MICROCHIP

AN541

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

Author: Don Lekei - NII Norsat International Inc.

INTRODUCTION

The 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^{2}C 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^{2}C 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^{2}C specification is supported, which can be controlled by the same software which would talk to a Microchip 24LCXX series EEPROM.

THE I^{2}C BUS

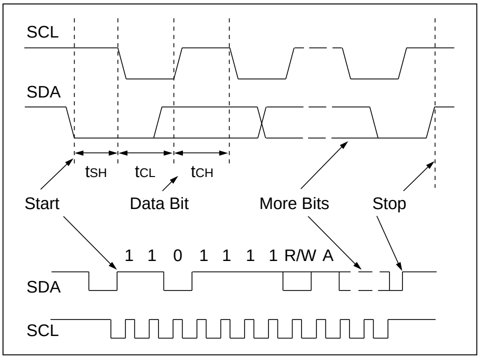
The I^{2}C bus is a master-slave two-wire interface, consisting of a clock line (SCL) and a data line (SDA). Bi-directional communication (and in a full, multi-master system, collision detection and clock synchronization) is facilitated through the use of a wire-and (ie. active-low, passive high) connection.

The standard-mode I^{2}C bus supports SCL clock frequency up to 100 KHz. The newly released fast-mode I^{2}C bus supports clock rate up to 400 KHz. This application note will support 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, SDA is only permitted to change while SCL is low, thus providing two violation conditions (see 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^{2}C 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 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^{2}C definition otherwise the master would be required to operate in slow mode, which most software implementations of I^{2}C 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 send a stop and retry.

If the direction bit is "0" the next byte is considered the sub-address (this is an extension to I^{2}C 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.

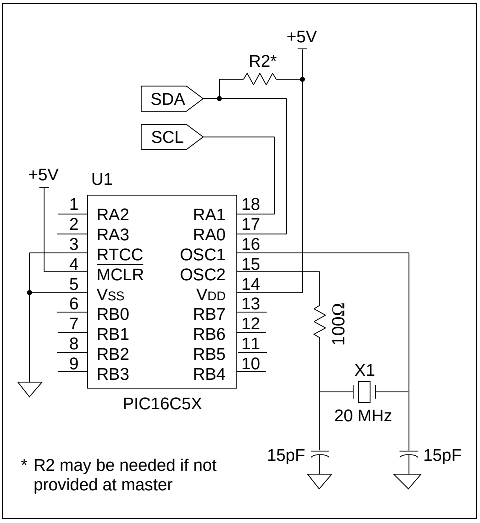
If the direction bit is "1", the slave will transfer successive bytes to the master (while the master holds the line at '1'), while the master acknowledges each byte with a "0" in the ninth bit. The master can terminate the transfer by not acknowledging the last byte, while the slave can stop the transfer 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^{2}C CONNECTIONS

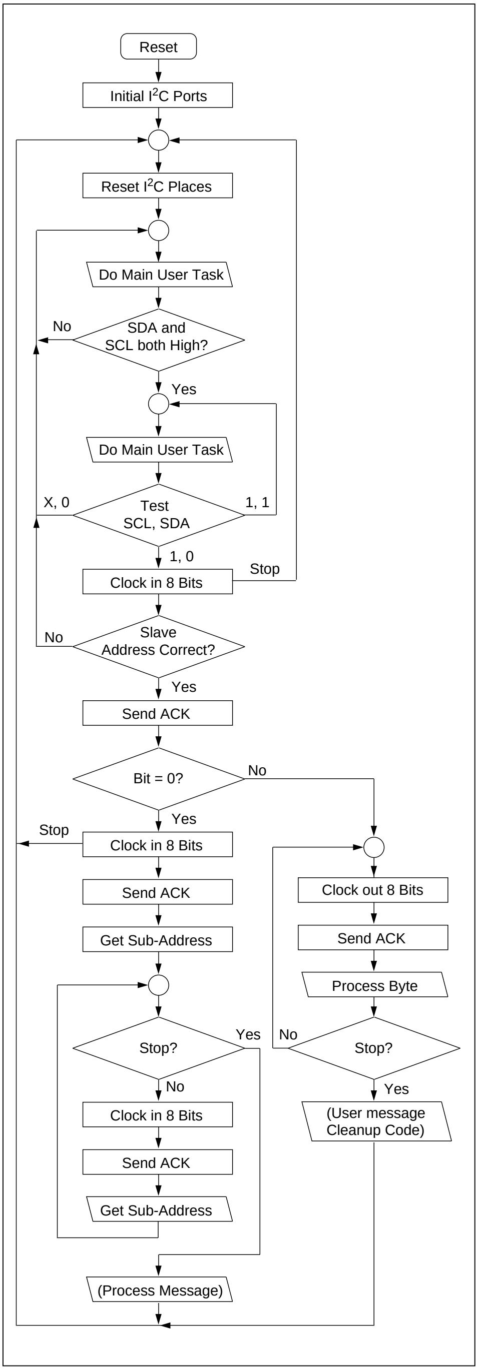


* R2 may be needed if not provided at master

The connections to the device are shown in Figure 2. The use of RA0 for data in 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^{2}C device with 8 registers, accessed as sub-addresses 1 through 8 (modulo 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^{2}C DEVICE FLOWCHART



When the initial sub-address 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 port B, 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, tSH of 4µs will be violated (see Fig. 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 4µs 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, Phillips Components, 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 (I2C) 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:

```
MPASM B0.54                                                             PAGE  1



                             LIST    P=16C54, C=80, N=0, R=DEC

0676                 CPU     EQU     1654
0000                 SIM     EQU     0               ;Change timing constants for simulator


                             IF      (CPU==1654) || (CPU==1655)
01FF                 _RESVEC EQU     01FFH           ;16c54 start address
                             ENDIF

                             IF      CPU==1656
                     _RESVEC EQU     03FFH           ;16C56 start address
                             ENDIF

                             IF      CPU==1657
                     _RESVEC EQU     07FFH           ;16C57 start address
                             ENDIF

                     ;*** Reset Vector *****************************************


                             ORG     _RESVEC         ;
                     RESVEC                          ;
01FF 0A0B                    GOTO    INIT            ;

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


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

                     ;* Macros to set/clear/branch/skip on bits
                     ;* These macros define and use synthetic “bit labels”
                     ;* Bit labels contain the address and bit of a location
                     ;*
                     ;**********************************************************

                     ;*      Usage                    Description
                     ;*      ___________  _____________

                     ;*      BIT     label,bit,file  ;Define a bit label
                     ;*      SEB     label           ;set bit using bit label
                     ;*      CLB     label           ;clear bit using bit label

                     ;*      SKBS    label           ;SKIP on bit set
                     ;*      SKBC    label           ;SKIP on bit clear
                     ;*      BBS     label,address   ;BRANCH on bit set
                     ;*      BBC     label,address   ;BRANCH on bit clear
                     ;*      CBS     label,address   ;CALL on bit set
                     ;*      CBC     label,address   ;CALL on bit clear
                     ;*
                     ;**********************************************************


                     BIT     MACRO   label,bit,file  ;Define a bit label
                     label   EQU     file<<8|bit
                             ENDM                    ;

                     SEB     MACRO   label           ;Set bit
                             BSF     label>>8,label&7;(macro)
                             ENDM                    ;
                     CLB      MACRO   label                    ;Clear bit
                              BCF     label>>8,label&7         ;(macro)
                              ENDM                             ;

                     SKBS     MACRO   label                    ;Skip on bit set
                              BTFSS   label>>8,label&7         ;(macro)
                              ENDM

                     SKBC     MACRO   label                    ;Skip on bit clear
                              BTFSC   label>>8,label&7         ;(macro)
                              ENDM

                     BBS      MACRO   label,address            ;Branch on bit set
                              BTFSC   label>>8,label&7         ;(macro)
                              GOTO    address                  ;(macro)
                              ENDM                             ;

                     BBC      MACRO   label,address            ;Branch on bit clear
                              BTFSS   label>>8,label&7         ;(macro)
                              GOTO    address                  ;(macro)
                              ENDM

                     CBS      MACRO   label,address            ;Call on bit set
                              CALL    label>>8,label&7         ;(macro)
                              ENDM                             ;

                     CBC      MACRO   label,address            ;Call on bit clear
                              CALL    label>>8,label&7         ;(macro)
                              ENDM


                     ;For Assembler portability

0000                 W       EQU     0                         ;For file,W
0000                 w       EQU     0                         ;For file,W
0001                 F       EQU     1                         ;For file,F
0001                 f       EQU     1                         ;For file,F

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

                     ;* REGISTER DECLARATIONS
                     ;**********************************************************



                             ORG     0                         ;ORG for register declaration


0000  0001           ind     RES     1                         ;0=pseudo-reg 0 for indirect

0001  0001           RTCC    RES     1                         ;1=real time counter
0002  0001           PC      RES     1                         ;2=PC
0003  0001           STATUS  RES     1                         ;3=status reg

                     ;* Status reg bits

                              BIT    B_C,0,STATUS              ;Carry
0300                 B_C      EQU    STATUS<<8|0


                              BIT    B_DC,1,STATUS             ;Half carry
0301                 B_DC     EQU    STATUS<<8|1


                              BIT    B_Z,2,STATUS              ;Zero
0302                 B_Z      EQU    STATUS<<8|2
```

```
                              BIT     B_PD,3,STATUS   ;Power down
0303                 B_PD     EQU     STATUS<<8|3


                              BIT     B_TO,4,STATUS   ;Timeout
0304                 B_TO     EQU     STATUS<<8|4


                              BIT     B_PA0,5,STATUS  ;Page select (56/57 only)
0305                 B_PA0    EQU     STATUS<<8|5


                              BIT     B_PA1,6,STATUS  ;Page select (56/57 only)
0306                 B_PA1    EQU     STATUS<<8|6


                              BIT     B_PA2,7,STATUS  ;GP flag
0307                 B_PA2    EQU     STATUS<<8|7



0004  0001           FSR     RES     1               ;4=file select reg 0-4=indirect address
0005  0001           PORTA   RES     1               ;5=port A I/O register (4 bits)
0006  0001           PORTB   RES     1               ;6=port B I/O register

                             IF      (CPU==1655)||(CPU==1657)
                     PORTC   RES     1               ;7=I/O port C on 16C54/56 only
                             ENDIF

                                                     ;registers used by this code

0007  0001           I2CFLG  RES     1               ;I2C flag reg
                                                     ;—i2c flags———————————————————————

                              BIT     B_RD,0,I2CFLG   ;Flag: 1=read
0700                 B_RD     EQU     I2CFLG<<8|0


                              BIT     B_UA,1,I2CFLG   ;Flag: 0=reading unit address
0701                 B_UA     EQU     I2CFLG<<8|1


                              BIT     B_SA,2,I2CFLG   ;Flag: 1=reading subabbress
0702                 B_SA     EQU     I2CFLG<<8|2


                              BIT     B_ID,3,I2CFLG   ;Flag: 1=reading id
0703                 B_ID     EQU     I2CFLG<<8|3


                     ;—————————————————————————————

0008  0001           I2CREG  RES     1               ;I2C I/O register
0009  0001           I2CSUBA RES     1               ;Subaddress
000A  0001           I2CBITS RES     1               ;I2C xmit bit counter


                     ;**********************************************************
                     ;* 8 Pseudo registers accessed by sub-addresses 1-8
                     ;* (address 0 accesses the ID string)
                     ;* these are read-write registers
                     ;**********************************************************



000B                 I2CR0   EQU     $               ;Sub-address 8
000B  0001                   RES     1               ;8 pseudo registers

000C                 I2CR1   EQU     $               ;Sub-address 1
```

```
000C  0001                   RES     1

000D                 I2CR2   EQU     $               ;Sub-address 2
000D  0001                   RES     1

000E                 I2CR3   EQU     $               ;Sub-address 3
000E  0001                   RES     1

000F                 I2CR4   EQU     $               ;Sub-address 4
000F  0001                   RES     1

0010                 I2CR5   EQU     $               ;Sub-address 5
0010  0001                   RES     1

0011                 I2CR6   EQU     $               ;Sub-address 6
0011  0001                   RES     1

0012                 I2CR7   EQU     $               ;Sub-address 7
0012  0001                   RES     1

                     ;Constants used by program

00D6                 DEVICE_ADDRESS EQU      0D6H    ;I2C device address


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



00F7                 TAREAD   EQU     B’11110111'     ;TRISA register for SDA rea

00F6                 TAWRITE  EQU     B’11110110'     ;TRISA register for SDA wri

00F7                 TAINIT   EQU     TAREAD          ;Initial TRISA value

                              BIT     B_SDA,0,PORTA   ;I2C SDA (data) This must be bit 0!
0500                 B_SDA    EQU     PORTA<<8|0


                              BIT     B_SCL,1,PORTA   ;I2C SCL (clock)
0501                 B_SCL    EQU     PORTA<<8|1


                     ;spare           B_???,2,PORTA   ;not used
                     ;spare           B_???,3,PORTA   ;not used

                     ;**********************************************************
                     ;**
                     ;** Port B definition (Parallel out)
                     ;**
                     ;**********************************************************

0000                 TBINIT  EQU     B’00000000'     ;Port B tris (all output)
00FF                 PBINIT  EQU     B’11111111'     ;Port B init


                     ;**********************************************************
                     ;* Macros to contain user POLL loop code.
                     ;* These are implimented 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
                     ;*              I˝C Spec. More than 4 Ês may result in I˝C
                     ;*              full spec speed.
                     ;*
                     ;* USER_MSG:    Code to execute at receipt of I˝C 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

                                    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

                                     CALL    GETID                        ;get next byte from ID
                                     GOTO    OUT_CONT                     ;and send it
                     _NXO_notid
                                     ENDM

                     ;**********************************************************
                     ; START OF CODE
                     ;**********************************************************
                             ORG     0
                     ;**********************************************************

                     ;* Device ID Table (must be at start)
                     ;* TABLE FOR UNIT ID returns next char in W
                     ;**********************************************************

                     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 / subsequant write
                     ;*
                     ;* SDA   —|   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
                     ;* SCL  ——|  |—|  |—|  |—|  |—|  |—|  |—|  |—|  |
                     ;*           |—|  |—|  |—|  |—|  |—|  |—|  |—|  |—|
                     ;*
                     ;* STATE: 0 1 2  3  2  3  2  3  2  3  2  3  2  3  2  3  2
                     ;*
                     ;*            <============== subsequant reads ============
                     ;*
                     ;* 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
                     ;* SCL  —|  |—|  |—|  |—|  |—|  |—|  |—|  |—|  |—|
                     ;*        |—|  |—|  |—|  |—|  |—|  |—|  |—|  |—|  |
                     ;*
                     ;* STATE:  7  8  7  8  7  8  7  8  7  8  7  8  7  8  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
                     ;* SCL  —|  |—|  |—|  |—|  |—|  |—|  |—|  |—|  |—|
                     ;*        |—|  |—|  |—|  |—|  |—|  |—|  |—|  |—|  |
                     ;*
                     ;* STATE:  7  8  7  8  7  8  7  8  7  8  7  8  7  8  7  8
                     ;*
                     ;* 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&CLOC
                     ;***********************************************************
                     ;I2C interface uses PORTA
                     ;note SDA must be on PORTA,0 for code efficiency
                     ;***********************************************************
                     ;** INIT
                     ;** Hardware reset entry point
                     ;**
                     ;***********************************************************
                     INIT    ;Power-on entry
                     ;***********************************************************
                     ;** RESET
                     ;** software reset entry point
                     ;**
                     ;***********************************************************
                     RESET                                  ;Soft reset

000B 0CF7                    OVLW   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
                     ;
                     ;******************************************************
                     I2CWAIT
0011 0004                     CLRWDT                        ;Clear watchdog timer
                              CLB     B_UA                  ;Init state flags
0012 0427                     BCF     B_UA>>8,B_UA&7
                              CLB     B_SA                  ;Init state flags
0013 0447                     BCF     B_SA>>8,B_SA&7
                              CLB     B_RD                  ;Init state flags
0014 0407                     BCF     B_RD>>8,B_RD&7
                     loop1
0015 0004                    CLRWDT                         ;Clear watchdog timer
                     USER_MAIN                              ;Call user code while in idle state
                     ;*** This would be user code for idle loop
                              SKBC    B_SDA                 ;Wait for SDA&SCL=H
0016 0605                     BTFSC   B_SDA>>8,B_SDA&7
                     loop2
                              SKBS    B_SCL                 ;
0017 0725                     BTFSS   B_SCL>>8,B_SCL&7
0018 0A15                     GOTO    loop1                 ; No longer valid to wait f
0019 0004                     CLRWDT                        ;Clear watchdog timer
                              USER_MAIN                     ;Call user code while in idle state
                     ;*** This would be user code for idle loop
                     ;** wait for start **
                              SKBC    B_SCL                 ;Clock has dropped
001A 0625                     BTFSC   B_SCL>>8,B_SCL&7
                              SKBC    B_SDA                 ;Data dropped... Start!
001B 0605                     BTFSC   B_SDA>>8,B_SDA&7
001C 0A17                     GOTO    loop2
                     ;** START RECEIVED! — wait for first bit!
                     loop3
                              BBS     B_SDA,I2CWAIT         ;Data raised before clock dropped —
001D 0605                     BTFSC   B_SDA>>8,B_SDA&7
001E 0A11                     GOTO    I2CWAIT
                              BBS     B_SCL,loop3           ;Wait for clock low
001F 0625                     BTFSC   B_SCL>>8,B_SCL&7
0020 0A1D                     GOTO    loop3
                     NEXTBYTE
0021 0004                     CLRWDT                        ;Clear watchdog timer
0022 0C01                     MOVLW    1                    ;Init receive byte so bit f
0023 0028                     MOVWF    I2CREG
                     ;** Shift bits! — external poll may be executed during low
                     ;* ENABLE line is checked for loss of enable ONLY during HI
                     ;*** CLOCK IS LOW — DATA MAY CHANGE HERE
                     ;*** We have at least 4 Ês before any change can occur
                     loop4
                              USER_Q
                     ;*** This would be quick user code
                     loop4A
                             BBC      B_SCL,loop4A          ;Wait for clock high
0024 0725                    BTFSS    B_SCL>>8,B_SCL&7
0025 0A24                    GOTO     loop4A
                     ;*** CLOCK IS HIGH — SHIFT BIT - then watch for change
0026 0305                    RRF     PORTA,W                ;Move RA0 into C
0027 0368                    RLF     I2CREG,F               ;Shift in bit
0028 0603                    SKPNC                          ;Skip if not done
0029 0A36                    GOTO    ACK_I2C                ;Acknowledge byte
002A 0608                    BTFSC   I2CREG,0               ;Skip if data bit was 0
002B 0A31                    GOTO    ii_1                   ;This bit was set
                     ii_0
                             BBC      B_SCL,loop4           ;Wait for clock  low
002C 0725                    BTFSS    B_SCL>>8,B_SCL&7
002D 0A24                    GOTO     loop4
                             SKBS     B_SDA                 ;Data low-high == stop
002E 0705                    BTFSS    B_SDA>>8,B_SDA&7
002F 0A2C                    GOTO     ii_0
                             I2CSTOP
                             USER_MSG                       ;process completed message!
                     ;*** This would be user code to process a message
0030 0A11                    GOTO     I2CWAIT               ;back to main loop
```

```
                     ii_1    BBC     B_SDA,I2CWAIT          ;Data high-low == start
0031 0705                    BTFSS   B_SDA>>8,B_SDA&7
0032 0A11                    GOTO    I2CWAIT
                             BBC     B_SCL,loop4            ;Wait for clock  low
0033 0725                    BTFSS   B_SCL>>8,B_SCL&7
0034 0A24                    GOTO    loop4
0035 0A31                    GOTO    ii_1
                     ACK_I2C
                             BBC     B_UA,ACK_UA            ;Not addressed - check unit address
0036 0727                    BTFSS   B_UA>>8,B_UA&7
0037 0A8B                    GOTO    ACK_UA
                             BBS     B_SA,ACK_SA            ;Reading secondary address
0038 0647                    BTFSC   B_SA>>8,B_SA&7
0039 0A97                    GOTO    ACK_SA
                     ;****
                     ;** Do what must be done with new data bytes here (before A

                     ;** Don’t ack if byte can’t be processed!
                     ;****
                     ;____
                     USER_RECV
                     ;*** This would be user code to process a received byte
                     ;*** example code sends sub-address 0 to port b
003A 0767                    BTFSS   B_ID>>8,B_ID&7
003B 0A3F                    GOTO    _NXI_notid
003C 0208                    MOVFW   I2CREG                 ;get received byte
003D 0026                    MOVWF   PORTB                  ;and write it on portb
003E 0A47                    GOTO    IN_CONT
                     _NXI_notid
003F 0C07                    MOVLW   07H                    ;Register count
0040 0169                    ANDWF   I2CSUBA,f              ;Limit register count
0041 0C0B                    MOVLW   I2CR0                  ;Pseudo-registers
0042 01C9                    ADDWF   I2CSUBA,W              ;Offset from buffer start
0043 02A9                    INCF    I2CSUBA                ;Next sub-address
0044 0024                    MOVWF   FSR                    ;Indirect address
0045 0208                    MOVFW   I2CREG
0046 0020                    MOVWF   ind                    ;Put data into register
                     IN_CONT                                ;continue point for interce
                     ACKloop
                             BBS     B_SCL,ACKloop          ;Wait for clock low
0047 0625                    BTFSC   B_SCL>>8,B_SCL&7
0048 0A47                    GOTO    ACKloop
                             CLB     B_SDA                  ;Set ACK
0049 0405                    BCF     B_SDA>>8,B_SDA&7
004A 0CF6                    MOVLW   TAWRITE
004B 0005                    TRIS    PORTA
                             CLB     B_SDA                  ;Set ACK (just in case docs are wrong)
004C 0405                    BCF     B_SDA>>8,B_SDA&7
                     ACKloop2
                            USER_Q
                     ;*** This would be quick user code
                             BBC     B_SCL,ACKloop2         ;Wait for clock high
004D 0725                    BTFSS   B_SCL>>8,B_SCL&7
004E 0A4D                    GOTO    ACKloop2
                     ACKloop3
                            USER_Q
                     ;*** This would be quick user code
                             BBS     B_SCL,ACKloop3         ;Wait for clock low
004F 0625                    BTFSC   B_SCL>>8,B_SCL&7
0050 0A4F                    GOTO    ACKloop3
0051 0CF7                    MOVLW   TAREAD                 ;End ACK
0052 0005                    TRIS    PORTA
                             BBC     B_RD,NEXTBYTE          ;Skip if read (we were acking address on
0053 0707                    BTFSS   B_RD>>8,B_RD&7
0054 0A21                    GOTO    NEXTBYTE
                     ;**********************************************************
                     ; I2C Readback (I2C read request)
                     ; Application specific code to get bytes to send may be add
                     ; This routine gets data from location pointed to by I2CSUB
                     ; sends it to I2C. Subsequent reads get sequential addresse
                     ; AND’s the register # with 7 to limit to 8 registers (for
                     ; could be modified to do a comparison to an ablolute numbe
                     ;
                     ;**********************************************************
                     NEXTOUT
                     ;*** <<< PUT NEXT BYTE INTO I2CREG HERE NOW! >>> ***
                     USER_XMIT
                     ;*** This would be user code to prepare an output byte
                     ;*** example code sends id string to output
0055 0767                    BTFSS  B_ID>>8,B_ID&7
0056 0A59                    GOTO   _NXO_notid
0057 0900                    CALL    GETID                  ;get next byte from ID chan
0058 0A60                    GOTO    OUT_CONT               ;and send it
                     _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                ;Next sub-address
005E 0024                    MOVWF   FSR                    ;Indirect address
005F 0200                    MOVFW   ind                    ;Get data from register
                     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 c
                     ;                may also be executed during high cycle if
                     ;* ENABLE line is checked for loss of enable ONLY during HI
                     ;*** CLOCK IS LOW — CHANGE DATA HERE FIRST!
                     ;*** loop 1: data was 1
                     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                     BCF     B_SDA>>8,B_SDA&7
0067 0CF6                     MOVLW   TAWRITE
0068 0005                     TRIS    PORTA
                              CLB     B_SDA                 ;Set data (just in case docs are
0069 0405                     BCF     B_SDA>>8,B_SDA&7
                     iiOUT_0
006A 0004                     CLRWDT                        ;Clear watchdog timer
                     USER_Q
                     ;*** This would be quick user code
                     iiOUT_loop_02
                              BBC     B_SCL,iiOUT_loop_02   ;Wait for clock high
006B 0725                     BTFSS  B_SCL>>8,B_SCL&7
006C 0A6B                     GOTO   iiOUT_loop_02
                     USER_Q
                     ;*** This would be quick user code
                     iiOUT_loop_03
                              BBS    B_SCL,iiOUT_loop_03    ;Wait for clock low
006D 0625                     BTFSC  B_SCL>>8,B_SCL&7
006E 0A6D                     GOTO   iiOUT_loop_03
006F 02EA                     DECFSZ  I2CBITS               ;Count bits
0070 0A74                     GOTO    iiOUT_loop_0          ;Loop for last bit 0
0071 0CF7                     MOVLW   TAREAD                ;Done with last bit 0... Se
0072 0005                     TRIS    PORTA
0073 0A80                     GOTO    iiOUT_ack             ;Get ACK
                     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
```

```
                     iiOUT_1
0079 0004                     CLRWDT                        ;Clear watchdog timer
                     USER_Q
                     ;*** This would be quick user code
                     iiOUT_loop_12
                             BBC     B_SCL,iiOUT_loop_12    ;Wait for clock high
007A 0725                            BTFSS  B_SCL>>8,B_SCL&7
007B 0A7A                    GOTO    iiOUT_loop_12
                     USER_Q
                     ;*** This would be quick user code
                     iiOUT_loop_13
                             BBS     B_SCL,iiOUT_loop_13    ;Wait for clock low
007C 0625                    BTFSC   B_SCL>>8,B_SCL&7
007D 0A7C                    GOTO    iiOUT_loop_13
007E 02EA                    DECFSZ  I2CBITS                ;Count bits
007F 0A63                    GOTO    iiOUT_loop_1           ;Loop for last bit 1
                     iiOUT_ack                              ;Get acknowledge
0080 02A9                    INCF    I2CSUBA                ;Next sub-address
                     iiOUT_loop_a2
                             BBC     B_SCL,iiOUT_loop_a2    ;Wait for clock high
0081 0725                    BTFSS   B_SCL>>8,B_SCL&7
0082 0A81                    GOTO    iiOUT_loop_a2
                             BBS     B_SDA,I2CWAIT          ;No ACK — wait for restart!
0083 0605                    BTFSC   B_SDA>>8,B_SDA&7
0084 0A11                    GOTO    I2CWAIT
                     ;— prepare next character here!
                     iiOUT_loop_a3
                             BBC     B_SCL,NEXTOUT          ;Wait for clock low - output next!
0085 0725                    BTFSS   B_SCL>>8,B_SCL&7
0086 0A55                    GOTO    NEXTOUT
                             BBS      B_SDA,iiOUT_loop_a3   ;Watch out for new start condition!
0087 0605                    BTFSC   B_SDA>>8,B_SDA&7
0088 0A85                    GOTO    iiOUT_loop_a3
0089 0A11                    GOTO    I2CWAIT                ;Stop received!
008A 0A11                    GOTO    I2CWAIT
                     ;**********************************************************
                     ;* Unit address received - check for valid address
                     ;*
                     ;**********************************************************
                     ACK_UA
                              SEB     B_UA                  ;Flag unit address received
008B 0527                     BSF     B_UA>>8,B_UA&7
008C 0608                     BTFSC   I2CREG,0              ;Skip if data coming in
                              SEB     B_RD                  ;Flag - reading from slave
008D 0507                     BSF     B_RD>>8,B_RD&7
008E 0208                     MOVF    I2CREG,W              ;Get address
008F 0EFE                     ANDLW   0FEH                  ;Mask direction flage before compare
0090 0FD6                     XORLW   DEVICE_ADDRESS        ;Device address
0091 0743                     BNZ     I2CWAIT               ;Not for me! (skip rest of message)
0092 0A11                     BBS     B_RD,ACKloop          ;Read - no secondary address
0093 0607                     BTFSC  B_RD>>8,B_RD&7
0094 0A47                     GOTO   ACKloop
                              SEB     B_SA                  ;Next is secondary address
0095 0547                     BSF    B_SA>>8,B_SA&7
0096 0A47                     GOTO    ACKloop               ;Yes! ACK address and continue
                     ;**********************************************************
                     ;* Secondary address received - stow it!
                     ;* SA = 0 is converted to 128 to facilitate ID read
                     ;**********************************************************
                              ACK_SA
                              CLB     B_SA                  ;Flag second address received
0097 0447                     BCF    B_SA>>8,B_SA&7
                              CLB     B_ID
0098 0467                     BCF    B_ID>>8,B_ID&7
0099 0208                     MOVFW   I2CREG                ;Get subaddress
009A 0643                     SKPNZ                         ;Not 0
                              SEB     B_ID                  ;Flag - id area selected
009B 0567                     BSF    B_ID>>8,B_ID&7
```

```
009C 0029                      MOVWF   I2CSUBA              ;Set subaddress
009D 0A47                      GOTO    ACKloop
                               END


Errors   :    0
Warnings :    0
```

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

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights. The Microchip logo and name are registered trademarks of Microchip Technology Inc. All rights reserved. All other trademarks mentioned herein are the property of their respective companies.