INTRODUCTION

The MCP2120 is a cost effective and easy to use device for sending and receiving IR serial data. The MCP2120 encodes an asynchronous serial data stream, converting each data bit to the corresponding Infrared (IR) formatted pulse. IR pulses that are received are decoded into the corresponding UART formatted serial data. The MCP2120 may be used to add IR capability to any embedded application where serial data is present. The encoding/decoding function in the MCP2120 is performed as specified in the physical layer component of the IrDA® standard. This part of the standard is referred to as “IrPHY”. A detailed discussion of this standard is beyond the scope of this Application Note, but a discussion regarding the encoding and decoding is in order. More detailed information is available from the IrDA website (www.IrDA.org). The vendor list later in this document also has web-links to more information. Figure 1 shows typical implementation of the MCP2120 in an embedded system.

FIGURE 1: SYSTEM BLOCK DIAGRAM

IrDA is a registered trademark of the Infrared Data Association.
SYSTEM HARDWARE

Figure 2 shows that very few components are needed to implement an IrDA standard compatible subsystem. The IR light pulses are converted to electrical pulses by the optical transceiver. The MCP2120 is connected directly to the optical transceiver. Resistor, R₁ and capacitor, C₁ are used to decouple the power supply of the optical transceiver from the rest of the system, since some transceivers have limited tolerance for power supply noise. This circuit will reduce 10 kHz power supply ripple by about 30 dB, if a good quality tantalum capacitor is used. Resistor, R₂ is used to limit the current of the emitter LED. Most transceivers use an external resistor for this purpose. Many infrared transceivers will emit an IR pulse when the transmit pin (TXD) is high, and will indicate a bit received by setting the receive pin (RXD) low.

The output impedance of the transceiver receive circuit may be 4 kΩ or more, so the MCP2120 should be located as close to the transceiver as possible. A ground plane under the transceiver will improve electromagnetic interference (EMI) performance and reduce susceptibility to EMI.

For battery powered applications, it may be an advantage to turn off power to the MCP2120. If power is turned off completely, care should be taken so that none of the I/O pins are exposed to a signal greater than Vss ± 0.6V. In some systems, it may be preferable to shut down the MCP2120 and leave other parts of the system active, thus exposing the MCP2120 to active signals while shut down. If this is the case, then the EN input pin should be used. If the EN pin (pin 6) is low, the device becomes disabled. The current consumption in this mode will be typically less than 1 µA and active I/O signals from the rest of the system do not need to be isolated from the MCP2120.
ENCODING

Figure 3 shows one-half (1st half) of an asynchronous serial byte sent by the MCP2120. Data to be transmitted is input to the MCP2120 on the TX pin (pin 12). The upper trace in Figure 3 shows a data word being sent. The first falling edge of the TX pin is the beginning of the start bit. The MCP2120 will then encode the following eight data bits according to the currently set data rate. The parameters for an IrDA standard transmission are: Start bit, eight data bits, no parity, and one stop bit.

Note 1: The sampling of the TX pin is level sensitive, not edge sensitive.
2: The MCP2120 does not indicate over-run errors. Care should be exercised to make sure the TX pin is low during the stop bit time.
3: An extended time period where TX is low (a BREAK), will result in the MCP2120 sending a string of 00h bytes as long as the TX pin is low.

The IrDA standard does not support other communication parameters. The MCP2120 has a fixed IR transmit pulse width which is equal to or greater than 1.6 µs.

Increasing Transmit Distance

The IrDA standard calls for a transmission distance of 1 m, with the emitter and received mis-aligned up to ±15 degrees. Some applications require a greater distance. This can be achieved with an increase in emitter power, a lens for the receiver, or both. Figure 4 shows how adding LEDs can be used to increase the transmission distance.

Note 1: For every doubling of distance the emitter power must be increased by a factor of 4. Thus if a transmission distance of 2 m is needed, three emitter LEDs of similar efficiency to the LED built into the transceiver, would need to be added. For 4 m distance, 15 LEDs would be need to be added.
2: Few IR LEDs are fast enough for use in IrDA standard compatible applications. The TON and TOFF for the LED device should be less than 100 ns.

The emitters used should have a wavelength centered at 875 nm. The author has used the Vishay/Temic TSSF4500 with excellent results. Typically, LEDs used in television-type remote controls have a wavelength of 950 nm and a TON and TOFF of 2 µs or more. These type of LEDs are not recommended for IrDA standard applications.

FIGURE 3: IR TRANSMISSION

FIGURE 4: USING ADDITIONAL LEDs FOR GREATER DISTANCE
DECODING

Figure 5 shows the reception of an IR byte. Many illumination sources, such as fluorescent lamps or sunlight can introduce light noise that can interfere with proper data reception. For best results, the IR transceiver should not be pointed directly at a visible light source. Also, sunlight is rich in IR light. If the ambient IR light level is too high, then the IR data source may not be sufficient to trigger the receiver. For best results, IR communications should not take place in direct sunlight.

**FIGURE 5: IR DATA RECEPTION**

<table>
<thead>
<tr>
<th>BITCLK (CLK)</th>
<th>RXIR</th>
<th>RX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Bit</td>
<td>16 CLK</td>
<td>0 1 0 0 1 ≥1.6 μs 16 CLK (or ≤50.5 μs typical) 8 CLK</td>
</tr>
<tr>
<td>Data bit 0</td>
<td></td>
<td>16 CLK 16 CLK 16 CLK 16 CLK 16 CLK 16 CLK</td>
</tr>
<tr>
<td>Data bit 1</td>
<td></td>
<td>16 CLK 16 CLK 16 CLK 16 CLK 16 CLK 16 CLK</td>
</tr>
<tr>
<td>Data bit 2</td>
<td></td>
<td>16 CLK 16 CLK 16 CLK 16 CLK 16 CLK 16 CLK</td>
</tr>
<tr>
<td>Data bit ...</td>
<td></td>
<td>16 CLK 16 CLK 16 CLK 16 CLK 16 CLK 16 CLK</td>
</tr>
</tbody>
</table>
HARDWARE DATA RATE SELECTION

The MCP2120 will encode and decode serial data at the currently selected data rate, or baud rate. The selection of this data rate is flexible and easy to use. Figure 6 shows how to use the BAUD2:BAUD0 input pins to implement hardware select mode. Jumpers or I/O signals from another controller may be used, or these inputs may be tied directly to fixed voltage levels, if the data rate does not have to change.

After the MCP2120 is reset, the BAUD2:BAUD0 input pins are sampled. If all three of these inputs are high, then software select mode is used. For any other inputs, hardware select mode is active. This setting is latched when the device is reset, either from the RESET pin or a power-on reset. After a device reset, changing the value of the BAUD2:BAUD0 pins has no effect on the device’s baud rate.

From Table 1, if a 9.6 kBaud data rate is desired with the device frequency at 7.3728 MHz, the BAUD2:BAUD0 pins should all be low.

FIGURE 6: USING HARDWARE DATA RATE SELECTION

TABLE 1: HARDWARE MODE - BAUD RATE SELECTION

<table>
<thead>
<tr>
<th>BAUD2:BAUD0</th>
<th>Fosc Frequency (MHz)</th>
<th>Bit Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.6144 (1)</td>
<td>2.000</td>
</tr>
<tr>
<td>000</td>
<td>800</td>
<td>2604</td>
</tr>
<tr>
<td>001</td>
<td>1600</td>
<td>5208</td>
</tr>
<tr>
<td>010</td>
<td>3200</td>
<td>10417</td>
</tr>
<tr>
<td>011</td>
<td>4800</td>
<td>15625</td>
</tr>
<tr>
<td>100</td>
<td>9600</td>
<td>31250</td>
</tr>
</tbody>
</table>

Note 1: An external clock is recommended for frequencies below 2 MHz.

2: For frequencies above 7.5 MHz, the TXIR pulse width (MCP2120 Data Sheet, Electrical Specifications, parameter IR121) will be shorter than the 1.6 µs IrDA standard specification.
SOFTWARE DATA RATE SELECTION

Software data rate selection is intended for use with systems where switching data rates must be changed frequently or when a minimum number of connections are needed between the MCP2120 and the embedded host as shown in Figure 7. Hardware data rate selection can be implemented with three signals. Software data rate selection requires five signals, in addition to using the RESET pin whenever a rate change is needed. The software Baud mode is compatible with one of the IR drivers published by Microsoft® for Microsoft Windows®.

Note: The Software Data select mode is compatible with the Microsoft CRYSTAL.VXD driver. See TB048, “Connecting the MCP2150 to the Windows Operating System” for more information.

In software baud mode, the MCP2120 differentiates between data and commands. This is controlled via the MODE pin. The command mode and data mode are summarized in Table 2. For select frequencies, the command/baud rate selected is shown in Table 3.

TABLE 2: SOFTWARE OPERATION

<table>
<thead>
<tr>
<th>Mode Pin State</th>
<th>Operation</th>
<th>Echo</th>
<th>Transmit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Command</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>1</td>
<td>Data</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Data sent to the MCP2120 will be encoded and transmitted via the IR transceiver. Commands are not intended to be transmitted. Commands are used to change data rates. When in command mode, the data sent to the MCP2120 will be echoed back to the embedded host.

The MODE pin is used to switch between command and data modes. When the MODE pin is low, the MCP2120 is in command mode, when the MODE pin is high, the MCP2120 is in data mode. The MODE pin is sampled during the start bit. Changing the state of the MODE pin after the start bit will have no effect. Be sure to allow for propagation delays to insure that the MODE pin is in the intended state before the start bit begins. If the MCP2120 is used with Microsoft Windows or other operating systems, the MODE pin is usually connected to the DTR signal of the host serial port. In this context, the host RTS signal is usually connected to the device reset as shown in Figure 7.

TABLE 3: SOFTWARE MODE - BAUD RATE SELECTION

<table>
<thead>
<tr>
<th>Hex Command (3, 4)</th>
<th>Fosc Frequency (MHz)</th>
<th>Bit Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.6144 2000 3.6864 4.9152 7.3728 14.7456 20.000</td>
<td></td>
</tr>
<tr>
<td>0x87</td>
<td>800 2604 4800 6400 9600 19200 26042</td>
<td>Fosc / 768</td>
</tr>
<tr>
<td>0x8B</td>
<td>1600 5208 9600 12800 19200 38400 52083</td>
<td>Fosc / 384</td>
</tr>
<tr>
<td>0x85</td>
<td>3200 10417 19200 25600 38400 76800 104167</td>
<td>Fosc / 192</td>
</tr>
<tr>
<td>0x83</td>
<td>4800 15625 28800 38400 57600 115200 156250</td>
<td>Fosc / 128</td>
</tr>
<tr>
<td>0x81</td>
<td>9600 31250 57600 78600 115200 230400 312500</td>
<td>Fosc / 64</td>
</tr>
</tbody>
</table>

Note 1: An external clock is recommended for frequencies below 2 MHz.

Note 2: For frequencies above 7.3728 MHz, the TXIR pulse width (MCP2120 Data Sheet, Electrical Specifications, parameter IR121) will be shorter than the 1.6 µs IrDA standard specification.

Note 3: Command 0x11 is used to change to the new baud rate.

Note 4: All other command codes are reserved.
SOFTWARE RATE SELECT COMMANDS

Two commands are supported: the “Next Data Rate” and the “Change Data Rate”. To use these commands, the MODE pin should be held low, then the one byte command codes sent. Table 4 shows these command codes.

<table>
<thead>
<tr>
<th>Command Value (hex)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x87</td>
<td>FOSC / 768 is next data rate</td>
</tr>
<tr>
<td>0x8B</td>
<td>FOSC / 384 is next data rate</td>
</tr>
<tr>
<td>0x85</td>
<td>FOSC / 192 is next data rate</td>
</tr>
<tr>
<td>0x83</td>
<td>FOSC / 128 is next data rate</td>
</tr>
<tr>
<td>0x81</td>
<td>FOSC / 64 is next data rate</td>
</tr>
<tr>
<td>0x11</td>
<td>Change to new rate</td>
</tr>
</tbody>
</table>

To change the data rate, two bytes must be sent. The first command loads the desired data rate. The second command changes the data rate to the value previously loaded. The “Change Data Rate” command will be echoed back at the current data rate. The next byte sent/received after the “Change Data Rate” command will be received/sent, or echoed at the new data rate. The MCP2120 requires that the stop bit of the “Change Data Rate” command byte finish at the currently selected data rate. If the current data rate is 9.6 k baud, then the required delay is 100 µs before data is sent or received. In addition, a delay of 200 µs should be used after any “Change Data Rate” command.

TURNAROUND LATENCY

An IR link can be compared to a one-wire data connection. The IR transceiver can transmit or receive, but not both at the same time. A delay of one bit time is suggested between the time a byte is received and another byte is transmitted.

USING THE MCP2120 DEVELOPER’S BOARD

Figure 8 shows two examples of how to use the MCP2120 with PICmicro® microcontrollers. The first example shows how wireless IR communication can be added to a minimum system using the PIC16F84. The PIC16F84 sends an IR message of “Hello World” when switch S3 is pressed. IR bytes received by the PIC16F84 are displayed in binary form. This example uses hardware select mode and a firmware UART for the PIC16F84. Another example shows a PIC16F84 using its internal hardware UART and software select mode.

FIGURE 8: EMBEDDED IrDA STANDARD EXAMPLE
REFERENCES

The IrDA Standards download page can be found at:
http://www.irda.org/standards/specifications

Manufacturers of Optical Transceivers are shown in Table 5.

TABLE 5: OPTICAL TRANSCEIVER MANUFACTURERS

<table>
<thead>
<tr>
<th>Company</th>
<th>Company Web Site Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infineon</td>
<td><a href="http://www.infineon.com">www.infineon.com</a></td>
</tr>
<tr>
<td>Agilent</td>
<td><a href="http://www.agilent.com">www.agilent.com</a></td>
</tr>
<tr>
<td>Vishay/Temic</td>
<td><a href="http://www.vishay.com">www.vishay.com</a></td>
</tr>
<tr>
<td>Rohm</td>
<td><a href="http://www.rohm.com">www.rohm.com</a></td>
</tr>
</tbody>
</table>

MEMORY USAGE

The PIC16F84 program that uses the Hardware Select of the baud rate (Appendix A) uses the following resources:

<table>
<thead>
<tr>
<th>Program Memory</th>
<th>135 words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Memory</td>
<td>9 bytes</td>
</tr>
</tbody>
</table>

The PIC16F84 program that uses the Software Select of the baud rate (Appendix B) uses the following resources:

<table>
<thead>
<tr>
<th>Program Memory</th>
<th>163 words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Memory</td>
<td>9 bytes</td>
</tr>
</tbody>
</table>

SUMMARY

The MCP2120 has a uniquely flexible combination of hardware, software, or Fosc selection of the data rate. The high integration, low power, and Windows compatibility make the MCP2120 well suited to implementing infrared solutions in consumer, industrial, automotive, and telecommunications applications.
Software License Agreement

The software supplied herewith by Microchip Technology Incorporated (the “Company”) for its PICmicro® Microcontroller is intended and supplied to you, the Company’s customer, for use solely and exclusively on Microchip PICmicro Microcontroller products.

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.

APPENDIX A: PIC16F84 HARDWARE SELECT SOURCE CODE

EXAMPLE A-1: PIC16F84 Hardware Select Code

;*****************************************************************
;               MCP2120 Demo with PicDem
;               Use with PIC16F84, 3.6864Hz clock
;               Checksum=9383 (cp on)
;*****************************************************************

; Revision History
;  1.0    04/05/01    Initial Release
;
;*****************************************************************

; Notes:
; This demo code sends/receives serial data at a fixed
data rate. This rate can be from 9.6 to 38.4KB. The
; bitreg delay values for the various data rates are given
; below. The data sent is a string which is stored in a table. The
; string is sent when the PICDEM RA1 button is pressed.
; Any bytes received are displayed on the PortB LEDs.
; This version of the code assumes that the MCP2120 is using
; hardware setup and the jumpers have been set to match the
; data rate of this code.
;*****************************************************************

LIST   C=132
include    P16F84.inc
#define    reset    H'00'    ;Reset vector

; Configuration Bits
__CONFIG _CP_OFF & _PWRTE_ON & _XT_OSC & _WDT_OFF
__IDLOCS H'0010'

; PortA Bits

#define    rts    porta,0    ;output, set high to reset MCP2120
#define    swt    porta,1    ;input, low when switch pressed
#define    rxd    porta,2    ;input, serial data from MCP2120
#define    txd    porta,3    ;output, serial data to MCP2120
#define    mode   porta,4    ;output, high for data mode, low for cmd mode

; cfga      equ    B'00000110' ;configuration for porta
cfgb      equ    H'00'       ;portb is an output port
cfgopt    equ    B'11001000' ;option reg setup
**Example A-1: PIC16F84 Hardware Select Code - Page 2**

```
;**********************************************************************
;  Constants
;
bytesz    equ     D'08'        ;there are 8 bits per byte
bitval    equ     D'08'        ;data bit delay
;
;Data Rate Constants
;   Rate  cyc  Bitval
;    9.6   96   20
;   19.2   48   08
;   38.4   24   02
;
;**********************************************************************

; Registers
;
cblock    H'0C'
areg                   ;GP scratchpad
breg                   ;GP scratchpad
bitreg                 ;storage for data bit delay
baudreg                ;storage for baud rate
cmdreg                 ;reg for commands
delreg                 ;reg for timing delays & scratchpad
bitcnt                 ;bit counter
flags
state                  ;reg for state counter
endc

;**********************************************************************
org      H'00'             ;use 00h as reset vector
    goto    start

;**********************************************************************

; String Table
; This table stores a string, breg is the offset. The string
; is terminated by a null.
;**********************************************************************
string clrf    pclath         ;this routine is on page 0
movf    breg,w         ;get the offset
addwf   pcl,f          ;add the offset to PC
DT      "Hello World"
DT      H'0D', H'0A'   ;the string also contains a CR+LF
DT      H'00'          ;terminate with 00h

;**********************************************************************

; 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
```
Example A-1: PIC16F84 Hardware Select Code - Page 3

;*****************************************************************
; Transmit serial Routine
; This routine sends the areg byte to the serial port at 19.2KB
;*****************************************************************

txser    bcf    txd        ;begin the start bit
    nop
    nop
    nop
    nop

; txdb    movf    bitreg,w
    call    delay
    nop
    nop
    btfsc   areg,0       ;if bit=0 then rxd=0
    goto    txdb1        ;if bit=1 then rxd=1

txdb0    nop
    nop
    bcf    txd        ;ir detected, bit=0
    rrf     areg,f       ;rotate the byte
    decfsz  bitcnt,f     ;all bits rev'd?
    goto    txdb         ;ir recv'd, toggle routine
    goto    txsp

; txdb1    nop
    bsf     txd
    rrf     areg,f       ;rotate the byte
    decfsz  bitcnt,f     ;all bits rev'd?
    goto    txdb
    goto    txsp

; txsp    nop
    nop
    movlw   bytesz       ;delay until the end of the 8th data bit
    movwf   bitcnt
    movf    bitreg,w
    call    delay
    bcf    txd        ;8th data bit ends here
    movf    bitreg,w
    call    delay
    movf    bitreg,w
    call    delay
    retlw   0

;
;*****************************************************************
; Receive Serial Routine
; This routine gets an incoming serial byte and stuffs it
; into areg
;*****************************************************************
rxser  nop ;delay from the beginning of the start bit
nop
nop
nop
nop
nop
nop

; rxdb  movf  bitreg,w
call  delay
nop
nop
rrf  areg,f ;rotate the byte
btfsc  rxd ;if rxd=0 then the bit=0
goto  rxdb1 ;if rxd=1 then bit=1
rxdb0  nop
nop
bcf  areg,7 ;clear the bit
decfsz  bitcnt,f ;all bits rev'd?
goto  rxdb ;ir recv'd, toggle routine
goto  rxsp

; rxdb1  nop
bsf  areg,7 ;set the bit
decfsz  bitcnt,f ;all bits rev'd?
goto  rxdb ;
goto  rxsp

; rxsp  movlw  bytesz ;reset the bit counter
movwf  bitcnt
movf  bitreg,w ;do the stop bit delay
call  delay
retlw  0

;
Example A-1: PIC16F84 Hardware Select Code - Page 5

;****************************************************************
;  Start Routine
;  The post-reset setup is done here
;****************************************************************
start   movlw    trisa       ;setup I/O
          movwf    fsr
          movlw    cfga
          movwf    indf
;
          movlw    trisb
          movwf    fsr
          movlw    cfgb
          movwf    indf
;
          movlw    option_reg  ;setup option reg
          movwf    fsr
          movlw    cfgopt
          movwf    indf
;
          movlw    H'00'       ;clear outputs
          movwf    portb
          bsf      txd         ;setup quiescent state
          bsf      mode
          bcf      rts
;
          movlw    bitval      
          movwf    bitreg
          movlw    bytesz      ;setup bit count
          movwf    bitcnt
;
          goto     main
;
;****************************************************************
; Main Routine
;****************************************************************
main    btfss    swt         ;check for keypress
          goto     send         ;key is pressed, send the bytes
          btfss    rxd         ;check for an incoming byte from MCP2120
          goto     getser      ;there's an incoming byte, go get it
          goto     main
;****************************************************************
; Send routine
; This routine sends the data found in sndtab
;****************************************************************
send    clrf    breg           ;clear the offset
sndlp   call    string1        ;get the byte
        movwf   areg           ;save the byte
        incf    breg,f         ;increment the table pointer
        movf    areg,f         ;move the byte to test it
        btfsc   status,z       ;if z=1 then we're done
        goto    sendex         ;we're done, do the exit
        call    txser          ;send the byte in areg
        goto    sndlp
;
sendex  btfss   swt            ;check for key release
        goto    sendex         ;key is pressed, wait for release
        movlw   H'FF'          ;do a debounce delay
        call    delay          ;return to waiting
;
;
;****************************************************************
; Get serial routine
; This routine gets a serial byte and displays the value
; on the PICDEM portb leds.
;****************************************************************
getser  call    rxser          ;get the serial byte
        movf    areg,w         ;w = serial byte
        movwf    portb         ;move the byte to the output
        goto    main
;
;
end
Software License Agreement

The software supplied herewith by Microchip Technology Incorporated (the “Company”) for its PICmicro® Microcontroller is intended and supplied to you, the Company’s customer, for use solely and exclusively on Microchip PICmicro Microcontroller products.

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.

APPENDIX B: PIC16F84 SOFTWARE SELECT SOURCE CODE

EXAMPLE B-1: PIC16F84 Software Select Code

;*****************************************************************
; MCP2120 Demo with PicDem
; Demonstration of Software Setup Mode of the MCP2120
; Use with PIC16F84, 3.6864Hz clock
; Checksum=9383 (cp on)
;*****************************************************************

; Revision History
; 1.0  04/05/01  Initial Release
;
;****************************************************************

; Notes:
; This demo code sends/receives serial data at a changeable
; data rate. This code can be easily changed from 9.6 to 38.4KB. The
; bitreg delay values for the various data rates are given
; below. The data sent is a string which is stored in a table. The
; string is sent when the PICDEM RA1 button is pressed.
; Any bytes received are displayed on the PortB LEDs.
; This version of the code assumes that the MCP2120 BAUD inputs
; have been set to software setup, BAUDx=B'111'. NOTE: When the
; MCP2120 powers up the optical transceiver may emit some garbage.
; The PICDEM board should be reset with the MCLR button before
; doing the demo.
; This demo starts with the MCP2120 and the host (this code)
; at 9.6KB. When the RA1 button is pressed the MCP2120 and the
; host change to 19.2KB and a "Hello World" message is transmitted
; via Ir at 19.2KB. Any data received by the MCP2120 is displayed on
; the PICDEM PortB LEDs. NOTE: The PICDEM serial port can be used to
; monitor the communication between this code and the MCP2120.
;****************************************************************

LIST    C=132
include P16F84.inc
#define reset H'00'

Configuration Bits
__CONFIG _CP_OFF & _PWRT_ON & _XT_OSC & _WDT_OFF
__IDLOCS H'0010'

;*****************************************************************
EXAMPLE B-1: PIC16F84 Software Select Code - Page 2

;******************************************************************************
;       PortA Bits
;
#definerts porta,0 ;output, set high to reset MCP2120
#defineswt porta,1 ;input, low when switch pressed
#definerxd porta,2 ;input, serial data from MCP2120
#definetxd porta,3 ;output, serial data to MCP2120
#definemode porta,4 ;output, high for data mode, low for cmd mode
;
;configa equ B'00000110' ;configuration for porta
configbequ H'00' ;portb is an output port
configopt equ B'11001000' ;option reg setup
;
;******************************************************************************
;       Constants
;
byteszequ D'08' ;there are 8 bits per byte
bitval equ D'20' ;data bit delay
bit96equ D'20'
bit19equ D'08'
bit38equ D'02'
;
;Data Rate Constants
; Rate   cyc  Bitval
;  9.6    96   20
;  19.2   48   08
;  38.4   24   02
;
;******************************************************************************
;       Registers
;
cblock H'0C'
;GP scratchpad
areg
;GP scratchpad
breg
;storage for data bit delay
bitreg
;storage for baud rate
baudreg
;reg for commands
cmdreg
delreg ;reg for timing delays & scratchpad
bitcnt
;bit counter
flags
;reg for state counter
state
endc
;
EXAMPLE B-1: PIC16F84 Software Select Code - Page 3

;******************************************************************
org     H'00'        ;use 00h as reset vector
;goto    start
;
;******************************************************************

; String Table
; This table stores a string, breg is the offset. The string
; is terminated by a null.
;******************************************************************
string1 clrf    pclath       ;this routine is on page 0
movf    breg,w       ;get the offset
addwf   pcl,f        ;add the offset to PC
DT      "Hello World"
DT      H'0D',H'0A'  ;the string also contains a CR+LF
DT      H'00'        ;terminate with 00h
;
;******************************************************************

;       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
;more to delay?
goto    dellp
retlw   0
;
;******************************************************************

;       Long Delay
;       This routine is used to delay past the POR time of the MCP2120.
;       The delay is 100ms.
;******************************************************************
ldelay  movlw   D'100'       ;the inner loop is 1ms, do it 100 times
movwf   breg         ;breg is the loop counter
ldellp  movlw   D'230'       ;delreg of 230 = 1ms
call    delay
decfsz  breg,f
;more to delay?
goto    ldellp
retlw   0
;
EXAMPLE B-1: PIC16F84 Software Select Code - Page 4

;**********************************************
;   Transmit serial Routine
;   This routine sends the areg byte to the serial port at 19.2KB
;**********************************************

txser   bcf     txd            ;begin the start bit
   nop
   nop
   nop

; txd
   movf    bitreg,w
   call    delay
   nop
   nop
   btfs   areg,0         ;if bit=0 then rxd=0
   goto    txdbl
          ;if bit=1 then rxd=1
   txd0
   nop
   nop
   bcf     txd            ;ir detected, bit=0
   rrf     areg,f         ;rotate the byte
   decfsz  bitcnt,f       ;all bits rev'd?
   goto    txd
   goto    txsp

; txdbl
   nop
   bsf     txd            ;8th data bit ends here
   rrf     areg,f         ;rotate the byte
   decfsz  bitcnt,f       ;all bits rev'd?
   goto    txd
   goto    txsp

; txsp
   nop
   nop
   nop
   movlw   bytesz         ;delay until the end of the 8th data bit
   movwf   bitcnt
   movf    bitreg,w
   call    delay
   bcf     txd            ;8th data bit ends here
   movf    bitreg,w
   call    delay
   movf    bitreg,w
   call    delay
   movf    bitreg,w
   call    delay
   retlw   0

;
EXAMPLE B-1: PIC16F84 Software Select Code - Page 5

;****************************************************************
; Receive Serial Routine
; This routine gets an incoming serial byte and stuffs it
; into areg
;****************************************************************

rxser  nop           ;delay from the beginning of the start bit
       nop
       nop
       nop
       nop
       nop

rxdb    movf    bitreg,w
       call    delay
       nop
       nop
       rrf     areg,f    ;rotate the byte
       btfsc   rxd       ;if rxd=0 then the bit=0
       goto    rxdb1     ;if rxd=1 then bit=1

rxdb0   nop
       nop
       bcf     areg,7    ;clear the bit
       decfsz  bitcnt,f  ;all bits rev'd?
       goto    rxdb      ;ir recv'd, toggle routine
       goto    rxsp

rxdb1   nop
       bsf     areg,7    ;set the bit
       decfsz  bitcnt,f  ;all bits rev'd?
       goto    rxdb
       goto    rxsp

rxsp    movlw   bytesz       ;reset the bit counter
       movwf   bitcnt
       movf    bitreg,w  ;do the stop bit delay
       call    delay
       retlw   0
EXAMPLE B-1: PIC16F84 Software Select Code - Page 6

;****************************************************************
;  Start Routine
;  The post-reset setup is done here
;****************************************************************

start   movlw  trisa          ;setup I/O
        movwf  fsr
        movlw  cfga
        movwf  indf
;
        movlw  trisb
        movwf  fsr
        movlw  cfgb
        movwf  indf
;
        movlw  option_reg     ;setup option reg
        movwf  fsr
        movlw  cfgopt
        movwf  indf
;
        movlw  H'00'          ;clear outputs
        movwf  portb
        bsf   txd            ;setup quiescent state
        bsf   mode
        bcf   rts
;
        movlw  bitval         ;
        movwf  bitreg
        movlw  bytesz         ;setup bit count
        movwf  bitcnt
;
        goto   main
;
;****************************************************************
;       Main Routine
;****************************************************************

main    btfss   swt            ;check for keypress
        goto   send           ;key is pressed, send the bytes
        btfss   rxd            ;check for an incoming byte from MCP2120
        goto   getser         ;there's an incoming byte, go get it
        goto   main
;

EXAMPLE B-1: PIC16F84 Software Select Code - Page 7

;****************************************************************
; Send routine
; This routine sends the data found in sndtab using the following
; procedure:
; 1.) The host (this code) resets to 9.6KB
; 2.) The MCP2120 is reset, reverting to 9.6KB as well.
; 3.) The MCP2120 is placed in command mode.
; 4.) The commands are sent to change the MCP2120 to 19.2KB
; 5.) The host changes to 19.2KB.
; 6.) The test data is sent
;
; Note: The mode output is RA4 which has an open-collector structure.
; This output is pulled up by R5 and loaded by C5 on the PICDEM
; board. To get around this we assume a 20us fall time and a
; 60us rise time. These mode delays will not be needed in
; any production production code. The MCP2120 does not
; require this delay.
;****************************************************************

send    movlw   bit96        ;change the host data rate to 9.6KB
movwf   bitreg
bsf     rts          ;assert the MCP2120 reset
movlw   D'25'        ;do the reset for 100us
call    delay        ;
bcf     rts          ;release the reset
call    ldelay       ;delay past the POR of the MCP2120

bcf     mode         ;change to command mode
movlw   D'10'        ;delay for the Mode fall time
call    delay        ;
movlw   H'8B'        ;8B loads the 19.2KB rate into the MCP2120
movwf   areg
call    txser
movlw   H'11'        ;11h changes the BAUD rate to the new value
movwf   areg
call    txser        ;send the command
bsf     mode         ;back to data mode
movlw   D'25'        ;delay for mode rise time
call    delay        ;
movlw   bit19        ;change the host data rate to 19.2KB
movwf   bitreg
;
clr breg        ;clear the offset
EXAMPLE B-1: PIC16F84 Software Select Code - Page 8

```
sndlp   call    string1        ;get the byte
         movwf   areg           ;save the byte
         incf    breg,f         ;increment the table pointer
         movf    areg,f         ;move the byte to test it
         btfsc   status,z       ;if z=1 then we're done
         goto    sendex         ;we're done, do the exit
         call    txser          ;send the byte in areg
         goto    sndlp

; sendex   btfss   swt            ;check for key release
         goto    sendex         ;key is pressed, wait for release
         movlw   H'FF'          ;do a debounce delay
         call    delay
         goto    main           ;return to waiting

;****************************************************************
;  Get serial routine
;  This routine gets a serial byte and displays the value
;  on the PICDEM portb leds.
;****************************************************************
getser  call    rxser          ;get the serial byte
         movf    areg,w         ;w = serial byte
         movwf   portb          ;move the byte to the output
         goto    main

;end
```
Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip’s products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, PIC, PICmicro, PICMASTER, PICSTART, PRO MATE, KeeLog, SEEVAL, MPLAB and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

Total Endurance, ICSP, In-Circuit Serial Programming, Filter-Lab, MXDEV, microID, FlexROM, fuzzyLAB, MPASM, MPLINK, MPLIB, PICC, PICDEM, PICDEM.net, ICEPIC, Migratable Memory, FanSense, ECONOMONITOR, Select Mode and microPort are trademarks of Microchip Technology Incorporated in the U.S.A.

Serialized Quick Term Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2001, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Printed on recycled paper.

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company’s quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELog® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip’s quality system for the design and manufacture of development systems is ISO 9001 certified.