AN541

Using a PIC16C5X as a Smart I^2C™ Peripheral

Author: Don Lekei
NII Norsat International Inc.

INTRODUCTION

PIC16C5X microcontrollers from Microchip are ideally suited for use as smart peripheral devices under the control of the main processors in systems, due to their low cost and high speed. They are capable of performing tasks which would simply overload a conventional microprocessor, or require considerable logic circuitry, at a cost competitive with lower mid-range PLDs. To minimize the engineering overhead of adding multiple controllers to a product, it is convenient for the auxiliary controllers to emulate standard I/O peripherals.

A common interface found in existing products is the I^2C bus. This efficient, two-wire, bi-directional interface allows the designer to connect multiple devices together, with the microprocessor able to send data to and receive data from any device on the bus. This interface is found on a variety of components, such as PLLs, DACs, video controllers, and EEPROMs. If a product already contains one or more I^2C devices, it is simple to add a PIC16C5X emulating a compatible component.

This application note describes the implementation of a standard slave device with multiple, bi-directional registers. A subset of the full I^2C specification is supported, which can be controlled by the same software which would talk to a Microchip 24LCXX series EEPROM.

THE I^2C BUS

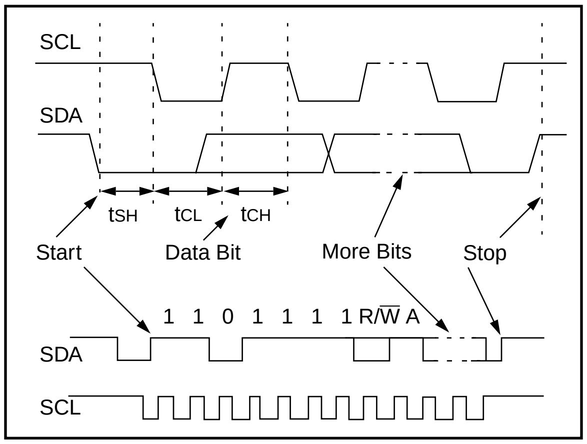
The I^2C bus is a master-slave two-wire interface, consisting of a clock line (SCL) and a data line (SDA). Bi-directional communication (and in a full, multi-master system, collision detection, and clock synchronization) is facilitated through the use of a “wire-and” (i.e., active-low, passive-high) connection.

The standard mode I^2C bus supports SCL clock frequencies up to 100 kHz. The fast-mode I^2C bus supports clock rates up to 400 kHz. This application note will support the 100 kHz (standard-mode) clock rate.

Each device has a unique seven bit address, which the master uses to access each individual slave device.

During normal communication, the SDA line is only permitted to change while the SCL line is low, thus providing two violation conditions (Figure 1) which are used to signal a start condition (SDA drops while SCL is high) and a stop condition (SDA rises while SCL is high), which frame a message.

FIGURE 1: I^2C TIMING



Each byte of a transfer is 9-bits long (see timing chart in the program listing). The talker sends 8 data bits followed by a '1' bit. The listener acknowledges the receipt of the byte and gives permission to send the next byte by inserting a '0' bit over the trailing '1'. The listener may indicate "not ready for data" by leaving the acknowledge bit as a '1'.

The clock is generated by the master only. The slave device must respond to the master within the timing specifications of the I^2C definition otherwise the master would be required to operate in slow mode, which most software implementations of I^2C masters do not actually support. The specified (standard-mode) t_{CL} is 4.7 µs, and t_{CH} is only 4 µs, so it would be extremely difficult to achieve the timing of a hardware slave device with a conventional microcontroller.

MESSAGE FORMAT

A message is always initiated by the master, and begins with a start condition, followed by a slave address (7 MSbs) and direction bit (LSb = '1' for READ, '0' for WRITE). The addressed slave must acknowledge this byte if it is ready to communicate any data. If the slave fails to respond, the master should initiate a stop condition and retry.

If the direction bit is '0' the next byte is considered the sub-address (this is an extension to I^2C used by most multi-register devices). The sub-address selects which "register" or “function” subsequent read or write operations will affect. Any additional bytes will be received and stored in consecutive locations until a stop is sent. If the slave is unable to process more data, it could terminate transfer by not acknowledging the last byte.

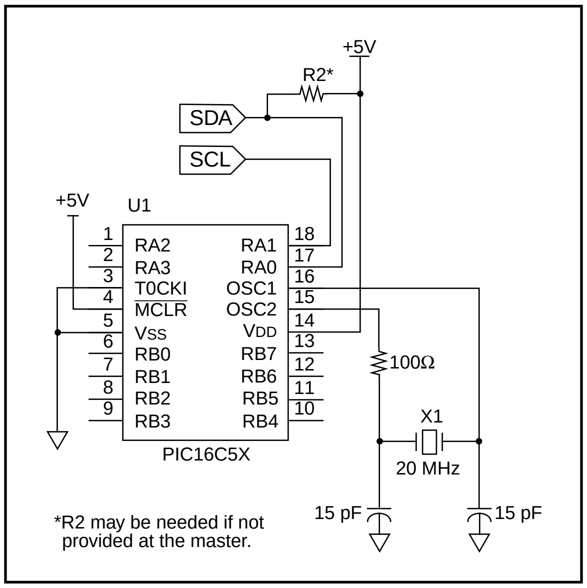
If the direction bit is '1', the slave will transfer successive bytes to the master (the master holds the line at '1'), while the master acknowledges each byte with a '0' in the ninth bit. The master can terminate the transfer by not acknowledging the last byte, while the slave can stop the transfer by generating a stop condition.

The start address of a read operation is set by sending a write request with a sub-address only (no data bytes). For a detailed set of timing diagrams and different communication modes, consult any of the Microchip 24LCXX EEPROM specifications. This program communicates using the same formats.

IMPLEMENTATION

The chip will respond to slave address "DEVICE_ADDRESS", which by default is $D6_{16}$ ($D7_{16}$ for read). This address was chosen because it is the fourth optional address of a Philips PCF8573 clock /calender or a TDA8443 tipple video switch (unlikely that a product would contain four of those).

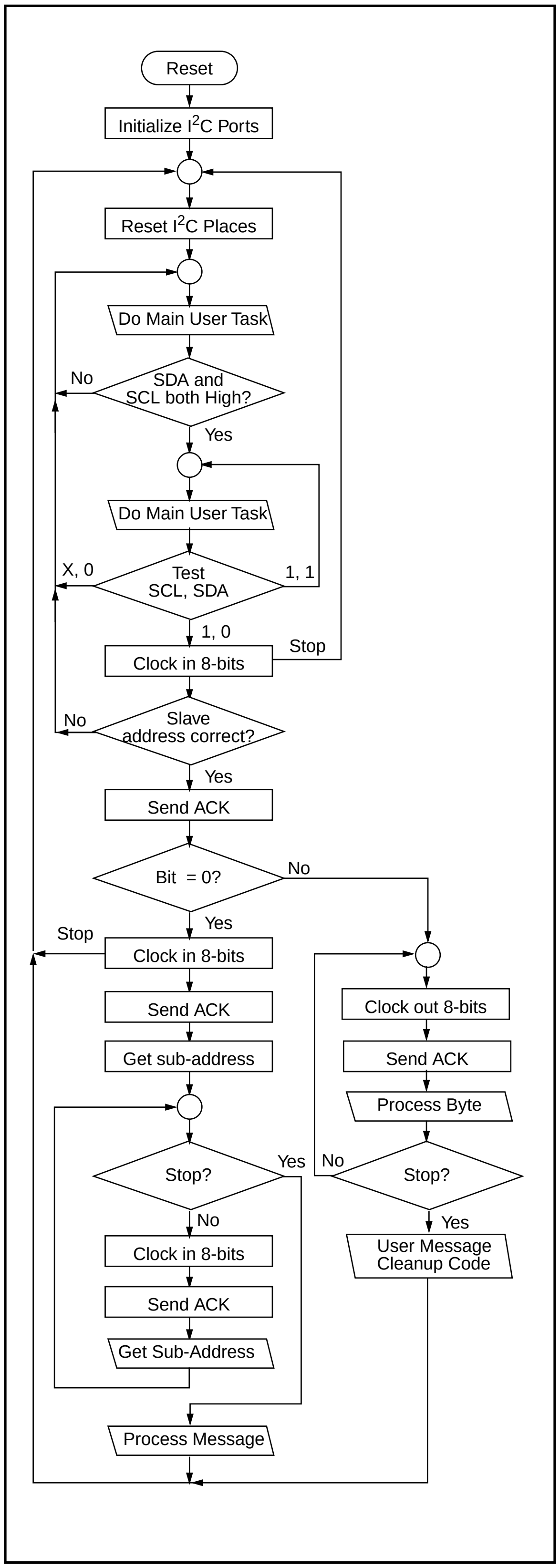
FIGURE 2: SCHEMATIC OF I^2C CONNECTIONS



The connections to the device are shown in Figure 2. The use of RA0 for data input is required. Data is shifted directly out of the port. The code could be modified to make it port independent, but the loss of efficiency may hinder some real-time applications.

This application emulates an I^2C device with 8 registers, accessed as sub-addresses 1 through 8 (module 7), plus a data channel (0). The example code returns an ID string when the data channel is accessed. When bytes are written to sub-addresses other than 0, they are stored in I2CR0-I2CR7 (I2CR0 gets data written to sub-address 8).

FIGURE 3: I^2C DEVICE FLOWCHART



When the initial subaddress is 0, the flag B_ID is set. This is used to indicate access to a special channel. In this case, the data channel is used to return an ID message, or output data to PORTB, however the natural extension would be to use this as a data I/O channel.

To make the basic device routines easily adaptable to a variety of uses, macros are used to implement the application specific code. This allows the developer the option of using subroutine calls, or in-line code to avoid the 4 clock cycle overhead and use of the precious stack.

Macro	User Code Function
USER_MAIN	Code to execute in the main loop while not in a message. If this code takes too long, t_{SH} of 4 µs will be violated (Figure 1). The slave will simply miss the address, not acknowledge, and the master will retry.
USER_Q	This would be quick user code to implement real-time processes. In most applications, this macro would be empty. If used, this routine should be kept under 4 µs if possible.
USER_MSG	This would be user code to process a message. It is inserted after a message is successfully received.
USER_RECV	This would be user code to process a received byte. It allows the user to add extra code to implement special purpose sub-addresses such as FIFOs.
USER_XMIT	This would be user code to prepare an output byte. In the default routine, it traps sub-address 0 and calls the ID string function.

References:

I^{2}C Bus Specification, Philips Corporation, December 1988.

The I^{2}C bus and how to use it (including specification), Signetics/Philips Semiconductors, January 1992.

Fenger, Carl, "The Inter-Integrated Circuit (I^{2}C) Serial Bus: Theory and Practical Consideration", Application Note 168, Philips Components, December 1988.

"24C16 16K CMOS Serial Electrically Erasable PROM", Microchip Data Book (1992).

About the Author:

Don Lekei has been designing microprocessor based products over 14 years. He has developed many software and hardware products for a wide variety of applications. Mr. Lekei is Manager of Advanced Technologies at NII Norsat International Inc. at their Canadian headquarters in Surrey, British Columbia. Norsat designs and manufactures products to receive broadcast communications from satellites, terrestrial broadcasting systems and optical fibre. Norsat develops technologies and products for satellite entertainment television, broadcast music and data networks.

APPENDIX A: I2C.ASM

```
MPASM 01.40 Released              I2C.ASM   1-16-1997  14:51:28         PAGE  1


LOC  OBJECT CODE  LINE SOURCE TEXT
  VALUE

               00001         LIST    P=16C54, C=80, N=0, R=DEC
               00002 ;
               00003 ;*****************************************************************
               00004 ;
               00005 ;       Program:          I2C.ASM
               00006 ;       Revision Date:
               00007 ;                         1-13-97      Compatibility with MPASMWIN 1.40
               00008 ;
               00009 ;***************************************************************
               00010 ;
  00000676     00011 CPU     EQU     1654
  00000000     00012 SIM     EQU     0               ;Change timing constants for simulator
               00013
               00014         IF      (CPU==1654) || (CPU==1655)
  000001FF     00015 _RESVEC EQU     01FFH           ;16c54 start address
               00016         ENDIF
               00017
               00018         IF      CPU==1656
               00019 _RESVEC EQU     03FFH           ;16C56 start address
               00020         ENDIF
               00021
               00022         IF      CPU==1657
               00023 _RESVEC EQU     07FFH           ;16C57 start address
               00024         ENDIF
               00025
               00026 ;*** Reset Vector ******************************************
               00027
01FF           00028         ORG     _RESVEC         ;
01FF           00029 RESVEC                          ;
01FF 0A0B      00030         GOTO    INIT            ;
               00031
               00032 ;************************************************************
               00033
               00034 ;************************************************************
               00035 ;* Macros to set/clear/branch/skip on bits
               00036 ;* These macros define and use synthetic "bit labels"
               00037 ;* Bit labels contain the address and bit of a location
               00038 ;*
               00039 ;*******************************************************************
               00040 ;*      Usage                    Description
               00041 ;*      -----------------------  ---------------------
               00042 ;*      BIT     label,bit,file  ;Define a bit label
               00043 ;*      SEB     label           ;set bit using bit label
               00044 ;*      CLB     label           ;clear bit using bit label
               00045 ;*      SKBS    label           ;SKIP on bit set
               00046 ;*      SKBC    label           ;SKIP on bit clear
               00047 ;*      BBS     label,address   ;BRANCH on bit set
               00048 ;*      BBC     label,address   ;BRANCH on bit clear
               00049 ;*      CBS     label,address   ;CALL on bit set
               00050 ;*      CBC     label,address   ;CALL on bit clear
               00051 ;*
               00052 ;*******************************************************************
               00053
               00054 BIT     MACRO   label,bit,file  ;Define a bit label
               00055 label           EQU    file<<8|bit         ;(macro)
               00056        ENDM                               ;
               00057
               00058 SEB     MACRO   label                     ;Set bit
               00059                 BSF    label>>8,label &7   ;(macro)
               00060        ENDM                               ;
               00061
               00062 CLB     MACRO   label                     ;Clear bit
               00063                 BCF    label>>8,label &7   ;(macro)
               00064        ENDM                               ;
               00065
               00066 SKBS    MACRO   label                     ;Skip on bit set
               00067                 BTFSS  label>>8,label &7   ;(macro)
               00068        ENDM
               00069
               00070 SKBC    MACRO   label                     ;Skip on bit clear
               00071                 BTFSC  label>>8,label &7   ;(macro)
               00072        ENDM
               00073
               00074 BBS     MACRO   label,address             ;Branch on bit set
               00075                 BTFSC  label>>8,label &7   ;(macro)
               00076                 GOTO   address             ;(macro)
               00077        ENDM                               ;
               00078
               00079 BBC     MACRO   label,address             ;Branch on bit clear
               00080                 BTFSS  label>>8,label &7   ;(macro)
               00081                 GOTO   address             ;(macro)
               00082        ENDM
               00083
               00084 CBS     MACRO   label,address             ;Call on bit set
               00085                 CALL   label>>8,label  &7  ;(macro)
               00086        ENDM                    ;
               00087
               00088 CBC     MACRO   label,address             ;Call on bit clear
               00089                 CALL   label>>8,label   &7 ;(macro)
               00090        ENDM
               00091
               00092
               00093 ;For Assembler portability
               00094
  00000000     00095 W       EQU     0                         ;For file,W
  00000000     00096 w       EQU     0                         ;For file,W
  00000001     00097 F       EQU     1                         ;For file,F
  00000001     00098 f       EQU     1                         ;For file,F
               00099
               00100 ;*******************************************************************
               00101 ;* REGISTER DECLARATIONS
               00102 ;*******************************************************************
               00103
               00104
0000           00105         ORG     0                  ;ORG for register declarations
               00106
0000           00107 ind     RES     1                  ;0=pseudo-reg 0 for in direct (FSR)
0001           00108 TMR0    RES     1                  ;1=real time counter
0002           00109 PC      RES     1                  ;2=PC
0003           00110 STATUS  RES     1                  ;3=status reg
               00111
               00112 ;* Status reg bits
               00113
               00114         BIT     B_C,0,STATUS       ;Carry
  00000300         M B_C             EQU    STATUS<<8|0  ;(macro)
               00115         BIT     B_DC,1,STATUS      ;Half carry
  00000301         M B_DC             EQU    STATUS<<8|1 ;(macro)
               00116         BIT     B_Z,2,STATUS       ;Zero
  00000302         M B_Z             EQU    STATUS<<8|2  ;(macro)
               00117         BIT     B_PD,3,STATUS      ;Power down
  00000303        M B_PD            EQU    STATUS<<8|3  ;(macro)
              00118         BIT     B_TO,4,STATUS       ;Timeout
  00000304        M B_TO            EQU    STATUS<<8|4  ;(macro)
              00119         BIT     B_PA0,5,STATUS      ;Page select (56/57 only)
  00000305        M B_PA0           EQU    STATUS<<8|5  ;(macro)
              00120         BIT     B_PA1,6,STATUS      ;Page select (56/57 only)
  00000306        M B_PA1           EQU    STATUS<      ;(macro)
              00121         BIT     B_PA2,7,STATUS      ;GP flag
  00000307        M B_PA2           EQU    STATUS<<8|7  ;(macro)
              00122
0004          00123 FSR     RES     1                   ;4=file select reg 0-4 =indirect address
0005          00124 PORTA   RES     1                   ;5=port A I/O register (4 bits)
0006          00125 PORTB   RES     1                   ;6=port B I/O register
              00126
              00127         IF      (CPU==1655)||(CPU==1657)
              00128 PORTC   RES     1                   ;7=I/O port C on 16C54/56 only
              00129         ENDIF
              00130
              00131 ;registers used by this code
              00132
0007          00133 I2CFLG  RES     1                    ;I2C flag reg
              00134 ;--i2c flags----------------------------------------------
              00135         BIT     B_RD,0,I2CFLG       ;Flag: 1=read
  00000700        M B_RD            EQU    I2CFLG<<8|0  ;(macro)
              00136         BIT     B_UA,1,I2CFLG       ;Flag: 0=reading unit address
  00000701        M B_UA            EQU    I2CFLG<<8|1  ;(macro)
              00137         BIT     B_SA,2,I2CFLG       ;Flag: 1=reading subabbress
  00000702        M B_SA            EQU    I2CFLG<<8|2  ;(macro)
              00138         BIT     B_ID,3,I2CFLG       ;Flag: 1=reading id
  00000703        M B_ID            EQU    I2CFLG<<8|3  ;(macro)
              00139 ;---------------------------------------------------------
0008          00140 I2CREG  RES     1                   ;I2C I/O register
0009          00141 I2CSUBA RES     1                   ;Subaddress
000A          00142 I2CBITS RES     1                   ;I2C xmit bit counter
              00143
              00144
              00145 ;**********************************************************************
              00146 ;* 8 Pseudo registers accessed by sub-addresses 1-8
              00147 ;* (address 0 accesses the ID string)
              00148 ;* these are read-write registers
              00149 ;**********************************************************************
              00150
              00151
  0000000B    00152 I2CR0   EQU     $                   ;Sub-address 8
000B          00153         RES     1                   ;8 pseudo registers
              00154
  0000000C    00155 I2CR1   EQU     $                   ;Sub-address 1
000C          00156         RES     1
              00157
  0000000D    00158 I2CR2   EQU     $                   ;Sub-address 2
000D          00159         RES     1
              00160
  0000000E    00161 I2CR3   EQU     $                   ;Sub-address 3
000E          00162         RES     1
              00163
  0000000F    00164 I2CR4   EQU     $                   ;Sub-address 4
000F          00165         RES     1
              00166
  00000010    00167 I2CR5   EQU     $                   ;Sub-address 5
0010          00168         RES     1
              00169
  00000011    00170 I2CR6   EQU     $                   ;Sub-address 6
0011          00171         RES     1
              00172
  00000012    00173 I2CR7   EQU     $                   ;Sub-address 7
0012          00174         RES     1
```

```

                ;Constants used by program

000000D6        DEVICE_ADDRESS EQU  0D6H  ;I2C device address (device_address+1 = read)

                ;***************************************************************
                ;** PORTA DEFINITIONS
                ;** I2C interface uses PORTA
                ;** note SDA goes to A0 for code efficiency
                ;**
                ;****************************************************************


000000F7        TAREAD  EQU     B’11110111’     ;TRISA register for SDA read
000000F6        TAWRITE EQU     B’11110110’     ;TRISA register for SDA write
000000F7        TAINIT  EQU     TAREAD          ;Initial TRISA value

                        BIT     B_SDA,0,PORTA   ;I2C SDA (data) This must be bit 0!
00000500      M B_SDA           EQU PORTA<<8|0  ;(macro)
                        BIT     B_SCL,1,PORTA   ;I2C SCL (clock)
00000501      M B_SCL           EQU PORTA<<8|1  ;(macro)
                ;spare          B_???,2,PORTA   ;not used
                ;spare          B_???,3,PORTA   ;not used

                ;***************************************************************
                ;**
                ;** Port B definition (Parallel out)
                ;**
                ;*****************************************************************
00000000        TBINIT  EQU     B’00000000’     ;Port B tris (all output)
000000FF        PBINIT  EQU     B’11111111’     ;Port B init


                ;***********************************************************************
                ;* Macros to contain user POLL loop code.
                ;* These are implemented as macros to allow ease of modification,
                ;* especially in real-time applications. The functions could be coded
                ;* as in-line code or as subroutines depending on ROM/time tradeoffs.
                ;*
                ;* USER_MAIN:   Decision or code to perform at idle time
                ;*
                ;* USER_Q:      ‘Quick’ code for use during transfer - max 8 Ês for
                ;*              full I2C Spec. More than 4s may result in I2C
                ;*              retries (at full spec speed.
                ;*
                ;* USER_MSG:    Code to execute at receipt of I2C command.
                ;*
                ;***********************************************************************

                USER_MAIN       MACRO
                ;*** This would be user code for idle loop
                        ENDM

                USER_Q          MACRO
                ;*** This would be quick user code
                        ENDM

                USER_MSG        MACRO
                ;*** This would be user code to process a message
                        ENDM

                USER_RECV       MACRO
                ;*** This would be user code to process a received byte
                ;*** example code sends sub-address 0 to port b
                   BBC     B_ID,_NXI_notid ;Channel 0! Bit set if INITIAL address was 0
                   MOVFW   I2CREG          ;get received byte
```

```
                  MOVWF   PORTB           ;and write it on portb
                       GOTO    IN_CONT
               _NXI_notid
                  ENDM

               USER_XMIT       MACRO
               ;*** This would be user code to prepare an output byte
               ;*** example code sends id string to output
                  BBC     B_ID,_NXO_notid ;Channel 0! Bit set if INITIAL address was 0
                  CALL    GETID           ;get next byte from ID channel
                  GOTO    OUT_CONT        ;and send it
               _NXO_notid
                  ENDM

               ;*******************************************************************
               ; START OF CODE
               ;*******************************************************************
0000                   ORG     0
               ;*******************************************************************
               ;* Device ID Table (must be at start)
               ;* TABLE FOR UNIT ID returns next char in W
               ;*******************************************************************
0000           GETID
0000 0209          MOVFW   I2CSUBA         ;W=I2CSUBA
0001 0E07          ANDLW   07H             ;Limit to 8 locations
0002 01E2          ADDWF   PC,F

               ;**********************************************************************
               ;* Device ID text: read starting at sub-address 0
               ;**********************************************************************

0003 0850          RETLW   'P'
0004 0849          RETLW   'I'
0005 0843          RETLW   'C'
0006 0849          RETLW   'I'
0007 0832          RETLW   '2'
0008 0843          RETLW   'C'
0009 0800          RETLW   0
000A 0800          RETLW   0


               ;**********************************************************************
               ;* I2C Device routines
               ;*
               ;* Enable must be HIGH, else state goes to 0
               ;* write is to me, read is from me.
               ;*
               ;*      <============== first byte / subsequent writes =====> <end>
               ;*
               ;* SDA   --|X-----X-----X-----X-----X-----X-----X-----X---X------|  |--
               ;*      |---X-----X-----X-----X-----X-----X-----X-----X---X******|--|
               ;* (bit)   s    7     6     5     4     3     2     1     0   ackout
               ;* SCL  -----|  |--|  |--|  |--|  |--|  |--|  |--|  |--|  |--|
               ;*           |--|  |--|  |--|  |--|  |--|  |--|--|  |--|  |--|
               ;*
               ;* STATE: 0 1 2  3  2  3  2  3  2  3  2  3  2  3  2  3  2  4  5  6  2  3 0
               ;*
               ;*            <============== subsequant reads ==================>
               ;*
               ;* SDA     X-----X-----X-----X-----X-----X-----X-----X----X------X-----
               ;*       --X-----X-----X-----X-----X-----X-----X-----X----X******X-----
               ;* (bit)ack   7     6     5     4     3     2     1     0   ackin
               ;* SCL  --|  |--|  |--|  |--|  |--|  |--|  |--|  |--|  |--|  |--|  |--
               ;*         |--|  |--|  |--|  |--|  |--|  |--|  |--|  |--|  |--|  |--|
               ;*
               ;* STATE:  7  8  7  8  7  8  7  8  7  8  7  8  7  8  7  8  9  A  7  8
```

```
             ;*
             ;*            <============== Final READ ========================>
             ;*
             ;* SDA     X-----X-----X-----X-----X-----X-----X-----X----X-|**|-------
             ;*       --X-----X-----X-----X-----X-----X-----X-----X----X*|--|
             ;* (bit)ack   7     6     5     4     3     2     1   0   ackin
             ;* SCL  --|  |--|  |--|  |--|  |--|  |--|  |--|  |--|  |--|  |---------
             ;*        |--|  |--|  |--|  |--|  |--|  |--|  |--|  |--|  |--|
             ;*
             ;* STATE:  7  8  7  8  7  8  7  8  7  8  7  8  7  8  7  8  9  A  0  0
             ;*
             ;* STATE B is an ignore bit state for non-addressed bits
             ;* STATE C indicates last sample had ENA low on rising
             ;*  edge of ENA, DATA LOW = low voltage, DATA&CLOCK low = RESET
             ;*****************************************************************

             ;I2C interface uses PORTA
             ;note SDA must be on PORTA,0 for code efficiency

             ;*****************************************************************
             ;** INIT
             ;** Hardware reset entry point
             ;**
             ;*****************************************************************
000B         INIT    ;Power-on entry

             ;*****************************************************************
             ;** RESET
             ;** software reset entry point
             ;**
             ;*****************************************************************
000B         RESET   ;Soft reset

000B 0CF7            MOVLW   TAINIT          ;Init ports
000C 0005            TRIS    PORTA
000D 0C00            MOVLW   TBINIT
000E 0006            TRIS    PORTB
000F 0CFF            MOVLW   PBINIT
0010 0026            MOVWF   PORTB

             ;*****************************************************************
             ; Main wait loop while idle. POLL loop should be called here
             ;
             ;*****************************************************************

0011         I2CWAIT
0011 0004            CLRWDT                  ;Clear watchdog timer
                     CLB    B_UA             ;Init state flags
0012 0427  M         BCF    B_UA>>8,B_UA&7   ;(macro)
                     CLB    B_SA             ;Init state flags
0013 0447  M         BCF    B_SA>>8,B_SA&7   ;(macro)
                     CLB    B_RD             ;Init state flags
0014 0407  M         BCF    B_RD>>8,B_RD&7   ;(macro)

0015         loop1
0015 0004            CLRWDT                  ;Clear watchdog timer

                     USER_MAIN       ;Call user code while in idle state
           M ;*** This would be user code for idle loop

                     SKBC   B_SDA            ;Wait for SDA&SCL=H
0016 0605  M         BTFSC  B_SDA>>8,B_SDA&7 ;(macro)
0017         loop2
                     SKBS   B_SCL              ;
0017 0725  M         BTFSS  B_SCL>>8,B_SCL&7   ;(macro)
0018 0A15            GOTO   loop1              ; No longer valid to wait for start!
```

```
               00365
0019 0004      00366        CLRWDT                    ;Clear watchdog timer
               00367
               00368        USER_MAIN       ;Call user code while in idle state
                   M ;*** This would be user code for idle loop
               00369
               00370 ;** wait for start **
               00371        SKBC   B_SCL              ;Clock has dropped
001A 0625          M        BTFSC  B_SCL>>8,B_SCL&7   ;(macro)
               00372        SKBC   B_SDA              ;Data dropped... Start!
001B 0605          M        BTFSC  B_SDA>>8,B_SDA&7   ;(macro)
001C 0A17      00373        GOTO   loop2
               00374
               00375 ;** START RECEIVED! --- wait for first bit!
001D           00376 loop3
               00377        BBS    B_SDA,I2CWAIT      ;Data raised before clock dropped -- abort
001D 0605          M        BTFSC  B_SDA>>8,B_SDA&7   ;(macro)
001E 0A11          M        GOTO   I2CWAIT            ;(macro)
               00378        BBS    B_SCL,loop3        ;Wait for clock low
001F 0625          M        BTFSC  B_SCL>>8,B_SCL&7   ;(macro)
0020 0A1D          M        GOTO   loop3              ;(macro)
               00379
               00380
0021           00381 NEXTBYTE
0021 0004      00382        CLRWDT             ;Clear watchdog timer
0022 0C01      00383        MOVLW   1          ;Init receive byte so bit falls off at end!
0023 0028      00384        MOVWF   I2CREG
               00385
               00386 ;*Shift bits! external poll may be executed during low clock cycle only!
               00387 ;*ENABLE line is checked for loss of enable ONLY during HIGH CLOCK
               00388
               00389 ;*** CLOCK IS LOW -- DATA MAY CHANGE HERE
               00390 ;*** We have at least 4 Ês before any change can occur
               00391
0024           00392 loop4
               00393        USER_Q
                   M ;*** This would be quick user code
0024           00394 loop4A
               00395        BBC     B_SCL,loop4A      ;Wait for clock high
0024 0725          M        BTFSS   B_SCL>>8,B_SCL&7  ;(macro)
0025 0A24          M        GOTO    loop4A            ;(macro)
               00396
               00397 ;*** CLOCK IS HIGH -- SHIFT BIT - then watch for chang
                          e
0026 0305      00398        RRF     PORTA,W           ;Move RA0 into C
0027 0368      00399        RLF     I2CREG,F          ;Shift in bit
0028 0603      00400        SKPNC                     ;Skip if not done
0029 0A36      00401        GOTO    ACK_I2C           ;Acknowledge byte
               00402
002A 0608      00403        BTFSC   I2CREG,0          ;Skip if data bit was 0
002B 0A31      00404        GOTO    ii_1              ;This bit was set
002C           00405 ii_0
               00406        BBC     B_SCL,loop4       ;Wait for clock  low
002C 0725          M        BTFSS   B_SCL>>8,B_SCL&7  ;(macro)
002D 0A24          M        GOTO    loop4             ;(macro)
               00407        SKBS    B_SDA             ;Data low-high == stop
002E 0705          M        BTFSS   B_SDA>>8,B_SDA&7  ;(macro)
002F 0A2C      00408        GOTO    ii_0
0030           00409 I2CSTOP
               00410        USER_MSG                  ;process completed message!
                   M ;*** This would be user code to process a message
0030 0A11      00411        GOTO    I2CWAIT           ;back to main loop
               00412
               00413 ii_1   BBC     B_SDA,I2CWAIT     ;Data high-low == start
0031 0705          M        BTFSS   B_SDA>>8,B_SDA&7  ;(macro)
0032 0A11          M        GOTO    I2CWAIT           ;(macro)
```

```
               00414         BBC     B_SCL,loop4       ;Wait for clock  low
0033 0725          M         BTFSS   B_SCL>>8,B_SCL&7  ;(macro)
0034 0A24          M         GOTO    loop4             ;(macro)
0035 0A31      00415         GOTO    ii_1
               00416
0036           00417 ACK_I2C
               00418         BBC     B_UA,ACK_UA       ;Not addressed - check unit address
0036 0727          M         BTFSS   B_UA>>8,B_UA&7    ;(macro)
0037 0A8B          M         GOTO    ACK_UA            ;(macro)
               00419         BBS     B_SA,ACK_SA       ;Reading secondary address
0038 0647          M         BTFSC   B_SA>>8,B_SA&7    ;(macro)
0039 0A97          M         GOTO    ACK_SA            ;(macro)
               00420
               00421 ;****
               00422 ;** Do what must be done with new data bytes here (before ACKloop)
               00423 ;** Don’t ack if byte can’t be processed!
               00424 ;****
               00425 ;--------
               00426
               00427         USER_RECV
                   M ;*** This would be user code to process a received byte
                   M ;*** example code sends sub-address 0 to port b
                   M         BBC     B_ID,_NXI_notid ;Channel 0! Bit set if INITIAL address was 0
003A 0767          M         BTFSS   B_ID>>8,B_ID&7    ;(macro)
003B 0A3F          M         GOTO    _NXI_notid        ;(macro)
003C 0208          M         MOVFW   I2CREG            ;get received byte
003D 0026          M         MOVWF   PORTB             ;and write it on portb
003E 0A47          M         GOTO    IN_CONT
003F               M _NXI_notid
               00428
003F 0C07      00429         MOVLW   07H               ;Register count
0040 0169      00430         ANDWF   I2CSUBA,f         ;Limit register count
0041 0C0B      00431         MOVLW   I2CR0             ;Pseudo-registers
0042 01C9      00432         ADDWF   I2CSUBA,W         ;Offset from buffer start
0043 02A9      00433         INCF    I2CSUBA, F        ;Next sub-address
0044 0024      00434         MOVWF   FSR               ;Indirect address
0045 0208      00435         MOVFW   I2CREG
0046 0020      00436         MOVWF   ind               ;Put data into register
               00437
0047           00438 IN_CONT                           ;continue point for intercepted bytes
               00439
0047           00440 ACKloop
               00441         BBS     B_SCL,ACKloop     ;Wait for clock low
0047 0625          M         BTFSC   B_SCL>>8,B_SCL&7  ;(macro)
0048 0A47          M         GOTO    ACKloop           ;(macro)
               00442
               00443         CLB     B_SDA             ;Set ACK
0049 0405          M         BCF     B_SDA>>8,B_SDA&7  ;(macro)
004A 0CF6      00444         MOVLW   TAWRITE
004B 0005      00445         TRIS    PORTA
               00446         CLB     B_SDA             ;Set ACK (just in case docs are wrong)
004C 0405          M         BCF     B_SDA>>8,B_SDA &7 ;(macro)
               00447
               00448
004D           00449 ACKloop2
               00450         USER_Q
                   M ;*** This would be quick user code
               00451         BBC     B_SCL,ACKloop2    ;Wait for clock high
004D 0725          M         BTFSS   B_SCL>>8,B_SCL&7  ;(macro)
004E 0A4D          M         GOTO    ACKloop2          ;(macro)
               00452
004F           00453 ACKloop3
               00454         USER_Q
                   M ;*** This would be quick user code
               00455         BBS     B_SCL,ACKloop3    ;Wait for clock low
004F 0625          M         BTFSC   B_SCL>>8,B_SCL&7  ;(macro)
```

```
0050 0A4F          M         GOTO    ACKloop3          ;(macro)

0051 0CF7                    MOVLW   TAREAD            ;End ACK
0052 0005                    TRIS    PORTA

                             BBC     B_RD,NEXTBYTE   ;Skip if read (we were  acking address only)
0053 0707          M         BTFSS   B_RD>>8,B_RD&7         ;(macro)
0054 0A21          M         GOTO    NEXTBYTE               ;(macro)


                     ;*********************************************************************
                     ; I2C Readback (I2C read request)
                     ; Application specific code to get bytes to send may be added here.
                     ; This routine gets data from location pointed to by I2CSUBA and
                     ; sends it to I2C. Subsequent reads get sequential addresses. This
                     ; version AND's the register # with 7 to limit to 8 registers (for
                     ; speed). Thiscould be modified to do a comparison to an Absolute
                     ; number.
                     ;*********************************************************************

0055                 NEXTOUT

                     ;*** <<< PUT NEXT BYTE INTO I2CREG HERE NOW! >>> ***

                             USER_XMIT
                   M ;*** This would be user code to prepare an output byte
                   M ;*** example code sends id string to output
                   M         BBC     B_ID,_NXO_notid ;Channel 0! Bit set if INITIAL address was 0
0055 0767          M         BTFSS   B_ID>>8,B_ID&7  ;(macro)
0056 0A59          M         GOTO    _NXO_notid      ;(macro)
0057 0900          M         CALL    GETID           ;get next byte from ID channel
0058 0A60          M         GOTO    OUT_CONT        ;and send it
0059               M _NXO_notid

0059 0C07                    MOVLW   07H             ;Register count
005A 0169                    ANDWF   I2CSUBA,f       ;Limit register count
005B 0C0B                    MOVLW   I2CR0           ;Pseudo-registers
005C 01C9                    ADDWF   I2CSUBA,W       ;Offset from buffer start
005D 02A9                    INCF    I2CSUBA, F      ;Next sub-address
005E 0024                    MOVWF   FSR             ;Indirect address
005F 0200                    MOVFW   ind             ;Get data from register

0060                 OUT_CONT
0060 0028                    MOVWF   I2CREG
                     ;-- add code here to init I2CREG! when B_ID is clear!
0061 0C08                    MOVLW   8               ;Bit counter
0062 002A                    MOVWF   I2CBITS

                     ;** OUT bits! -- external poll may be executed during low clock cycle,
                     ;   but may also be executed during high cycle if necessary.

                     ;* ENABLE line is checked for loss of enable ONLY during HIGH CLOCK

                     ;*** CLOCK IS LOW -- CHANGE DATA HERE FIRST!

                     ;*** loop 1: data was 1
0063                 iiOUT_loop_1
0063 0368                    RLF     I2CREG,F        ;Shift data out, MSB first
0064 0603                    SKPNC                   ;1->0: change
0065 0A79                    GOTO    iiOUT_1         ;Output another 1!
                             CLB     B_SDA           ;Output 0
0066 0405          M         BCF     B_SDA>>8,B_SDA&7 ;(macro)
0067 0CF6                    MOVLW   TAWRITE
0068 0005                    TRIS    PORTA
                             CLB     B_SDA           ;Set data (just in case docs are wrong)
0069 0405          M         BCF     B_SDA>>8,B_SDA&7 ;(macro)
```

```
                   
006A               iiOUT_0
006A 0004                  CLRWDT                  ;Clear watchdog timer
                   
                           USER_Q
                 M ;*** This would be quick user code
                   
006B               iiOUT_loop_02
                           BBC     B_SCL,iiOUT_loop_02 ;Wait for clock high
006B 0725        M         BTFSS   B_SCL>>8,B_SCL&7 ;(macro)
006C 0A6B        M         GOTO    iiOUT_loop_02    ;(macro)
                   
                           USER_Q
                 M ;*** This would be quick user code
                   
006D               iiOUT_loop_03
                           BBS     B_SCL,iiOUT_loop_03 ;Wait for clock low
006D 0625        M         BTFSC   B_SCL>>8,B_SCL&7    ;(macro)
006E 0A6D        M         GOTO    iiOUT_loop_03       ;(macro)
                   
006F 02EA                  DECFSZ  I2CBITS, F      ;Count bits
0070 0A74                  GOTO    iiOUT_loop_0    ;Loop for last bit 0
0071 0CF7                  MOVLW   TAREAD          ;Done with last bit 0... Set to 1 for ACK
0072 0005                  TRIS    PORTA
0073 0A80                  GOTO    iiOUT_ack       ;Get ACK
                   
0074               iiOUT_loop_0
0074 0368                  RLF     I2CREG,F        ;Shift data out, MSB first
0075 0703                  SKPC                    ;0->1: change
0076 0A6A                  GOTO    iiOUT_0         ;Output another 0!
                   
0077 0CF7                  MOVLW   TAREAD          ;Set to 1
0078 0005                  TRIS    PORTA
                   
0079               iiOUT_1
0079 0004                  CLRWDT                  ;Clear watchdog timer
                   
                   
                           USER_Q
                 M ;*** This would be quick user code
                   
007A               iiOUT_loop_12
                           BBC     B_SCL,iiOUT_loop_12 ;Wait for clock high
007A 0725        M         BTFSS   B_SCL>>8,B_SCL&7     ;(macro)
007B 0A7A        M         GOTO    iiOUT_loop_12        ;(macro)
                   
                   
                           USER_Q
                 M ;*** This would be quick user code
                   
007C               iiOUT_loop_13
                           BBS     B_SCL,iiOUT_loop_13 ;Wait for clock low
007C 0625        M         BTFSC   B_SCL>>8,B_SCL&7     ;(macro)
007D 0A7C        M         GOTO    iiOUT_loop_13        ;(macro)
                   
007E 02EA                  DECFSZ  I2CBITS, F           ;Count bits
007F 0A63                  GOTO    iiOUT_loop_1         ;Loop for last bit 1
                   
                   
0080               iiOUT_ack ;Get acknowledge
0080 02A9                  INCF    I2CSUBA, F           ;Next sub-address
                   
0081               iiOUT_loop_a2
                           BBC     B_SCL,iiOUT_loop_a2 ;Wait for clock high
0081 0725        M         BTFSS   B_SCL>>8,B_SCL&7    ;(macro)
0082 0A81        M         GOTO    iiOUT_loop_a2       ;(macro)
```

```
               00561
               00562         BBS     B_SDA,I2CWAIT   ;No ACK --- wait for restart!
0083 0605          M         BTFSC   B_SDA>>8,B_SDA&7    ;(macro)
0084 0A11          M         GOTO    I2CWAIT             ;(macro)
               00563
               00564 ;-- prepare next character here!
               00565
0085           00566 iiOUT_loop_a3
               00567         BBC     B_SCL,NEXTOUT   ;Wait for clock low - output next char!
0085 0725          M         BTFSS   B_SCL>>8,B_SCL&7    ;(macro)
0086 0A55          M         GOTO    NEXTOUT             ;(macro)
               00568
               00569         BBS     B_SDA,iiOUT_loop_a3 ;Watch out for new start condition!
0087 0605          M         BTFSC   B_SDA>>8,B_SDA&7    ;(macro)
0088 0A85          M         GOTO    iiOUT_loop_a3       ;(macro)
0089 0A11      00570         GOTO    I2CWAIT             ;Stop received!
008A           00571 USER_READ                           ;user code to process data sent
               00572
008A 0A11      00573         GOTO    I2CWAIT
               00574
               00575
               00576 ;*********************************************************************
               00577 ;* Unit address received - check for valid address
               00578 ;*
               00579 ;*********************************************************************
008B           00580 ACK_UA
               00581         SEB     B_UA                ;Flag unit address received
008B 0527          M         BSF     B_UA>>8,B_UA&7      ;(macro)
008C 0608      00582         BTFSC   I2CREG,0            ;Skip if data coming in
               00583         SEB     B_RD                ;Flag - reading from slave
008D 0507          M         BSF     B_RD>>8,B_RD&7      ;(macro)
008E 0208      00584         MOVF    I2CREG,W            ;Get address
008F 0EFE      00585         ANDLW   0FEH                ;Mask direction flage before compare
0090 0FD6      00586         XORLW   DEVICE_ADDRESS      ;Device address
0091 0743 0A11 00587         BNZ     I2CWAIT             ;Not for me! (skip rest of message)
               00588         BBS     B_RD,ACKloop        ;Read - no secondary address
0093 0607          M         BTFSC   B_RD>>8,B_RD&7      ;(macro)
0094 0A47          M         GOTO    ACKloop             ;(macro)
               00589         SEB     B_SA                ;Next is secondary address
0095 0547          M         BSF     B_SA>>8,B_SA&7      ;(macro)
0096 0A47      00590         GOTO    ACKloop             ;Yes! ACK address and continue!
               00591
               00592 ;*********************************************************************
               00593 ;* Secondary address received - stow it!
               00594 ;* SA = 0 is converted to 128 to facilitate ID read
               00595 ;*********************************************************************
0097           00596 ACK_SA
               00597         CLB     B_SA                ;Flag second address received
0097 0447          M         BCF     B_SA>>8,B_SA&7      ;(macro)
               00598         CLB     B_ID
0098 0467          M         BCF     B_ID>>8,B_ID&7      ;(macro)
0099 0208      00599         MOVFW   I2CREG              ;Get subaddress
009A 0643      00600         SKPNZ                       ;Not 0
               00601         SEB     B_ID                ;Flag - id area selected
009B 0567          M         BSF     B_ID>>8,B_ID&7      ;(macro)
009C 0029      00602         MOVWF   I2CSUBA             ;Set subaddress
009D 0A47      00603         GOTO    ACKloop
               00604
               00605
               00606         END
```

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

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

All other memory blocks unused.

Program Memory Words Used:   159
Program Memory Words Free:   353


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

MICROCHIP

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