MICROCHIP

AN558

Using the 24XX65 and 24XX32 with Stand Alone PIC16C54 Code

USING THE SMART SERIAL™ SERIES

With the advent of CMOS silicon devices came the battery powered application. The battery powered application required more functionality and thus more power from the microcontroller which required more power from the batteries which cost more. Just as nature has a definite cycle, so it seems, does the portable application. Nowhere has this been more evident than in the areas of personal communication and data acquisition. Never before in the history of the industrial revolution have the type of applications seen emerging today been possible. Every man-made object that requires human or machine interface, has the potential to be controlled by embedded circuits and to be powered by the advanced technology batteries available today. Along with this increased functionality comes the requirement for MORE MEMORY. More in a smaller package, at a better price requiring less power. The majority of the hand-held embedded applications either use what little non-volatile memory that is available on the controllers or they use external devices. Parallel EEPROMs require too many I/O connects, which is the major source of device active current. Serial EEPROMs are typically the answer for these applications and the industry standard I^2C™ is the industry leader (>70 % worldwide). The I^2C standard specifies a maximum 16K bit address space. How does the personal communications system designer, who requires more memory, solve this problem?

THE I^2C DILEMMA

The I^2C serial bus has many advantages over other common serial interfaces for serial embedded devices. The I^2C bus with level-triggered inputs offers better noise immunity over edge triggered technology. Opcodes are not needed to communicate with storage devices because all interfaces are intuitive and comparable to parallel devices. The I^2C protocol assigns a slave address for each unique device or device family. Microchip Technology, and most non-volatile suppliers, use the slave address of 1010 for serial electrical erasable programmable read only memory (Serial EEPROM). The protocol also facilitates up to a maximum of 16K bytes of memory on the bus via the 8-bit address and the three device or memory block select pins A0, A1, and A2. Herein lies the dilemma: with the advent of the more sophisticated personal communication devices such as cellular and full featured phones, personal digital assistants and palm-top computers, 16K bytes is not enough!

Smart Serial Products

The Smart Serial concept grew from the industry need for increased memory requirements in I^2C embedded applications, smarter endurance performance, security needs, and the need for more functionality at lower power demands. Currently, Microchip Technology Inc. has the 24XX65 and 24XX32 devices available. All comments in this application note pertain primarily to the 24XX65 but the routines and general architectural comments apply to the 24XX32 as well.

The user should reference the individual data sheets for specific differences.

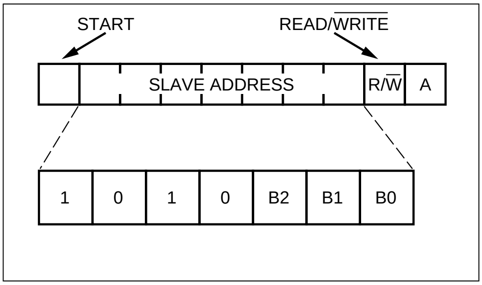
The Microchip 24XX65 is a serial memory device with 64K bits (8Kx8) capacity and additional patent pending unique features not found any where else. First let's look at the current I^2C addressing scheme, the cascadable solution, and finally the Microchip total embedded systems solution.

I^2C ADDRESSING

The I^2C protocol utilizes a master/slave bi-directional communication bus. The master, usually a microcontroller, which controls the bus, generates the serial clock (SCL) and originates the start and stop conditions. A Serial EEPROM is considered a slave device and is defined as a transmitter during read operations and generates acknowledges when receiving data from the master. The start and stop bits are utilized to control the bus. Normal operation begins with a start bit and ends with a stop bit. Following a start, commands begin with an 8 bit 'control' byte originated by the master. The control byte identifies the slave device to be addressed and defines the operation to take place. A typical control byte for a Serial EEPROM (slave address = 1010) is shown in Figure 1. The control byte, therefore, consists of a start bit, a four-bit slave address, a read/write bit and an acknowledge. The slave address consists of the 1010 identifying address plus the three block or chip select bits.

8

FIGURE 1 - CONTROL BYTE ALLOCATION



All memory and peripheral devices on the I²C bus conform to this sequence for identification and selection. There are 128 assigned slave addresses in the standard protocol. There is a 10-bit extension to the protocol for 1024 additional slave addresses.

THE CASCADABLE SOLUTION

Cascading is an addressing scheme used in the 24164 (2Kx8 SEEPROM) to enable using more than the 16K limit set forth by the standard. In this method the A0, A1, and A2 pins are mapped into bits 2, 3, and 4 of the 8-bit slave address. This approach allows the system designer to use the non-standard 24164 to increase the total memory of a given memory subsystem. This solution is definitely workable, but does not offer the user the system design flexibility needed for the newer and more complex systems. Most system architects would prefer to have a linear address space for program and data memory. Programmers also find the linear solution more attractive. The overhead required for bank switching and chip selection usually requires additional overhead and hardware.

THE ULTIMATE SOLUTION BY MICROCHIP

Microchip Technology has designed an addressing scheme based on the standard I²C protocol and device addresses but incorporating an additional address byte for enabling the designer to use up to 256K bits per device and add from 1 to 8 devices on the system bus. This flexibility allows for future memory expansion and more advanced features in a smaller, more cost effective, design. This enhanced addressing combined with the many advanced and patent pending features of the 24XX65 make the 24XX65 an exciting and innovative device. It is the first in a family of sophisticated **Smart Serial** EEPROMs from Microchip Technology Inc.

24XX65 Features

The 24XX65 has an advanced architecture with the following features:

- 15-bit, 2-byte address field (14 bit for the 24XX32)
- 4K-bit High Endurance Block - 1 Million E/W cycles typical (Fixed at the last 4k bit block in the array)
- Programmable write protect security features with up to a 15 blocks of 4K bits
- 8 byte by 8 line input write cache for I²C Fast Mode, burst mode capability, and use as a capture buffer

24XX65 Addressing

For the first byte or control byte, the 24XX65 adheres to the I²C protocol (reference Figure 2). This is the first byte received following the start condition from the master device. The control byte consists of a four-bit control code for the 24XX65 (this is assigned as 1010 binary for read and write operations). The next three bits of the control byte are the device select bits (A2, A1, A0). They are used by the master device to select which of the eight devices are to be accessed. These bits are in effect the three most significant bits of the word address. The last bit of the control byte defines the operation to be performed. When set to a "1" a read operation is selected, when set to a "0" a write operation is selected. The least significant 13 bits of the next 2 bytes define the address of the first byte within the 8K block. The most significant byte is transferred first. Following the start condition, the 24XX65 monitors the SDA bus, checking the device type identifier being transmitted, upon receiving a 1010 code and appropriate device select bits, the slave device outputs an acknowledge signal on the SDA line. Depending on the state of the R/W bit, the 24XX65 will select a read or write operation.

The addressing scheme uses the standard first byte slave address format of the I²C standard with the Microchip Technology-assigned 1010 slave address. The internal bus controller scans the next byte for bit 7 to be asserted indicating that a security operation is to take place. This will be further explained later. The remainder of the byte is composed of the most significant address bits. The next byte is the least significant address byte. See Figure 2 for graphical representation of this sequence. After receiving a valid 2 byte address and a stop bit, the 24XX65 will process this address according to bit "0" of the control byte and either wait for data to be written, if in a write sequence, or present the requested data if in a read sequence.

ADVANCED FEATURES

Programmable security

The 24XX65 has a sophisticated security mechanism by which selected blocks of memory may be write protected by the user. The write sequence includes a bit for enabling the security protection scheme, bit 7 of the first address byte. When this bit is set to a one, the first byte following the address during a write sequence defines the security block. This includes a pointer to the starting 4K block to be protected (the write address), a write/erase flag, and a non-zero four bit code for determining the number of 4K blocks to protect up to the maximum of 15 (60K). The 4K blocks must be contiguous. The high endurance block cannot be protected. In a normal application the security block or blocks would be set after all code or look-up table data has been finalized. THIS OPERATION CAN ONLY BE PERFORMED ONCE. (See Figures 2 and 3.)

Total Endurance™

When defining endurance, we need to look at a few common definitions and possible misconceptions. Endurance with respect to EEPROMs is defined in number of Erase/Write (E/W) Cycles and is the most common rating referred to when discussing or specifying endurance. E/W ratings are based on the environmental and operating conditions of voltage, temperature, cycling mode and rate, for each byte in the application, not on the number of opcodes or control byte commands, and is never based on any read functions whether they be a data read or configuration read. If a part is rated at 100K E/W cycles, then each individual byte can be erased and written 100,000 times. This is probably the most common misinterpretation made by system designers. Endurance is thus an interactive application-specific reliability parameter. It is not a typical data sheet specification, such as a parametric AC/DC specification with benchmark standards for measurement. Microchip has done extensive predictive laboratory studies on Microchip 2- and 3-wire Serial EEPROMs. Applying the predictive data from the 24LC04B, which has similar characteristics, to the 4K high endurance block and assuming the following:

- a five-year life for a personal communication device
- an expected E/W cycles of 10 times per day
- a last number redial function of 11 bytes

Operational specifications:

Device	24XX65
Voltage	5
Temperature	25 C
Bytes/Cycle	11
E/W/Day	10
App. Life (Yr.)	5
Cycling Mode	BYTE
Data Pattern	RANDOM

The 4K HE block with 1 M E/W cycles typical, in this application, should yield the following results:

FIT	1.0
PPM	6
Time	5.0
Write cycles	18,250

FIGURE 2 - CONTROL BYTE ALLOCATION

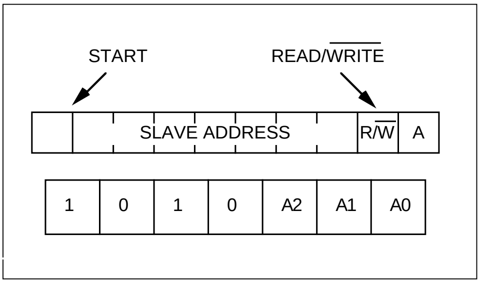
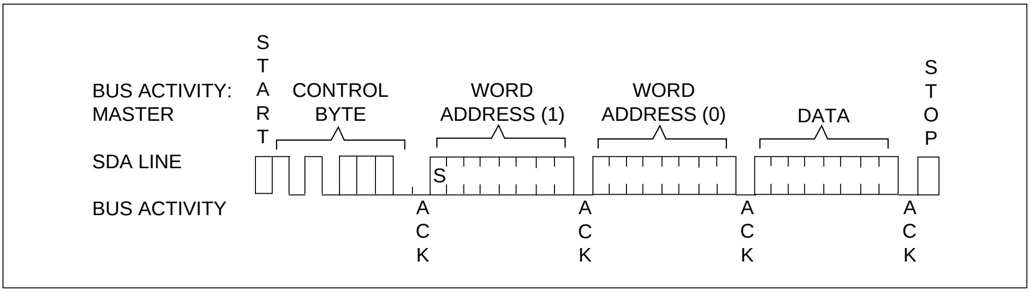
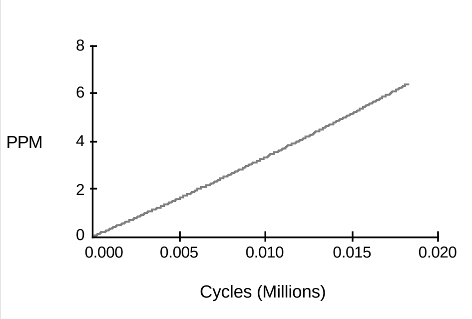


FIGURE 3 - BYTE WRITE



8

FIGURE 4 - PPM RATE IN MILLIONS OF CYCLES



The results shown are predictive in nature and should reflect an accurate representation of the expected results. For a more detailed description of endurance see the related application notes AN537 and AN562 contained in this volume. (See Figure 4.)

64 Byte Write Cache

The cache is arranged in 8-byte pages by 8 lines each. This yields a total of 64 bytes. The interface to the 24XX65 supports both the standard 100KHz mode and FAST mode at 400KHz. The input cache can therefore support a burst write option of up to 64 bytes. When using the I^2C protocol, an end of a page is defined by the transmission of a stop bit by the master. This sequence in the 24XX65 could be used to define pages from 8 to 64 bytes in length. The 8 byte by 8 line cache will roll over if a write is attempted past byte 8 of line 8, thus it can be used as a 64 byte capture or snapshot buffer. Each line is over written during a subsequent write to the same line. Faster memory and controller interfaces will become increasingly important in applications incorporating the ACCESS.bus™ interface standard.

Power Management

Increasingly, power management is becoming a predominant requirement for hand-held devices where a limited amount of power is available from the total power budget for a particular function. The 24XX65 has built-in power saving features such that the entire device is put into standby mode upon receiving a stop bit or an abort when in a read sequence, and after the completion of writing the data in the cache lines to the array when in an erase/write sequence. When the device is in standby mode, the only active circuit is the input circuit for the I^2C clock. This yields a standby current that is the current consumption of this lone input and the normal leakage current of the silicon and typically will be less than 2 µA.

The 24XX65 is the first device available in the Smart Serial product line. The features and capabilities initiated with this device will become standard on all future Microchip Technology Inc. serials and some features will be enhanced, such as the security options.

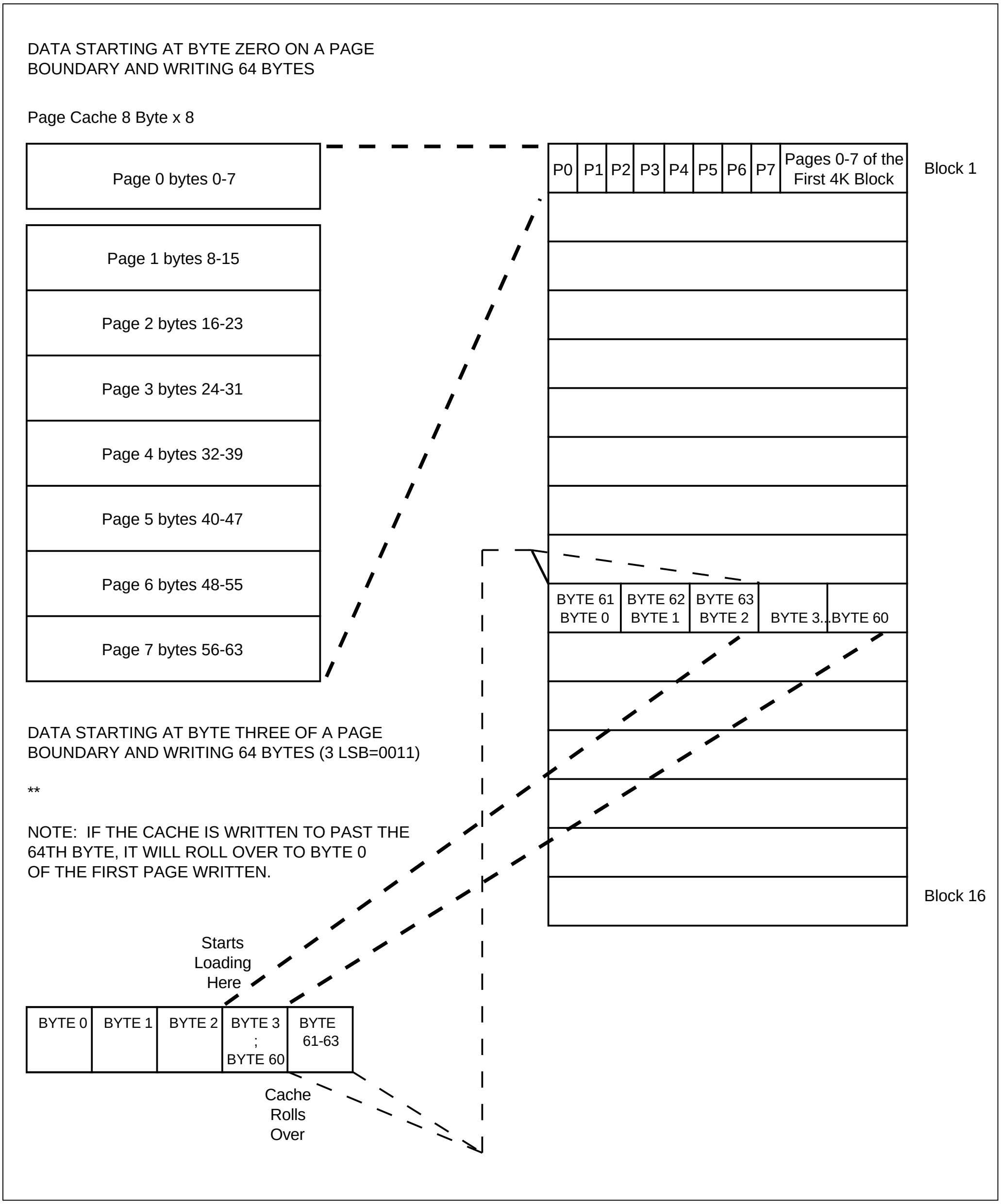
Appendix A of this application note contains the required PIC16C54 assembly routines for setting the security function, addressing the Smart Serial devices and using the most common addressing, read, write, and data polling functions for the I^2C bus.

All of the list programs are complete stand-alone programs.

Authors: *Richard J. Fisher*
Memory Products Division

Bruce Negley
Memory Products Division

FIGURE 5 - CACHE WRITE



APPENDIX A

```
16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:27 1994  Page 1

Line   PC    Opcode

                          LIST P=16C54,c=132
                   ;************************************************************
                   ;       64K Byte Read Program (138 bytes)
                   ;
                   ;       This program demonstrates how to interface a
                   ;       Microchip PIC16C54 to a 24XX32/65 Serial EE device
                   ;       and perform a random read operation. This program
                   ;       will read 8 consecutive addresses in the ‘random
                   ;       read’ mode.  This entails setting the address pointer
                   ;       before doing the read command for each address.
                   ;
                   ;       Another, more efficient method of reading consecutive
                   ;       addresses is the ‘sequential read’ mode.  This involves
                   ;       sending the control byte and address for the first byte
                   ;       to read, then continuing to provide clocks for the next
                   ;       addresses.  The device will automatically increment the
                   ;       address.  An example of the sequential read mode is
                   ;       provided in the ’64kseqr.asm’ file.
                   ;
                   ;       Timing is based on using the PIC16C54 in ‘XT’ mode
                   ;       using a 4Mhz crystal.  Clock speeds to the serial EE
                   ;       will be approximately 60 kHz for this setup.
                   ;
                   ;       As an option, the user may connect a LED to pin 18
                   ;       on the PIC16C54 (use about a 1K resistor in series) as an
                   ;       acknowledge fail indicator.  This LED will come on if
                   ;       the serial EE fails to acknowledge correctly.
                   ;
                   ;        PIC16C54 to Serial EE Connections:
                   ;
                   ;         PIC16C54       Serial EE
                   ;       ______    ______
                   ;       Pin 12 (RB6) –>  SCLK
                   ;       Pin 13 (RB7) –>  SDATA
                   ;
                   ;       PIN 18 (RA1) –>  Acknowledge fail LED  (Optional)
                   ;
                   ;
                   ;************************************************************
                   ;           Register Definitions
                   ;************************************************************
             0003  status  equ    03h     ; status register
             0005  port_a  equ    05h     ; port 5 (port a) used for LED display
             0006  port_b  equ    06h     ; port 6 (port b) used for data and
                                          ; clock lines
             000A  eeprom  equ    0ah     ; bit buffer
             000B  bycnt   equ    0bh     ; byte counter for read mode
             000C  addr    equ    0ch     ; word 0 address counter
             000D  datai   equ    0dh     ; data input register
             000E  datao   equ    0eh     ; data output register
             000F  slave   equ    0fh     ; device address 1010xxx0)

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:27 1994  Page 2

Line   PC    Opcode

             0010  txbuf   equ    10h     ; transmit buffer
             0011  count   equ    11h     ; bit counter
             0012  bcount  equ    12h     ; byte counter
             0015  loops   equ    15h     ; delay loop counter
             0016  loops2  equ    16h     ; delay loop counter
             0017  addr1   equ    17h     ; word 1 address counter
                   ;************************************************************
```

```
                ;           Bit Definitions
                ;************************************************************
        0007    di      equ    7       ; eeprom input bit
        0006    do      equ    6       ; eeprom output bit
        0007    sdata   equ    7       ; serial EE data line (port_b,pin 13)
        0006    sclk    equ    6       ; serial EE clock line (port_b,pin 12)
        0001    ackf    equ    1       ; acknowledge fail LED (port_a)
                ;************************************************************
                ;
        0000            org    01ffh   ; set reset vector
01FF    0A6B            goto   PWRUP
        0000            org    000h    ;
0000    0A6B            goto   PWRUP
                ;
                ;*************************************************************
                ;       DELAY ROUTINE
                ;       This routine takes the value in 'loops'
                ;       and multiplies it times 1 millisecond to
                ;       determine delay time.
                ;*************************************************************
                WAIT
                ;
0001    0C6E    top      movlw   .110           ; timing adjustment variable
0002    0036             movwf   loops2
0003    0000    top2     nop                    ; sit and wait
0004    0000             nop
0005    0000             nop
0006    0000             nop
0007    0000             nop
0008    0000             nop
0009    02F6             decfsz  loops2         ; inner loops complete?
000A    0A03             goto    top2           ; no, go again
                                                ;
000B    02F5             decfsz  loops          ; outer loops complete?
000C    0A01             goto    top            ; no, go again
000D    0800             retlw   0              ; yes, return from sub
                ;
                ;**************************************************************
                ;       Start Bit Subroutine
                ;       this routine generates a start bit
                ;       (Low going data line while clock is high)
                ;**************************************************************
                ;
                BSTART

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:27 1994  Page 3

Line   PC     Opcode

000E    05E6             bsf     port_b,sdata   ; make sure data is high
000F    0C3F             movlw   b'00111111'
0010    0006             tris    port_b         ; set data and clock lines for output
0011    04C6             bcf     port_b,sclk    ; make sure clock is low
0012    0000             nop
0013    05C6             bsf     port_b,sclk    ; set clock high
0014    0000             nop
0015    0000             nop
0016    0000             nop
0017    0000             nop
0018    0000             nop
0019    04E6             bcf     port_b,sdata   ; data line goes low during
                                                ; high clock for start bit
001A    0000             nop
001B    0000             nop
001C    0000             nop
001D    0000             nop
001E    0000             nop                    ; timing adjustment
001F    04C6             bcf     port_b,sclk    ; start clock train
0020    0000             nop
```

8

```
0021  0000           nop
0022  0800           retlw   0
                     ;
            ;************************************************************
            ;       Stop Bit Subroutine
            ;       This routine generates a stop bit
            ;       (High going data line while clock is high)
            ;************************************************************
            BSTOP
0023  04E6           bcf     port_b,sdata   ; make sure data line is low
0024  0C3F           movlw   b'00111111'    ;
0025  0006           tris    port_b         ; set data/clock lines as outputs
0026  04E6           bcf     port_b,sdata   ; make sure data line is low
0027  0000           nop
0028  0000           nop
0029  0000           nop
002A  05C6           bsf     port_b,sclk    ; set clock high
002B  0000           nop
002C  0000           nop
002D  0000           nop
002E  05E6           bsf     port_b,sdata   ; data goes high while clock high
                                            ; for stop bit
002F  0000           nop
0030  0000           nop
0031  04C6           bcf     port_b,sclk    ; set clock low again
0032  0000           nop
0033  0000           nop
0034  0000           nop
0035  0800           retlw   0
            ;
            ;*************************************************************

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:27 1994  Page 4

Line   PC    Opcode

            ;       BITOUT routine takes the bit of data in 'do' and
            ;       transmits it to the serial EE device
            ;*************************************************************
            BITOUT
0036  0C3F           movlw   b'00111111'    ; set data,clock as outputs
0037  0006           tris    port_b
0038  07CA           btfss   eeprom,do      ; check for state of data bit to xmit
0039  0A3C           goto    bitlow         ; low? go set data line low
003A  05E6           bsf     port_b,sdata   ; high? set data line high
003B  0A3D           goto    clkout         ; go toggle the clock

003C  04E6  bitlow   bcf     port_b,sdata   ; output a low bit
003D  05C6  clkout   bsf     port_b,sclk    ; set clock line high
003E  0000           nop
003F  0000           nop
0040  0000           nop
0041  0000           nop
0042  04C6           bcf     port_b,sclk    ; return clock line low
0043  0800           retlw   0
            ;
            ;**************************************************************
            ;       BITIN routine reads one bit of data from the
            ;       serial EE device and stores it in the bit 'di'
            ;**************************************************************
            BITIN
0044  05EA           bsf     eeprom,di      ; assume input bit is high
0045  0CBF           movlw   b'10111111'    ; make sdata an input line
0046  0006           tris    port_b
0047  05C6           bsf     port_b,sclk    ; set clock line high
0048  0000           nop                    ; just sit here a sec
0049  0000           nop
004A  0000           nop
004B  0000           nop
```

```
004C  0000          nop                     ;
004D  07E6          btfss   port_b,sdata    ; read the data bit
004E  04EA          bcf     eeprom,di       ; input bit was low, set 'di' accordingly
004F  04C6          bcf     port_b,sclk     ; set clock line low
                                            ;
0050  0800          retlw   0               ;
               ;
               ;****************************************************************
               ;       Transmit Data Subroutine
               ;       This routine takes the byte of data stored in the
               ;       'datao' register and transmits it to the serial EE device.
               ;       It will then send 1 more clock to the serial EE for the
               ;       acknowledge bit.  If the ack bit from the part was low
               ;       then the transmission was successful.  If it is high, then
               ;       the device did not send a proper ack bit and the ack
               ;       fail LED will be turned on.
               ;****************************************************************
               TX

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:27 1994  Page 5

Line   PC    Opcode

0051  0C08          movlw   .8
0052  0031          movwf   count           ; set the #bits to 8
                    ;
               TXLP
0053  04CA          bcf     eeprom,do       ; assume bit out is low
0054  06F0          btfsc   txbuf,7         ; is bit out really low?
0055  05CA          bsf     eeprom,do       ; no, set it high
0056  0936          call    BITOUT          ; send the bit to serial EE
0057  0370          rlf     txbuf           ; rotate txbuf left
0058  02F1          decfsz  count           ; 8 bits done?
0059  0A53          goto    TXLP            ; no - go again
005A  0944          call    BITIN           ; read ack bit
005B  06EA          btfsc   eeprom,di       ; check ack bit
005C  0525          bsf     port_a,ackf     ; set acknowledge fail LED if the
                                            ; device did not pull data low
                    ;
005D  0800          retlw   0
               ;
               ;****************************************************************
               ;       Receive data routine
               ;       This routine reads one byte of data from the part
               ;       into the 'datai' register.  It then sends a high
               ;       ack bit to indicate that no more data is to be read
               ;****************************************************************
               RX
005E  006D          clrf    datai           ; clear input buffer
005F  0C08          movlw   .8              ; set # bits to 8
0060  0031          movwf   count
0061  0403          bcf     status,0        ; make sure carry bit is low
0062  036D     RXLP rlf     datai           ; rotate datai 1 bit left
0063  0944          call    BITIN           ; read a bit
0064  06EA          btfsc   eeprom,di
0065  050D          bsf     datai,0         ; set bit 0 if necessary
0066  02F1          decfsz  count           ; 8 bits done?
0067  0A62          goto    RXLP            ; no, do another
0068  05CA          bsf     eeprom,do       ; set ack bit = 1
0069  0936          call    BITOUT          ; to finish transmission
006A  0800          retlw   0
               ;
               ;****************************************************************
               ;       Power up routine
               ;       This is the program entry point.  I/O line status is
               ;       set for port A here.
               ;****************************************************************
               PWRUP
006B  0C00           movlw  b'00000000'
```

8

```
006C  0005             tris   port_a        ; set port A as all output
006D  0065             clrf   port_a        ; all output lines low
006E  0A6F             goto   READ
            ;
            ;**************************************************************

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:27 1994  Page 6

PC    Opcode

            ;       READ (read routine)
            ;       This routine reads 8 consecutive addresses of the
            ;       serial EE device starting at address 00 in the
            ;       random access mode (non-sequential read). Reading
            ;       the device using the random access mode
            ;       requires that the address pointer be set for every
            ;       byte before the read takes place. The address pointer
            ;       is set by sending a ‘write mode’ control byte with the
            ;       address of the location to read.
            ;***************************************************************
            READ
            ;
006F  0425           bcf     port_a,ackf ; clear the ack fail LED if on
0070  0C08           movlw   .8          ; set number of bytes to read as 8
0071  0032           movwf   bcount      ;
0072  0077           clrf    addr1       ; set starting high address byte to 00
0073  006C           clrf    addr        ; set starting low address byte to 00
                                         ;
0074  090E  rbyte    call    BSTART      ; generate start bit
                                         ;
                                         ; now send the write control byte and
                                         ; address to set the pointer
                                         ;
0075  0CA0           movlw   b’10100000' ; get slave address (write mode)
0076  0030           movwf   txbuf       ; into transmit buffer
0077  0951           call    TX          ; and send it
0078  0217           movf    addr1,w     ; get word 1 address
0079  0030           movwf   txbuf       ; into transmit buffer
007A  0951           call    TX          ; and send it
007B  020C           movf    addr,w      ; get word 0 address
007C  0030           movwf   txbuf       ; into transmit buffer
007D  0951           call    TX          ; and send it
                                         ;
                                         ; now read one byte from the part
                                         ;
007E  090E           call    BSTART      ; generate start bit
007F  0CA1           movlw   b’10100001' ; get slave address and read mode
0080  0030           movwf   txbuf       ; into transmit buffer
0081  0951           call    TX          ; and transmit it
0082  095E           call    RX          ; read 1 byte from serial EE
0083  0923           call    BSTOP       ; send stop bit to end transmission
0084  02AC           incf    addr        ; add 1 to address counter
0085  02F2           decfsz  bcount      ; yes, are all 8 bytes read?
0086  0A74           goto    rbyte       ; no, do another byte

0087  0CFF           movlw   .255        ; long delay for scope
0088  0035           movwf   loops       ; trigger purposes only
0089  0901           call    wait
008A  0A6F           goto    READ        ; yes, start all over
            ;
      0000  END
```



```
16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:40 1994  Page 1

Line   PC    Opcode

                        LIST P=16C54,c=132
                  ;************************************************************
                  ;       64K Byte Write Program (127 bytes)
                  ;
                  ;       This program demonstrates how to interface a
                  ;       Microchip PIC16C54 to the 24XX32/65 Serial EE device
                  ;       and perform a byte write operation on 8 consecutive
                  ;       addresses.
                  ;
                  ;       After each byte is written, time must be given to the
                  ;       device for it to complete the write cycle before
                  ;       the next command can be sent.  The easiest solution
                  ;       is to consult the data book for the maximum write
                  ;       cycle time and just wait that long before the next
                  ;       command is sent(10ms for this device).  This program
                  ;       demonstrates that solution.
                  ;
                  ;       Another, more efficient method of determining when the
                  ;       write cycle is complete is called 'data polling.'  This
                  ;       method is demonstrated in the program "64kdpoll."
                  ;
                  ;       As an option, the user may connect a LED to pin 18
                  ;       on the PIC16C54 (use about a 1K resistor in series) as an
                  ;       acknowledge fail indicator.  This LED will come on if
                  ;       the serial EE fails to acknowledge correctly.
                  ;
                  ;       Timing is based on using the PIC16C54 in 'XT' mode
                  ;       using a 4MHz crystal.  Clock speeds to the serial EE
                  ;       will be approximately 60 kHz for this setup.
                  ;
                  ;        PIC16C54 to Serial EE Connections:
                  ;
                  ;        PIC16C54        Serial EE
                  ;       ______    ______
                  ;       Pin 12 (RB6) ->  SCLK
                  ;       Pin 13 (RB7) ->  SDATA
                  ;
                  ;       PIN 18 (RA1) ->  Acknowledge fail LED  (Optional)
                  ;
                  ;************************************************************
                  ;           Register Definitions
                  ;************************************************************
             0005  port_a  equ     5h     ; port 5 (port_a) used for LEDs
             0006  port_b  equ     6h     ; port 6 (port b) used for data and
                                          ; clock lines
             000A  eeprom  equ    0ah     ; bit buffer
             000B  bycnt   equ    0bh     ; byte counter for read mode
             000C  addr    equ    0ch     ; word 0 address counter
             000D  datai   equ    0dh     ; data input register
             000E  datao   equ    0eh     ; data output register
             000F  slave   equ    0fh     ; device address (1010xxx0)

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:40 1994  Page 2

Line   PC    Opcode

             0010  txbuf   equ    10h     ; transmit buffer
             0011  count   equ    11h     ; bit counter
             0012  bcount  equ    12h     ; byte counter
             0015  loops   equ    15h     ; delay loop counter
             0016  loops2  equ    16h     ; delay loop counter
             0017  addr1   equ    17h     ; word 1 address counter
                  ;************************************************************
                  ;           Bit Definitions
                  ;************************************************************
```

8

```
        0007  di      equ    7       ; eeprom input bit
        0006  do      equ    6       ; eeprom output bit
        0007  sdata   equ    7       ; serial EE data line (port_b,pin 13)
        0006  sclk    equ    6       ; serial EE clock line (port_b,pin 12)
        0001  ackf    equ    1       ; acknowledge fail LED (port_a,pin 18)
              ;************************************************************
              ;
              ;
        0000          org    01ffh   ; set reset vector
  01FF  0A5E          goto   PWRUP
        0000          org    000h    ;
  0000  0A5E          goto   PWRUP
              ;
              ;*************************************************************
              ;       DELAY ROUTINE
              ;       This routine takes the value in ‘loops’
              ;       and multiplies it times 1 millisecond to
              ;       determine delay time.
              ;*************************************************************
              WAIT
              ;
  0001  0C6E  top      movlw   .110           ; timing adjustment variable
  0002  0036           movwf   loops2
  0003  0000  top2     nop                    ; sit and wait
  0004  0000           nop
  0005  0000           nop
  0006  0000           nop
  0007  0000           nop
  0008  0000           nop
  0009  02F6           decfsz  loops2         ; inner loops complete?
  000A  0A03           goto    top2           ; no, go again
                                              ;
  000B  02F5           decfsz  loops          ; outer loops complete?
  000C  0A01           goto    top            ; no, go again
  000D  0800           retlw   0              ; yes, return from sub
              ;
              ;**************************************************************
              ;       Start Bit Subroutine
              ;       this routine generates a start bit
              ;       (Low going data line while clock is high)
              ;**************************************************************
              ;

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:40 1994  Page 3

Line   PC    Opcode

              BSTART
  000E  05E6           bsf     port_b,sdata   ; make sure data is high
  000F  0C3F           movlw   b’00111111'
  0010  0006           tris    port_b         ; set data and clock lines for output
  0011  04C6           bcf     port_b,sclk    ; make sure clock is low
  0012  0000           nop
  0013  05C6           bsf     port_b,sclk    ; set clock high
  0014  0000           nop
  0015  0000           nop
  0016  0000           nop
  0017  0000           nop
  0018  0000           nop
  0019  04E6           bcf     port_b,sdata   ; data line goes low during
                                              ; high clock for start bit
  001A  0000           nop
  001B  0000           nop
  001C  0000           nop
  001D  0000           nop
  001E  0000           nop                    ; timing adjustment
  001F  04C6           bcf     port_b,sclk    ; start clock train
  0020  0000           nop
  0021  0000           nop
```

```
0022  0800           retlw   0
                ;
                ;***********************************************************
                ;       Stop Bit Subroutine
                ;       This routine generates a stop bit
                ;       (High going data line while clock is high)
                ;***********************************************************
                BSTOP
0023  0C3F           movlw   b'00111111'    ;
0024  0006           tris    port_b         ; set data/clock lines as outputs
0025  04E6           bcf     port_b,sdata   ; make sure data line is low
0026  0000           nop
0027  0000           nop
0028  0000           nop
0029  05C6           bsf     port_b,sclk    ; set clock high
002A  0000           nop
002B  0000           nop
002C  0000           nop
002D  05E6           bsf     port_b,sdata   ; data goes high while clock high
                                            ; for stop bit
002E  0000           nop
002F  0000           nop
0030  04C6           bcf     port_b,sclk    ; set clock low again
0031  0000           nop
0032  0000           nop
0033  0000           nop
0034  0800           retlw   0
                ;
                ;************************************************************

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:40 1994  Page 4

Line   PC    Opcode

                ;       BITOUT routine takes one bit of data in 'do' and
                ;       transmits it to the serial EE device
                ;************************************************************
                BITOUT
0035  0C3F           movlw   b'00111111'    ; set data,clock as outputs
0036  0006           tris    port_b
0037  07CA           btfss   eeprom,do      ; check for state of data bit to xmit
0038  0A3B           goto    bitlow         ;
0039  05E6           bsf     port_b,sdata   ; set data line high
003A  0A3C           goto    clkout         ; go toggle the clock

003B  04E6      bitlow  bcf     port_b,sdata   ; output a low bit
003C  05C6      clkout  bsf     port_b,sclk    ; set clock line high
003D  0000           nop
003E  0000           nop
003F  0000           nop
0040  0000           nop
0041  04C6           bcf     port_b,sclk    ; return clock line low
0042  0800           retlw   0
                ;
                ;*************************************************************
                ;       BITIN routine reads one bit of data from the
                ;       serial EE device and stores it in 'di'
                ;*************************************************************
                BITIN
0043  05EA           bsf     eeprom,di      ; assume input bit is high
0044  0CBF           movlw   b'10111111'    ; make sdata an input line
0045  0006           tris    port_b
0046  05E6           bsf     port_b,sdata   ; set sdata line for input
0047  05C6           bsf     port_b,sclk    ; set clock line high
0048  0000           nop                    ; just sit here a sec
0049  0000           nop
004A  0000           nop
004B  0000           nop
004C  0000           nop                    ;
```

8

```
004D  07E6           btfss   port_b,sdata   ; read the data bit
004E  04EA           bcf     eeprom,di      ; input bit was low
004F  04C6           bcf     port_b,sclk    ; set clock line low
                                            ;
0050  0800           retlw   0              ;
               ;
               ;****************************************************************
               ;       Transmit Data Subroutine
               ;       This routine takes the byte of data stored in the
               ;       'datao' register and transmits it to the serial EE device.
               ;       It will then send 1 more clock to the serial EE for the
               ;       acknowledge bit.  If the ack bit from the part was low
               ;       then the transmission was successful.  If it is high, then
               ;       the device did not send a proper ack bit and the ack
               ;       fail LED will be turned on.
               ;****************************************************************

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:40 1994  Page 5

Line   PC    Opcode

               TX
0051  0C08           movlw   .8
0052  0031           movwf   count          ; set the #bits to 8
                     ;
               TXLP
0053  04CA           bcf     eeprom,do      ; assume bit out is low
0054  06F0           btfsc   txbuf,7        ; is bit out really low?
0055  05CA           bsf     eeprom,do      ; otherwise data bit =1
0056  0935           call    BITOUT         ; serial data out
0057  0370           rlf     txbuf          ; rotate txbuf left
0058  02F1           decfsz  count          ; 8 bits done?
0059  0A53           goto    TXLP           ; no - go again
005A  0943           call    BITIN          ; read ack bit
005B  06EA           btfsc   eeprom,di      ; check ack bit
005C  0525           bsf     port_a,ackf    ; set acknowledge fail LED if the
                     ;
005D  0800           retlw   0
               ;
               ;****************************************************************
               ;       Power up routine
               ;       This is the program entry point, which in this case simply
               ;       sets the port_a I/O lines and directs control to the
               ;       byte write routine.
               ;****************************************************************
               PWRUP
005E  0C00            movlw  b'00000000'
005F  0005            tris   port_a         ; set port A as all output
0060  0065            clrf   port_a         ; all output lines low
0061  0A62            goto   WRBYTE

               ;****************************************************************
               ;       Byte Write Routine
               ;       This routine writes the data in "datao" to
               ;       8 consecutive bytes in the serial EE device starting
               ;       at address 00.  This routine waits 10mS after every
               ;       byte to give the device time to do the write.  This
               ;       program repeats forever.
               ;*****************************************************************
               ;
               WRBYTE
               ;
0062  0065           clrf    port_a      ; clear all LEDs
0063  0CA0           movlw   b'10100000' ; set slave address and write mode
0064  002F           movwf   slave
0065  0C55           movlw   b'01010101' ; set data to write as 55h
0066  002E           movwf   datao
               ;
0067  0C08           movlw   .8          ; set number of bytes
```

```
       0068  0032            movwf    bcount       ; to write as to 8
       0069  0077            clrf     addr1        ; set high address byte to 00
       006A  006C            clrf     addr         ; set low address byte to 00

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:40 1994  Page 6

Line   PC    Opcode

                                                   ;
       006B  090E  byte      call     BSTART       ; generate start bit
       006C  020F            movf     slave,w      ; move slave address
       006D  0030            movwf    txbuf        ; into transmit buffer
       006E  0951            call     TX           ; and send it
       006F  0217            movf     addr1,w      ; move word 1 address
       0070  0030            movwf    txbuf        ; into transmit buffer
       0071  0951            call     TX           ; and send it
       0072  020C            movf     addr,w       ; move word 0 address
       0073  0030            movwf    txbuf        ; into transmit buffer
       0074  0951            call     TX           ; and send it
       0075  020E            movf     datao,w      ; move data byte
       0076  0030            movwf    txbuf        ; to transmit buffer
       0077  0951            call     TX           ; and transmit it
       0078  0923            call     BSTOP        ; generate stop bit
                                                   ;
       0079  0C0A            movlw    .10
       007A  0035            movwf    loops        ; set delay time to give
       007B  0901            call     WAIT         ; 10 ms wait after every byte
       007C  02AC            incf     addr         ; add 1 to low address counter
       007D  02F2            decfsz   bcount       ; all 8 bytes written?
       007E  0A6B            goto     byte         ; no, do another byte
                                                   ;
       007F  0A62            goto     wrbyte       ; start over
                   ;
                   ;
             0000  END
```

```
16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:54 1994  Page 1

Line   PC    Opcode

                          LIST P=16C54,c=132
                   ;*************************************************************
                   ;       64K Byte Write With Data Polling Program (133 bytes)
                   ;
                   ;       This program demonstrates how to interface a
                   ;       Microchip PIC16C54 to a 24XX32/65 Serial EE device.
                   ;       This program performs a byte write operation on 8
                   ;       consecutive addresses using the data polling method
                   ;       to determine when the write cycle is complete.
                   ;
                   ;       When writing to a serial E≤ device, there are 2
                   ;       ways to handle the internal timed write cycle
                   ;       time of the device.  The simplest method is
                   ;       simply to wait until the maximum cycle time
                   ;       is exceeded before attempting another command.
                   ;       The other, more efficient method is known as “data
                   ;       polling.”  Data polling is done by sending a start
                   ;       bit and control byte to the part after the write
                   ;       cycle has been initiated by a stop bit.  If the ack bit
                   ;       is low, then the device is through writing, otherwise
                   ;       the the sequence is repeated.  If no low acknowledge
                   ;       is found within 40 attempts (about 10 milliseconds)
                   ;       then the routine times out and sets the timeout
                   ;       LED (pin 1) high.
                   ;
                   ;       As an option, the user can connect a LED to pin 1
                   ;       on the PIC16C54 (use about a 1K resistor in series) as a
                   ;       timeout indicator.  This LED will come on if the
                   ;       data polling is unsuccessful and the device being
                   ;       programmed does not respond.  This can be tested by
                   ;       simply running the program with no part in the socket.
                   ;
                   ;       Timing is based on using the PIC16C54 in ‘XT’ mode
                   ;       using a 4MHz crystal.  Clock speeds to the serial EE
                   ;       will be approximately 40 kHz for this setup.
                   ;
                   ;        PIC16C54 to Serial EE Connections:
                   ;
                   ;        PIC16C54       Serial EE
                   ;       ______    ______
                   ;       Pin 12 (RB6) —>  SCLK
                   ;       Pin 13 (RB7) —>  SDATA
                   ;
                   ;       Pin 1  (RA2) —>  Write cycle timeout fail LED (optional)
                   ;
                   ;*************************************************************
                   ;           Register Definitions
                   ;*************************************************************
             0005  port_a  equ     5h     ; port 5 (port_a) used LEDs
             0006  port_b  equ     6h     ; port 6 (port_b) used for data and
                                          ; clock lines

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:54 1994  Page 2

Line   PC    Opcode

             000A  eeprom  equ    0ah     ; bit buffer
             000B  bycnt   equ    0bh     ; byte counter for read mode
             000C  addr    equ    0ch     ; address counter
             000D  datai   equ    0dh     ; data input register
             000E  datao   equ    0eh     ; data output register
             000F  slave   equ    0fh     ; device address (1010xxx0)
             0010  txbuf   equ    10h     ; transmit buffer
             0011  count   equ    11h     ; bit counter
             0012  bcount  equ    12h     ; byte counter
```

```
       0015  loops   equ    15h     ; delay loop counter
       0016  loops2  equ    16h     ; delay loop counter
       0017  pollcnt equ    17h     ; data polling counter
       0018  addr1   equ    18h     ; word 1 address counter
             ;************************************************************
             ;           Bit Definitions
             ;************************************************************
       0007  di      equ    7       ; eeprom input bit
       0006  do      equ    6       ; eeprom output bit
       0007  sdata   equ    7       ; serial EE data line (port_b,pin 13)
       0006  sclk    equ    6       ; serial EE clock line (port_b,pin 12)
       0002  timeout equ    2       ; write cycle timeout fail LED, port_a (pin 1)
       0001  ackf    equ    1       ; acknowledge fail LED, port_a (pin 18)
             ;************************************************************
             ;
       0000          org    01ffh   ; set reset vector
01FF   0A5C          goto   PWRUP
       0000          org    000h    ;
0000   0A5C          goto   PWRUP
             ;
             ;*************************************************************
             ;       DELAY ROUTINE
             ;       This routine takes the value in ‘loops’
             ;       and multiplies it times 1 millisecond to
             ;       determine delay time.
             ;*************************************************************
             WAIT
             ;
0001   0C6E  top      movlw   .110           ; timing adjustment variable
0002   0036           movwf   loops2
0003   0000  top2     nop                    ; sit and wait
0004   0000           nop
0005   0000           nop
0006   0000           nop
0007   0000           nop
0008   0000           nop
0009   02F6           decfsz  loops2         ; inner loops complete?
000A   0A03           goto    top2           ; no, go again
                                             ;
000B   02F5           decfsz  loops          ; outer loops complete?
000C   0A01           goto    top            ; no, go again
000D   0800           retlw   0              ; yes, return from sub

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:54 1994  Page 3

Line   PC    Opcode

             ;
             ;**************************************************************
             ;       Start Bit Subroutine
             ;       this routine generates a start bit
             ;       (Low going data line while clock is high)
             ;**************************************************************
             ;
             BSTART
000E   05E6           bsf     port_b,sdata   ; make sure data is high
000F   0C3F           movlw   b’00111111'
0010   0006           tris    port_b         ; set data and clock lines for output
0011   04C6           bcf     port_b,sclk    ; make sure clock is low
0012   0000           nop
0013   05C6           bsf     port_b,sclk    ; set clock high
0014   0000           nop
0015   0000           nop
0016   0000           nop
0017   0000           nop
0018   0000           nop
0019   04E6           bcf     port_b,sdata   ; data line goes low during
                                             ; high clock for start bit
001A   0000           nop
```

```
       001B  0000           nop
       001C  0000           nop
       001D  0000           nop
       001E  0000           nop                     ; timing adjustment
       001F  04C6           bcf     port_b,sclk     ; start clock train
       0020  0000           nop
       0021  0000           nop
       0022  0800           retlw   0
                            ;
                   ;************************************************************
                   ;       Stop Bit Subroutine
                   ;       This routine generates a stop bit
                   ;       (High going data line while clock is high)
                   ;************************************************************
                   BSTOP
       0023  0C3F           movlw   b'00111111'     ;
       0024  0006           tris    port_b          ; set data/clock lines as outputs
       0025  04E6           bcf     port_b,sdata    ; make sure data line is low
       0026  0000           nop
       0027  0000           nop
       0028  0000           nop
       0029  05C6           bsf     port_b,sclk     ; set clock high
       002A  0000           nop
       002B  0000           nop
       002C  0000           nop
       002D  05E6           bsf     port_b,sdata    ; data goes high while clock high
                                                    ; for stop bit
       002E  0000           nop
       002F  0000           nop

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:54 1994  Page 4

Line   PC    Opcode

       0030  04C6           bcf     port_b,sclk     ; set clock low again
       0031  0000           nop
       0032  0000           nop
       0033  0000           nop
       0034  0800           retlw   0
                   ;
                   ;*************************************************************
                   ;       BITOUT routine takes one bit of data in 'do' and
                   ;       transmits it to the serial EE device
                   ;*************************************************************
                   BITOUT
       0035  0C3F           movlw   b'00111111'     ; set data,clock as outputs
       0036  0006           tris    port_b
       0037  07CA           btfss   eeprom,do       ; check for state of data bit to xmit
       0038  0A3B           goto    bitlow          ;
       0039  05E6           bsf     port_b,sdata    ; set data line high
       003A  0A3C           goto    clkout          ; go toggle the clock
                                                    ;
       003B  04E6  bitlow   bcf     port_b,sdata    ; output a low bit
       003C  05C6  clkout   bsf     port_b,sclk     ; set clock line high
       003D  0000           nop
       003E  0000           nop
       003F  0000           nop
       0040  0000           nop
       0041  04C6           bcf     port_b,sclk     ; return clock line low
       0042  0800           retlw   0
                   ;
                   ;                  End of Subroutine
                   ;
                   ;**************************************************************
                   ;       BITIN routine reads one bit of data from the
                   ;       serial EE device and stores it in 'di'
                   ;**************************************************************
                   BITIN
       0043  05EA           bsf     eeprom,di       ; assume input bit is high
```

```
0044  0CBF            movlw   b'10111111'    ; make sdata an input line
0045  0006            tris    port_b
0046  05E6            bsf     port_b,sdata   ; set sdata line for input
0047  05C6            bsf     port_b,sclk    ; set clock line high
0048  0000            nop                    ; just sit here a sec
0049  0000            nop
004A  0000            nop
004B  0000            nop
004C  0000            nop                    ;
004D  07E6            btfss   port_b,sdata   ; read the data bit
004E  04EA            bcf     eeprom,di      ; input bit was low
004F  04C6            bcf     port_b,sclk    ; set clock line low
                                             ;
0050  0800            retlw   0              ;
                  ;
                  ;*****************************************************************

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:54 1994  Page 5

Line   PC    Opcode

                  ;       Transmit Data Subroutine
                  ;       This routine takes the byte of data stored in the
                  ;       'datao' register and transmits it to the serial EE device.
                  ;       It will then send 1 more clock to the serial EE for the
                  ;       acknowledge bit.  If the ack bit from the part was low
                  ;       then the transmission was successful.  If it is high, then
                  ;       the device did not send a proper ack bit and the ack
                  ;       fail LED will be turned on.
                  ;*****************************************************************
                  TX
0051  0C08            movlw   .8
0052  0031            movwf   count          ; set the #bits to 8
                      ;
                  TXLP
0053  04CA            bcf     eeprom,do      ; assume bit out is low
0054  06F0            btfsc   txbuf,7        ; is bit out really low?
0055  05CA            bsf     eeprom,do      ; otherwise data bit =1
0056  0935            call    BITOUT         ; serial data out
0057  0370            rlf     txbuf          ; rotate txbuf left
0058  02F1            decfsz  count          ; 8 bits done?
0059  0A53            goto    TXLP           ; no - go again
005A  0943            call    BITIN          ; read ack bit
                      ;
005B  0800            retlw   0
                  ;                    ;
                  ;*****************************************************************
                  ;       Power up routine
                  ;       This is the program entry point, which in this case simply
                  ;       sets the port_a I/O lines and directs control to the
                  ;       write routine.
                  ;*****************************************************************
                  PWRUP
005C  0C00             movlw  b'00000000'
005D  0005             tris   port_a         ; set port A as all output
005E  0065             clrf   port_a         ; all output lines low
005F  0A60             goto   WRBYTE

                  ;*****************************************************************
                  ;       Byte Write Routine with data polling technique
                  ;
                  ;       This routine writes the data in "datao" to
                  ;       8 consecutive bytes in the serial EE device starting
                  ;       at address 00. To determine when the write cycle time
                  ;       of the device, the 'data polling' method is used.  This
                  ;       involves sending a start bit and control byte to the part
                  ;       and checking for an acknowledge.  If the ack bit is low, then
                  ;       the device is through writing, otherwise the the sequence
                  ;       is repeated.  If no low acknowledge is found within 40
```

8

```
                  ;       attempts (about 10 milliseconds) then the routine times
                  ;       out and sets the timeout LED (pin 18) high.  This program
                  ;       will repeat forever.

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:54 1994  Page 6

Line   PC    Opcode

                  ;
                  ;*****************************************************************
                  ;
                  WRBYTE
                  ;
       0060  0065          clrf    port_a          ; all LEDs off
       0061  0CA0          movlw   b'10100000'     ; set slave address and write mode
       0062  002F          movwf   slave
       0063  0CAA          movlw   b'10101010'     ; set data to write as AAh
       0064  002E          movwf   datao
       0065  0C08          movlw   .8              ; set number of bytes
       0066  0032          movwf   bcount          ; to write as to 8
       0067  0078          clrf    addr1           ; set starting high address to 00
       0068  006C          clrf    addr            ; set starting low address to 00
                                                   ;
       0069  090E  byte    call    BSTART          ; generate start bit
       006A  020F          movf    slave,w         ; move slave address
       006B  0030          movwf   txbuf           ; into transmit buffer
       006C  0951          call    TX              ; and send it
       006D  0218          movf    addr1,w         ; move word 1 address
       006E  0030          movwf   txbuf           ; into transmit buffer
       006F  0951          call    TX              ; and send it
       0070  020C          movf    addr,w          ; move word 0 address
       0071  0030          movwf   txbuf           ; into transmit buffer
       0072  0951          call    TX              ; and send it
       0073  020E          movf    datao,w         ; move data byte
       0074  0030          movwf   txbuf           ; to tranmit buffer
       0075  0951          call    TX              ; and transmit it
       0076  0923          call    BSTOP           ; generate stop bit
                                                   ;
                                                   ; now start polling for a low ack bit
                                                   ;
       0077  0C28          movlw   .40             ;
       0078  0037          movwf   pollcnt         ; set max number of times to poll as 40
       0079  090E  poll    call    BSTART          ; generate start bit
       007A  0CA0          movlw   b'10100000'     ; move slave address (write mode)
       007B  0030          movwf   txbuf           ; into transmit buffer
       007C  0951          call    TX              ; and send it
       007D  07EA          btfss   eeprom,di       ; was the ack bit low?
       007E  0A82          goto    exitpoll        ; yes, do another byte
       007F  02F7          decfsz  pollcnt         ; is poll counter down to zero?
       0080  0A79          goto    poll            ; no, poll again.  Other wise the part is
       0081  0545          bsf     port_a,timeout  ; not responding in time so set timeout
                                                   ; LED and continue on
                                                   ;
       0082  02AC  exitpoll incf   addr            ; add 1 to address counter
       0083  02F2          decfsz  bcount          ; all 8 bytes written?
       0084  0A69          goto    byte            ; no, do another byte
       0085  0A60          goto    WRBYTE          ; yes, start over
                  ;
                  ;

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:54 1994  Page 7

Line   PC    Opcode

             0000  END
```

```
16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:48 1994  Page 1

Line   PC    Opcode

                            LIST P=16C54,c=132
                   ;************************************************************
                   ;       64K Page Write Program (126 bytes)
                   ;
                   ;       The 24XX32/65 has a page length of 8 bytes.  This page
                   ;       can be used to write up to 8 bytes of data into the
                   ;       part before initiating the write cycle.  Since the
                   ;       write cycle is timed the same for 1 byte or 8 bytes
                   ;       it is more efficient to use the page mode when
                   ;       consecutive addresses are being written to.
                   ;
                   ;       When using page mode, the control byte, upper and lower
                   ;       addresses are sent for the first address only. After the
                   ;       data byte for the first address is sent, the data for the
                   ;       next consecutive address is clocked in.  This is
                   ;       repeated as many times as needed (as long as the page
                   ;       length is not exceeded) and then a stop bit is sent.
                   ;       The device will still acknowledge between every byte of
                   ;       data.  After the stop bit is sent, the part will
                   ;       initiate the self timed write cycle.
                   ;
                   ;       This routine waits approximately 10mS for the write
                   ;       cycle to complete for each page.  A more efficient
                   ;       method of determining when the write cycle is complete
                   ;       is called “data polling.”  The data polling method is
                   ;       explained in the program “64kdpoll.asm.”  That
                   ;       particular program uses data polling in a byte write
                   ;       mode but it can be used in exactly the same way for
                   ;       a page mode write.
                   ;
                   ;       As an option, the user can connect a LED to pin 18
                   ;       on the PIC16C54 (use about a 1K resistor in series) as a
                   ;       acknowledge fail indicator.  This LED will come on
                   ;       if the device being programmed does not send a low
                   ;       acknowledge bit at the proper times.  This can be
                   ;       tested by simply running the program with no part
                   ;       in the socket.
                   ;
                   ;       Timing is based on using the PIC16C54 in ‘XT’ mode
                   ;       using a 4MHz crystal.  Clock speeds to the serial EE
                   ;       will be approximately 40 kHz for this setup.
                   ;
                   ;
                   ;        PIC16C54 to Serial EE Connections:
                   ;
                   ;        PIC16C54        Serial EE
                   ;       ______    ______
                   ;       Pin 12 (RB6) —>  SCLK
                   ;       Pin 13 (RB7) —>  SDATA
                   ;
                   ;       PIN 18 (RA1) —>  Acknowledge fail LED  (Optional)

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:48 1994  Page 2

Line   PC    Opcode

                   ;
                   ;
                   ;************************************************************
                   ;           Register Definitions
                   ;************************************************************
             0005  port_a  equ     5h     ; port 5 (port_a) used for LEDs
             0006  port_b  equ     6h     ; port 6 (port b) used for data and
                                          ; clock lines
             000A  eeprom  equ    0ah     ; bit buffer
```

8

```
0061        000B  bycnt   equ    0bh     ; byte counter for read mode
0062        000C  addr    equ    0ch     ; word 0 address counter
0063        000D  datai   equ    0dh     ; data input register
0064        000E  datao   equ    0eh     ; data output register
0065        000F  slave   equ    0fh     ; device address (1010xxx0)
0066        0010  txbuf   equ    10h     ; transmit buffer
0067        0011  count   equ    11h     ; bit counter
0068        0012  bcount  equ    12h     ; byte counter
0069        0015  loops   equ    15h     ; delay loop counter
0070        0016  loops2  equ    16h     ; delay loop counter
0071        0017  addr1   equ    17h     ; word 1 address counter
0072              ;************************************************************
0073              ;           Bit Definitions
0074              ;************************************************************
0075        0007  di      equ    7       ; eeprom input bit
0076        0006  do      equ    6       ; eeprom output bit
0077        0007  sdata   equ    7       ; serial EE data line (port_b,pin 13)
0078        0006  sclk    equ    6       ; serial EE clock line (port_b,pin 12)
0079        0001  ackf    equ    1       ; acknowledge fail LED (port_a,pin 18)
0080              ;************************************************************
0081              ;
0082              ;
0083        0000          org    01ffh   ; set reset vector
0084  01FF  0A5E          goto   PWRUP
0085        0000          org    000h    ;
0086  0000  0A5E          goto   PWRUP
0087              ;
0088              ;*************************************************************
0089              ;       DELAY ROUTINE
0090              ;       This routine takes the value in ‘loops’
0091              ;       and multiplies it times 1 millisecond to
0092              ;       determine delay time.
0093              ;*************************************************************
0094              WAIT
0095              ;
0096  0001  0C6E  top      movlw   .110           ; timing adjustment variable
0097  0002  0036           movwf   loops2
0098  0003  0000  top2     nop                    ; sit and wait
0099  0004  0000           nop
0100  0005  0000           nop
0101  0006  0000           nop
0102  0007  0000           nop

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:48 1994  Page 3

Line  PC    Opcode

0103  0008  0000           nop
0104  0009  02F6           decfsz  loops2         ; inner loops complete?
0105  000A  0A03           goto    top2           ; no, go again
0106                                              ;
0107  000B  02F5           decfsz  loops          ; outer loops complete?
0108  000C  0A01           goto    top            ; no, go again
0109  000D  0800           retlw   0              ; yes, return from sub
0110              ;
0111              ;**************************************************************
0112              ;       Start Bit Subroutine
0113              ;       this routine generates a start bit
0114              ;       (Low going data line while clock is high)
0115              ;**************************************************************
0116              ;
0117              BSTART
0118  000E  05E6           bsf     port_b,sdata   ; make sure data is high
0119  000F  0C3F           movlw   b’00111111'
0120  0010  0006           tris    port_b         ; set data and clock lines for output
0121  0011  04C6           bcf     port_b,sclk    ; make sure clock is low
0122  0012  0000           nop
0123  0013  05C6           bsf     port_b,sclk    ; set clock high
0124  0014  0000           nop
```

```
0015  0000              nop
0016  0000              nop
0017  0000              nop
0018  0000              nop
0019  04E6              bcf     port_b,sdata   ; data line goes low during
                                               ; high clock for start bit
001A  0000              nop
001B  0000              nop
001C  0000              nop
001D  0000              nop
001E  0000              nop                    ; timing adjustment
001F  04C6              bcf     port_b,sclk    ; start clock train
0020  0000              nop
0021  0000              nop
0022  0800              retlw   0
                ;
                ;************************************************************
                ;       Stop Bit Subroutine
                ;       This routine generates a stop bit
                ;       (High going data line while clock is high)
                ;************************************************************
                BSTOP
0023  0C3F              movlw   b'00111111'    ;
0024  0006              tris    port_b         ; set data/clock lines as outputs
0025  04E6              bcf     port_b,sdata   ; make sure data line is low
0026  0000              nop
0027  0000              nop
0028  0000              nop
0029  05C6              bsf     port_b,sclk    ; set clock high

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:48 1994  Page 4

Line   PC    Opcode

002A  0000              nop
002B  0000              nop
002C  0000              nop
002D  05E6              bsf     port_b,sdata   ; data goes high while clock high
                                               ; for stop bit
002E  0000              nop
002F  0000              nop
0030  04C6              bcf     port_b,sclk    ; set clock low again
0031  0000              nop
0032  0000              nop
0033  0000              nop
0034  0800              retlw   0
                ;
                ;*************************************************************
                ;       BITOUT routine takes one bit of data in 'do' and
                ;       transmits it to the serial EE device
                ;*************************************************************
                BITOUT
0035  0C3F              movlw   b'00111111'    ; set data,clock as outputs
0036  0006              tris    port_b
0037  07CA              btfss   eeprom,do      ; check for state of data bit to xmit
0038  0A3B              goto    bitlow         ;
0039  05E6              bsf     port_b,sdata   ; set data line high
003A  0A3C              goto    clkout         ; go toggle the clock

003B  04E6      bitlow  bcf     port_b,sdata   ; output a low bit
003C  05C6      clkout  bsf     port_b,sclk    ; set clock line high
003D  0000              nop
003E  0000              nop
003F  0000              nop
0040  0000              nop
0041  04C6              bcf     port_b,sclk    ; return clock line low
0042  0800              retlw   0
                ;
                ;**************************************************************
```

8

```
                  ;       BITIN routine reads one bit of data from the
                  ;       serial EE device and stores it in 'di'
                  ;**************************************************************
                  BITIN
0043  05EA                 bsf     eeprom,di      ; assume input bit is high
0044  0CBF                 movlw   b'10111111'    ; make sdata an input line
0045  0006                 tris    port_b
0046  05E6                 bsf     port_b,sdata   ; set sdata line for input
0047  05C6                 bsf     port_b,sclk    ; set clock line high
0048  0000                 nop                    ; just sit here a sec
0049  0000                 nop
004A  0000                 nop
004B  0000                 nop
004C  0000                 nop                    ;
004D  07E6                 btfss   port_b,sdata   ; read the data bit
004E  04EA                 bcf     eeprom,di      ; input bit was low

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:48 1994  Page 5

Line   PC    Opcode

004F  04C6                 bcf     port_b,sclk    ; set clock line low
                                                  ;
0050  0800                 retlw   0              ;
                  ;
                  ;***************************************************************
                  ;       Transmit Data Subroutine
                  ;       This routine takes the byte of data stored in the
                  ;       'datao' register and transmits it to the serial EE device.
                  ;       It will then send 1 more clock to the serial EE for the
                  ;       acknowledge bit.  If the ack bit from the part was low
                  ;       then the transmission was successful.  If it is high, then
                  ;       the device did not send a proper ack bit and the ack
                  ;       fail LED will be turned on.
                  ;***************************************************************
                  TX
0051  0C08                 movlw   .8
0052  0031                 movwf   count          ; set the #bits to 8
                           ;
                  TXLP
0053  04CA                 bcf     eeprom,do      ; assume bit out is low
0054  06F0                 btfsc   txbuf,7        ; is bit out really low?
0055  05CA                 bsf     eeprom,do      ; otherwise data bit =1
0056  0935                 call    BITOUT         ; serial data out
0057  0370                 rlf     txbuf          ; rotate txbuf left
0058  02F1                 decfsz  count          ; 8 bits done?
0059  0A53                 goto    TXLP           ; no - go again
005A  0943                 call    BITIN          ; read ack bit
005B  06EA                 btfsc   eeprom,di      ; check ack bit
005C  0525                 bsf     port_a,ackf    ; set acknowledge fail LED if the
                           ;
005D  0800                 retlw   0
                  ;
                  ;***************************************************************
                  ;       Power up routine
                  ;       This is the program entry point, which in this case simply
                  ;       sets the port_a I/O lines and directs control to the
                  ;       byte write routine.
                  ;***************************************************************
                  PWRUP
005E  0C00                  movlw  b'00000000'
005F  0005                  tris   port_a         ; set port A as all output
0060  0065                  clrf   port_a         ; all output lines low
0061  0A62                  goto   pwrite         ; go do the page write

                  ;***************************************************************
                  ;       Page Write Routine
                  ;       This routine writes the data in "datao" to 8 consecutive
                  ;       bytes using page write mode starting at address 00.
```

```
                ;       This routine waits 10mS after every page to give the device
                ;       time to do the write.  This program repeats forever.
                ;*****************************************************************

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:48 1994  Page 6

Line   PC    Opcode

                ;
                pwrite
                ;
       0062  0065         clrf    port_a      ; clear all LEDs
       0063  0CA0         movlw   b'10100000' ; set slave address and write mode
       0064  002F         movwf   slave
       0065  0C55         movlw   b'01010101' ; set data to write as 55h
       0066  002E         movwf   datao
                ;
       0067  0C08         movlw   .8          ; set number of bytes
       0068  0032         movwf   bcount      ; to write as to 8
       0069  0077         clrf    addr1       ; set high address byte to 00
       006A  006C         clrf    addr        ; set low address byte to 00
                                              ;
       006B  090E         call    BSTART      ; generate start bit
       006C  020F         movf    slave,w     ; move slave address
       006D  0030         movwf   txbuf       ; into transmit buffer
       006E  0951         call    TX          ; and send it
       006F  0217         movf    addr1,w     ; move word 1 address
       0070  0030         movwf   txbuf       ; into transmit buffer
       0071  0951         call    TX          ; and send it
       0072  020C         movf    addr,w      ; move word 0 address
       0073  0030         movwf   txbuf       ; into transmit buffer
       0074  0951         call    TX          ; and send it

       0075  020E  byte   movf    datao,w     ; move data byte
       0076  0030         movwf   txbuf       ; to transmit buffer
       0077  0951         call    TX          ; and transmit it
       0078  02F2         decfsz  bcount      ; all 8 bytes written?
       0079  0A75         goto    byte        ; no, do another byte
                                              ;
       007A  0923         call    BSTOP       ; generate stop bit
       007B  0C0A         movlw   .10
       007C  0035         movwf   loops       ; set delay time to give
       007D  0901         call    WAIT        ; 10 ms wait after every byte
       007E  0A62         goto    pwrite      ; start over
                ;
                ;
             0000  END
```

```
16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:33 1994  Page 1

Line   PC    Opcode

                            LIST P=16C54,c=132
                   ;************************************************************
                   ;       64K Sequential Read Program (142 bytes)
                   ;
                   ;       This program demonstrates how to interface a
                   ;       Microchip PIC16C54 to a 24XX32/65 Serial EE device
                   ;       and perform a sequential read operation. This program
                   ;       will read 8 consecutive addresses in the 'sequential
                   ;       read' mode.  This entails setting the address pointer
                   ;       for the first address only and then clocking out as many
                   ;       bytes of data as needed.  The device will automatically
                   ;       increment the address.
                   ;
                   ;       Timing is based on using the PIC16C54 in 'XT' mode
                   ;       using a 4Mhz crystal.  Clock speeds to the serial EE
                   ;       will be approximately 60 kHz for this setup.
                   ;
                   ;       As an option, the user may connect a LED to pin 18
                   ;       on the PIC16C54 (use about a 1K resistor in series) as an
                   ;       acknowledge fail indicator.  This LED will come on if
                   ;       the serial EE fails to acknowledge correctly.
                   ;
                   ;        PIC16C54 to Serial EE Connections:
                   ;
                   ;        PIC16C54        Serial EE
                   ;       ______    ______
                   ;       Pin 12 (RB6) —>  SCLK
                   ;       Pin 13 (RB7) —>  SDATA
                   ;
                   ;       PIN 18 (RA1) —>  Acknowledge fail LED  (Optional)
                   ;
                   ;
                   ;************************************************************
                   ;           Register Definitions
                   ;************************************************************
             0003  status  equ    03h     ; status register
             0005  port_a  equ    05h     ; port 5 (port a) used for LED display
             0006  port_b  equ    06h     ; port 6 (port b) used for data and
                                          ; clock lines
             000A  eeprom  equ    0ah     ; bit buffer
             000B  bycnt   equ    0bh     ; byte counter for read mode
             000C  addr    equ    0ch     ; word 0 address counter
             000D  datai   equ    0dh     ; data input register
             000E  datao   equ    0eh     ; data output register
             000F  slave   equ    0fh     ; device address 1010xxx0)
             0010  txbuf   equ    10h     ; transmit buffer
             0011  count   equ    11h     ; bit counter
             0012  bcount  equ    12h     ; byte counter
             0015  loops   equ    15h     ; delay loop counter
             0016  loops2  equ    16h     ; delay loop counter
             0017  addr1   equ    17h     ; word 1 address counter

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:33 1994  Page 2

Line   PC    Opcode

                   ;************************************************************
                   ;           Bit Definitions
                   ;************************************************************
             0007  di      equ    7       ; eeprom input bit
             0006  do      equ    6       ; eeprom output bit
             0007  sdata   equ    7       ; serial EE data line (port_b,pin 13)
             0006  sclk    equ    6       ; serial EE clock line (port_b,pin 12)
             0001  ackf    equ    1       ; acknowledge fail LED (port_a)
                   ;************************************************************
```

```
0061                   ;
0062         0000              org    01ffh   ; set reset vector
0063   01FF  0A69              goto   PWRUP
0064         0000              org    000h    ;
0065   0000  0A69              goto   PWRUP
0066                   ;
0067                   ;*************************************************************
0068                   ;       DELAY ROUTINE
0069                   ;       This routine takes the value in 'loops'
0070                   ;       and multiplies it times 1 millisecond to
0071                   ;       determine delay time.
0072                   ;*************************************************************
0073                   WAIT
0074                   ;
0075   0001  0C6E  top      movlw   .110           ; timing adjustment variable
0076   0002  0036           movwf   loops2
0077   0003  0000  top2     nop                    ; sit and wait
0078   0004  0000           nop
0079   0005  0000           nop
0080   0006  0000           nop
0081   0007  0000           nop
0082   0008  0000           nop
0083   0009  02F6           decfsz  loops2         ; inner loops complete?
0084   000A  0A03           goto    top2           ; no, go again
0085                                               ;
0086   000B  02F5           decfsz  loops          ; outer loops complete?
0087   000C  0A01           goto    top            ; no, go again
0088   000D  0800           retlw   0              ; yes, return from sub
0089                   ;
0090                   ;**************************************************************
0091                   ;       Start Bit Subroutine
0092                   ;       this routine generates a start bit
0093                   ;       (Low going data line while clock is high)
0094                   ;**************************************************************
0095                   ;
0096                   BSTART
0097   000E  05E6           bsf     port_b,sdata   ; make sure data is high
0098   000F  0C3F           movlw   b'00111111'
0099   0010  0006           tris    port_b         ; set data and clock lines for output
0100   0011  04C6           bcf     port_b,sclk    ; make sure clock is low
0101   0012  0000           nop
0102   0013  05C6           bsf     port_b,sclk    ; set clock high

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:33 1994  Page 3

Line   PC    Opcode

0103   0014  0000           nop
0104   0015  0000           nop
0105   0016  0000           nop
0106   0017  0000           nop
0107   0018  0000           nop
0108   0019  04E6           bcf     port_b,sdata   ; data line goes low during
0109                                               ; high clock for start bit
0110   001A  0000           nop
0111   001B  0000           nop
0112   001C  0000           nop
0113   001D  0000           nop
0114   001E  0000           nop                    ; timing adjustment
0115   001F  04C6           bcf     port_b,sclk    ; start clock train
0116   0020  0000           nop
0117   0021  0000           nop
0118   0022  0800           retlw   0
0119                        ;
0120                   ;************************************************************
0121                   ;       Stop Bit Subroutine
0122                   ;       This routine generates a stop bit
0123                   ;       (High going data line while clock is high)
0124                   ;************************************************************
```

8

```
                BSTOP
0023  04E6              bcf     port_b,sdata   ; make sure data line is low
0024  0C3F              movlw   b'00111111'    ;
0025  0006              tris    port_b         ; set data/clock lines as outputs
0026  04E6              bcf     port_b,sdata   ; make sure data line is low
0027  0000              nop
0028  0000              nop
0029  0000              nop
002A  05C6              bsf     port_b,sclk    ; set clock high
002B  0000              nop
002C  0000              nop
002D  0000              nop
002E  05E6              bsf     port_b,sdata   ; data goes high while clock high
                                               ; for stop bit
002F  0000              nop
0030  0000              nop
0031  04C6              bcf     port_b,sclk    ; set clock low again
0032  0000              nop
0033  0000              nop
0034  0000              nop
0035  0800              retlw   0
                ;
                ;************************************************************
                ;       BITOUT routine takes the bit of data in 'do' and
                ;       transmits it to the serial EE device
                ;************************************************************
                BITOUT
0036  0C3F              movlw   b'00111111'    ; set data,clock as outputs
0037  0006              tris    port_b

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:33 1994  Page 4

Line   PC    Opcode

0038  07CA              btfss   eeprom,do      ; check for state of data bit to xmit
0039  0A3C              goto    bitlow         ; low? go set data line low
003A  05E6              bsf     port_b,sdata   ; high? set data line high
003B  0A3D              goto    clkout         ; go toggle the clock

003C  04E6      bitlow  bcf     port_b,sdata   ; output a low bit
003D  05C6      clkout  bsf     port_b,sclk    ; set clock line high
003E  0000              nop
003F  0000              nop
0040  0000              nop
0041  0000              nop
0042  04C6              bcf     port_b,sclk    ; return clock line low
0043  0800              retlw   0
                ;
                ;*************************************************************
                ;       BITIN routine reads one bit of data from the
                ;       serial EE device and stores it in the bit 'di'
                ;*************************************************************
                BITIN
0044  05EA              bsf     eeprom,di      ; assume input bit is high
0045  0CBF              movlw   b'10111111'    ; make sdata an input line
0046  0006              tris    port_b
0047  05C6              bsf     port_b,sclk    ; set clock line high
0048  0000              nop                    ; just sit here a sec
0049  0000              nop
004A  0000              nop
004B  0000              nop
004C  0000              nop                    ;
004D  07E6              btfss   port_b,sdata   ; read the data bit
004E  04EA              bcf     eeprom,di      ; input bit was low, set 'di' accordingly
004F  04C6              bcf     port_b,sclk    ; set clock line low
                                               ;
0050  0800              retlw   0              ;
                ;
                ;***************************************************************
```

```
                ;       Transmit Data Subroutine
                ;       This routine takes the byte of data stored in the
                ;       'datao' register and transmits it to the serial EE device.
                ;       It will then send 1 more clock to the serial EE for the
                ;       acknowledge bit.  If the ack bit from the part was low
                ;       then the transmission was successful.  If it is high, then
                ;       the device did not send a proper ack bit and the ack
                ;       fail LED will be turned on.
                ;*****************************************************************
                TX
0051  0C08               movlw   .8
0052  0031               movwf   count           ; set the #bits to 8
                         ;
                TXLP
0053  04CA               bcf     eeprom,do       ; assume bit out is low
0054  06F0               btfsc   txbuf,7         ; is bit out really low?

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:33 1994  Page 5

Line   PC    Opcode

0055  05CA               bsf     eeprom,do       ; no, set it high
0056  0936               call    BITOUT          ; send the bit to serial EE
0057  0370               rlf     txbuf           ; rotate txbuf left
0058  02F1               decfsz  count           ; 8 bits done?
0059  0A53               goto    TXLP            ; no - go again
005A  0944               call    BITIN           ; read ack bit
005B  06EA               btfsc   eeprom,di       ; check ack bit
005C  0525               bsf     port_a,ackf     ; set acknowledge fail LED if the
                                                 ; device did not pull data low
                         ;
005D  0800               retlw   0
                ;
                ;*****************************************************************
                ;       Receive data routine
                ;       This routine reads one byte of data from the part
                ;       into the 'datai' register.  It then sends a high
                ;       ack bit to indicate that no more data is to be read
                ;*****************************************************************
                RX
005E  006D               clrf    datai           ; clear input buffer
005F  0C08               movlw   .8              ; set # bits to 8
0060  0031               movwf   count
0061  0403               bcf     status,0        ; make sure carry bit is low
0062  036D      RXLP     rlf     datai           ; rotate datai 1 bit left
0063  0944               call    BITIN           ; read a bit
0064  06EA               btfsc   eeprom,di
0065  050D               bsf     datai,0         ; set bit 0 if necessary
0066  02F1               decfsz  count           ; 8 bits done?
0067  0A62               goto    RXLP            ; no, do another
0068  0800               retlw   0
                ;
                ;*****************************************************************
                ;       Power up routine
                ;       This is the program entry point.  I/O line status is
                ;       set for port A here.
                ;*****************************************************************
                PWRUP
0069  0C00                movlw  b'00000000'
006A  0005                tris   port_a          ; set port A as all output
006B  0065                clrf   port_a          ; all output lines low
006C  0A6D                goto   READ
                ;
                ;****************************************************************
                ;       READ (read routine)
                ;       This routine reads 8 consecutive addresses of the
                ;       serial EE device starting at address 00 in the
                ;       sequential read mode. Reading in this mode is more
                ;       efficient than the random read mode as the control byte
```

```
                  ;       and address have to be sent only once at the beginning
                  ;       of the sequence.  As many consecutive addresses as
                  ;       needed can then be read from the part until a stop bit is

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:49:33 1994  Page 6

Line   PC    Opcode

                  ;       sent.  In the read mode, the 16C54 must send the acknowledge
                  ;       bit after every 8 data bits from the device.  When the
                  ;       last byte needed has been read, then the controller will
                  ;       send a high acknowledge bit and then a stop bit to halt
                  ;       transmission from the device.
                  ;***************************************************************
                  READ
                  ;
       006D  0425          bcf     port_a,ackf ; clear the ack fail LED if on
       006E  0C08          movlw   .8
       006F  0032          movwf   bcount      ; set number of bytes to read as 8
       0070  0CA0          movlw   b'10100000' ; set slave address and write mode
       0071  002F          movwf   slave
       0072  0077          clrf    addr1       ; set starting high address to 00
       0073  006C          clrf    addr        ; set starting low address to 00
                                               ;
       0074  090E          call    BSTART      ; generate start bit
       0075  020F          movf    slave,w     ; get slave address
       0076  0030          movwf   txbuf       ; into transmit buffer
       0077  0951          call    TX          ; and send it
       0078  0217          movf    addr1,w     ; get word 1 address
       0079  0030          movwf   txbuf       ; into transmit buffer
       007A  0951          call    TX          ; and send it
       007B  020C          movf    addr,w      ; get word 0 address
       007C  0030          movwf   txbuf       ; into transmit buffer
       007D  0951          call    TX          ; and send it
       007E  090E          call    BSTART      ; generate start bit
       007F  0CA1          movlw   b'10100001' ; get slave address and read mode
       0080  0030          movwf   txbuf       ; into transmit buffer
       0081  0951          call    TX          ; and transmit it
                                               ;
       0082  095E  rbyte   call    RX          ; read 1 byte from device
       0083  02F2          decfsz  bcount      ; are all 8 bytes read?
       0084  0A8C          goto    lowack      ; no, send low ack and do another
       0085  05CA          bsf     eeprom,do   ; yes, send high ack bit
       0086  0936          call    BITOUT      ; to stop transmission
       0087  0923          call    BSTOP       ; and send a stop bit

       0088  0CFF          movlw   .255        ; long delay for scope
       0089  0035          movwf   loops       ; trigger purposes only
       008A  0901          call    wait
       008B  0A6D          goto    READ        ; start all over

       008C  04CA  lowack  bcf     eeprom,do   ; send low ack bit
       008D  0936          call    BITOUT      ; to continue transmission
       008E  0A82          goto    rbyte       ; and read another byte

                  ;
             0000  END
```

```
16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:50:10 1994  Page 1

Line   PC    Opcode

                             LIST P=16C54,c=132
                   ;************************************************************
                   ;       Program to set the high endurance block on the 64K
                   ;       device (24XX65). This routine does not apply to the 24XX32.(122 bytes)
                   ;
                   ;       The 24LC65 is a 64K device, 4K of which is error
                   ;       corrected to increase the endurance of that portion
                   ;       of the array.  The location of this high endurance
                   ;       'block' can be set by the user.  When the device
                   ;       comes from the factory, the high endurance block
                   ;       will be set as the uppermost block in the array.
                   ;
                   ;       This program sets the high endurance block as the
                   ;       first block in the array (address range 00h to 1FFh).
                   ;
                   ;       The high endurance block is set by sending the following
                   ;       sequence to the device:
                   ;
                   ;       SB 10100000 1XXAAAAX XXXXXXXX 00XX0000 STB
                   ;
                   ;       Where SB=start bit
                   ;              1=data high
                   ;              0=data low
                   ;              X=don't care
                   ;           AAAA=4 bit high endurance block address
                   ;            STB=stop bit
                   ;
                   ;       As an option, the user may connect a LED to pin 18
                   ;       on the PIC16C54 (use about a 1K resistor in series) as an
                   ;       acknowledge fail indicator.  This LED will come on if
                   ;       the serial EE fails to acknowledge correctly.
                   ;
                   ;       Timing is based on using the PIC16C54 in 'XT' mode
                   ;       using a 4MHz crystal.  Clock speeds to the serial EE
                   ;       will be approximately 60 kHz for this setup.
                   ;
                   ;        PIC16C54 to Serial EE Connections:
                   ;
                   ;         PIC16C54       Serial EE
                   ;       ______    ______
                   ;       Pin 12 (RB6) —>  SCLK
                   ;       Pin 13 (RB7) —>  SDATA
                   ;
                   ;       PIN 18 (RA1) —>  Acknowledge fail LED  (Optional);
                   ;************************************************************
                   ;           Register Definitions
                   ;************************************************************
             0005  port_a  equ     5h     ; port 5 (port_a) used for LEDs
             0006  port_b  equ     6h     ; port 6 (port b) used for data and
                                          ; clock lines
             000A  eeprom  equ    0ah     ; bit buffer

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:50:10 1994  Page 2

Line   PC    Opcode

             000B  bycnt   equ    0bh     ; byte counter for read mode
             000C  addr    equ    0ch     ; word 0 address counter
             000D  datai   equ    0dh     ; data input register
             000E  datao   equ    0eh     ; data output register
             000F  slave   equ    0fh     ; device address (1010xxx0)
             0010  txbuf   equ    10h     ; transmit buffer
             0011  count   equ    11h     ; bit counter
             0012  bcount  equ    12h     ; byte counter
             0015  loops   equ    15h     ; delay loop counter
```

8

```
      0016  loops2  equ    16h     ; delay loop counter
      0017  addr1   equ    17h     ; word 1 address counter
      0018  he_blk  equ    18h     ; high endurance block address
            ;**********************************************************
            ;            Bit Definitions
            ;**********************************************************
      0007  di      equ    7       ; eeprom input bit
      0006  do      equ    6       ; eeprom output bit
      0007  sdata   equ    7       ; serial EE data line (port_b,pin 13)
      0006  sclk    equ    6       ; serial EE clock line (port_b,pin 12)
      0001  ackf    equ    1       ; acknowledge fail LED (port_a,pin 18)
            ;**********************************************************
            ;
            ;
      0000          org    01ffh   ; set reset vector
01FF  0A5E          goto   PWRUP
      0000          org    000h    ;
0000  0A5E          goto   PWRUP
            ;
            ;***********************************************************
            ;       DELAY ROUTINE
            ;       This routine takes the value in 'loops'
            ;       and multiplies it times 1 millisecond to
            ;       determine delay time.
            ;***********************************************************
            WAIT
            ;
0001  0C6E  top      movlw   .110           ; timing adjustment variable
0002  0036           movwf   loops2
0003  0000  top2     nop                    ; sit and wait
0004  0000           nop
0005  0000           nop
0006  0000           nop
0007  0000           nop
0008  0000           nop
0009  02F6           decfsz  loops2         ; inner loops complete?
000A  0A03           goto    top2           ; no, go again
                                            ;
000B  02F5           decfsz  loops          ; outer loops complete?
000C  0A01           goto    top            ; no, go again
000D  0800           retlw   0              ; yes, return from sub
            ;

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:50:10 1994  Page 3

Line   PC    Opcode

            ;************************************************************
            ;       Start Bit Subroutine
            ;       this routine generates a start bit
            ;       (Low going data line while clock is high)
            ;************************************************************
            ;
            BSTART
000E  05E6           bsf     port_b,sdata   ; make sure data is high
000F  0C3F           movlw   b'00111111'
0010  0006           tris    port_b         ; set data and clock lines for output
0011  04C6           bcf     port_b,sclk    ; make sure clock is low
0012  0000           nop
0013  05C6           bsf     port_b,sclk    ; set clock high
0014  0000           nop
0015  0000           nop
0016  0000           nop
0017  0000           nop
0018  0000           nop
0019  04E6           bcf     port_b,sdata   ; data line goes low during
                                            ; high clock for start bit
001A  0000           nop
001B  0000           nop
```

```
        001C  0000           nop
        001D  0000           nop
        001E  0000           nop                   ; timing adjustment
        001F  04C6           bcf     port_b,sclk    ; start clock train
        0020  0000           nop
        0021  0000           nop
        0022  0800           retlw   0
                      ;
                      ;***********************************************************
                      ;       Stop Bit Subroutine
                      ;       This routine generates a stop bit
                      ;       (High going data line while clock is high)
                      ;***********************************************************
                      BSTOP
        0023  0C3F           movlw   b'00111111'    ;
        0024  0006           tris    port_b         ; set data/clock lines as outputs
        0025  04E6           bcf     port_b,sdata   ; make sure data line is low
        0026  0000           nop
        0027  0000           nop
        0028  0000           nop
        0029  05C6           bsf     port_b,sclk    ; set clock high
        002A  0000           nop
        002B  0000           nop
        002C  0000           nop
        002D  05E6           bsf     port_b,sdata   ; data goes high while clock high
                                                    ; for stop bit
        002E  0000           nop
        002F  0000           nop
        0030  04C6           bcf     port_b,sclk    ; set clock low again

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:50:10 1994  Page 4

Line   PC    Opcode

        0031  0000           nop
        0032  0000           nop
        0033  0000           nop
        0034  0800           retlw   0
                      ;
                      ;************************************************************
                      ;       BITOUT routine takes one bit of data in 'do' and
                      ;       transmits it to the serial EE device
                      ;************************************************************
                      BITOUT
        0035  0C3F           movlw   b'00111111'    ; set data,clock as outputs
        0036  0006           tris    port_b
        0037  07CA           btfss   eeprom,do      ; check for state of data bit to xmit
        0038  0A3B           goto    bitlow         ;
        0039  05E6           bsf     port_b,sdata   ; set data line high
        003A  0A3C           goto    clkout         ; go toggle the clock

        003B  04E6    bitlow  bcf     port_b,sdata   ; output a low bit
        003C  05C6    clkout  bsf     port_b,sclk    ; set clock line high
        003D  0000           nop
        003E  0000           nop
        003F  0000           nop
        0040  0000           nop
        0041  04C6           bcf     port_b,sclk    ; return clock line low
        0042  0800           retlw   0
                      ;
                      ;*************************************************************
                      ;       BITIN routine reads one bit of data from the
                      ;       serial EE device and stores it in 'di'
                      ;*************************************************************
                      BITIN
        0043  05EA           bsf     eeprom,di      ; assume input bit is high
        0044  0CBF           movlw   b'10111111'    ; make sdata an input line
        0045  0006           tris    port_b
        0046  05E6           bsf     port_b,sdata   ; set sdata line for input
```

```
0047  05C6           bsf     port_b,sclk    ; set clock line high
0048  0000           nop                    ; just sit here a sec
0049  0000           nop
004A  0000           nop
004B  0000           nop
004C  0000           nop                    ;
004D  07E6           btfss   port_b,sdata   ; read the data bit
004E  04EA           bcf     eeprom,di      ; input bit was low
004F  04C6           bcf     port_b,sclk    ; set clock line low
                                            ;
0050  0800           retlw   0              ;
              ;
              ;****************************************************************
              ;       Transmit Data Subroutine
              ;       This routine takes the byte of data stored in the
              ;       'datao' register and transmits it to the serial EE device.

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:50:10 1994  Page 5

Line   PC    Opcode

              ;       It will then send 1 more clock to the serial EE for the
              ;       acknowledge bit.  If the ack bit from the part was low
              ;       then the transmission was successful.  If it is high, then
              ;       the device did not send a proper ack bit and the ack
              ;       fail LED will be turned on.
              ;****************************************************************
              TX
0051  0C08           movlw   .8
0052  0031           movwf   count          ; set the #bits to 8
                     ;
              TXLP
0053  04CA           bcf     eeprom,do      ; assume bit out is low
0054  06F0           btfsc   txbuf,7        ; is bit out really low?
0055  05CA           bsf     eeprom,do      ; otherwise data bit =1
0056  0935           call    BITOUT         ; serial data out
0057  0370           rlf     txbuf          ; rotate txbuf left
0058  02F1           decfsz  count          ; 8 bits done?
0059  0A53           goto    TXLP           ; no - go again
005A  0943           call    BITIN          ; read ack bit
005B  06EA           btfsc   eeprom,di      ; check ack bit
005C  0525           bsf     port_a,ackf    ; set acknowledge fail LED if the
                     ;
005D  0800           retlw   0
              ;
              ;****************************************************************
              ;       Power up routine
              ;       This is the program entry point, which in this case simply
              ;       sets the port_a I/O lines and directs control to the
              ;       byte write routine.
              ;****************************************************************
              PWRUP
005E  0C00            movlw  b'00000000'
005F  0005            tris   port_a         ; set port A as all output
0060  0065            clrf   port_a         ; all output lines low
0061  0A62            goto   setheblk       ; set go set the high endurance block

              ;****************************************************************
              ;       Set High Endurance Block Routine
              ;       This routine sets the high endurance block to the first
              ;       block in the array and then delays about a half second.  This
              ;       routine will repeat forever.
              ;*****************************************************************
              ;
              setheblk
              ;
0062  0065          clrf    port_a          ; clear all LEDs
0063  002F          movwf   slave
0064  0C00          movlw   .0
```

```
       0065  0038          movwf   he_blk          ; set the endurance block as first
                                                   ; block in array (block 0)
       0066  0378          rlf    he_blk           ; rotate starting block left 1 bit

16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:50:10 1994  Page 6

Line   PC    Opcode

                                                   ; to arrange it correctly
       0067  05F8          bsf     he_blk,7        ; set msb bit in block address byte
       0068  090E          call    BSTART          ; generate start bit
       0069  0CA0          movlw   b'10100000'     ; get slave address and write mode
       006A  0030          movwf   txbuf           ; into transmit buffer
       006B  0951          call    TX              ; and send it
       006C  0218          movf    he_blk,w        ; get the block address byte
       006D  0030          movwf   txbuf           ; into transmit buffer
       006E  0951          call    TX              ; and send it
       006F  0070          clrf    txbuf           ; clear buffer
       0070  0951          call    TX              ; send 8 don't care bits
       0071  0070          clrf    txbuf
       0072  0951          call    TX              ; send config byte(all 0's)
       0073  0923          call    BSTOP           ; send stop bit
       0074  0CFF          movlw   .255            ; long delay
       0075  0035          movwf   loops
       0076  0901          call    wait
       0077  0CFF          movlw   .255
       0078  0035          movwf   loops
       0079  0901          call    wait
       007A  0A62          goto    setheblk        ; go again
                   ;
                   ;
             0000  END
```

8

```
16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:50:03 1994  Page 1
Line   PC    Opcode

                                                         LIST  P=16C54,c=132
;************************************************************               ;      Program to
set the security option (write protect)
                   ;       on the 64K device (24LC65).   (125 bytes)
                   ;
                   ;       The security option (or write protect option) allows
                   ;       the user to write protect any number of consecutive
                   ;       4K blocks in the device.
                   ;
                   ;       This program sets the security option to protect the
                   ;       lower 4 blocks (2K bytes) in the array.
                   ;
                   ;       **********  CAUTIONS TO THE USER !!!! **********
                   ;
                   ;       1) THE SECURITY OPTION CAN ONLY BE SET ONCE!!
                   ;       2) THE HIGH ENDURANCE BLOCK CANNOT BE CHANGED
                   ;          AFTER THE SECURITY OPTION IS SET.
                   ;       3) THE HIGH ENDURANCE BLOCK CANNOT BE PROTECTED.
                   ;
                   ;       The security option is set by sending the following
                   ;       sequence to the device:
                   ;
                   ;       SB 10100000 1XXAAAAX XXXXXXXX 10XXNNNN STB
                   ;
                   ;       Where SB=start bit
                   ;              1=data high
                   ;              0=data low
                   ;              X=don't care
                   ;           AAAA=4 bit number to indicate first secure block (0-15)
                   ;           NNNN=4 bit number to indicate how many secure blocks(1-15)
                   ;            STB=stop bit
                   ;
                   ;       As an option, the user may connect a LED to pin 18
                   ;       on the PIC 16C54 (use about a 1K resistor in series) as an
                   ;       acknowledge fail indicator.  This LED will come on if
                   ;       the serial EE fails to acknowledge correctly.
                   ;
                   ;       Timing is based on using the PIC 16C54 in 'XT' mode
                   ;       using a 4Mhz crystal.  Clock speeds to the serial EE
                   ;       will be approximately 60 Khz for this setup.
                   ;
                   ;        PIC 16C54 to Serial EE Connections:
                   ;
                   ;        PIC 16C54        Serial EE
                   ;       ______     ______
                   ;       Pin 12 (RB6) —>  SCLK
                   ;       Pin 13 (RB7) —>  SDATA
                   ;
                   ;       PIN 18 (RA1) —>  Acknowledge fail LED  (Optional);
                   ;************************************************************
                   ;           Register Definitions
 16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:50:03 1994  Page 2
Line   PC    Opcode

                   ;************************************************************          0005
port_a     equ     5h     ; port 5 (port_a) used for LEDs
             0006  port_b     equ     6h     ; port 6 (port b) used for data and
                                          ; clock lines
             000A  eeprom     equ    0ah     ; bit buffer
             000B  bycnt      equ    0bh     ; byte counter for read mode
             000C  addr       equ    0ch     ; word 0 address counter
             000D  datai      equ    0dh     ; data input register
             000E  datao      equ    0eh     ; data output register
             000F  slave      equ    0fh     ; device address (1010xxx0)
             0010  txbuf      equ    10h     ; transmit buffer
             0011  count      equ    11h     ; bit counter
```

```
0055                                    ; clock lines
0056         000A  eeprom     equ    0ah     ; bit buffer
0057         000B  bycnt      equ    0bh     ; byte counter for read mode
0058         000C  addr       equ    0ch     ; word 0 address counter
0059         000D  datai      equ    0dh     ; data input register
0060         000E  datao      equ    0eh     ; data output register
0061         000F  slave      equ    0fh     ; device address (1010xxx0)
0062         0010  txbuf      equ    10h     ; transmit buffer
0063         0011  count      equ    11h     ; bit counter
0064         0012  bcount     equ    12h     ; byte counter
0065         0015  loops      equ    15h     ; delay loop counter
0066         0016  loops2     equ    16h     ; delay loop counter
0067         0017  addr1      equ    17h     ; word 1 address counter
0068         0018  strt_blk   equ    18h     ; starting block number for secure portion
0069         0019  sec_blks   equ    19h     ; number of secure blocks
0070               ;************************************************************
0071               ;           Bit Definitions
0072               ;************************************************************
0073         0007  di      equ    7       ; eeprom input bit
0074         0006  do      equ    6       ; eeprom output bit
0075         0007  sdata   equ    7       ; serial EE data line (port_b,pin 13)
0076         0006  sclk    equ    6       ; serail EE clock line (port_b,pin 12)
0077         0001  ackf    equ    1       ; acknowledge fail LED (port_a,pin 18)
0078               ;************************************************************
0079              ; 0080             ; 0081        0000         org    01ffh   ; set reset vector
0082   01FF  0A5E          goto   PWRUP
0083         0000          org    000h    ;
0084   0000  0A5E          goto   PWRUP
0085               ;
0086             ;************************************************************** 0087             ;
DELAY ROUTINE
0088               ;       This routine takes the value in 'loops'
0089               ;       and multiplies it times 1 millisecond to
0090               ;       determine delay time.
0091               ;*************************************************************
0092             WAIT 0093             ; 0094   0001  0C6E  top     movlw   .110          ; timing
adjustment variable
0095   0002  0036           movwf   loops2
0096   0003  0000  top2     nop                   ; sit and wait
0097   0004  0000           nop
0098   0005  0000           nop
0099   0006  0000           nop
0100   0007  0000           nop
0101   0008  0000           nop
0102   0009  02F6           decfsz  loops2        ; inner loops complete?
 16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:50:03 1994  Page 3
Line   PC    Opcode

0103   000A  0A03           goto    top2          ; no, go again
0104                                              ;
0105   000B  02F5           decfsz  loops         ; outer loops complete?
0106   000C  0A01           goto    top           ; no, go again
0107   000D  0800           retlw   0             ; yes, return from sub
0108               ;
0109            ;*************************************************************** 0110            ;
Start Bit Subroutine
0111               ;       this routine generates a start bit
0112               ;       (Low going data line while clock is high)
0113               ;**************************************************************
0114               ;
0115              BSTART 0116   000E  05E6          bsf     port_b,sdata   ; make sure data is high
0117   000F  0C3F           movlw   b'00111111'
0118   0010  0006           tris    port_b        ; set data and clock lines for output
0119   0011  04C6           bcf     port_b,sclk    ; make sure clock is low
0120   0012  0000           nop
0121   0013  05C6           bsf     port_b,sclk    ; set clock high
0122   0014  0000           nop
0123   0015  0000           nop
0124   0016  0000           nop
```

8

```
0017  0000            nop
0018  0000            nop
0019  04E6            bcf     port_b,sdata   ; data line goes low during
                                             ; high clock for start bit
001A  0000            nop
001B  0000            nop
001C  0000            nop
001D  0000            nop
001E  0000            nop                    ; timing adjustment
001F  04C6            bcf     port_b,sclk    ; start clock train
0020  0000            nop
0021  0000            nop
0022  0800            retlw   0
                 ;
                 ;************************************************************
                 ;       Stop Bit Subroutine
                 ;       This routine generates a stop bit
                 ;       (High going data line while clock is high)
                 ;************************************************************
                 BSTOP
0023  0C3F            movlw   b'00111111'    ;
0024  0006            tris    port_b         ; set data/clock lines as outputs
0025  04E6            bcf     port_b,sdata   ; make sure data line is low
0026  0000            nop
0027  0000            nop
0028  0000            nop
0029  05C6            bsf     port_b,sclk    ; set clock high
002A  0000            nop
002B  0000            nop
 16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:50:03 1994  Page 4
Line   PC    Opcode

002C  0000            nop
002D  05E6            bsf     port_b,sdata   ; data goes high while clock high
                                             ; for stop bit
002E  0000            nop
002F  0000            nop
0030  04C6            bcf     port_b,sclk    ; set clock low again
0031  0000            nop
0032  0000            nop
0033  0000            nop
0034  0800            retlw   0
                 ;
                 ;************************************************************
                 ;       BITOUT routine takes one bit of data in 'do' and
                 ;       transmits it to the serial EE device
                 ;************************************************************
                 BITOUT
0035  0C3F            movlw   b'00111111'    ; set data,clock as outputs
0036  0006            tris    port_b
0037  07CA            btfss   eeprom,do      ; check for state of data bit to xmit
0038  0A3B            goto    bitlow         ;
0039  05E6            bsf     port_b,sdata   ; set data line high
003A  0A3C            goto    clkout         ; go toggle the clock

003B  04E6  bitlow    bcf     port_b,sdata   ; output a low bit
003C  05C6  clkout    bsf     port_b,sclk    ; set clock line high
003D  0000            nop
003E  0000            nop
003F  0000            nop
0040  0000            nop
0041  04C6            bcf     port_b,sclk    ; return clock line low
0042  0800            retlw   0
                 ;
                 ;************************************************************
                 ;       BITIN routine reads one bit of data from the
                 ;       serial EE device and stores it in 'di'
                 ;************************************************************
                 BITIN
0043  05EA            bsf     eeprom,di      ; assume input bit is high
0044  0CBF            movlw   b'10111111'    ; make sdata an input line
0045  0006            tris    port_b
```

```
0194   0046  05E6           bsf     port_b,sdata   ; set sdata line for input
0195   0047  05C6           bsf     port_b,sclk    ; set clock line high
0196   0048  0000           nop                    ; just sit here a sec
0197   0049  0000           nop
0198   004A  0000           nop
0199   004B  0000           nop
0200   004C  0000           nop                    ;
0201   004D  07E6           btfss   port_b,sdata   ; read the data bit
0202   004E  04EA           bcf     eeprom,di      ; input bit was low
0203   004F  04C6           bcf     port_b,sclk    ; set clock line low
0204                                               ;
 16c5x/7x Cross-Assembler V4.12 Released  Mon Jun 06 10:50:03 1994  Page 5
Line   PC    Opcode

0205   0050  0800           retlw   0              ;
0206                  ;
0207                  ;*****************************************************************
0208                  ;       Transmit Data Subroutine
0209                  ;       This routine takes the byte of data stored in the
0210                  ;       'datao' register and transmits it to the serial EE device.
0211                  ;       It will then send 1 more clock to the serial EE for the
0212                  ;       acknowledge bit.  If the ack bit from the part was low
0213                  ;       then the transmission was sucessful.  If it is high, then
0214                  ;       the device did not send a proper ack bit and the ack
0215                  ;       fail LED will be turned on.
0216                  ;*****************************************************************
0217                  TX 0218   0051  0C08           movlw   .8
0219   0052  0031           movwf   count          ; set the #bits to 8
0220                        ;
0221                  TXLP 0222   0053  04CA           bcf     eeprom,do      ; assume bit out is low
0223   0054  06F0           btfsc   txbuf,7        ; is bit out really low?
0224   0055  05CA           bsf     eeprom,do      ; otherwise data bit =1
0225   0056  0935           call    BITOUT         ; serial data out
0226   0057  0370           rlf     txbuf          ; rotate txbuf left
0227   0058  02F1           decfsz  count          ; 8 bits done?
0228   0059  0A53           goto    TXLP           ; no - go again
0229   005A  0943           call    BITIN          ; read ack bit
0230   005B  06EA           btfsc   eeprom,di      ; check ack bit
0231   005C  0525           bsf     port_a,ackf    ; set acknowledge fail LED if the
0232                        ;
0233   005D  0800           retlw   0
0234                  ;
0235                  ;*****************************************************************
0236                  ;       Power up routine
0237                  ;       This is the program entry point, which in this case simply
0238                  ;       sets the port_a I/O lines and directs control to the
0239                  ;       byte write routine.
0240                  ;*****************************************************************
0241                  PWRUP 0242   005E  0C00           movlw  b'00000000'
0243   005F  0005            tris   port_a         ; set port A as all output
0244   0060  0065            clrf   port_a         ; all output lines low
0245   0061  0A62            goto   set_sec        ; set go set the high endurance block
0246
0247                  ;*****************************************************************
0248                  ;       Set Security Routine
0249                  ;       This routine sets the secure portion of the array to the
0250                  ;       lower 4 blocks (2K bytes) in the array and then delays
0251                  ;       about a half second.  This routine will repeat forever.
0252                  ;******************************************************************
0253                 ; 0254              set_sec 0255             ;                           16c5x/
7x Cross-Assembler V4.12 Released  Mon Jun 06 10:50:03 1994  Page 6
Line   PC    Opcode

0256   0062  0065          clrf    port_a          ; clear all LEDs
0257   0063  0C00          movlw   .0
0258   0064  0038          movwf   strt_blk        ; set the first protected block as
0259                                               ; block 0
0260   0065  0C04          movlw   .4
0261   0066  0039          movwf   sec_blks        ; set the number of protected blocks
```

8

```
                                                   ; as 4 (2048 bytes)
       0067  0378         rlf     strt_blk         ; rotate starting block left 1 bit
                                                   ; to arrange it correctly
       0068  05F8         bsf     strt_blk,7       ; set msb bit in starting block address
       0069  05F9         bsf     sec_blks,7       ; set msb bit block count byte
       006A  090E         call    BSTART           ; generate start bit
       006B  0CA0         movlw   b'10100000'      ; get slave address and write mode
       006C  0030         movwf   txbuf            ; into transmit buffer
       006D  0951         call    TX               ; and send it
       006E  0218         movf    strt_blk,w       ; get the starting block address
       006F  0030         movwf   txbuf            ; into transmit buffer
       0070  0951         call    TX               ; and send it
       0071  0070         clrf    txbuf            ; clear buffer
       0072  0951         call    TX               ; send 8 don't care bits
       0073  0219         movf    sec_blks,w       ; get secure blocks byte
       0074  0030         movwf   txbuf            ; into transmit buffer
       0075  0951         call    TX               ; and send it
       0076  0923         call    BSTOP            ; send stop bit
       0077  0CFF         movlw   .255             ; long delay
       0078  0035         movwf   loops
       0079  0901         call    wait
       007A  0CFF         movlw   .255
       007B  0035         movwf   loops
       007C  0901         call    wait
       007D  0A62         goto    set_sec          ; go again
                   ;
                   ;        0000  END              16c5x/7x Cross-Assembler V4.12 Released  Mon
Jun 06 10:50:03 1994  Page 7
```

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