AN510

Implementation of an Asynchronous Serial I/O

Author: Amar Palacherla
Microchip Technology Inc.

INTRODUCTION

The PIC16C5X series from Microchip Technology Inc., are 8-bit, high-speed, EPROM-based microcontrollers. This application note describes an implementation of an asynchronous serial I/O using a PIC16C5X microcontroller. These microcontrollers can operate at very high speeds with a minimum cycle time of 200 ns @ 20 MHz input clock. Many microcontroller applications require chip-to-chip serial data communications. Since the PIC16C5X series has no on-chip serial ports, serial communication has to be performed in software. For many cost-sensitive high volume applications, implementation of a serial I/O through software provides a more cost effective solution than dedicated logic. This application note provides code for the PIC16C5X to simulate a serial port using two I/O pins (one as input for reception and the other as output for transmission).

IMPLEMENTATION

Two programs are provided in this application note. One program (Appendix B) simulates full duplex RS-232 communication and the other (Appendix A) provides implementation of half-duplex communication. Using half-duplex, baud rates up to 19200 can be implemented using an 8 MHz input clock. For full-duplex, the software can handle up to 9600 baud at 8 MHz and 19200 baud at 20 MHz, one or two stop bits, eight or seven data bits, no Parity and can transmit or receive with either LSb first (normal mode) or MSb first (CODEC-like mode). It should be noted that the higher the input clock the better the resolution. These options should be set up during assembly time and not during run time. The user simply has to change the header file for the required communication options. The software does not provide any handshaking protocols. With minor modifications, the user may incorporate software handshaking using XON/XOFF. To implement hardware handshaking, an additional two digital I/O pins may be used as RTS (ready to send) and CTS (clear to send) lines.

Figure 1 shows a flowchart for serial transmission and Figure 2 shows a flowchart for reception. The flowcharts show case transmission/reception with LSb first and eight data bits. For reception, the data receive pin, DR, is polled approximately every B/2 seconds (52 µs for 9600 baud) to detect the start bit, where B is the time duration of one bit (B = 1/Baud). If a start bit is found, then the first data bit is checked for after 1.25B seconds. From then on, the other data bits are checked every B seconds (104 µs for 9600 baud).

In the case of transmission, first a start bit is sent by setting the transmit data pin, DX, to zero for B seconds, and from then on the DX pin is set/cleared corresponding to the data bit every B seconds. Assembly language code corresponding to the following flowcharts is given in Example 1 and Example 2.

FIGURE 1: TRANSMISSION FLOWCHART

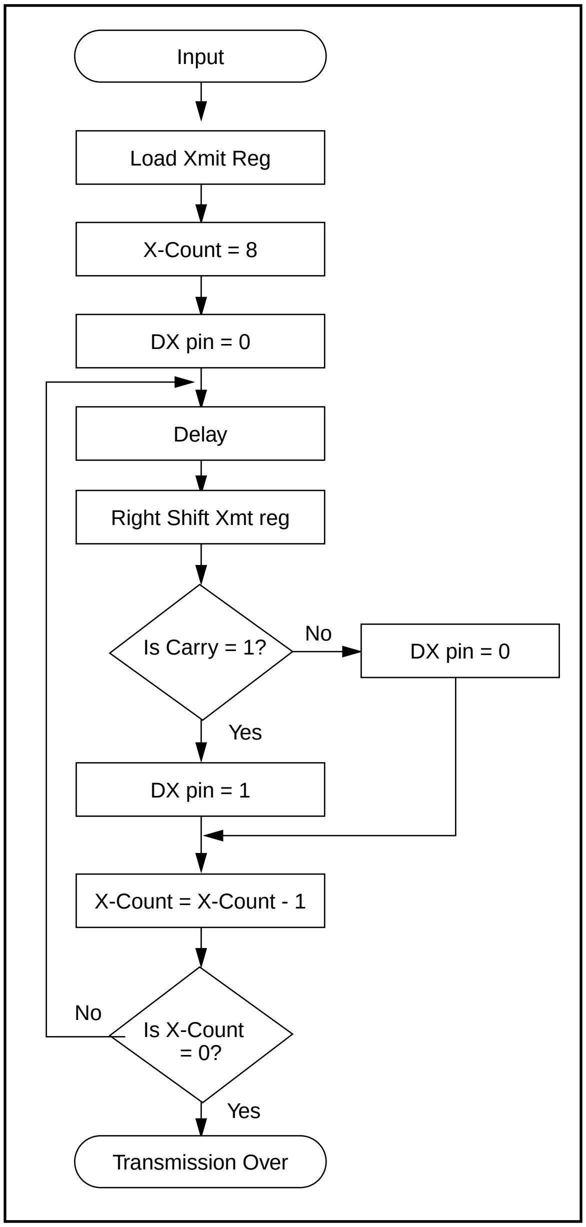
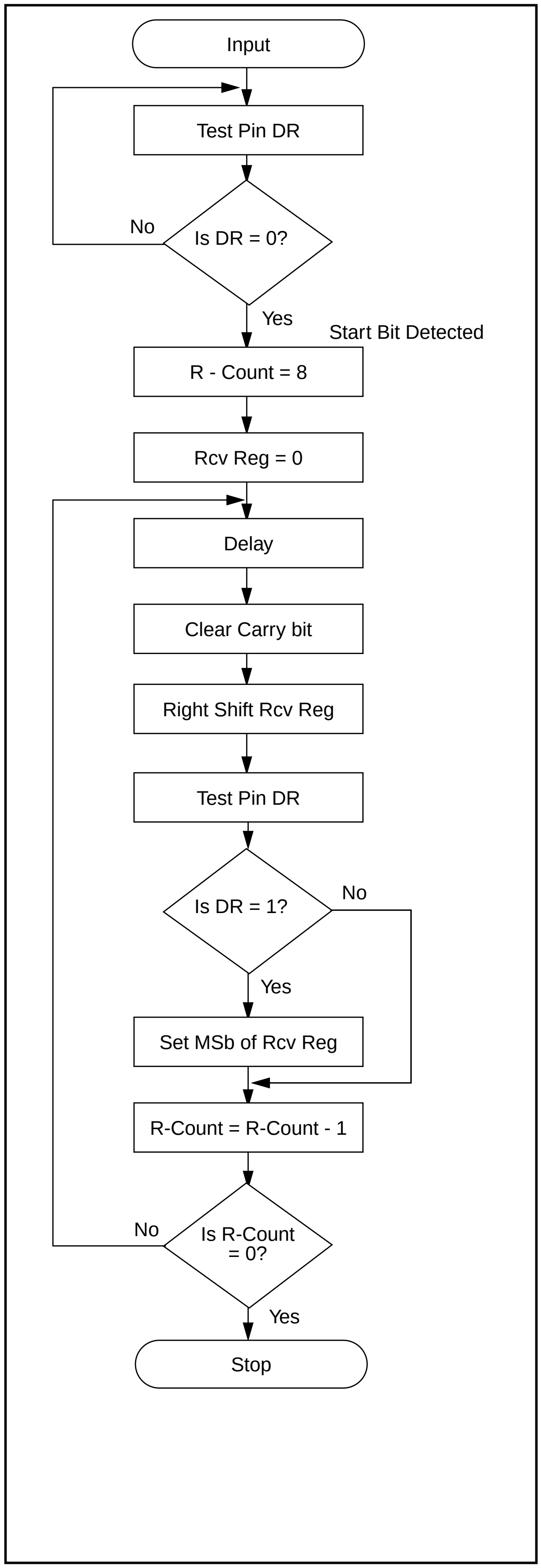


FIGURE 2: RECEPTION FLOWCHART



EXAMPLE 1: TRANSMIT ASSEMBLY CODE (CORRESPONDING TO FIGURE 1)

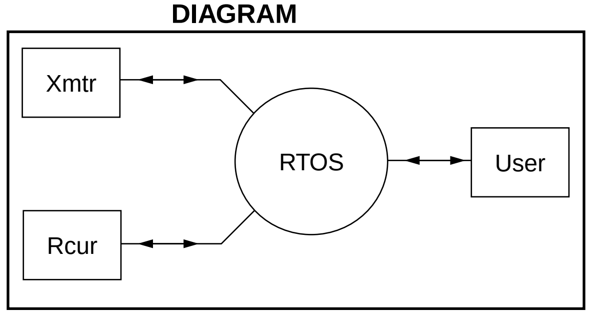
```
;******************** Transmitter****************************
Xmtr    movlw    8             ; Assume XmtReg contains data to be Xmted
        movwf    XCount        ; 8 data bits
        bcf      Port_A,DX     ; Send Start Bit
X_next  call     Delay         ; Delay for B/2 Seconds
        rrf      XmtReg
        btfsc    STATUS,CARRY  ; Test the bit to be transmitted
        bsf      Port_A,DX     ; Bit is a one
        btfss    STATUS,CARRY
        bcf      Port_A,DX     ; Bit is zero
        decfsz   Count         ; If count = 0, then transmit a stop bit
        goto     X_next        ; transmit next bit
;
X_Stop  call     Delay
        bsf      Port_A,DX     ; Send Stop Bit
X_Over  goto     X_Over
```

EXAMPLE 2: RECEIVE ASSEMBLY CODE (CORRESPONDING TO FIGURE 2)

```
;****************    Receiver  *********************
        ;
        Rcvr    btfsc   Port_A,DR      ; Test for Start Bit
                goto    Rcvr           ; Start Bit not found
                movlw   8              ; Start Bit Detected
                movwf   RCount         ; 8 Data Bits
                clrf    RcvReg         ; Receive Data Register
        R_next  call    Delay          ; Delay for B/2 Seconds, B=Time duration of
                                       ;  1bit
                bcf     STATUS,CARRY   ; Clear CARRY bit
                rrf     RcvReg         ; to set if MSB first or LSB first
                btfsc   Port_A,DR      ; Is the bit a zero or one ?
                bsf     RcvReg,MSB     ; Bit is a one
                call    Delay
                decfsz  RCount
                goto    R_next
        R_Over  goto    R_Over         ; Reception done
```

The software is organized such that the communication software acts as a Real-Time Operating System (RTOS) which gives control to the user routine for a certain time interval. After this predetermined time slot, the user must give back control to the Operating System. This is true only in the case of full-duplex implementation. Timing considerations are such that the user gets control for approximately half the time of the bit rate and the rest of the time is used up by the Operating System (and software delays). Please refer to Table 1 for the delay constants and the time the user gets using an 8 MHz input clock. Delay constants and the time that the user gets at 20 MHz and 4 MHz input clock speeds are given in the source code listing of the full-duplex routine. At frequencies other than 4, 8, or 20 MHz, the delay constants and the time the user gets can be computed from the equations given in Equation 1.

FIGURE 3: FULL-DUPLEX BLOCK DIAGRAM



EQUATION 1: EQUATIONS FOR DELAY CONSTANTS

```
Note: CLKOUT = Fosc/4
Baud_Cycles = Clkout/Baud;
User_time = Baud_Cycles • (float) 0.5;
K0 = (1.25 • Baud_Cycles - 2.0 • User_time - 89)/3.0; IF (K0 < 0)
   {
   K0 = 0.0;
   User_time = 0.50 • (1.25 • Baud_Cycles - 89.0) ;
   }
K1 = (1.25 • Baud_Cycles  - User_time - 59.0 - 3 • K0)/3.0 ;
K2 = (Baud_Cycles - User_time - 41.0 - 3 • K0)/3.0 ;
K3 = (Baud_Cycles - User_time - 61.0 - 3 • K0)/3.0  ;
K4 = (Baud_Cycles - User_time - 55.0 - 3 • K0)/3.0  ;
K5 = (Baud_Cycles - User_time - 55.0 - 3 • K0)/3.0 +1.0 ;
K6 = 0.0;
K7 = (1.25 • Baud_Cycles - User_time - 39.0 - 3 • K0)/3.0 ;
```

TABLE 1: DELAY CONSTANTS AT 8 MHz INPUT CLOCK

Constant	19200	9600	4800	2400	1200
K0	-	0	5	39	109
K1	-	39	80	150	288
K2	-	27	51	86	155
K3	-	21	44	80	148
K4	-	23	46	82	150
K5	-	24	47	83	151
K6	-	0	0	0	0
K7	-	45	86	156	295
User Cycles	-	86	208	416	832

TABLE 2: DELAY CONSTANTS AT 20 MHz INPUT CLOCK

Constant	19200	9600	4800	2400	1200
K0	0	13	57	143	317
K1	49	98	184	358	705
K2	34	60	103	191	364
K3	27	53	96	184	357
K4	29	55	98	186	359
K5	30	56	99	187	360
K6	0	0	0	0	0
K7	56	104	190	365	712
User Cycles	118	260	521	1042	2083

For example, if the baud rate selected is 9600 bps (@ 8 MHz), then the total time frame for one bit is approximately 104 µs. Out of this 104 µs, 61 µs are used by the Operating System and the other 43 µs are available to the user. It is the user's responsibility to return control to the Operating System exactly after the time specified in Table 1. For very accurate timing (with resolution up to one clock cycle) the user may set up Timer0 with the Prescaler option for calculating the real-time. With TMR0 configured to increment on internal CLKOUT (500 ns @ 8 MHz CLKIN) and the prescaler assigned to it, very accurate and long timing delay loops may be assigned. This method of attaining accurate delay loops is not used in the RS-232 code (RTOS), so that Timer0 is available to the user for other important functions. If Timer0 is not used for other functions, the user may modify the code to replace the software delay loops by counting TMR0. For an example of using TMR0 counting for exact timing delays, refer to the "user" routine in Full Duplex code (Appendix B).

The software uses minimal processor resources. Only six data RAM locations (File Registers) are used. The RTOS uses one level of stack, but it is freed once control is given back to the user. The Watchdog Timer (WDT) and Timer0 are not used. The user should clear the WDT at regular intervals, if the WDT is enabled.

The usage of the program is described in the following sections. The user should branch to location "Op_Sys" exactly after the time specified in Table 1 or as computed from equations in Equation 1.

Whereas, the transmission is totally under user control, the Reception is under the control of the Operating System. As long as the user does not set the X_flag, no transmission occurs. On the other hand the Operating System is constantly looking for a start bit and the user should not modify the R_done flag or the RcvReg.

TRANSMISSION

Transmit Data is output on the DX pin (bit0 of PORTA). In the user routine, the user should load the data to be transmitted in the XmtReg and set the X_flag (`bsf kwn FlagRX,X_flag`). This flag gets cleared after the transmission. The user should check this flag (X_flag) to see if a transmission is in progress. Modifying XmtReg when the X_flag is set will cause erroneous data to be transmitted.

RECEPTION

Data is received on pin DR (bit1 of PORTA). The user should constantly check the "R_done" flag to see if the reception is over. If a reception is in progress, the R_flag is set. If the reception is over, the "R_done" flag is set. The "R_done" flag gets cleared when the next start bit is detected. The user should constantly check the R_done flag, and if set, then the received word is in Register "RcvReg". This register gets cleared when a new start bit is detected. It is recommended that the RcvReg, be copied to another register after the R_done flag is set. The R_done flag also gets cleared when the next start bit is detected.

The user may modify the code to implement an N deep buffer (limited to the number of Data RAM locations available) for receive. Also, if receiving at high speeds, and if the N deep buffer is full, an XOFF signal (HEX 13) may be transmitted. When ready to receive more data, an XON signal (HEX 11) should be transmitted.

SUMMARY

The PIC16C5X family of microcontrollers allow users to implement half or full duplex RS-232 communication in software.

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX A: ASSEMBLY LANGUAGE FOR HALF DUPLEX

```
MPASM 01.40 Released         HALF_DUP.ASM   1-16-1997  11:48:17         PAGE  1


LOC  OBJECT CODE  LINE SOURCE TEXT
  VALUE

                    00001        LIST    P = 16C54, n = 66
                    00002 ;
                    00003 ;*********************************************************************
                    00004 ;                       RS-232 Communication With PIC16C54
                    00005 ;
                    00006 ;        Half Duplex Asynchronous Communication
                    00007 ;
                    00008 ;        This program has been tested at Bauds from 1200 to 19200 Baud
                    00009 ;        ( @ 8,16,20 Mhz CLKIN )
                    00010 ;
                    00011 ;       As a test, this program will echo back the data that has been
                    00012 ;       received.
                    00013 ;
                    00014 ;       Program:          HALF_DUP.ASM
                    00015 ;       Revision Date:
                    00016 ;                         1-13-97     Compatibility with MPASMWIN 1.40
                    00017 ;
                    00018 ;********************************************************************
                    00019 ;
                    00020        INCLUDE         <P16C5X.INC>
                    00001        LIST
                    00002 ; P16C5X.INC Standard Header File, Ver. 3.30 Microchip Technology, Inc.
                    00224        LIST
                    00021
  000001FF          00022 PIC54   equ 1FFH  ; Define Reset Vector
  00000001          00023 Same    equ 1
  00000007          00024 MSB     equ     7
                    00025
                    00026 ;*****************  Communication Parameters   **********************
                    00027 ;
  00000001          00028 X_MODE  equ 1 ; If ( X_MODE==1) Then transmit LSB first
                    00029               ; if ( X_MODE==0) Then transmit MSB first (CODEC like)
  00000001          00030 R_MODE  equ 1 ; If ( R_MODE==1) Then receive LSB first
                    00031               ; if ( X_MODE==0) Then receive MSB first (CODEC like)
  00000001          00032 X_Nbit  equ 1 ; if (X_Nbit==1) # of data bits (Transmission is 8 else 7
  00000001          00033 R_Nbit  equ 1 ; if (R_Nbit==1) # of data bits (Reception) is 8 else 7
                    00034 ;
  00000000          00035 Sbit2   equ 0 ; if Sbit2 = 0 then 1 Stop Bit else 2 Stop Bits
                    00036 ;
                    00037 ;********************************************************************
  00000005          00038 X_flag  equ PA0 ; Bit 5 of F3 ( PA0 )
  00000006          00039 R_flag  equ PA1 ; Bit 6 of F3 ( PA1 )
                    00040 ;
  00000000          00041 DX      equ 0   ; Transmit Pin ( Bit 0 of Port A )
  00000001          00042 DR      equ 1   ; Reciive Pin ( Bit 1 of Port A )
                    00043 ;
                    00044 ;
  00000044          00045 BAUD_1  equ .68  ; 3+3X = CLKOUT/Baud
  00000043          00046 BAUD_2  equ .67  ; 6+3X = CLKOUT/Baud
  00000022          00047 BAUD_3  equ .34  ; 3+3X = 0.5*CLKOUT/Baud
  00000056          00048 BAUD_4  equ .86  ; 3+3X = 1.25*CLKOUT/Baud
  00000042          00049 BAUD_X  equ .66  ; 11+3X = CLKOUT/Baud
  00000042          00050 BAUD_Y  equ .66  ; 9 +3X = CLKOUT/Baud
```

```
                    ;
                    ;************************  Data RAM Assignments  ********************
                    ;
0008                        ORG  08H    ; Dummy Origin
                    ;
0008                RcvReg  RES  1      ; Data received
0009                XmtReg  RES  1      ; Data to be transmitted
000A                Count   RES  1      ; Counter for #of Bits Transmitted
000B                DlyCnt  RES  1
                    ;***************************************************************
                    ;
0000                        ORG     0
                    ;
0000 0068           Talk    clrf    RcvReg          ; Clear all bits of RcvReg
0001 0625                   btfsc   PORTA,DR        ; check for a Start Bit
0002 0A30                   goto    User            ; delay for 104/2 uS
0003 0923                   call    Delay4          ; delay for 104+104/4
                    ;***************************************************************
                    ;       Receiver
                    ;
0004                Rcvr
                            IF      R_Nbit
0004 0C08                   movlw   8               ; 8 Data bits
                            ELSE
                            movlw   7               ; 7 data bits
                            ENDIF
                    ;
0005 002A                   movwf   Count
0006 0403           R_next  bcf     STATUS,C
                            IF      R_MODE
0007 0328                   rrf     RcvReg,Same     ; to set if MSB first or LSB first
                            ELSE
                            rlf     RcvReg,Same
                            ENDIF
0008 0625                   btfsc   PORTA,DR
                    ;
                            IF      R_MODE
                              IF      R_Nbit
0009 05E8                     bsf     RcvReg,MSB       ; Conditional Assembly
                              ELSE
                              bsf     RcvReg,MSB-1
                              ENDIF
                            ELSE
                            bsf     RcvReg,LSB
                            ENDIF
                    ;
000A 091F                   call    DelayY
000B 02EA                   decfsz  Count,Same
000C 0A06                   goto    R_next
                    ;****************************************************
000D 0208           R_over  movf    RcvReg,0        ; Send back What is Just Received
000E 0029                   movwf   XmtReg
                    ;****************************************************
                    ;       Transmitter
                    ;
000F                Xmtr
                            IF      X_Nbit
000F 0C08                   movlw   8
                            ELSE
                            movlw   7
                            ENDIF
0010 002A                   movwf   Count
                    ;
                            IF      X_MODE
                            ELSE
                              IF    X_Nbit
```

```
                           ELSE
                           rlf    XmtReg,Same
                           ENDIF
                         ENDIF
                   ;
0011 0405                bcf    PORTA,DX        ; Send Start Bit
0012 0925                call   Delay1
0013 0403          X_next  bcf    STATUS,C
                   ;
                         IF     X_MODE
0014 0329                rrf    XmtReg,Same      ; Conditional Assembly
                         ELSE                    ; to set if MSB first or LSB first
                         rlf    XmtReg,Same
                         ENDIF
                   ;
0015 0603                btfsc  STATUS,C
0016 0505                bsf    PORTA,DX
0017 0703                btfss  STATUS,C
0018 0405                bcf    PORTA,DX
0019 0921                call   DelayX
001A 02EA                decfsz Count,Same
001B 0A13                goto   X_next
001C 0505                bsf    PORTA,DX        ; Send Stop Bit
001D 0925                call   Delay1
                   ;
                         IF     Sbit2
                         bsf    PORTA,DX
                         call   Delay1
                         ENDIF
                   ;
001E 0A00                goto   Talk            ; Back To Reception & Transmision
                   ;
                   ;   End of Transmission
                   ;
001F 0C42          DelayY  movlw  BAUD_Y
0020 0A28                goto   save
0021 0C42          DelayX  movlw  BAUD_X
0022 0A28                goto   save
0023 0C56          Delay4  movlw  BAUD_4
0024 0A28                goto   save
0025 0C44          Delay1  movlw  BAUD_1          ; 104 uS for 9600 baud
0026 0A28                goto   save
0027 0C43          Delay2  movlw  BAUD_2
0028 002B          save    movwf  DlyCnt
0029 02EB          redo_1  decfsz DlyCnt,Same
002A 0A29                goto   redo_1
002B 0800                retlw  0
                   ;
002C 0C0E          main    movlw  0EH             ; Bit 0 of Port A is Output
002D 0005                tris   PORTA           ; Set PORTA.0 as output ( DX )
002E 0525                bsf    PORTA,DR
                   ;
002F 0A00                goto   Talk
                   ;
                   ;
0030 0C22          User    movlw  BAUD_3
0031 002B                movwf  DlyCnt
0032 02EB          redo_2  decfsz DlyCnt,Same
0033 0A32                goto   redo_2
0034 0A00                goto   Talk            ; Loop Until Start Bit Found
                   ;
                   ;
01FF                     ORG    PIC54
01FF 0A2C                goto   main
                   ;
                         END
```



```
MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXX-----------
01C0 : ---------------- ---------------- ---------------- ---------------X

All other memory blocks unused.

Program Memory Words Used:    54
Program Memory Words Free:   458


Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,     0 suppressed
```

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX B: ASSEMBLY LANGUAGE LISTING FOR FULL DUPLEX

```
MPASM 01.40 Released           RS232.ASM   1-16-1997  12:12:09        PAGE  1

LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                 LIST    P = 16C54, n = 66
           ;
           ;************************************************************
                 TITLE   "RS232 Communication Using PIC16C54"
           ;
           ;    Comments :
           ;               (1) Full Duplex
           ;               (2) Tested from 1200 to 9600 Baud( @ 8 Mhz )
           ;               (3) Tested from 1200 to 19200 Baud(@ 16 & 20 Mhz)
           ;
           ;       The User gets a total time as specified by the User Cycles
           ;       in the table ( or from equations ). The user routine has to
           ;       exactly use up this amount of time. After this time the User
           ;       routine has to give up the control to the Operating System.
           ;       If less than 52 uS is used, then the user should wait in a
           ;       delay loop, until exactly 52 uS.
           ;
           ;       Transmission :
           ;               Transmit Data is output on DX pin (Bit DX of PORTA).
           ;               In the user routine, the user should load the
           ;               data to be transmitted in the XmtReg and Set the
           ;               X_flag ( bsf FlagRX,X_flag ). This flag gets cleared
           ;               after the transmission.
           ;
           ;       Reception :
           ;               Data is received on pin DR ( bit DR of PORTA ).
           ;               The User should constantly check the "R_done" flag
           ;               to see if reception is over. If the reception is
           ;               in progress, R_flag is set to 1.
           ;               If the reception is over, "R_done" flag is set to 1.
           ;               The "R_done" flag gets reset to zero when a next start
           ;               bit is detected. So, the user should constantly check
           ;               the R_done flag, and if SET, then the received word
           ;               is in Register "RcvReg". This register gets cleared
           ;               when a new start bit is detected.
           ;
           ;    Program Memory :
           ;       Total Program Memory Locations Used ( except initialization
           ;       in "main" & User routine ) = 132 locations.
           ;
           ;    Data Memory :
           ;       Total Data memory locations (file registers used) = 6
           ;               2 File registers to hold Xmt Data & Rcv Data
           ;               1 File registers for Xmt/Rcv flag test bits
           ;               3 File registers for delay count & scratch pad
           ;
           ;    Stack :
           ;       Only one level of stack is used in the Operating System/RS232
           ;       routine. But this is freed as soon as the program returns to
           ;       the user routine.
           ;
           ;    Timer0 :     Not Used
           ;    WDT  :     Not Used
           ;
           ;       Program:          RS232.ASM
```

```
                            ;       Revision Date:
                            ;                        1-16-97    Compatibility with MPASMWIN 1.40
                            ;
                            ;*********************************************************************
                            ;
                                    INCLUDE         <P16C5X.INC>
                                    LIST
                            ; P16C5X.INC Standard Header File, Ver. 3.30 Microchip Technology, Inc.
                                    LIST

  000001FF                  PIC54   equ     1FFH    ; Define Reset Vector
  00000001                  Same    equ     1
  00000007                  MSB     equ     7

                                    INCLUDE         <RS232.H>
                            ;****************************************************************
                            ;                       RS232 Communication Parameters
                            ;
                            ;
  00000001                  X_MODE  equ 1  ; If (X_MODE==1) Then transmit LSB first
                                           ; if (X_MODE==0) Then transmit MSB first (CODEC like)
  00000001                  R_MODE  equ 1  ; If (R_MODE==1) Then receive LSB first
                                           ; if ( R_MODE==0) Then receive MSB first (CODEC like)
  00000001                  X_Nbit  equ 1  ; if (X_Nbit==1)#of data bits (Transmission) is 8 else 7
  00000001                  R_Nbit  equ 1  ; if (R_Nbit==1) # of data bits (Reception) is 8 else 7
                            ;
  00000000                  SB2     equ 0  ; if SB2 = 0 then 1 Stop Bit
                            ;              ; if SB2 = 1 then 2 Stop Bit
                            ;****************************************************************
                            ;       Transmit & Receive Test Bit Assignments
                            ;
  00000000                  X_flag  equ 0  ; Bit 0 of FlagRX
  00000002                  R_flag  equ 2  ; Bit 1 of FlagRX
  00000003                  S_flag  equ 3  ; Bit 2 of FlagRX
  00000004                  BitXsb  equ 4  ; Bit 3 of FlagRX
  00000005                  A_flag  equ 5
  00000006                  S_bit   equ 6  ; Xmt Stop Bit Flag( for 2/1 Stop bits )
                            ;
  00000001                  R_done  equ 1  ; When Reception complete, this bit is SET
  00000000                  X_done  equ X_flag  ; When Xmission complete, this bit is Cleared
                            ;
  00000000                  DX      equ 0  ; Transmit Pin ( Bit 0 of Port A )
  00000001                  DR      equ 1  ; Receive Pin ( Bit 1 of Port A )
                            ;
                            ;************************  Data RAM Assignments  ********************
                            ;
0008                                ORG     08H     ; Dummy Origin
                            ;
0008                        RcvReg  RES     1       ; Data received
0009                        XmtReg  RES     1       ; Data to be transmitted
000A                        Xcount  RES     1       ; Counter for #of Bits Transmitted
000B                        Rcount  RES     1       ; Counter for #of Bits to be Received
000C                        DlyCnt  RES     1       ; Counter for Delay constant
000D                        FlagRX  RES     1       ; Transmit & Receive test flag hold register
                            ;______________________________________________________________
                            ;    Constants       19200   9600    4800    2400    1200
                            ;  ( @ 20 Mhz )
                            ;______________________________________________________________
                            ;       K0             0      13       57     143      317*
                            ;       K1            49      98      184     358*     705*
                            ;       K2            34      60      103     191      364*
                            ;       K3            27      53       96     184      357*
                            ;       K4            29      55       98     186      359*
                            ;       K5            30      56       99     187      360*
                            ;       K6             0       0        0       0        0
                            ;       K7            56     104      190     365*     712*
```

```
                   ;
                   ;    User Cycles    118     260      521     1042     2083
                   ; *************************************************************
                   ;
                   ;
                   ;______________________________________________________________
                   ;    Constants       19200   9600    4800    2400    1200
                   ;  ( @ 8 Mhz )
                   ;______________________________________________________________
                   ;       K0            --      0        5      39      109
                   ;       K1            --     39       80     150      288*
                   ;       K2            --     27       51      86      155
                   ;       K3            --     21       44      80      148
                   ;       K4            --     23       46      82      150
                   ;       K5            --     24       47      83      151
                   ;       K6            --      0        0       0        0
                   ;       K7            --     45       86     156      295*
                   ;
                   ;    User_Cycles      --     86      208     416      832
                   ; *************************************************************
                   ;
                   ;______________________________________________________________
                   ;    Constants       19200   9600    4800    2400    1200
                   ;  ( @ 4 Mhz )
                   ;______________________________________________________________
                   ;       K0            --     --        0       5       39
                   ;       K1            --     --       39      80      150
                   ;       K2            --     --       27      51       86
                   ;       K3            --     --       21      44       80
                   ;       K4            --     --       23      46       82
                   ;       K5            --     --       24      47       83
                   ;       K6            --     --        0       0        0
                   ;       K7            --     --       45      86      156
                   ;
                   ;    User_Cycles      --     --       86     208      416
                   ; *************************************************************

                   ;
                   ; The constants marked " * " are >255. To implement these constants
                   ; in delay loops, the delay loop should be broken into 2 or more loops.
                   ; For example, 357 = 255+102. So 2 delay loops, one with 255 and
                   ; the other with 102 may be used.
                   ;*******************************************************************
                   ;       Set Delay Constants for 9600 Baud @ CLKIN = 8 Mhz
                   ;
  00000000         K0      EQU     .0
  00000027         K1      EQU     .39
  0000001B         K2      EQU     .27
  00000015         K3      EQU     .21
  00000017         K4      EQU     .23
  00000018         K5      EQU     .24
  00000000         K6      EQU     .0
  0000002D         K7      EQU     .45
                   ;
                   ;*******************************************************************

                   ;
0000                       ORG     0
                   ;*******************************************************************
                   ;
0000 0C01          Delay   movlw   K0+1
0001 002C                  movwf   DlyCnt        ; Total Delay = 3K+6
0002 02EC          redo    decfsz  DlyCnt,Same
0003 0A02                  goto    redo
0004 0800                  retlw   0
                   ;
```

```
0005 002C      Delay1  movwf   DlyCnt
0006 02EC      redo_1  decfsz  DlyCnt,Same    ;
0007 0A06              goto    redo_1
0008 0A8D              goto    User
               ;
0009 002C      Delay2  movwf   DlyCnt
000A 02EC      redo_2  decfsz  DlyCnt,Same     ; Delay =   = 260 Cycles
000B 0A0A              goto    redo_2
000C 0A67              goto    User_1
               ;
000D 0625      R_strt  btfsc   PORTA,DR        ; check for a Start Bit
000E 0A17              goto    ShellY          ; delay for 104/2 uS
000F 042D              bcf     FlagRX,R_done   ; Reset Receive done flag
0010 054D              bsf     FlagRX,R_flag   ; Set flag for Reception in Progress
0011 078D              btfss   FlagRX,BitXsb
0012 05AD              bsf     FlagRX,A_flag   ; A_flag is for start bit detected in R_strt
0013 0068              clrf    RcvReg          ; Clear all bits of RcvReg
                       IF      R_Nbit
0014 0C08              movlw   8               ; 8 Data bits
                       ELSE
                       movlw   7               ; 7 data bits
                       ENDIF
0015 002B              movwf   Rcount
0016 0A78              goto    Shell           ; delay for 104+104/4
               ;
0017 078D      ShellY  btfss   FlagRX,BitXsb
0018 0A78              goto    Shell
0019 054D              bsf     FlagRX,R_flag
001A 0A78              goto    Shell
               ;
001B 0403      R_next  bcf     STATUS,C
                       IF      R_MODE
001C 0328              rrf     RcvReg,Same     ; to set if MSB first or LSB first
                       ELSE
                       rlf     RcvReg,Same
                       ENDIF
001D 0625              btfsc   PORTA,DR
                       IF      R_MODE
                         IF      R_Nbit
001E 05E8                bsf     RcvReg,MSB    ; Conditional Assembly
                         ELSE
                         bsf     RcvReg,MSB-1
                         ENDIF
                       ELSE
                       bsf     RcvReg,LSB
                       ENDIF
001F 02EB              decfsz  Rcount,Same
0020 0A78              goto    Shell
0021 044D              bcf     FlagRX,R_flag
0022 056D              bsf     FlagRX,S_flag
0023 052D              bsf     FlagRX,R_done
0024 0A78              goto    Shell
               ;
               ;       Reception Done
               ;
0025 0405      X_strt  bcf     PORTA,DX        ; Send Start Bit
                       IF      X_Nbit
0026 0C08              movlw   8
                       ELSE
                       movlw   7
                       ENDIF
0027 002A              movwf   Xcount
                       IF      X_MODE
                       ELSE
                         IF    X_Nbit
                         ELSE
```



```
                  rlf    XmtReg,Same
                  ENDIF
                ENDIF
0028 0A50       goto     X_SB
          ;
0029 0403 X_next  bcf    STATUS,C
                IF       X_MODE
002A 0329       rrf      XmtReg,Same      ; Conditional Assembly
                ELSE                      ; to set if MSB first or LSB first
                rlf      XmtReg,Same
                ENDIF
002B 0603       btfsc    STATUS,C
002C 0505       bsf      PORTA,DX
002D 0703       btfss    STATUS,C
002E 0405       bcf      PORTA,DX
002F 00EA       decf     Xcount,Same
0030 0A52       goto     X_Data
          ;
0031 040D X_SB_1  bcf    FlagRX,X_flag    ; Xmt flag = 0 -- transmission over
0032 0C09       movlw    9
0033 002A       movwf    Xcount
0034 0505       bsf      PORTA,DX         ; Send Stop Bit
0035 0A60       goto     X_Stop
          ;
0036 0505 X_SB_2  bsf    PORTA,DX
0037 04CD       bcf      FlagRX,S_bit
0038 0A60       goto     X_Stop
          ;
          ;   End of Transmission
          ;
0039 076D R0_X0   btfss  FlagRX,S_flag
003A 0A8D       goto     User
003B 046D       bcf      FlagRX,S_flag
003C 0900       call     Delay
003D 0C2E       movlw    K7+1
003E 0A05       goto     Delay1
          ;
003F      R1_X0
003F 0900       call     Delay
0040 0C28       movlw    K1+1             ; delay for 1st bit is 104+104/4
0041 002C       movwf    DlyCnt
                IF       R_Nbit
0042 0C08       movlw    8                ; 8 Data bits
                ELSE
                movlw    7                ; 7 data bits
                ENDIF
0043 018B       xorwf    Rcount,W
0044 0643       btfsc    STATUS,Z
0045 0A06       goto     redo_1
0046 0C1C       movlw    K2+1
0047 0A05       goto     Delay1
          ;
0048      R1_X1                           ; same as R0_X1
0048 0C09 R0_X1   movlw  9
0049 008A       subwf    Xcount,W
004A 0643       btfsc    STATUS,Z
004B 0A25       goto     X_strt
004C 022A       movf     Xcount,Same      ; to check if All data bits Xmted
004D 0743       btfss    STATUS,Z
004E 0A29       goto     X_next
                IF       SB2
                  btfsc  FlagRX,S_bit
                  goto   X_SB_2
                  bsf    FlagRX,S_bit
                  goto   X_SB_1
                ELSE
```



```
004F 0A31                goto    X_SB_1
                       ENDIF
                 ;
                 ;
0050 0A51        X_SB    goto    cycle4
0051 0A52        cycle4  goto    X_Data
                 ;
0052 06AD        X_Data  btfsc   FlagRX,A_flag
0053 0A59                goto    SbDly
0054 068D                btfsc   FlagRX,BitXsb
0055 0A5D                goto    ABC
0056 0900                call    Delay
0057 0C16                movlw   K3+1
0058 0A09                goto    Delay2
                 ;
0059 04AD        SbDly   bcf     FlagRX,A_flag
005A 0900                call    Delay
005B 0C18                movlw   K4+1
005C 0A09                goto    Delay2
                 ;
005D 048D        ABC     bcf     FlagRX,BitXsb
005E 0900                call    Delay
005F 0A67                goto    User_1
                 ;
0060             X_Stop
0060 06AD                btfsc   FlagRX,A_flag
0061 0A59                goto    SbDly
0062 068D                btfsc   FlagRX,BitXsb
0063 0A5D                goto    ABC
0064 0900                call    Delay
0065 0C19                movlw   K5+1
0066 0A09                goto    Delay2
                 ;
0067 064D        User_1  btfsc   FlagRX,R_flag
0068 0A77                goto    Sync_1          ; Reception already in progress
0069 066D                btfsc   FlagRX,S_flag
006A 0A74                goto    Sync_3
006B 0625                btfsc   PORTA,DR       ; check for a Start Bit
006C 0A77                goto    Sync_2          ; No Start Bit - goto User routine
006D 042D                bcf     FlagRX,R_done   ; Reset Receive done flag
006E 044D                bcf     FlagRX,R_flag
006F 058D                bsf     FlagRX,BitXsb   ; Set flag for Reception in Progress
0070 0068                clrf    RcvReg          ; Clear all bits of RcvReg
                         IF      R_Nbit
0071 0C08                movlw   8               ; 8 Data bits
                         ELSE
                         movlw   7               ; 7 data bits
                         ENDIF
0072 002B                movwf   Rcount
0073 0A8D                goto    User
                 ;
0074 046D        Sync_3  bcf     FlagRX,S_flag
0075 0C01                movlw   K6+1
0076 0A05                goto    Delay1
                 ;
0077             Sync_1
0077 0A8D        Sync_2  goto    User
                 ;
                 ;********************************************************************
                 ;
0078 064D        Shell   btfsc   FlagRX,R_flag
0079 0A7D                goto    Chek_X
007A 060D                btfsc   FlagRX,X_flag
007B 0A48                goto    R0_X1
007C 0A39                goto    R0_X0            ; Case for R0_X0
007D 060D        Chek_X  btfsc   FlagRX,X_flag
```

```
007E 0A48               goto    R1_X1
007F 0A3F               goto    R1_X0
                 ;
                 ;
                 ;*********************************************************************
                 ;       Operating System
                 ;  The User routine after time = B/2, should branch Here
                 ;
0080 074D        Op_Sys  btfss   FlagRX,R_flag
0081 0A0D               goto    R_strt
0082 0A1B               goto    R_next
                 ;
                 ;*********************************************************************
                 ;
0083 0C0E        main    movlw   0EH             ; Bit 0 of Port A is Output
0084 0005               tris    PORTA           ; Set PORTA.0 as output ( DX )
                 ;                                 & PORTA.1 is input ( DR )
0085 0505               bsf     PORTA,DX
0086 0C09               movlw   9
0087 002A               movwf   Xcount          ; If Xcount == 9, Then send start bit
0088 006D               clrf    FlagRX          ; Clear All flag bits.
                        IF      SB2
                          bsf     FlagRX,S_bit  ; Set Xmt Stop bit flag(2 Stop Bits)
                        ELSE
0089 04CD                 bcf     FlagRX,S_bit  ; Clear Xmt Stop bit flag
                        ENDIF
008A 0C1F               movlw   1FH             ; Prescaler = 4
008B 0002               OPTION                  ; Set TMR0 increment on internal Clock
008C 0A80               goto    Op_Sys
                 ;
                 ;*********************************************************************
                 ;
                 ;       *******************  User Routine  ***************
                 ;  The User routine should use up time exactly = User time as given
                 ;  in the Constants Table ( or by Equations for constants ).
                 ;  At 9600, this 86 Clock Cycles. TMR0 timer is used here to count
                 ;  upto 86 cycles ( From 128-86 To 0 ) by examining Bit 7 of TMR0.
                 ;
  00000030       K_user  equ     .128+.6-.86
                 ;
008D 0C30        User    movlw   K_user
008E 0021               movwf   TMR0
008F 062D               btfsc   FlagRX,R_done
0090 0A97               goto    ErrChk
0091 060D        SetXmt  btfsc   FlagRX,X_flag
0092 0A9C               goto    Op
0093 0C41               movlw   41H
0094 0029               movwf   XmtReg
0095 050D               bsf     FlagRX,X_flag   ; Enable Xmission
0096 0A9C               goto    Op
                 ;
0097             ErrChk
0097 0C5A               movlw   “Z”
0098 0188               xorwf   RcvReg,W
0099 0643               btfsc   STATUS,Z
009A 0A91               goto    SetXmt
009B 0A9B        err1    goto    err1            ; Received word is not “Z”
                 ;
009C 07E1        Op      btfss   TMR0,MSB        ; Test for TMR0 bit 7
009D 0A9C               goto    Op              ; If Set, Then TMR0 has incremented
009E 0A80        Oflow   goto    Op_Sys          ; to 128.
                 ;
                 ; ***********************************************************
                 ;
01FF                    ORG     PIC54
01FF 0A83               goto    main
```

```
               00342
               00343         END

MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXX- ---------------- ----------------
01C0 : ---------------- ---------------- ---------------- ---------------X

All other memory blocks unused.

Program Memory Words Used:   160
Program Memory Words Free:   352

Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,     0 suppressed
```

Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products 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 PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be 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 product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

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, KEELOQ® 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.

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: http://www.microchip.com

Rocky Mountain
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-7456

Atlanta
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
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, Indiana 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

New York
150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

San Jose
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto
6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

Australia
Microchip Technology Australia Pty Ltd
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

China - Beijing
Microchip Technology Consulting (Shanghai) Co., Ltd., Beijing Liaison Office
Unit 915
Bei Hai Wan Tai Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Chengdu
Microchip Technology Consulting (Shanghai) Co., Ltd., Chengdu Liaison Office
Rm. 2401, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-6766200 Fax: 86-28-6766599

China - Fuzhou
Microchip Technology Consulting (Shanghai) Co., Ltd., Fuzhou Liaison Office
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506 Fax: 86-591-7503521

China - Shanghai
Microchip Technology Consulting (Shanghai) Co., Ltd.
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

China - Shenzhen
Microchip Technology Consulting (Shanghai) Co., Ltd., Shenzhen Liaison Office
Rm. 1315, 13/F, Shenzhen Kerry Centre,
Renminnan Lu
Shenzhen 518001, China
Tel: 86-755-2350361 Fax: 86-755-2366086

Hong Kong
Microchip Technology Hongkong Ltd.
Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

India
Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan
Microchip Technology Japan K.K.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

Korea
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Singapore
Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-6334-8870 Fax: 65-6334-8850

Taiwan
Microchip Technology Taiwan
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Denmark
Microchip Technology Nordic ApS
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France
Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - ler Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany
Microchip Technology GmbH
Gustav-Heinemann Ring 125
D-81739 Munich, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy
Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

United Kingdom
Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820

03/01/02