buffer Size of 64 Bytes -
;                  Ensure only 64 bytes are sent during MCP215x
;                  Transmit Window
; 115200        276 Exceeds MCP215x Buffer Size of 64 Bytes -
;                  Ensure only 64 bytes are sent during MCP215x
;                  Transmit Window
;
; SENDX64DATA                     ; Send until Buffer is full
; MOVLW TableCNTR       ; Send 256 Byte Table this # of times
; MOVLWF RXTableTXCount  ;  1x =  250 Bytes
;                              ;  4x = 1000 Bytes
;                              ;  8x = 2000 Bytes
;                              ; 10x = 2500 Bytes
; CLRF    B256CNTR        ; # of Tables Transmitted
;
; SENDX64DataLoop                   ; Send until Buffer is full
; clrf    MENUCNTR          ; MENU Counter = 0
; call    MENU              ; Get next byte of data from the MENU Data Table
; movwf   MENUBYTES         ; This is the # of bytes in the MENU
;                              ; (Menu size must be > 1, MENUBYTES = 0 --> 256 )
; INCF    B256CNTR, F       ; Increment the # of Tables Transmitted
;
; MENUX64LOOPRXBufLoad
; MOVLW 0x40              ; Load 64 into the Counter RXBufferSize
; MOVLWF RXBufferSize      ;
; if (MCP215X==H'55' || MCP215X==H'50')

X64CTSWait1
BTFSS cts ; Wait for CTS to be High
GOTO X64CTSWait1 ; (need to ensure that we detect CTS falling edge)
endif

X64CTSWait2
btfsc cts ; Wait for CTS to be Low
goto X64CTSWait2 ; (waiting to detect CTS falling edge)

MOVF B256CNTR, W ; Load value into PORTD
MOVWF PORTD ;

MENUX64LOOP1
incf MENUCNTR, F ; Point to next location in the MENU
call MENU ; Get next byte of data from the MENU Data Table
movwf hostdata ; Store this byte in register "hostdata"
FIGURE B-12: “215X TPUT V001.ASM” - PAGE 12

```assembly
if MCP215X==H'50'
    ; Conditional Assemble for MCP2150
    btfs cd ; Is the link still active?
endif ; End of Conditional Assemble for MCP2150
;
if (MCP215X==H'55' || MCP215X==H'40') ; Conditional Assemble for MCP2155/MCP2140
    btfs dsr ; Is the link still active?
endif ; End of Conditional Assemble for MCP2155/MCP2140
;
goto RESET215X ; NO, link closed for unknown reason,
    ; RESET MCP215x
BCF XferStrobe ; 240 Byte Strobe
call SERSNDX64 ; YES, Send the Data Byte
;
DECFSZ MENUBYTES, F ; Decrement the number of available bytes
    ; If MENUBYTES = 0, The complete MENU has been sent
GOTO TestRXBufSize ; MENUBYTES != 0, need to check RX Buffer Size
;
TestTXTableCNTR ; MENUBYTES = 0, Do we need to send the table some more?
    DECFSZ RXTableTXCount, F ; If the counter != 0, then we need to send the
        ; MENU table some more times,
        ; but lets check RX Buffer Size
GOTO LoadNextTableX64 ; TestRXBufSize != 0, more tables to send
BSF XferStrobe ; TestRXBufSize = 0, Close Data Transmit Strobe
RETURN ; Back to Program3 loop (S->P Transmission)
;
LoadNextTableX64
    clrf MENUCNTR ; MENU Counter = 0
call MENU ; Get next byte of data from the MENU Data Table
movwf MENUBYTES ; This is the # of bytes in the MENU
        ; (Menu size must be > 1, MENUBYTES = 0 --> 256 )
INCF B256CNTR, F
;
TestRXBufSize
    DECFSZ RXBufferSize, F ; Is RX Buffer Full for this window?
GOTO MENUX641LOOP1 ; NO, More of the MENU needs to be sent
;
TestLP1 GOTO TestLP1 ; YES, wait for next CTS low for more of the MENU to be sent
GOTO MENUX641LOPRXBufLoad ; YES, wait for next CTS low for more of the MENU to be sent
;
```
; Serial Send Routine
; This routine uses the uart to send a single data byte to
; the MCP215x with hardware handshake.
; Data is passed in register called "hostdata"
;
; Wait for UART to be ready for next byte to be loaded
; Ensure the MCP215x can still receive data (test CTS signal)
; Load data to send Data, then return
;
SERSNDX64
  bsf  STATUS, RP0  ; Bank 1
SERSLPX64
  btfss  TXSTA, TRMT  ; check if UART ready
  goto  SERSLPX64   ; not ready, wait
  bcf    STATUS, RP0  ; Bank 0

; if MCP215X==H'50'  ; Conditional Assemble for MCP2150
SERS1X64
  btfsc  cd           ; Is the link still active?
  endif           ; End of Conditional Assemble for MCP2150

; if (MCP215X==H'55' || MCP215X==H'40')  ; Conditional Assemble for MCP2155/MCP2140
SERS1X64
  btfsc  dsr          ; Is the link still active?
  endif          ; End of Conditional Assemble for MCP2155/MCP2140

; goto  RESET215X    ; NO, link closed for unknown reason,
  movf  hostdata,w   ; get the byte to send
  movwf  TXREG       ; send the byte
  BSF   ByteStrobe   ; Byte is sent - Start Strobe
  NOP             ; Some High Time (easier to count)
  NOP
  NOP
  NOP
  NOP
  NOP
  NOP
  BCF   ByteStrobe   ; end of Strobe
  RETURN           ;

; endif
if TXMode==H'0' ; assemble for Transfer while CTS is Low
;
; This code sends the data as while the CTS signal is Low. This does not
; give optimum data throughput, since the Receive Data window is not
; forced closed by having 64 bytes in the MCP215x Host UART Receive Buffer
;
; XferCTSHigh
;
if MCP215X==H'50' ; Conditional Assemble for MCP2150
 P3CTSWAITCD
 btfs c cd ; Has the MCP2150 made a link?
 goto P3CTSWAITCD ; NO, wait for a link to be established
 endif ; End of Conditional Assemble for MCP2150
;
if (MCP215X==H'55' || MCP215X==H'40') ; Conditional Assemble for MCP2155/MCP2140
 P3CTSWAITDSR
 btfs c dsr ; Has the MCP2155/MCP2140 made a link?
 goto P3CTSWAITDSR ; NO, wait for a link to be established
 bcf c cd ;*** Light the CD LED to show that DSR was low
 endif ; End of Conditional Assemble for MCP2155/MCP2140
;
bcf rts ; YES, Host can receive the "Dummy" byte
P3CTSRXWAIT1
 btfs c PIR1, RCIF ; Has a byte been received yet?
 goto P3CTSRXWAIT1 ; YES
;
if MCP215X==H'50' ; Conditional Assemble for MCP2150
 btfs c cd ; NO, so test if MCP2150 link still active?
 endif ; End of Conditional Assemble for MCP2150
;
if (MCP215X==H'55' || MCP215X==H'40') ; Conditional Assemble for MCP2155/MCP2140
 btfs c dsr ; NO, so test if MCP2155/MCP2140 link still active?
 endif ; End of Conditional Assemble for MCP2155/MCP2140
;
goto MAIN ; NO, Link was lost, so start over
goto P3CTSRXWAIT1 ; YES, Have not received a byte yet
P3CTSGOTBYTE1
 movf RCREG, W ; Get byte into W register and this clears
 movwf PORTD ; the RCIF flag. Link is established,
 ; display on PORTB then send bytes
 ;
call SENDCTSDATA ; Send the MENU character string
 ;
GOTO Program3 ; Program Completed transmission of
 ; characters, Start Program 3 over to
 ; see if want to transmit again
 ;
; Send String (MENU) routine
;
; This routine Transmits the String (MENU) Data to the MCP215x
; The First byte of the String (Menu) is the length of the Data
; MENUCNTR is pointer into Table MENU to get the Table lookup data
; MENUBYTES contains the number of bytes of the String (MENU) still to
; be transmitted. hostdata contains the value returned from MENU, to
; be transmitted
;
; Determine if PIC16F877 can transmit UART data (monitor CTS signal)
; After Calling Serial Send Routine, decrement the number of bytes to send
; Test to see if still more bytes to send.
;
; CTS Window 22ms.
; Baud Rate Max Bytes Transferred
; 9600 23
; 19200 46
; 57600 134 Exceeds MCP215x Buffer Size of 64 Bytes –
; Ensure only 64 bytes are sent during MCP215x
; Transmit Window
; 115200 276 Exceeds MCP215x Buffer Size of 64 Bytes –
; Ensure only 64 bytes are sent during MCP215x
; Transmit Window
;
SENDCTSDATA ; Send until CTS is High
MOVLW TableCNTR ; Send 256 Byte Table this # of times
MOVWF RXTableTXCount ; 1x = 250 Bytes
 ; 4x = 1000 Bytes
 ; 8x = 2000 Bytes
 ; 10x = 2500 Bytes
CLRF B256CNTR ; # of Tables Transmitted
;
SENDCTSDataLoop ; Send until Buffer is full
clrf MENUCNTR ; MENU Counter = 0
call MENU ; Get next byte of data from the MENU Data Table
movf MENUBYTES ; This is the # of bytes in the MENU
 ; (Menu size must be > 1, MENUBYTES = 0 --> 256 )
INCF B256CNTR, F ; Increment the # of Tables Transmitted
;
if (MCP215X = H'55' || MCP215X = H'50')

; CTS Wait 1
BTFSS cts ; Wait for CTS to be High
GOTO CTSWait1 ; (need to ensure that we detect CTS falling edge)
endif

; CTS Wait 2
btsc cts ; Wait for CTS to be Low
goto CTSWait2 ; (waiting to detect CTS falling edge)

MOVF B256CNTR, W ; Load value into PORTD
MOVWF PORTD ;
FIGURE B-16: “215X TPUT V001.ASM” - PAGE 16

```
MENUCTSLOOP1
    incf   MENUCNTR, F    ; Point to next location in the MENU
    call   MENU           ; Get next byte of data from the MENU Data Table
    movwf  hostdata       ; Store this byte in register "hostdata"
    
P3CTSMENULP1
    if MCP215X==H'50'      ; Conditional Assemble for MCP2150
        btfsc  cd          ; Is the link still active?
        endif            ; End of Conditional Assemble for MCP2150
    endif            ; Conditional Assemble for MCP2155/MCP2140
    if (MCP215X==H'55'   ||  MCP215X==H'40')
        btfsc  dsr         ; Is the link still active?
        endif           ; End of Conditional Assemble for MCP2155/MCP2140
    endif            ; More of the MENU needs to be sent been sent
    goto   RESET215X      ; NO, link closed for unknown reason,
           ;    RESET MCP215x
    btfsc  cts           ; YES, Can the Host can send Data?
    goto   P3CTSMENULP1   ; NO, wait for MCP215x to be ready for data
    
    BCF    XferStrobe
    
    call   SERSNDCTS      ; YES, Send the Data Byte
    
    DECFSZ  MENUBYTES, F  ; Decrement the number of available bytes
    goto   MENUCTSLOOP1    ; More of the MENU needs to be sent been sent
    
    DECFSZ  RXTableTXCount, F   ; MENUBYTES = 0, Decrement the Transmit
    
    goto   LoadNextTableCTS ; MENU needs to be sent out again
    
    BSF    XferStrobe     ; MENUBYTES = 0 and RXTableTXCount = 0,
           ;    All Data Bytes have been transmitted
    
    RETURN                  ; Back to main loop

LoadNextTableCTS
    clrf    MENUCNTR       ; MENU Counter = 0
    call    MENU           ; Get next byte of data from the MENU Data Table
    movwf   MENUBYTES      ; This is the # of bytes in the MENU
                               ;    (Menu size must be > 1, MENUBYTES = 0 --> 256 )
    INCF    B256CNTR, F    ;
    
    GOTO   MENUCTSLOOP1    ; NO, More of the MENU needs to be sent
```
; Serial Send Routine
; This routine uses the UART to send a single data byte to
; the MCP215x with hardware handshake.
; Data is passed in register called "hostdata"
;
; Wait for UART to be ready for next byte to be loaded
; Ensure the MCP215x can still receive data (test CTS signal)
; Load data to send Data, then return
;
SERSNDCTS
  bsf STAT, RP0       ; Bank 1
SERSLPCTS
  btfss TXSTA, TRMT   ; check if UART ready
  goto SERSLPCTS     ; not ready, wait
  bcf STAT, RP0       ; Bank 0
  
  if MCP215X==H'50'   ; Conditional Assemble for MCP2150
  SERS1CTS
    btfsc cd          ; Is the link still active?
  endif             ; End of Conditional Assemble for MCP2150
  
  if (MCP215X==H'55' || MCP215X==H'40') ; Conditional Assemble for MCP2155/MCP2140
  SERS1CTS
    btfsc dsr         ; Is the link still active?
  endif             ; End of Conditional Assemble for MCP2155/MCP2140
  
    goto RESET215X    ; NO, link closed for unknown reason,
    ; RESET MCP215x
  btfsc cts          ; YES, check the printer handshake
  goto SERS1CTS      ; if CTS=1 then do not print
  movf hostdata,w    ; get the byte to send
  movwf TXREG        ; send the byte
  BSF ByteStrobe     ; Byte is sent - Start Strobe
  NOP                 ; Some High Time (easier to count)
  NOP
  NOP
  NOP
  NOP
  NOP
  NOP
  BCF ByteStrobe     ; end of Strobe
  RETURN             ;
  endif
;
;******************************************************************************
;
; Delay Routine
; Each unit change of delay value changes the delay by 4 cycles.
; The delay value is passed in W.
;
;******************************************************************************
DELAY movwf delreg
DELLP nop
  decfsz delreg,f
  goto DELLP
retlw 0
;
FIGURE B-18: “215X TPUT V001.ASM” - PAGE 18

```
ORG H'0400' ; use 0400h as Start of String Table Routine

STRING TABLE - Used in PROGRAM2 (P->S) and PROGRAM3 (S->P)
This table stores the MENU string, MENUCNTR is the offset.
The string is terminated by a null.
Caution: Do not let MENU String cross 256 word boundary
(that is the reason for the ORG directive)

MENU MOVW HIGH (MENU) ; Get the upper address bits where this table
MOVWF PCLATH ; is located and load into the PCLATH
; register
MOVF MENUCNTR, W ; get the offset
ADDWF PCL, F ; add the offset to PC

MENU2 DT D'250' ; the first byte is the byte count (0 = 256)
; 1 Characters
DT "12345678", 0x0D, 0x0A ; 10 Characters - 10
DT "2BCDEFGH", 0x0D, 0x0A ; 10 Characters - 20
DT "32345678", 0x0D, 0x0A ; 10 Characters - 30
DT "4bcdefgh", 0x0D, 0x0A ; 10 Characters - 40
DT "52345678", 0x0D, 0x0A ; 10 Characters - 50
DT "6BCDEFGH", 0x0D, 0x0A ; 10 Characters - 60
DT "72345678", 0x0D, 0x0A ; 10 Characters - 70
DT "8bcdefgh", 0x0D, 0x0A ; 10 Characters - 80
DT "92345678", 0x0D, 0x0A ; 10 Characters - 90
DT "ABCDDEFGH", 0x0D, 0x0A ; 10 Characters - 100
DT "B2345678", 0x0D, 0x0A ; 10 Characters - 110
DT "Cbcdefgh", 0x0D, 0x0A ; 10 Characters - 120
DT "D2345678", 0x0D, 0x0A ; 10 Characters - 130
DT "EBCDEFGH", 0x0D, 0x0A ; 10 Characters - 140
DT "F2345678", 0x0D, 0x0A ; 10 Characters - 150
DT "1bcdefgh", 0x0D, 0x0A ; 10 Characters - 160
DT "22345678", 0x0D, 0x0A ; 10 Characters - 170
DT "3BCDEFGH", 0x0D, 0x0A ; 10 Characters - 180
DT "42345678", 0x0D, 0x0A ; 10 Characters - 190
DT "5bcdefgh", 0x0D, 0x0A ; 10 Characters - 200
DT "62345678", 0x0D, 0x0A ; 10 Characters - 210
DT "7BCDEFGH", 0x0D, 0x0A ; 10 Characters - 220
DT "82345678", 0x0D, 0x0A ; 10 Characters - 230
DT "9bcdefgh", 0x0D, 0x0A ; 10 Characters - 240
DT "a2345678", 0x0D, 0x0A ; 10 Characters - 250

; NOTE: 0x0D = Carriage Return, 0x0A = Line Feed

ENDPROG NOP

end
```
APPENDIX C: OSCILLOSCOPE SCREEN CAPTURES - MCP215X REV C

FIGURE C-1: 1000-BYTE DATA TRANSFER - TRANSMIT 64 BYTES PER CTS FALLING EDGE

The image shows an oscilloscope capture of a 1000-byte data transfer with 64 bytes transmitted per CTS falling edge. The duration of the transfer is marked as 454.092 ms.
FIGURE C-2: 1000-BYTE DATA TRANSFER - TRANSMIT WHILE CTS IS LOW

803.334 ms
APPENDIX D: OSCILLOSCOPE SCREEN CAPTURES - MCP215X REV B

FIGURE D-1: 1000-BYTE DATA TRANSFER - TRANSMIT 64 BYTES PER CTS FALLING EDGE
FIGURE D-2: 1000-BYTE DATA TRANSFER - TRANSMIT WHILE CTS IS LOW

801.675 ms
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.
## WORLDWIDE SALES AND SERVICE

### AMERICAS

**Corporate Office**  
2335 West Chandler Blvd.  
Chandler, AZ  85224-6199  
Tel: 480-792-7200  
Fax: 480-792-7277  
Technical Support: 480-792-7627  
Web Address: http://www.microchip.com

- **Atlanta**  
  3780 Mansell Road, Suite 130  
  Alpharetta, GA  30022  
  Tel: 770-640-0034  
  Fax: 770-640-0307

- **Boston**  
  2 Lan Drive, Suite 120  
  Westford, MA  01886  
  Tel: 978-692-3848  
  Fax: 978-692-3821

- **Chicago**  
  333 Pierce Road, Suite 180  
  Itasca, IL  60143  
  Tel: 630-285-0071  
  Fax: 630-285-0075

- **Dallas**  
  4570 Westgrove Drive, Suite 160  
  Addison, TX  75001  
  Tel: 972-818-7423  
  Fax: 972-818-2924

- **Detroit**  
  Tri-Atria Office Building  
  32255 Northwestern Highway, Suite 190  
  Farmington Hills, MI  48334  
  Tel: 248-538-2250  
  Fax: 248-538-2260

- **Kokomo**  
  2767 S. Albright Road  
  Kokomo, IN  46902  
  Tel: 765-864-8360  
  Fax: 765-864-8387

- **Los Angeles**  
  18201 Von Karman, Suite 1090  
  Irvine, CA  92612  
  Tel: 949-263-1888  
  Fax: 949-263-1338

- **San Jose**  
  1300 Terra Bella Avenue  
  Mountain View, CA  94043  
  Tel: 650-215-1444  
  Fax: 650-961-0286

- **Toronto**  
  6285 Northam Drive, Suite 108  
  Mississauga, Ontario L4V 1X5, Canada  
  Tel: 905-673-6099  
  Fax: 905-673-6509

### ASIA/PACIFIC

**Australia**  
Suite 22, 41 Lawson Street  
Epping 2121, NSW  
Australia  
Tel: 61-2-9868-6733  
Fax: 61-2-9868-6755

**China - Beijing**  
Unit 708B  
Wan Tai Bei Hai Bldg.  
No. 6 Chaoyangmen Bei Str.  
Beijing, 100027, China  
Tel: 86-10-85282100  
Fax: 86-10-85282104

**China - Chengdu**  
Rm. 2401-2402, 24th Floor,  
Ming Xing Financial Tower  
No. 88 TIDU Street  
Chengdu 610016, China  
Tel: 86-28-86766200  
Fax: 86-28-86766599

**China - Fuzhou**  
Unit 28F. World Trade Plaza  
No. 71 Wusi Road  
Fuzhou 350001, China  
Tel: 86-591-7503506  
Fax: 86-591-7503521

**China - Hong Kong SAR**  
Unit 901-6, Tower 2, Metroplaza  
223 Hing Fong Road  
Kowloon, N.T., Hong Kong  
Tel: 852-2401-1200  
Fax: 852-2401-3431

**China - Shenzhen**  
Rm. 1812, 18/F, Building A,  
United Plaza  
No. 5022 Binhe Road, Futian District  
Shenzhen 518033, China  
Tel: 86-755-82901380  
Fax: 86-755-8295-1393

**Korea**  
168-1, Youngbo Bldg., 3 Floor  
Samsung-Dong, Kangnam-Ku  
Seoul, Korea 135-862  
Tel: 82-2-554-7200  
Fax: 82-2-558-5932 or  
82-2-558-5934

**Singapore**  
200 Middle Road  
#07-02 Prime Centre  
Singapore, 189980  
Tel: 65-6334-8870  
Fax: 65-6334-8850

**Taiwan**  
Kaohsiung Branch  
30F - 1 No. 5  
Min Chuan 2nd Road  
Kaohsiung 806, Taiwan  
Tel: 886-7-536-4818  
Fax: 886-7-536-4803

**EUROPE**

**Austria**  
Duratolstrasse 2  
A-4800 Wels  
Austria  
Tel: 43-7242-2244-399  
Fax: 43-7242-2244-393

**France**  
Parc d’Activite du Moulin de Massy  
43 Rue du Saule Trapu  
Batiement A - ler Etage  
91300 Massy, France  
Tel: 33-1-69-53-63-20  
Fax: 33-1-69-30-90-79

**Germany**  
Steinheilstrasse 10  
D-85737 Ismaning, Germany  
Tel: 44-118-921-5869  
Fax: 44-118-921-5820

**Netherlands**  
P. A. De Biesbosch 14  
NL-5152 SC Drunen, Netherlands  
Tel: 31-416-690399  
Fax: 31-416-690340

**United Kingdom**  
505 Eskdale Road  
Winnersh Triangle  
Wokingham  
Berkshire, England RG41 5TU  
Tel: 44-118-921-5869  
Fax: 44-118-921-5820

02/17/04