Implementation of an Asynchronous Serial I/O

INTRODUCTION

The PIC16C5X series from Microchip Technology Inc., are 8-bit, high-speed, EPROM-based microcontrollers. This application note describes the implementation of an Asynchronous serial I/O using Microchip's PIC16C5X series of high-speed 8-bit microcontrollers. These EPROM based microcontrollers can operate at very high speeds with a minimum of 200 ns cycle time @ 20 MHz input clock. Many microcontroller applications require chip-to-chip serial data communications. Since the PIC16C5X series have 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 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 simulates a full duplex RS-232 communication and the other provides implementation of half duplex communication. Using Half-Duplex, rates up to 19200 baud can be implemented using an 8 MHz input clock. In case of 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 flow chart for serial transmission and Figure 2 shows a flow chart for reception. The flowcharts show cases for 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 in case of 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 (104µs in case of 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 Figures 3 and 4.

FIGURE 1 - TRANSMISSION FLOW CHART

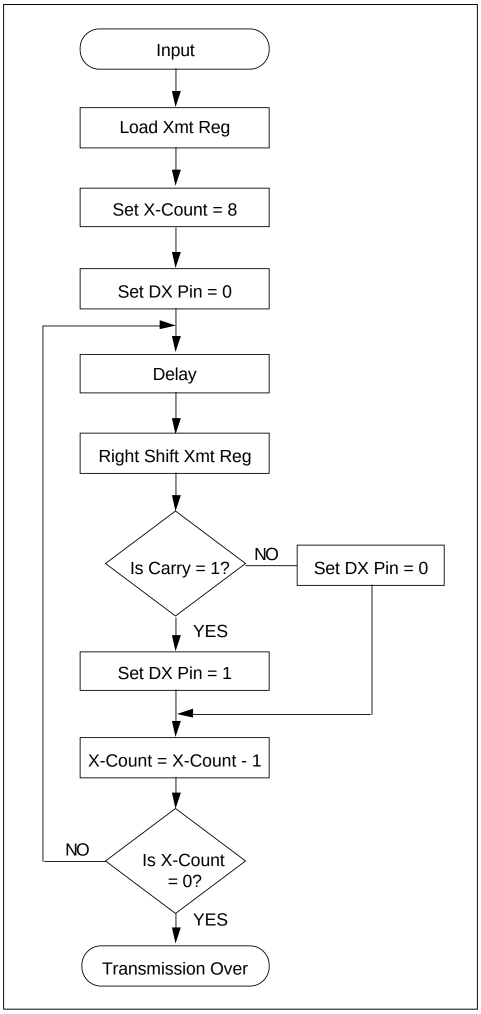

FIGURE 2 - RECEPTION FLOW CHART

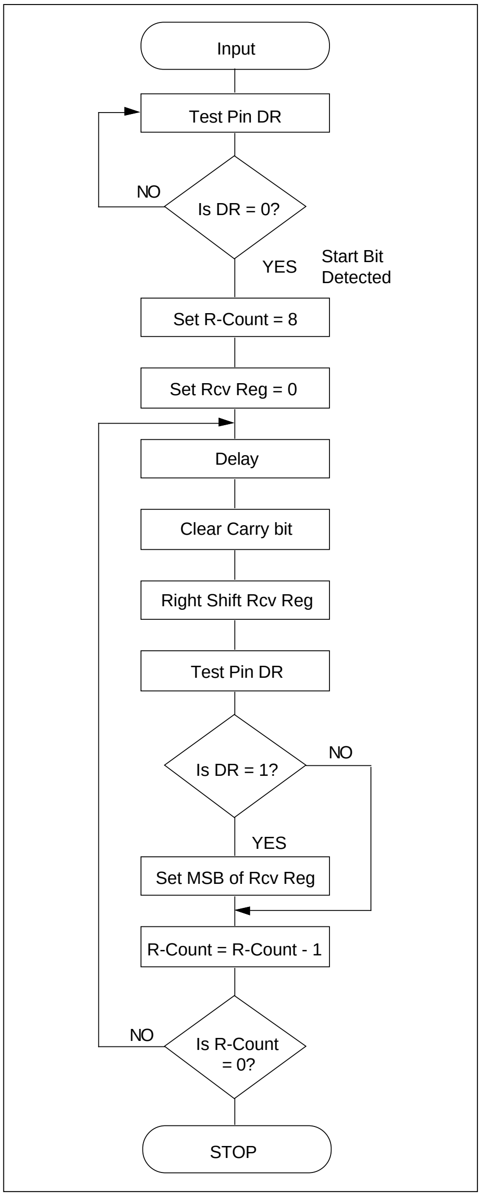

FIGURE 3 - TRANSMIT ASSEMBLY CODE (CORRESPONDING TO FIGURE 1)

```
;*********************  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
```

FIGURE 4 - 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 1
Bit
              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 the 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 half 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 at 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 Figure 6.

FIGURE 5 - FULL DUPLEX BLOCK DIAGRAM

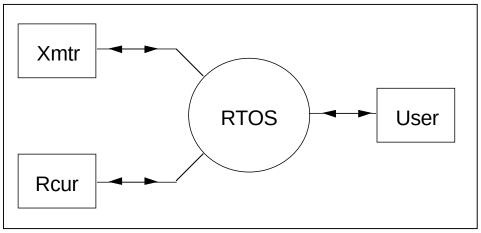

FIGURE 6 - EQUATIONS FOR DELAY CONSTANTS

```
  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-18 - 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 is used by the Operating System and the other 4 µs is 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 the RTCC timer with the Prescaler option for calculating the real time. With RTCC set 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 RS232 code (RTOS), so that the RTCC is available to the User for other important functions. If the RTCC is not used for other functions, the User may modify the code to replace the software delay loops, by counting the RTCC. For an example of using this method of counting 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 the control is given back to the user. The watchdog timer and RTCC are not used. The user should clear the watchdog timer at regular intervals, if the WDT is enabled.

The usage of the program is described below. The user should branch to location "Op_Sys" exactly after the time specified in Table 1 or as computed from Equations in Figure 6. 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 either R_done flag or RcvReg.

TRANSMISSION

Transmit Data is output on DX pin (Bit 0 of Port_A). 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. The user should check this flag (X_flag) to see if transmission is in progress. Modifying XmtReg when X_flag is set will cause erroneous data to be transmitted.

RECEPTION

Data is received on pin DR (Bit 1 of Port_A). 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. 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. 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 receive register 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

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

Implementation of an Asynchronous Serial I/O

Author: Amar Palacherla
Logic Products Division

2

APPENDIX A: ASSEMBLY LANGUAGE FOR HALF DUPLEX

```
MPASM 01.00.02 Alpha  C:\AP-NOTES\   6-24-1994  8:56:3                 PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT

                     0001 ;                       RS-232 Communication With PIC16C54
                     0002 ;
                     0003 ;        Half Duplex Asynchronous Communication
                     0004 ;
                     0005 ;        This program has been tested at Bauds from 1200 to 19200 Baud
                     0006 ;        ( @ 8,16,20 Mhz CLKIN )
                     0007 ;
                     0008 ;       As a test, this program will echo back the data that has been
                     0009 ;       received.
                     0010 ;
                     0011 ;
                     0012        LIST            P=16C54, T=ON
                     0013    INCLUDE         “PICREG.H”
                     0001 ;*************************  PIC16C5X Header    *************************
01FF                 0002 PIC54   equ     1FFH    ; Define Reset Vectors
01FF                 0003 PIC55   equ     1FFH
03FF                 0004 PIC56   equ     3FFH
07FF                 0005 PIC57   equ     7FFH
                     0006 ;
0001                 0007 RTCC    equ     1
0002                 0008 PC      equ     2
0003                 0009 STATUS  equ     3       ; F3 Reg is STATUS Reg.
0004                 0010 FSR     equ     4
                     0011 ;
0005                 0012 Port_A  equ     5
0006                 0013 Port_B  equ     6       ; I/O Port Assignments
0007                 0014 Port_C  equ     7
                     0015 ;
                     0016 ;***********************************************************************
                     0017 ;
                     0018 ;                       ; STATUS REG. Bits
0000                 0019 CARRY   equ     0       ; Carry Bit is Bit.0 of F3
0000                 0020 C       equ     0
0001                 0021 DCARRY  equ     1
0001                 0022 DC      equ     1
0002                 0023 Z_bit   equ     2       ; Bit 2 of F3 is Zero Bit
0002                 0024 Z       equ     2
0003                 0025 P_DOWN  equ     3
0003                 0026 PD      equ     3
0004                 0027 T_OUT   equ     4
0004                 0028 TO      equ     4
0005                 0029 PA0     equ     5
```

```
0006                 0030 PA1     equ     6
0007                 0031 PA2     equ     7
                     0032 ;
0001                 0033 Same    equ     1
0000                 0034 W       equ     0
                     0035 ;
0000                 0036 LSB     equ     0
0007                 0037 MSB     equ     7
                     0038 ;
0001                 0039 TRUE    equ     1
0001                 0040 YES     equ     1
0000                 0041 FALSE   equ     0
0000                 0042 NO      equ     0
                     0043 ;
                     0044 ;***********************************************************************
                     0045
                     0013
                     0014 ;*****************  Communication Parameters   **************************
                     0015 ;
0001                 0016 X_MODE  equ     1       ; If ( X_MODE==1) Then transmit LSB first
                     0017 ;                         if ( X_MODE==0) Then transmit MSB first ( CODEC like )
0001                 0018 R_MODE  equ     1       ; If ( R_MODE==1) Then receive LSB first
                     0019 ;                         if ( X_MODE==0) Then receive MSB first ( CODEC like )
0001                 0020 X_Nbit  equ     1       ; if (X_Nbit==1) # of data bits ( Transmission ) is 8 else 7
0001                 0021 R_Nbit  equ     1       ; if (R_Nbit==1) # of data bits ( Reception ) is 8 else 7
                     0022 ;
0000                 0023 Sbit2   equ     0       ; if Sbit2 = 0 then 1 Stop Bit else 2 Stop Bits
                     0024 ;
                     0025 ;***********************************************************************
0005                 0026 X_flag  equ     PA0     ; Bit 5 of F3 ( PA0 )
0006                 0027 R_flag  equ     PA1     ; Bit 6 of F3 ( PA1 )
                     0028 ;
0000                 0029 DX      equ     0       ; Transmit Pin ( Bit 0 of Port A )
0001                 0030 DR      equ     1       ; Reciive Pin ( Bit 1 of Port A )
                     0031 ;
                     0032 ;
0044                 0033 BAUD_1  equ     .68     ; 3+3X = CLKOUT/Baud
0043                 0034 BAUD_2  equ     .67     ; 6+3X = CLKOUT/Baud
0022                 0035 BAUD_3  equ     .34     ; 3+3X = 0.5*CLKOUT/Baud
0056                 0036 BAUD_4  equ     .86     ; 3+3X = 1.25*CLKOUT/Baud
0042                 0037 BAUD_X  equ     .66     ; 11+3X = CLKOUT/Baud
0042                 0038 BAUD_Y  equ     .66     ; 9 +3X = CLKOUT/Baud
                     0039 ;
                     0040 ;***********************  Data RAM Assignments  *********************
                     0041 ;
                     0042         ORG     08H     ; Dummy Origin
                     0043 ;
0008  0001           0044 RcvReg  RES     1       ; Data received
```

```
0009  0001        XmtReg  RES     1               ; Data to be transmitted
000A  0001        Count   RES     1               ; Counter for #of Bits Transmitted
000B  0001        DlyCnt  RES     1
                 ;*************************************************************************
                 ;
                          ORG     0
                 ;
0000 0068         Talk    clrf    RcvReg           ; Clear all bits of RcvReg
0001 0625                 btfsc   Port_A,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
                 ;
                  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,CARRY
                          IF      R_MODE
0007 0328                 rrf     RcvReg,Same      ; to set if MSB first or LSB first
                          ELSE
                          rlf     RcvReg,Same
                          ENDIF
0008 0625                 btfsc   Port_A,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
```

```
                     0093 ;
                     0094 Xmtr
                     0095         IF      X_Nbit
000F 0C08            0096         movlw   8
                     0097         ELSE
                     0098         movlw   7
                     0099         ENDIF
0010 002A            0100         movwf   Count
                     0101 ;
                     0102         IF      X_MODE
                     0103         ELSE
                     0104           IF    X_Nbit
                     0105           ELSE
                     0106           rlf   XmtReg,Same
                     0107           ENDIF
                     0108         ENDIF
                     0109 ;
0011 0405            0110         bcf     Port_A,DX       ; Send Start Bit
0012 0925            0111         call    Delay1
0013 0403            0112 X_next  bcf     STATUS,CARRY
                     0113 ;
                     0114         IF      X_MODE
0014 0329            0115         rrf     XmtReg,Same     ; Conditional Assembly
                     0116         ELSE                    ; to set if MSB first or LSB first
                     0117         rlf     XmtReg,Same
                     0118         ENDIF
                     0119 ;
0015 0603            0120         btfsc   STATUS,CARRY
0016 0505            0121         bsf     Port_A,DX
0017 0703            0122         btfss   STATUS,CARRY
0018 0405            0123         bcf     Port_A,DX
0019 0921            0124         call    DelayX
001A 02EA            0125         decfsz  Count,Same
001B 0A13            0126         goto    X_next
001C 0505            0127         bsf     Port_A,DX       ; Send Stop Bit
001D 0925            0128         call    Delay1
                     0129 ;
                     0130         IF      Sbit2
                     0131         bsf     Port_A,DX
                     0132         call    Delay1
                     0133         ENDIF
                     0134 ;
001E 0A00            0135         goto    Talk            ; Back To Reception & Transmision
                     0136 ;
                     0137 ;   End of Transmission
                     0138 ;
001F 0C42            0139 DelayY  movlw   BAUD_Y
0020 0A28            0140         goto    save
```

```
0021 0C42           0141 DelayX  movlw   BAUD_X
0022 0A28           0142         goto    save
0023 0C56           0143 Delay4  movlw   BAUD_4
0024 0A28           0144         goto    save
0025 0C44           0145 Delay1  movlw   BAUD_1          ; 104 uS for 9600 baud
0026 0A28           0146         goto    save
0027 0C43           0147 Delay2  movlw   BAUD_2
0028 002B           0148 save    movwf   DlyCnt
0029 02EB           0149 redo_1  decfsz  DlyCnt,Same
002A 0A29           0150         goto    redo_1
002B 0800           0151         retlw   0
                    0152 ;
002C 0C0E           0153 main    movlw   0EH             ; Bit 0 of Port A is Output
002D 0005           0154         tris    Port_A          ; Set Port_A.0 as output ( DX )
002E 0525           0155         bsf     Port_A,DR
                    0156 ;
002F 0A00           0157         goto    Talk
                    0158 ;
                    0159 ;
0030 0C22           0160 User    movlw   BAUD_3
0031 002B           0161         movwf   DlyCnt
0032 02EB           0162 redo_2  decfsz  DlyCnt,Same
0033 0A32           0163         goto    redo_2
0034 0A00           0164         goto    Talk            ; Loop Until Start Bit Found
                    0165 ;
                    0166 ;
                    0167         ORG     PIC54
01FF 0A2C           0168         goto    main
                    0169 ;
                    0170         END
                    0171
MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXX-----------
0040 : ---------------- ---------------- ---------------- ----------------

0180 : ---------------- ---------------- ---------------- ----------------
01C0 : ---------------- ---------------- ---------------- ---------------X

All other memory blocks unused.
Errors   :    0
Warnings :    0
```

APPENDIX B: ASSEMBLY LANGUAGE LISTING FOR FULL DUPLEX

```
MPASM 01.00.02 Alpha  C:\AP-NOTES\   6-24-1994  9:8:43                 PAGE  1
RS232 Communication Using PIC16C54

LOC  OBJECT CODE     LINE SOURCE TEXT

                     0001 ;************************************************************************
                     0002        TITLE   “RS232 Communication Using PIC16C54”
                     0003 ;
                     0004 ;    Comments :
                     0005 ;               (1) Full Duplex
                     0006 ;               (2) Tested from 1200 to 9600 Baud( @ 8 Mhz )
                     0007 ;               (3) Tested from 1200 to 19200 Baud(@ 16 & 20 Mhz)
                     0008 ;
                     0009 ;       The User gets a total time as specified by the User Cycles
                     0010 ;       in the table ( or from equations ). The user routine has to
                     0011 ;       exactly use up this amount of time. After this time the User
                     0012 ;       routine has to give up the control to the Operating System.
                     0013 ;       If less than 52 uS is used, then the user should wait in a
                     0014 ;       delay loop, until exactly 52 uS.
                     0015 ;
                     0016 ;       Transmission :
                     0017 ;               Transmit Data is output on DX pin (Bit DX of Port_A).
                     0018 ;               In the user routine, the user should load the
                     0019 ;               data to be transmitted in the XmtReg and Set the
                     0020 ;               X_flag ( bsf FlagRX,X_flag ). This flag gets cleared
                     0021 ;               after the transmission.
                     0022 ;
                     0023 ;       Reception :
                     0024 ;               Data is received on pin DR ( bit DR of Port_A ).
                     0025 ;               The User should constantly check the “R_done” flag
                     0026 ;               to see if reception is over. If the reception is
                     0027 ;               in progress, R_flag is set to 1.
                     0028 ;               If the reception is over, “R_done” flag is set to 1.
                     0029 ;               The “R_done” flag gets reset to zero when a next start
                     0030 ;               bit is detected. So, the user should constantly check
                     0031 ;               the R_done flag, and if SET, then the received word
                     0032 ;               is in Register “RcvReg”. This register gets cleared
                     0033 ;               when a new start bit is detected.
                     0034 ;
                     0035 ;    Program Memory :
                     0036 ;       Total Program Memory Locations Used ( except initialization
                     0037 ;       in “main” & User routine ) = 132 locations.
                     0038 ;
                     0039 ;    Data Memory :
                     0040 ;       Total Data memory locations (file registers used) = 6
                     0041 ;               2 File registers to hold Xmt Data & Rcv Data
                     0042 ;               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.
                     ;
                     ;    RTCC :     Not Used
                     ;    WDT  :     Not Used
                     ;
                             LIST            P=16C54, T=ON
                         INCLUDE         "PICREG.H"
                     ;**************************  PIC16C5X Header    *************************
01FF                 PIC54   equ     1FFH    ; Define Reset Vectors
01FF                 PIC55   equ     1FFH
03FF                 PIC56   equ     3FFH
07FF                 PIC57   equ     7FFH
                     ;
0001                 RTCC    equ     1
0002                 PC      equ     2
0003                 STATUS  equ     3       ; F3 Reg is STATUS Reg.
0004                 FSR     equ     4
                     ;
0005                 Port_A  equ     5
0006                 Port_B  equ     6       ; I/O Port Assignments
0007                 Port_C  equ     7
                     ;
                     ;*************************************************************************
                     ;
                     ;                       ; STATUS REG. Bits
0000                 CARRY   equ     0       ; Carry Bit is Bit.0 of F3
0000                 C       equ     0
0001                 DCARRY  equ     1
0001                 DC      equ     1
0002                 Z_bit   equ     2       ; Bit 2 of F3 is Zero Bit
0002                 Z       equ     2
0003                 P_DOWN  equ     3
0003                 PD      equ     3
0004                 T_OUT   equ     4
0004                 TO      equ     4
0005                 PA0     equ     5
0006                 PA1     equ     6
0007                 PA2     equ     7
                     ;
0001                 Same    equ     1
0000                 W       equ     0
                     ;
0000                 LSB     equ     0
```

```
0007                0037 MSB      equ     7
                    0038 ;
0001                0039 TRUE     equ     1
0001                0040 YES      equ     1
0000                0041 FALSE    equ     0
0000                0042 NO       equ     0
                    0043 ;
                    0044 ;************************************************************************
                    0045
                    0054
                    0055     INCLUDE         "RS232.H"
                    0001 ;************************************************************************
                    0002 ;                       RS232 Communication Parameters
                    0003 ;
                    0004 ;
0001                0005 X_MODE   equ     1       ; If ( X_MODE==1) Then transmit LSB first
                    0006 ;                           if ( X_MODE==0) Then transmit MSB first ( CODEC like )
0001                0007 R_MODE   equ     1       ; If ( R_MODE==1) Then receive LSB first
                    0008 ;                           if ( R_MODE==0) Then receive MSB first ( CODEC like )
0001                0009 X_Nbit   equ     1       ; if ( X_Nbit==1) # of data bits ( Transmission ) is 8 else 7
0001                0010 R_Nbit   equ     1       ; if ( R_Nbit==1) # of data bits ( Reception ) is 8 else 7
                    0011 ;
0000                0012 SB2      equ     0       ; if SB2 = 0 then 1 Stop Bit
                    0013 ;                        ; if SB2 = 1 then 2 Stop Bit
                    0014 ;**********************************************************************
                    0015 ;       Transmit & Receive Test Bit Assignments
                    0016 ;
0000                0017 X_flag   equ     0       ; Bit 0 of FlagRX
0002                0018 R_flag   equ     2       ; Bit 1 of FlagRX
0003                0019 S_flag   equ     3       ; Bit 2 of FlagRX
0004                0020 BitXsb   equ     4       ; Bit 3 of FlagRX
0005                0021 A_flag   equ     5
0006                0022 S_bit    equ     6       ; Xmt Stop Bit Flag( for 2/1 Stop bits )
                    0023 ;
0001                0024 R_done   equ     1       ; When Reception complete, this bit is SET
0000                0025 X_done   equ     X_flag  ; When Xmission complete, this bit is Cleared
                    0026 ;
0000                0027 DX       equ     0       ; Transmit Pin ( Bit 0 of Port A )
0001                0028 DR       equ     1       ; Reciive Pin ( Bit 1 of Port A )
                    0029 ;
                    0030 ;***********************  Data RAM Assignments  *********************
                    0031 ;
                    0032          ORG     08H     ; Dummy Origin
                    0033 ;
0008  0001          0034 RcvReg   RES     1       ; Data received
0009  0001          0035 XmtReg   RES     1       ; Data to be transmitted
000A  0001          0036 Xcount   RES     1       ; Counter for #of Bits Transmitted
000B  0001          0037 Rcount   RES     1       ; Counter for #of Bits to be Received
```

```
000C  0001              0038 DlyCnt  RES     1       ; Counter for Delay constant
000D  0001              0039 FlagRX  RES     1       ; Transmit & Receive test flag hold register
                        0040 ;_____________________________________________________________
                        0041 ;    Constants       19200   9600    4800    2400    1200
                        0042 ;  ( @ 20 Mhz )
                        0043 ;_____________________________________________________________
                        0044 ;       K0             0      13       57     143      317*
                        0045 ;       K1            49      98      184     358*     705*
                        0046 ;       K2            34      60      103     191      364*
                        0047 ;       K3            27      53       96     184      357*
                        0048 ;       K4            29      55       98     186      359*
                        0049 ;       K5            30      56       99     187      360*
                        0050 ;       K6             0       0        0       0        0
                        0051 ;       K7            56     104      190     365*     712*
                        0052 ;
                        0053 ;    User Cycles    118     260      521     1042     2083
                        0054 ; ************************************************************
                        0055 ;
                        0056 ;
                        0057 ;_____________________________________________________________
                        0058 ;    Constants       19200   9600    4800    2400    1200
                        0059 ;  ( @ 8 Mhz )
                        0060 ;_____________________________________________________________
                        0061 ;       K0             —       0        5      39      109
                        0062 ;       K1             —      39       80     150      288*
                        0063 ;       K2             —      27       51      86      155
                        0064 ;       K3             —      21       44      80      148
                        0065 ;       K4             —      23       46      82      150
                        0066 ;       K5             —      24       47      83      151
                        0067 ;       K6             —       0        0       0        0
                        0068 ;       K7             —      45       86     156      295*
                        0069 ;
                        0070 ;    User_Cycles      —      86      208     416      832
                        0071 ; ************************************************************
                        0072 ;
                        0073 ;_____________________________________________________________
                        0074 ;    Constants       19200   9600    4800    2400    1200
                        0075 ;  ( @ 4 Mhz )
                        0076 ;_____________________________________________________________
                        0077 ;       K0             —       —        0       5       39
                        0078 ;       K1             —       —       39      80      150
                        0079 ;       K2             —       —       27      51       86
                        0080 ;       K3             —       —       21      44       80
                        0081 ;       K4             —       —       23      46       82
                        0082 ;       K5             —       —       24      47       83
                        0083 ;       K6             —       —        0       0        0
                        0084 ;       K7             —       —       45      86      156
                        0085 ;
```

```
                     0086 ;    User_Cycles      —      —      86     208      416
                     0087 ; *************************************************************
                     0088
                     0089 ;
                     0090 ; The constants marked “ * “ are >255. To implement these constants
                     0091 ; in delay loops, the delay loop should be broken into 2 or more loops.
                     0092 ; For example, 357 = 255+102. So 2 delay loops, one with 255 and
                     0093 ; the other with 102 may be used.
                     0094 ;************************************************************************
                     0095 ;       Set Delay Constants for 9600 Baud @ CLKIN = 8 Mhz
                     0096 ;
0000                 0097 K0      EQU     .0
0027                 0098 K1      EQU     .39
001B                 0099 K2      EQU     .27
0015                 0100 K3      EQU     .21
0017                 0101 K4      EQU     .23
0018                 0102 K5      EQU     .24
0000                 0103 K6      EQU     .0
002D                 0104 K7      EQU     .45
                     0105 ;
                     0106 ;************************************************************************
                     0107
                     0055
                     0056 ;
                     0057         ORG     0
                     0058 ;************************************************************************
                     0059 ;
0000 0C01            0060 Delay   movlw   K0+1
0001 002C            0061         movwf   DlyCnt          ; Total Delay = 3K+6
0002 02EC            0062 redo    decfsz  DlyCnt,Same
0003 0A02            0063         goto    redo
0004 0800            0064         retlw   0
                     0065 ;
0005 002C            0066 Delay1  movwf   DlyCnt
0006 02EC            0067 redo_1  decfsz  DlyCnt,Same     ;
0007 0A06            0068         goto    redo_1
0008 0A8D            0069         goto    User
                     0070 ;
0009 002C            0071 Delay2  movwf   DlyCnt
000A 02EC            0072 redo_2  decfsz  DlyCnt,Same     ; Delay =   = 260 Cycles
000B 0A0A            0073         goto    redo_2
000C 0A67            0074         goto    User_1
                     0075 ;
000D 0625            0076 R_strt  btfsc   Port_A,DR       ; check for a Start Bit
000E 0A17            0077         goto    ShellY          ; delay for 104/2 uS
000F 042D            0078         bcf     FlagRX,R_done   ; Reset Receive done flag
0010 054D            0079         bsf     FlagRX,R_flag   ; Set flag for Reception in Progress
0011 078D            0080         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,CARRY
                            IF      R_MODE
001C 0328                   rrf     RcvReg,Same     ; to set if MSB first or LSB first
                            ELSE
                            rlf     RcvReg,Same
                            ENDIF
001D 0625                   btfsc   Port_A,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     Port_A,DX       ; Send Start Bit
                            IF      X_Nbit
0026 0C08                   movlw   8
                            ELSE
                            movlw   7
                            ENDIF
0027 002A                   movwf   Xcount
                            IF      X_MODE
                    0129          ELSE
                    0130            IF    X_Nbit
                    0131            ELSE
                    0132            rlf   XmtReg,Same
                    0133            ENDIF
                    0134          ENDIF
0028 0A50           0135          goto    X_SB
                    0136 ;
0029 0403           0137 X_next   bcf     STATUS,CARRY
                    0138          IF      X_MODE
002A 0329           0139          rrf     XmtReg,Same      ; Conditional Assembly
                    0140          ELSE                     ; to set if MSB first or LSB first
                    0141          rlf     XmtReg,Same
                    0142          ENDIF
002B 0603           0143          btfsc   STATUS,CARRY
002C 0505           0144          bsf     Port_A,DX
002D 0703           0145          btfss   STATUS,CARRY
002E 0405           0146          bcf     Port_A,DX
002F 00EA           0147          decf    Xcount,Same
0030 0A52           0148          goto    X_Data
                    0149 ;
0031 040D           0150 X_SB_1   bcf     FlagRX,X_flag    ; Xmt flag = 0 — transmission over
0032 0C09           0151          movlw   9
0033 002A           0152          movwf   Xcount
0034 0505           0153          bsf     Port_A,DX        ; Send Stop Bit
0035 0A60           0154          goto    X_Stop
                    0155 ;
0036 0505           0156 X_SB_2   bsf     Port_A,DX
0037 04CD           0157          bcf     FlagRX,S_bit
0038 0A60           0158          goto    X_Stop
                    0159 ;
                    0160 ;   End of Transmission
                    0161 ;
0039 076D           0162 R0_X0    btfss   FlagRX,S_flag
003A 0A8D           0163          goto    User
003B 046D           0164          bcf     FlagRX,S_flag
003C 0900           0165          call    Delay
003D 0C2E           0166          movlw   K7+1
003E 0A05           0167          goto    Delay1
                    0168 ;
                    0169 R1_X0
003F 0900           0170          call    Delay
0040 0C28           0171          movlw   K1+1             ; delay for 1st bit is 104+104/4
0041 002C           0172          movwf   DlyCnt
                    0173          IF      R_Nbit
0042 0C08           0174          movlw   8                ; 8 Data bits
                    0175          ELSE
                    0176          movlw   7                ; 7 data bits
```

```
                          0177          ENDIF
 0043 018B                0178          xorwf   Rcount,W
 0044 0643                0179          btfsc   STATUS,Z_bit
 0045 0A06                0180          goto    redo_1
 0046 0C1C                0181          movlw   K2+1
 0047 0A05                0182          goto    Delay1
                          0183 ;
                          0184 R1_X1                          ; same as R0_X1
 0048 0C09                0185 R0_X1    movlw   9
 0049 008A                0186          subwf   Xcount,W
 004A 0643                0187          btfsc   STATUS,Z_bit
 004B 0A25                0188          goto    X_strt
 004C 022A                0189          movf    Xcount,Same     ; to check if All data bits Xmted
 004D 0743                0190          btfss   STATUS,Z_bit
 004E 0A29                0191          goto    X_next
                          0192          IF      SB2
                          0193            btfsc FlagRX,S_bit
                          0194            goto  X_SB_2
                          0195            bsf   FlagRX,S_bit
                          0196            goto  X_SB_1
                          0197          ELSE
 004F 0A31                0198            goto    X_SB_1
                          0199          ENDIF
                          0200 ;
                          0201 ;
 0050 0A51                0202 X_SB     goto    cycle4
 0051 0A52                0203 cycle4   goto    X_Data
                          0204 ;
 0052 06AD                0205 X_Data   btfsc   FlagRX,A_flag
 0053 0A59                0206          goto    SbDly
 0054 068D                0207          btfsc   FlagRX,BitXsb
 0055 0A5D                0208          goto    ABC
 0056 0900                0209          call    Delay
 0057 0C16                0210          movlw   K3+1
 0058 0A09                0211          goto    Delay2
                          0212 ;
 0059 04AD                0213 SbDly    bcf     FlagRX,A_flag
 005A 0900                0214          call    Delay
 005B 0C18                0215          movlw   K4+1
 005C 0A09                0216          goto    Delay2
                          0217 ;
 005D 048D                0218 ABC      bcf     FlagRX,BitXsb
 005E 0900                0219          call    Delay
 005F 0A67                0220          goto    User_1
                          0221 ;
                          0222 X_Stop
 0060 06AD                0223          btfsc   FlagRX,A_flag
 0061 0A59                0224          goto    SbDly
```

```
0062 068D          0225         btfsc   FlagRX,BitXsb
0063 0A5D          0226         goto    ABC
0064 0900          0227         call    Delay
0065 0C19          0228         movlw   K5+1
0066 0A09          0229         goto    Delay2
                   0230 ;
0067 064D          0231 User_1  btfsc   FlagRX,R_flag
0068 0A77          0232         goto    Sync_1          ; Reception already in progress
0069 066D          0233         btfsc   FlagRX,S_flag
006A 0A74          0234         goto    Sync_3
006B 0625          0235         btfsc   Port_A,DR       ; check for a Start Bit
006C 0A77          0236         goto    Sync_2          ; No Start Bit - goto User routine
006D 042D          0237         bcf     FlagRX,R_done   ; Reset Receive done flag
006E 044D          0238         bcf     FlagRX,R_flag
006F 058D          0239         bsf     FlagRX,BitXsb   ; Set flag for Reception in Progress
0070 0068          0240         clrf    RcvReg          ; Clear all bits of RcvReg
                   0241         IF      R_Nbit
0071 0C08          0242         movlw   8               ; 8 Data bits
                   0243         ELSE
                   0244         movlw   7               ; 7 data bits
                   0245         ENDIF
0072 002B          0246         movwf   Rcount
0073 0A8D          0247         goto    User
                   0248 ;
0074 046D          0249 Sync_3  bcf     FlagRX,S_flag
0075 0C01          0250         movlw   K6+1
0076 0A05          0251         goto    Delay1
                   0252 ;
                   0253 Sync_1
0077 0A8D          0254 Sync_2  goto    User
                   0255 ;
                   0256 ;*************************************************************************
                   0257 ;
0078 064D          0258 Shell   btfsc   FlagRX,R_flag
0079 0A7D          0259         goto    Chek_X
007A 060D          0260         btfsc   FlagRX,X_flag
007B 0A48          0261         goto    R0_X1
007C 0A39          0262         goto    R0_X0            ; Case for R0_X0
007D 060D          0263 Chek_X  btfsc   FlagRX,X_flag
007E 0A48          0264         goto    R1_X1
007F 0A3F          0265         goto    R1_X0
                   0266 ;
                   0267 ;
                   0268 ;*************************************************************************
                   0269 ;       Operating System
                   0270 ;  The User routine after time = B/2, should branch Here
                   0271 ;
0080 074D          0272 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    Port_A           ; Set Port_A.0 as output ( DX )
                 ;                                   & Port_A.1 is input ( DR )
0085 0505                 bsf     Port_A,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 RTCC 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. RTCC timer is used here to count
                 ;  upto 86 cycles ( From 128-86 To 0 ) by examining Bit 7 of RTCC.
                 ;
0030             K_user   equu    .128+.6-.86
                 ;
008D 0C30        User     movlw   K_user
008E 0021                 movwf   RTCC
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
                 ;
                 ErrChk
0097 0C5A                 movlw   "Z"
0098 0188                 xorwf   RcvReg,W
0099 0643                 btfsc   STATUS,Z_bit
009A 0A91                 goto    SetXmt
009B 0A9B        error    goto    error            ; Received word is not "Z"
```

```
                     0321 ;
009C 07E1            0322 Op      btfss   RTCC,MSB        ; Test for RTCC bit 7
009D 0A9C            0323         goto    Op              ; If Set, Then RTCC has incremented
009E 0A80            0324 Oflow   goto    Op_Sys          ; to 128.
                     0325 ;
                     0326 ; ***********************************************************
                     0327 ;
                     0328         ORG     PIC54
01FF 0A83            0329         goto    main
                     0330
                     0331         END
                     0332
                     0333

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

0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX

0080 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXX- ———————— ————————
00C0 : ———————— ———————— ———————— ————————

0180 : ———————— ———————— ———————— ————————
01C0 : ———————— ———————— ———————— ———————X

All other memory blocks unused.


Errors   :    0
Warnings :    0
```

NOTES:

WORLDWIDE SALES & SERVICE

AMERICAS

Corporate Office
Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 602 786-7200 Fax: 602 786-7277
Technical Support: 602 786-7627
Web: http://www.mchip.com/microhip

Atlanta
Microchip Technology Inc.
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770 640-0034 Fax: 770 640-0307

Boston
Microchip Technology Inc.
5 Mount Royal Avenue
Marlborough, MA 01752
Tel: 508 480-9990 Fax: 508 480-8575

Chicago
Microchip Technology Inc.
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 708 285-0071 Fax: 708 285-0075

Dallas
Microchip Technology Inc.
14651 Dallas Parkway, Suite 816
Dallas, TX 75240-8809
Tel: 214 991-7177 Fax: 214 991-8588

Dayton
Microchip Technology Inc.
35 Rockridge Road
Englewood, OH 45322
Tel: 513 832-2543 Fax: 513 832-2841

Los Angeles
Microchip Technology Inc.
18201 Von Karman, Suite 455
Irvine, CA 92715
Tel: 714 263-1888 Fax: 714 263-1338

New York
Microchip Technology Inc.
150 Motor Parkway, Suite 416
Hauppauge, NY 11788
Tel: 516 273-5305 Fax: 516 273-5335

AMERICAS (continued)

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

ASIA/PACIFIC

Hong Kong
Microchip Technology
Unit No. 3002-3004, Tower 1
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T. Hong Kong
Tel: 852 2 401 1200 Fax: 852 2 401 3431

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

Singapore
Microchip Technology
200 Middle Road
#10-03 Prime Centre
Singapore 188980
Tel: 65 334 8870 Fax: 65 334 8850

Taiwan
Microchip Technology
10F-1C 207
Tung Hua North Road
Taipei, Taiwan, ROC
Tel: 886 2 717 7175 Fax: 886 2 545 0139

EUROPE

United Kingdom
Arizona Microchip Technology Ltd.
Unit 6, The Courtyard
Meadow Bank, Furlong Road
Bourne End, Buckinghamshire SL8 5AJ
Tel: 44 0 1628 851077 Fax: 44 0 1628 850259

France
Arizona Microchip Technology SARL
2 Rue du Buisson aux Fraises
91300 Massy - France
Tel: 33 1 69 53 63 20 Fax: 33 1 69 30 90 79

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

Italy
Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Pegaso Ingresso No. 2
Via Paracelso 23, 20041
Agrate Brianza (MI) Italy
Tel: 39 039 689 9939 Fax: 39 039 689 9883

JAPAN

Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shin Yokohama
Kohoku-Ku, Yokohama
Kanagawa 222 Japan
Tel: 81 45 471 6166 Fax: 81 45 471 6122

9/22/95