



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 I²C™ is the industry leader (>70 % worldwide). The I²C standard specifies a maximum 16K bit address space. How does the personal communications system designer, who requires more memory, solve this problem?

THE I²C DILEMMA

The I²C serial bus has many advantages over other common serial interfaces for serial embedded devices. The I²C 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 I²C 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 I²C 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.

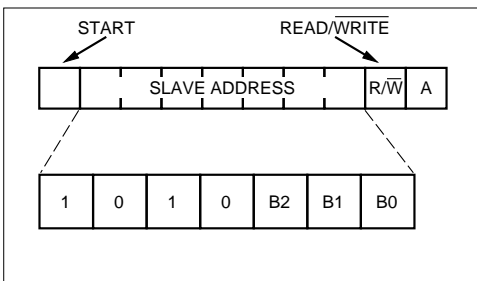
The Microchip 24XX65 is a serial memory device with 64K bits (8Kx8) capacity and additional patent pending unique features not found any where else. First let's look at the current I²C addressing scheme, the cascadable solution, and finally the Microchip total embedded systems solution.

I²C ADDRESSING

The I²C 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.

Using the 24XX65 and 24XX32

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 EEPROM) 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

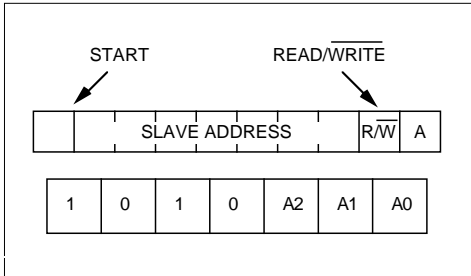
written 100,000 times. This is probably the most common misinterpretation made by system designers. Endurance is thus an interactive application-specific reliability parameter. It is not a typical data sheet specification, such as a parametric AC/DC specification with benchmark standards for measurement. 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

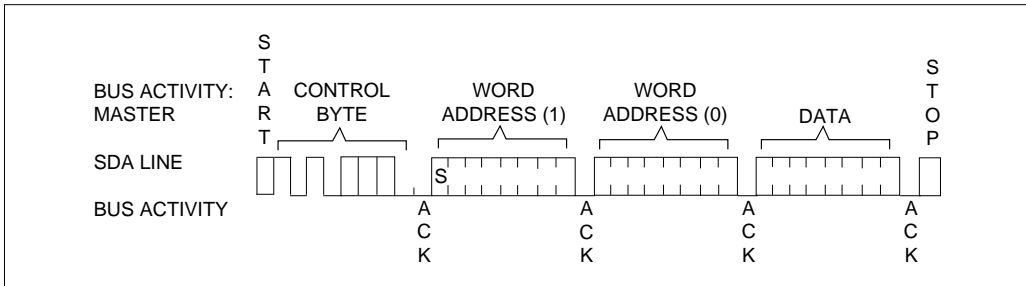
FIGURE 2 - CONTROL BYTE ALLOCATION



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

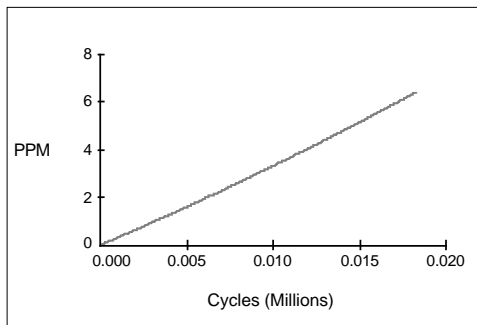
FIT	1.0
PPM	6
Time	5.0
Write cycles	18,250

FIGURE 3 - BYTE WRITE



Using the 24XX65 and 24XX32

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 I²C 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 I²C 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 I²C bus.

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

*Authors: Richard J. Fisher
Memory Products Division
Bruce Negley
Memory Products Division*

Using the 24XX65 and 24XX32

APPENDIX A

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

```
Line   PC   Opcode

0001                LIST P=16C54,c=132
0002                ;*****
0003                ;      64K Byte Read Program (138 bytes)
0004                ;
0005                ;      This program demonstrates how to interface a
0006                ;      Microchip PIC16C54 to a 24XX32/65 Serial EE device
0007                ;      and perform a random read operation. This program
0008                ;      will read 8 consecutive addresses in the 'random
0009                ;      read' mode. This entails setting the address pointer
0010                ;      before doing the read command for each address.
0011                ;
0012                ;      Another, more efficient method of reading consecutive
0013                ;      addresses is the 'sequential read' mode. This involves
0014                ;      sending the control byte and address for the first byte
0015                ;      to read, then continuing to provide clocks for the next
0016                ;      addresses. The device will automatically increment the
0017                ;      address. An example of the sequential read mode is
0018                ;      provided in the '64kseqr.asm' file.
0019                ;
0020                ;      Timing is based on using the PIC16C54 in 'XT' mode
0021                ;      using a 4Mhz crystal. Clock speeds to the serial EE
0022                ;      will be approximately 60 kHz for this setup.
0023                ;
0024                ;      As an option, the user may connect a LED to pin 18
0025                ;      on the PIC16C54 (use about a 1K resistor in series) as an
0026                ;      acknowledge fail indicator. This LED will come on if
0027                ;      the serial EE fails to acknowledge correctly.
0028                ;
0029                ;      PIC16C54 to Serial EE Connections:
0030                ;
0031                ;      PIC16C54      Serial EE
0032                ;      _____  _____
0033                ;      Pin 12 (RB6) -> SCLK
0034                ;      Pin 13 (RB7) -> SDATA
0035                ;
0036                ;      PIN 18 (RA1) -> Acknowledge fail LED (Optional)
0037                ;
0038                ;
0039                ;*****
0040                ;      Register Definitions
0041                ;*****
0042                0003 status equ 03h      ; status register
0043                0005 port_a equ 05h      ; port 5 (port a) used for LED display
0044                0006 port_b equ 06h      ; port 6 (port b) used for data and
0045                ;      clock lines
0046                000A eeprom equ 0ah      ; bit buffer
0047                000B bycnt equ 0bh      ; byte counter for read mode
0048                000C addr equ 0ch      ; word 0 address counter
0049                000D datai equ 0dh      ; data input register
0050                000E datao equ 0eh      ; data output register
0051                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

0052                0010 txbuf equ 10h      ; transmit buffer
0053                0011 count equ 11h      ; bit counter
0054                0012 bcount equ 12h      ; byte counter
0055                0015 loops equ 15h      ; delay loop counter
0056                0016 loops2 equ 16h      ; delay loop counter
0057                0017 addr1 equ 17h      ; word 1 address counter
0058                ;*****
```

Using the 24X65 and 24X32

```
0059          ;          Bit Definitions
0060          ;*****
0061      0007      di      equ      7          ; eeprom input bit
0062      0006      do      equ      6          ; eeprom output bit
0063      0007      sdata   equ      7          ; serial EE data line (port_b,pin 13)
0064      0006      sclk   equ      6          ; serial EE clock line (port_b,pin 12)
0065      0001      ackf   equ      1          ; acknowledge fail LED (port_a)
0066          ;*****
0067          ;
0068      0000          org      01ffh      ; set reset vector
0069      01FF 0A6B      goto     PWRUP
0070      0000          org      000h      ;
0071      0000 0A6B      goto     PWRUP
0072          ;
0073          ;*****
0074          ;          DELAY ROUTINE
0075          ;          This routine takes the value in 'loops'
0076          ;          and multiplies it times 1 millisecond to
0077          ;          determine delay time.
0078          ;*****
0079      WAIT
0080          ;
0081      0001 0C6E      top      movlw   .110          ; timing adjustment variable
0082      0002 0036      movwf   loops2
0083      0003 0000      top2     nop
0084      0004 0000      nop
0085      0005 0000      nop
0086      0006 0000      nop
0087      0007 0000      nop
0088      0008 0000      nop
0089      0009 02F6      decfsz  loops2          ; inner loops complete?
0090      000A 0A03      goto     top2          ; no, go again
0091          ;
0092      000B 02F5      decfsz  loops          ; outer loops complete?
0093      000C 0A01      goto     top          ; no, go again
0094      000D 0800      retlw   0          ; yes, return from sub
0095          ;
0096          ;*****
0097          ;          Start Bit Subroutine
0098          ;          this routine generates a start bit
0099          ;          (Low going data line while clock is high)
0100          ;*****
0101          ;
0102      BSTART
```

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

Line	PC	Opcod			
0103	000E	05E6	bsf	port_b,sdata	; make sure data is high
0104	000F	0C3F	movlw	b'00111111'	
0105	0010	0006	tris	port_b	; set data and clock lines for output
0106	0011	04C6	bcf	port_b,sclk	; make sure clock is low
0107	0012	0000	nop		
0108	0013	05C6	bsf	port_b,sclk	; set clock high
0109	0014	0000	nop		
0110	0015	0000	nop		
0111	0016	0000	nop		
0112	0017	0000	nop		
0113	0018	0000	nop		
0114	0019	04E6	bcf	port_b,sdata	; data line goes low during
0115					; high clock for start bit
0116	001A	0000	nop		
0117	001B	0000	nop		
0118	001C	0000	nop		
0119	001D	0000	nop		
0120	001E	0000	nop		; timing adjustment
0121	001F	04C6	bcf	port_b,sclk	; start clock train
0122	0020	0000	nop		

Using the 24XX65 and 24XX32

```
0123 0021 0000      nop
0124 0022 0800      retlw 0
0125                ;
0126                ;*****
0127                ;      Stop Bit Subroutine
0128                ;      This routine generates a stop bit
0129                ;      (High going data line while clock is high)
0130                ;*****
0131      BSTOP
0132 0023 04E6      bcf    port_b,sdata ; make sure data line is low
0133 0024 0C3F      movlw  b'00111111' ;
0134 0025 0006      tris   port_b      ; set data/clock lines as outputs
0135 0026 04E6      bcf    port_b,sdata ; make sure data line is low
0136 0027 0000      nop
0137 0028 0000      nop
0138 0029 0000      nop
0139 002A 05C6      bsf    port_b,sclk  ; set clock high
0140 002B 0000      nop
0141 002C 0000      nop
0142 002D 0000      nop
0143 002E 05E6      bsf    port_b,sdata ; data goes high while clock high
0144                ; for stop bit
0145 002F 0000      nop
0146 0030 0000      nop
0147 0031 04C6      bcf    port_b,sclk  ; set clock low again
0148 0032 0000      nop
0149 0033 0000      nop
0150 0034 0000      nop
0151 0035 0800      retlw 0
0152                ;
0153                ;*****
```

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

```
Line  PC  Opcode
0154                ;      BITOUT routine takes the bit of data in 'do' and
0155                ;      transmits it to the serial EE device
0156                ;*****
0157      BITOUT
0158 0036 0C3F      movlw  b'00111111' ; set data,clock as outputs
0159 0037 0006      tris   port_b
0160 0038 07CA      btffs  eeprom,do   ; check for state of data bit to xmit
0161 0039 0A3C      goto   bitlow     ; low? go set data line low
0162 003A 05E6      bsf    port_b,sdata ; high? set data line high
0163 003B 0A3D      goto   clkout     ; go toggle the clock
0164
0165 003C 04E6      bitlow  bcf    port_b,sdata ; output a low bit
0166 003D 05C6      clkout  bsf    port_b,sclk  ; set clock line high
0167 003E 0000      nop
0168 003F 0000      nop
0169 0040 0000      nop
0170 0041 0000      nop
0171 0042 04C6      bcf    port_b,sclk  ; return clock line low
0172 0043 0800      retlw 0
0173                ;
0174                ;*****
0175                ;      BITIN routine reads one bit of data from the
0176                ;      serial EE device and stores it in the bit 'di'
0177                ;*****
0178      BITIN
0179 0044 05EA      bsf    eeprom,di   ; assume input bit is high
0180 0045 0CBF      movlw  b'10111111' ; make sdata an input line
0181 0046 0006      tris   port_b
0182 0047 05C6      bsf    port_b,sclk ; set clock line high
0183 0048 0000      nop                ; just sit here a sec
0184 0049 0000      nop
0185 004A 0000      nop
0186 004B 0000      nop
```

Using the 24X65 and 24X32

```
0187 004C 0000      nop                ;
0188 004D 07E6      btfs    port_b,sdata ; read the data bit
0189 004E 04EA      bcf     eeeprom,di   ; input bit was low, set 'di' accordingly
0190 004F 04C6      bcf     port_b,sclk  ; set clock line low
0191                ;
0192 0050 0800      retlw   0            ;
0193                ;
0194                ;*****
0195                ; Transmit Data Subroutine
0196                ; This routine takes the byte of data stored in the
0197                ; 'data0' register and transmits it to the serial EE device.
0198                ; It will then send 1 more clock to the serial EE for the
0199                ; acknowledge bit. If the ack bit from the part was low
0200                ; then the transmission was successful. If it is high, then
0201                ; the device did not send a proper ack bit and the ack
0202                ; fail LED will be turned on.
0203                ;*****
0204                TX
```

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

```
Line  PC  Opcode
0205  0051  0C08      movlw   .8
0206  0052  0031      movwf   count        ; set the #bits to 8
0207                ;
0208                TXLP
0209  0053  04CA      bcf     eeeprom,do   ; assume bit out is low
0210  0054  06F0      btfs    txbuf,7      ; is bit out really low?
0211  0055  05CA      bsf     eeeprom,do   ; no, set it high
0212  0056  0936      call   BITOUT        ; send the bit to serial EE
0213  0057  0370      rlf     txbuf        ; rotate txbuf left
0214  0058  02F1      decfsz  count        ; 8 bits done?
0215  0059  0A53      goto   TXLP          ; no - go again
0216  005A  0944      call   BITIN         ; read ack bit
0217  005B  06EA      btfs    eeeprom,di   ; check ack bit
0218  005C  0525      bsf     port_a,ackf   ; set acknowledge fail LED if the
0219                ; device did not pull data low
0220                ;
0221  005D  0800      retlw   0
0222                ;
0223                ;*****
0224                ; Receive data routine
0225                ; This routine reads one byte of data from the part
0226                ; into the 'data1' register. It then sends a high
0227                ; ack bit to indicate that no more data is to be read
0228                ;*****
0229                RX
0230  005E  006D      clrf    data1        ; clear input buffer
0231  005F  0C08      movlw   .8           ; set # bits to 8
0232  0060  0031      movwf   count
0233  0061  0403      bcf     status,0     ; make sure carry bit is low
0234  0062  036D      rlf     data1        ; rotate data1 1 bit left
0235  0063  0944      call   BITIN         ; read a bit
0236  0064  06EA      btfs    eeeprom,di   ; read ack bit
0237  0065  050D      bsf     data1,0      ; set bit 0 if necessary
0238  0066  02F1      decfsz  count        ; 8 bits done?
0239  0067  0A62      goto   RXLP         ; no, do another
0240  0068  05CA      bsf     eeeprom,do   ; set ack bit = 1
0241  0069  0936      call   BITOUT        ; to finish transmission
0242  006A  0800      retlw   0
0243                ;
0244                ;*****
0245                ; Power up routine
0246                ; This is the program entry point. I/O line status is
0247                ; set for port A here.
0248                ;*****
0249                PWRUP
0250  006B  0C00      movlw   b'00000000'
```

Using the 24XX65 and 24XX32

```
0251 006C 0005      tris   port_a      ; set port A as all output
0252 006D 0065      clr   port_a      ; all output lines low
0253 006E 0A6F      goto  READ
0254                ;
0255                ;*****
```

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

```
Line  PC  Opcode
0256                ;      READ (read routine)
0257                ;      This routine reads 8 consecutive addresses of the
0258                ;      serial EE device starting at address 00 in the
0259                ;      random access mode (non-sequential read). Reading
0260                ;      the device using the random access mode
0261                ;      requires that the address pointer be set for every
0262                ;      byte before the read takes place. The address pointer
0263                ;      is set by sending a 'write mode' control byte with the
0264                ;      address of the location to read.
0265                ;*****
0266      READ
0267                ;
0268 006F 0425      bcf   port_a,ackf ; clear the ack fail LED if on
0269 0070 0C08      movlw .8          ; set number of bytes to read as 8
0270 0071 0032      movwf bcount     ;
0271 0072 0077      clr   addr1     ; set starting high address byte to 00
0272 0073 006C      clr   addr      ; set starting low address byte to 00
0273                ;
0274 0074 090E rbyte  call  BSTART     ; generate start bit
0275                ;
0276                ; now send the write control byte and
0277                ; address to set the pointer
0278                ;
0279 0075 0CA0      movlw b'10100000' ; get slave address (write mode)
0280 0076 0030      movwf txbuf     ; into transmit buffer
0281 0077 0951      call  TX        ; and send it
0282 0078 0217      movf  addr1,w   ; get word 1 address
0283 0079 0030      movwf txbuf     ; into transmit buffer
0284 007A 0951      call  TX        ; and send it
0285 007B 020C      movf  addr,w   ; get word 0 address
0286 007C 0030      movwf txbuf     ; into transmit buffer
0287 007D 0951      call  TX        ; and send it
0288                ;
0289                ; now read one byte from the part
0290                ;
0291 007E 090E      call  BSTART     ; generate start bit
0292 007F 0CA1      movlw b'10100001' ; get slave address and read mode
0293 0080 0030      movwf txbuf     ; into transmit buffer
0294 0081 0951      call  TX        ; and transmit it
0295 0082 095E      call  RX        ; read 1 byte from serial EE
0296 0083 0923      call  BSTOP     ; send stop bit to end transmission
0297 0084 02AC      incf  addr      ; add 1 to address counter
0298 0085 02F2      decfsz bcount   ; yes, are all 8 bytes read?
0299 0086 0A74      goto  rbyte     ; no, do another byte
0300
0301 0087 0CFF      movlw .255     ; long delay for scope
0302 0088 0035      movwf loops    ; trigger purposes only
0303 0089 0901      call  wait     ;
0304 008A 0A6F      goto  READ     ; yes, start all over
0305                ;
0306                0000  END
```

Using the 24XX65 and 24XX32

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

```
Line  PC  Opcode
0001                LIST P=16C54,c=132
0002                ;*****
0003                ;      64K Byte Write Program (127 bytes)
0004                ;
0005                ;      This program demonstrates how to interface a
0006                ;      Microchip PIC16C54 to the 24XX32/65 Serial EE device
0007                ;      and perform a byte write operation on 8 consecutive
0008                ;      addresses.
0009                ;
0010                ;      After each byte is written, time must be given to the
0011                ;      device for it to complete the write cycle before
0012                ;      the next command can be sent.  The easiest solution
0013                ;      is to consult the data book for the maximum write
0014                ;      cycle time and just wait that long before the next
0015                ;      command is sent(10ms for this device).  This program
0016                ;      demonstrates that solution.
0017                ;
0018                ;      Another, more efficient method of determining when the
0019                ;      write cycle is complete is called 'data polling.'  This
0020                ;      method is demonstrated in the program "64kdpoll."
0021                ;
0022                ;      As an option, the user may connect a LED to pin 18
0023                ;      on the PIC16C54 (use about a 1K resistor in series) as an
0024                ;      acknowledge fail indicator.  This LED will come on if
0025                ;      the serial EE fails to acknowledge correctly.
0026                ;
0027                ;      Timing is based on using the PIC16C54 in 'XT' mode
0028                ;      using a 4MHz crystal.  Clock speeds to the serial EE
0029                ;      will be approximately 60 kHz for this setup.
0030                ;
0031                ;      PIC16C54 to Serial EE Connections:
0032                ;
0033                ;      PIC16C54      Serial EE
0034                ;
0035                ;      Pin 12 (RB6) -> SCLK
0036                ;      Pin 13 (RB7) -> SDATA
0037                ;
0038                ;      PIN 18 (RA1) -> Acknowledge fail LED (Optional)
0039                ;
0040                ;*****
0041                ;      Register Definitions
0042                ;*****
0043    0005    port_a    equ     5h      ; port 5 (port_a) used for LEDs
0044    0006    port_b    equ     6h      ; port 6 (port b) used for data and
0045                ;      clock lines
0046    000A    eeprom    equ     0ah     ; bit buffer
0047    000B    bycnt     equ     0bh     ; byte counter for read mode
0048    000C    addr      equ     0ch     ; word 0 address counter
0049    000D    datai     equ     0dh     ; data input register
0050    000E    datao     equ     0eh     ; data output register
0051    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
0052    0010    txbuf    equ     10h     ; transmit buffer
0053    0011    count    equ     11h     ; bit counter
0054    0012    bcount    equ     12h     ; byte counter
0055    0015    loops     equ     15h     ; delay loop counter
0056    0016    loops2    equ     16h     ; delay loop counter
0057    0017    addr1     equ     17h     ; word 1 address counter
0058                ;*****
0059                ;      Bit Definitions
0060                ;*****
```

Using the 24XX65 and 24XX32

```
0061      0007 di      equ   7      ; eeprom input bit
0062      0006 do      equ   6      ; eeprom output bit
0063      0007 sdata   equ   7      ; serial EE data line (port_b,pin 13)
0064      0006 sclk    equ   6      ; serial EE clock line (port_b,pin 12)
0065      0001 ackf     equ   1      ; acknowledge fail LED (port_a,pin 18)
0066      ;*****
0067      ;
0068      ;
0069      0000          org   01fffh ; set reset vector
0070  01FF 0A5E          goto  PWRUP
0071      0000          org   000h   ;
0072  0000 0A5E          goto  PWRUP
0073      ;
0074      ;*****
0075      ;      DELAY ROUTINE
0076      ;      This routine takes the value in 'loops'
0077      ;      and multiplies it times 1 millisecond to
0078      ;      determine delay time.
0079      ;*****
0080      WAIT
0081      ;
0082  0001 0C6E top      movlw  .110      ; timing adjustment variable
0083  0002 0036          movwf  loops2
0084  0003 0000 top2    nop                ; sit and wait
0085  0004 0000          nop
0086  0005 0000          nop
0087  0006 0000          nop
0088  0007 0000          nop
0089  0008 0000          nop
0090  0009 02F6          decfsz loops2      ; inner loops complete?
0091  000A 0A03          goto   top2        ; no, go again
0092      ;
0093  000B 02F5          decfsz loops      ; outer loops complete?
0094  000C 0A01          goto   top        ; no, go again
0095  000D 0800          retlw  0          ; yes, return from sub
0096      ;
0097      ;*****
0098      ;      Start Bit Subroutine
0099      ;      this routine generates a start bit
0100      ;      (Low going data line while clock is high)
0101      ;*****
0102      ;
```

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

Line	PC	Opcode			
0103			BSTART		
0104	000E	05E6	bsf	port_b,sdata	; make sure data is high
0105	000F	0C3F	movlw	b'00111111'	
0106	0010	0006	tris	port_b	; set data and clock lines for output
0107	0011	04C6	bcf	port_b,sclk	; make sure clock is low
0108	0012	0000	nop		
0109	0013	05C6	bsf	port_b,sclk	; set clock high
0110	0014	0000	nop		
0111	0015	0000	nop		
0112	0016	0000	nop		
0113	0017	0000	nop		
0114	0018	0000	nop		
0115	0019	04E6	bcf	port_b,sdata	; data line goes low during
0116					; high clock for start bit
0117	001A	0000	nop		
0118	001B	0000	nop		
0119	001C	0000	nop		
0120	001D	0000	nop		
0121	001E	0000	nop		; timing adjustment
0122	001F	04C6	bcf	port_b,sclk	; start clock train
0123	0020	0000	nop		
0124	0021	0000	nop		

Using the 24XX65 and 24XX32

```
0125 0022 0800          retlw  0
0126                ;
0127                ;*****
0128                ;   Stop Bit Subroutine
0129                ;   This routine generates a stop bit
0130                ;   (High going data line while clock is high)
0131                ;*****
0132                BSTOP
0133 0023 0C3F          movlw  b'00111111'  ;
0134 0024 0006          tris   port_b      ; set data/clock lines as outputs
0135 0025 04E6          bcf   port_b,sdata ; make sure data line is low
0136 0026 0000          nop
0137 0027 0000          nop
0138 0028 0000          nop
0139 0029 05C6          bsf   port_b,sclk  ; set clock high
0140 002A 0000          nop
0141 002B 0000          nop
0142 002C 0000          nop
0143 002D 05E6          bsf   port_b,sdata ; data goes high while clock high
0144                ; for stop bit
0145 002E 0000          nop
0146 002F 0000          nop
0147 0030 04C6          bcf   port_b,sclk  ; set clock low again
0148 0031 0000          nop
0149 0032 0000          nop
0150 0033 0000          nop
0151 0034 0800          retlw  0
0152                ;
0153                ;*****
```

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

```
Line  PC  Opcode
0154                ;   BITOUT routine takes one bit of data in 'do' and
0155                ;   transmits it to the serial EE device
0156                ;*****
0157                BITOUT
0158 0035 0C3F          movlw  b'00111111'  ; set data,clock as outputs
0159 0036 0006          tris   port_b
0160 0037 07CA          btfss  eeprom,do  ; check for state of data bit to xmit
0161 0038 0A3B          goto   bitlow      ;
0162 0039 05E6          bsf   port_b,sdata ; set data line high
0163 003A 0A3C          goto   clkout      ; go toggle the clock
0164
0165 003B 04E6          bitlow  bcf   port_b,sdata ; output a low bit
0166 003C 05C6          clkout  bsf   port_b,sclk  ; set clock line high
0167 003D 0000          nop
0168 003E 0000          nop
0169 003F 0000          nop
0170 0040 0000          nop
0171 0041 04C6          bcf   port_b,sclk  ; return clock line low
0172 0042 0800          retlw  0
0173                ;
0174                ;*****
0175                ;   BITIN routine reads one bit of data from the
0176                ;   serial EE device and stores it in 'di'
0177                ;*****
0178                BITIN
0179 0043 05EA          bsf   eeprom,di  ; assume input bit is high
0180 0044 0CBF          movlw  b'10111111'  ; make sdata an input line
0181 0045 0006          tris   port_b
0182 0046 05E6          bsf   port_b,sdata ; set sdata line for input
0183 0047 05C6          bsf   port_b,sclk  ; set clock line high
0184 0048 0000          nop                ; just sit here a sec
0185 0049 0000          nop
0186 004A 0000          nop
0187 004B 0000          nop
0188 004C 0000          nop                ;
```

Using the 24XX65 and 24XX32

```
0189 004D 07E6      btfss port_b,sdata ; read the data bit
0190 004E 04EA      bcf     eeprom,di  ; input bit was low
0191 004F 04C6      bcf     port_b,sclk ; set clock line low
0192
0193 0050 0800      retlw  0           ;
0194
0195 ;*****
0196 ; Transmit Data Subroutine
0197 ; This routine takes the byte of data stored in the
0198 ; 'datao' register and transmits it to the serial EE device.
0199 ; It will then send 1 more clock to the serial EE for the
0200 ; acknowledge bit. If the ack bit from the part was low
0201 ; then the transmission was successful. If it is high, then
0202 ; the device did not send a proper ack bit and the ack
0203 ; fail LED will be turned on.
0204 ;*****
```

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

```
Line  PC  Opcode
0205                      TX
0206 0051 0C08      movlw  .8
0207 0052 0031      movwf  count      ; set the #bits to 8
0208
0209                      TXLPL
0210 0053 04CA      bcf     eeprom,do  ; assume bit out is low
0211 0054 06F0      btfsc  txbuf,7    ; is bit out really low?
0212 0055 05CA      bsf     eeprom,do  ; otherwise data bit =1
0213 0056 0935      call   BITOUT     ; serial data out
0214 0057 0370      rlf    txbuf      ; rotate txbuf left
0215 0058 02F1      decfsz count     ; 8 bits done?
0216 0059 0A53      goto   TXLPL     ; no - go again
0217 005A 0943      call   BITIN     ; read ack bit
0218 005B 06EA      btfsc  eeprom,di  ; check ack bit
0219 005C 0525      bsf    port_a,ackf ; set acknowledge fail LED if the
0220
0221 005D 0800      retlw  0
0222
0223 ;*****
0224 ; Power up routine
0225 ; This is the program entry point, which in this case simply
0226 ; sets the port_a I/O lines and directs control to the
0227 ; byte write routine.
0228 ;*****
0229 PWRUP
0230 005E 0C00      movlw  b'00000000'
0231 005F 0005      tris  port_a      ; set port A as all output
0232 0060 0065      clrf  port_a      ; all output lines low
0233 0061 0A62      goto  WRBYTE
0234
0235 ;*****
0236 ; Byte Write Routine
0237 ; This routine writes the data in "datao" to
0238 ; 8 consecutive bytes in the serial EE device starting
0239 ; at address 00. This routine waits 10mS after every
0240 ; byte to give the device time to do the write. This
0241 ; program repeats forever.
0242 ;*****
0243
0244 WRBYTE
0245
0246 0062 0065      clrf  port_a      ; clear all LEDs
0247 0063 0CA0      movlw b'10100000' ; set slave address and write mode
0248 0064 002F      movwf slave
0249 0065 0C55      movlw b'01010101' ; set data to write as 55h
0250 0066 002E      movwf datao
0251
0252 0067 0C08      movlw  .8         ; set number of bytes
```

Using the 24XX65 and 24XX32

```
0253 0068 0032      movwf  bcount      ; to write as to 8
0254 0069 0077      clrfs  addr1       ; set high address byte to 00
0255 006A 006C      clrfs  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
0256
0257 006B 090E  byte  call  BSTART      ; generate start bit
0258 006C 020F      movf  slave,w     ; move slave address
0259 006D 0030      movwf txbuf       ; into transmit buffer
0260 006E 0951      call  TX          ; and send it
0261 006F 0217      movf  addr1,w     ; move word 1 address
0262 0070 0030      movwf txbuf       ; into transmit buffer
0263 0071 0951      call  TX          ; and send it
0264 0072 020C      movf  addr,w      ; move word 0 address
0265 0073 0030      movwf txbuf       ; into transmit buffer
0266 0074 0951      call  TX          ; and send it
0267 0075 020E      movf  data0,w     ; move data byte
0268 0076 0030      movwf txbuf       ; to transmit buffer
0269 0077 0951      call  TX          ; and transmit it
0270 0078 0923      call  BSTOP       ; generate stop bit
0271
0272 0079 0C0A      movlw .10
0273 007A 0035      movwf loops       ; set delay time to give
0274 007B 0901      call  WAIT        ; 10 ms wait after every byte
0275 007C 02AC      incf  addr        ; add 1 to low address counter
0276 007D 02F2      decfsz bcount     ; all 8 bytes written?
0277 007E 0A6B      goto  byte        ; no, do another byte
0278
0279 007F 0A62      goto  wrbyte      ; start over
0280
0281
0282          0000  END
```

Using the 24XX65 and 24XX32

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

```
Line   PC   Opcode

0001                               LIST P=16C54,c=132
0002   ;*****
0003   ;       64K Byte Write With Data Polling Program (133 bytes)
0004   ;
0005   ;       This program demonstrates how to interface a
0006   ;       Microchip PIC16C54 to a 24XX32/65 Serial EE device.
0007   ;       This program performs a byte write operation on 8
0008   ;       consecutive addresses using the data polling method
0009   ;       to determine when the write cycle is complete.
0010   ;
0011   ;       When writing to a serial EE device, there are 2
0012   ;       ways to handle the internal timed write cycle
0013   ;       time of the device. The simplest method is
0014   ;       simply to wait until the maximum cycle time
0015   ;       is exceeded before attempting another command.
0016   ;       The other, more efficient method is known as "data
0017   ;       polling." Data polling is done by sending a start
0018   ;       bit and control byte to the part after the write
0019   ;       cycle has been initiated by a stop bit. If the ack bit
0020   ;       is low, then the device is through writing, otherwise
0021   ;       the the sequence is repeated. If no low acknowledge
0022   ;       is found within 40 attempts (about 10 milliseconds)
0023   ;       then the routine times out and sets the timeout
0024   ;       LED (pin 1) high.
0025   ;
0026   ;       As an option, the user can connect a LED to pin 1
0027   ;       on the PIC16C54 (use about a 1K resistor in series) as a
0028   ;       timeout indicator. This LED will come on if the
0029   ;       data polling is unsuccessful and the device being
0030   ;       programmed does not respond. This can be tested by
0031   ;       simply running the program with no part in the socket.
0032   ;
0033   ;       Timing is based on using the PIC16C54 in 'XT' mode
0034   ;       using a 4MHz crystal. Clock speeds to the serial EE
0035   ;       will be approximately 40 kHz for this setup.
0036   ;
0037   ;       PIC16C54 to Serial EE Connections:
0038   ;
0039   ;       PIC16C54       Serial EE
0040   ;       _____
0041   ;       Pin 12 (RB6) -> SCLK
0042   ;       Pin 13 (RB7) -> SDATA
0043   ;
0044   ;       Pin 1 (RA2) -> Write cycle timeout fail LED (optional)
0045   ;
0046   ;*****
0047   ;       Register Definitions
0048   ;*****
0049   0005 port_a equ    5h    ; port 5 (port_a) used LEDs
0050   0006 port_b equ    6h    ; port 6 (port_b) used for data and
0051                               ; clock lines
```

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

```
Line   PC   Opcode

0052   000A eeprom equ    0ah    ; bit buffer
0053   000B bycnt equ    0bh    ; byte counter for read mode
0054   000C addr equ    0ch    ; address counter
0055   000D datai equ    0dh    ; data input register
0056   000E datao equ    0eh    ; data output register
0057   000F slave equ    0fh    ; device address (1010xxx0)
0058   0010 txbuf equ    10h    ; transmit buffer
0059   0011 count equ    11h    ; bit counter
0060   0012 bcount equ    12h    ; byte counter
```

Using the 24XX65 and 24XX32

```

0061      0015 loops equ 15h ; delay loop counter
0062      0016 loops2 equ 16h ; delay loop counter
0063      0017 pollcnt equ 17h ; data polling counter
0064      0018 addr1 equ 18h ; word 1 address counter
0065      ;*****
0066      ;          Bit Definitions
0067      ;*****
0068      0007 di equ 7 ; eeprom input bit
0069      0006 do equ 6 ; eeprom output bit
0070      0007 sdata equ 7 ; serial EE data line (port_b,pin 13)
0071      0006 sclk equ 6 ; serial EE clock line (port_b,pin 12)
0072      0002 timeout equ 2 ; write cycle timeout fail LED, port_a (pin 1)
0073      0001 ackf equ 1 ; acknowledge fail LED, port_a (pin 18)
0074      ;*****
0075      ;
0076      0000 org 01ffh ; set reset vector
0077      01FF 0A5C goto PWRUP
0078      0000 org 000h ;
0079      0000 0A5C goto PWRUP
0080      ;
0081      ;*****
0082      ;          DELAY ROUTINE
0083      ;          This routine takes the value in 'loops'
0084      ;          and multiplies it times 1 millisecond to
0085      ;          determine delay time.
0086      ;*****
0087      WAIT
0088      ;
0089      0001 0C6E top movlw .110 ; timing adjustment variable
0090      0002 0036 movwf loops2
0091      0003 0000 top2 nop ; sit and wait
0092      0004 0000 nop
0093      0005 0000 nop
0094      0006 0000 nop
0095      0007 0000 nop
0096      0008 0000 nop
0097      0009 02F6 decfsz loops2 ; inner loops complete?
0098      000A 0A03 goto top2 ; no, go again
0099      ;
0100      000B 02F5 decfsz loops ; outer loops complete?
0101      000C 0A01 goto top ; no, go again
0102      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		
0103			;	
0104			;*****	
0105			; Start Bit Subroutine	
0106			; this routine generates a start bit	
0107			; (Low going data line while clock is high)	
0108			;*****	
0109			;	
0110			BSTART	
0111	000E	05E6	bsf	port_b,sdata ; make sure data is high
0112	000F	0C3F	movlw	b'00111111'
0113	0010	0006	tris	port_b ; set data and clock lines for output
0114	0011	04C6	bcf	port_b,sclk ; make sure clock is low
0115	0012	0000	nop	
0116	0013	05C6	bsf	port_b,sclk ; set clock high
0117	0014	0000	nop	
0118	0015	0000	nop	
0119	0016	0000	nop	
0120	0017	0000	nop	
0121	0018	0000	nop	
0122	0019	04E6	bcf	port_b,sdata ; data line goes low during
0123				; high clock for start bit
0124	001A	0000	nop	

Using the 24XX65 and 24XX32

```
0125 001B 0000      nop
0126 001C 0000      nop
0127 001D 0000      nop
0128 001E 0000      nop          ; timing adjustment
0129 001F 04C6      bcf      port_b,sclk   ; start clock train
0130 0020 0000      nop
0131 0021 0000      nop
0132 0022 0800      retlw   0
0133                ;
0134                ;*****
0135                ;      Stop Bit Subroutine
0136                ;      This routine generates a stop bit
0137                ;      (High going data line while clock is high)
0138                ;*****
0139      BSTOP
0140 0023 0C3F      movlw   b'00111111'   ;
0141 0024 0006      tris   port_b      ; set data/clock lines as outputs
0142 0025 04E6      bcf      port_b,sdata ; make sure data line is low
0143 0026 0000      nop
0144 0027 0000      nop
0145 0028 0000      nop
0146 0029 05C6      bsf      port_b,sclk   ; set clock high
0147 002A 0000      nop
0148 002B 0000      nop
0149 002C 0000      nop
0150 002D 05E6      bsf      port_b,sdata ; data goes high while clock high
0151                ; for stop bit
0152 002E 0000      nop
0153 002F 0000      nop
```

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

```
Line  PC  Opcode
-----
0154 0030 04C6      bcf      port_b,sclk   ; set clock low again
0155 0031 0000      nop
0156 0032 0000      nop
0157 0033 0000      nop
0158 0034 0800      retlw   0
0159                ;
0160                ;*****
0161                ;      BITOUT routine takes one bit of data in 'do' and
0162                ;      transmits it to the serial EE device
0163                ;*****
0164      BITOUT
0165 0035 0C3F      movlw   b'00111111'   ; set data,clock as outputs
0166 0036 0006      tris   port_b
0167 0037 07CA      btfs    eeprom,do     ; check for state of data bit to xmit
0168 0038 0A3B      goto   bitlow        ;
0169 0039 05E6      bsf      port_b,sdata ; set data line high
0170 003A 0A3C      goto   clkout        ; go toggle the clock
0171                ;
0172 003B 04E6      bitlow  bcf      port_b,sdata ; output a low bit
0173 003C 05C6      clkout  bsf      port_b,sclk   ; set clock line high
0174 003D 0000      nop
0175 003E 0000      nop
0176 003F 0000      nop
0177 0040 0000      nop
0178 0041 04C6      bcf      port_b,sclk   ; return clock line low
0179 0042 0800      retlw   0
0180                ;
0181                ;      End of Subroutine
0182                ;
0183                ;*****
0184                ;      BITIN routine reads one bit of data from the
0185                ;      serial EE device and stores it in 'di'
0186                ;*****
0187      BITIN
0188 0043 05EA      bsf      eeprom,di    ; assume input bit is high
```

Using the 24X65 and 24X32

```
0189 0044 0CBF      movlw  b'10111111'    ; make sdata an input line
0190 0045 0006      tris   port_b
0191 0046 05E6      bsf    port_b,sdata   ; set sdata line for input
0192 0047 05C6      bsf    port_b,sclk    ; set clock line high
0193 0048 0000      nop                                ; just sit here a sec
0194 0049 0000      nop
0195 004A 0000      nop
0196 004B 0000      nop
0197 004C 0000      nop                                ;
0198 004D 07E6      btfss  port_b,sdata   ; read the data bit
0199 004E 04EA      bcf    eeprom,di     ; input bit was low
0200 004F 04C6      bcf    port_b,sclk    ; set clock line low
0201                                ;
0202 0050 0800      retlw  0              ;
0203                                ;
0204                                ;*****
```

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

```
Line  PC  Opcode
0205          ;      Transmit Data Subroutine
0206          ;      This routine takes the byte of data stored in the
0207          ;      'datao' register and transmits it to the serial EE device.
0208          ;      It will then send 1 more clock to the serial EE for the
0209          ;      acknowledge bit.  If the ack bit from the part was low
0210          ;      then the transmission was successful.  If it is high, then
0211          ;      the device did not send a proper ack bit and the ack
0212          ;      fail LED will be turned on.
0213          ;*****
0214          TX
0215 0051 0C08      movlw  .8
0216 0052 0031      movwf  count          ; set the #bits to 8
0217          ;
0218          TXLP
0219 0053 04CA      bcf    eeprom,do     ; assume bit out is low
0220 0054 06F0      btfsc  txbuf,7       ; is bit out really low?
0221 0055 05CA      bsf    eeprom,do     ; otherwise data bit =1
0222 0056 0935      call   BITOUT        ; serial data out
0223 0057 0370      rlf   txbuf          ; rotate txbuf left
0224 0058 02F1      decfz  count         ; 8 bits done?
0225 0059 0A53      goto  TXLP          ; no - go again
0226 005A 0943      call   BITIN        ; read ack bit
0227          ;
0228 005B 0800      retlw  0
0229          ;
0230          ;*****
0231          ;      Power up routine
0232          ;      This is the program entry point, which in this case simply
0233          ;      sets the port_a I/O lines and directs control to the
0234          ;      write routine.
0235          ;*****
0236          PWRUP
0237 005C 0C00      movlw  b'00000000'
0238 005D 0005      tris  port_a         ; set port A as all output
0239 005E 0065      clrf  port_a         ; all output lines low
0240 005F 0A60      goto  WRBYTE
0241          ;
0242          ;*****
0243          ;      Byte Write Routine with data polling technique
0244          ;
0245          ;      This routine writes the data in "datao" to
0246          ;      8 consecutive bytes in the serial EE device starting
0247          ;      at address 00. To determine when the write cycle time
0248          ;      of the device, the 'data polling' method is used. This
0249          ;      involves sending a start bit and control byte to the part
0250          ;      and checking for an acknowledge.  If the ack bit is low, then
0251          ;      the device is through writing, otherwise the the sequence
0252          ;      is repeated.  If no low acknowledge is found within 40
```

Using the 24XX65 and 24XX32

```
0253 ; attempts (about 10 milliseconds) then the routine times
0254 ; out and sets the timeout LED (pin 18) high. This program
0255 ; will repeat forever.
```

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

```
Line PC Opcode
0256 ;
0257 ;*****
0258 ;
0259 WRBYTE
0260 ;
0261 0060 0065 clrf port_a ; all LEDs off
0262 0061 0CA0 movlw b'10100000' ; set slave address and write mode
0263 0062 002F movwf slave
0264 0063 0CAA movlw b'10101010' ; set data to write as AAh
0265 0064 002E movwf datao
0266 0065 0C08 movlw .8 ; set number of bytes
0267 0066 0032 movwf bcount ; to write as to 8
0268 0067 0078 clrf addr1 ; set starting high address to 00
0269 0068 006C clrf addr ; set starting low address to 00
0270 ;
0271 0069 090E byte call BSTART ; generate start bit
0272 006A 020F movf slave,w ; move slave address
0273 006B 0030 movwf txbuf ; into transmit buffer
0274 006C 0951 call TX ; and send it
0275 006D 0218 movf addr1,w ; move word 1 address
0276 006E 0030 movwf txbuf ; into transmit buffer
0277 006F 0951 call TX ; and send it
0278 0070 020C movf addr,w ; move word 0 address
0279 0071 0030 movwf txbuf ; into transmit buffer
0280 0072 0951 call TX ; and send it
0281 0073 020E movf datao,w ; move data byte
0282 0074 0030 movwf txbuf ; to transmit buffer
0283 0075 0951 call TX ; and transmit it
0284 0076 0923 call BSTOP ; generate stop bit
0285 ;
0286 ; now start polling for a low ack bit
0287 ;
0288 0077 0C28 movlw .40 ;
0289 0078 0037 movwf pollcnt ; set max number of times to poll as 40
0290 0079 090E poll call BSTART ; generate start bit
0291 007A 0CA0 movlw b'10100000' ; move slave address (write mode)
0292 007B 0030 movwf txbuf ; into transmit buffer
0293 007C 0951 call TX ; and send it
0294 007D 07EA btfs eeprom,di ; was the ack bit low?
0295 007E 0A82 goto exitpoll ; yes, do another byte
0296 007F 02F7 decfsz pollcnt ; is poll counter down to zero?
0297 0080 0A79 goto poll ; no, poll again. Other wise the part is
0298 0081 0545 bsf port_a,timeout ; not responding in time so set timeout
0299 ; LED and continue on
0300 ;
0301 0082 02AC exitpoll incf addr ; add 1 to address counter
0302 0083 02F2 decfsz bcount ; all 8 bytes written?
0303 0084 0A69 goto byte ; no, do another byte
0304 0085 0A60 goto WRBYTE ; yes, start over
0305 ;
0306 ;
```

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

```
Line PC Opcode
0307 0000 END
```

Using the 24X65 and 24X32

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

```
Line   PC   Opcode
0001                LIST P=16C54,c=132
0002                ;*****
0003                ;    64K Page Write Program (126 bytes)
0004                ;
0005                ;    The 24X32/65 has a page length of 8 bytes. This page
0006                ;    can be used to write up to 8 bytes of data into the
0007                ;    part before initiating the write cycle. Since the
0008                ;    write cycle is timed the same for 1 byte or 8 bytes
0009                ;    it is more efficient to use the page mode when
0010                ;    consecutive addresses are being written to.
0011                ;
0012                ;    When using page mode, the control byte, upper and lower
0013                ;    addresses are sent for the first address only. After the
0014                ;    data byte for the first address is sent, the data for the
0015                ;    next consecutive address is clocked in. This is
0016                ;    repeated as many times as needed (as long as the page
0017                ;    length is not exceeded) and then a stop bit is sent.
0018                ;    The device will still acknowledge between every byte of
0019                ;    data. After the stop bit is sent, the part will
0020                ;    initiate the self timed write cycle.
0021                ;
0022                ;    This routine waits approximately 10mS for the write
0023                ;    cycle to complete for each page. A more efficient
0024                ;    method of determining when the write cycle is complete
0025                ;    is called "data polling." The data polling method is
0026                ;    explained in the program "64kdpoll.asm." That
0027                ;    particular program uses data polling in a byte write
0028                ;    mode but it can be used in exactly the same way for
0029                ;    a page mode write.
0030                ;
0031                ;    As an option, the user can connect a LED to pin 18
0032                ;    on the PIC16C54 (use about a 1K resistor in series) as a
0033                ;    acknowledge fail indicator. This LED will come on
0034                ;    if the device being programmed does not send a low
0035                ;    acknowledge bit at the proper times. This can be
0036                ;    tested by simply running the program with no part
0037                ;    in the socket.
0038                ;
0039                ;    Timing is based on using the PIC16C54 in 'XT' mode
0040                ;    using a 4MHz crystal. Clock speeds to the serial EE
0041                ;    will be approximately 40 kHz for this setup.
0042                ;
0043                ;
0044                ;    PIC16C54 to Serial EE Connections:
0045                ;
0046                ;    PIC16C54      Serial EE
0047                ;    _____
0048                ;    Pin 12 (RB6) ->  SCLK
0049                ;    Pin 13 (RB7) ->  SDATA
0050                ;
0051                ;    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
0052                ;
0053                ;
0054                ;*****
0055                ;    Register Definitions
0056                ;*****
0057                0005 port_a equ    5h    ; port 5 (port_a) used for LEDs
0058                0006 port_b equ    6h    ; port 6 (port b) used for data and
0059                ;    clock lines
0060                000A eeprom equ   0ah    ; bit buffer
```

Using the 24XX65 and 24XX32

```
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
```

Using the 24X65 and 24X32

```
0125 0015 0000      nop
0126 0016 0000      nop
0127 0017 0000      nop
0128 0018 0000      nop
0129 0019 04E6      bcf      port_b,sdata ; data line goes low during
0130                                ; high clock for start bit
0131 001A 0000      nop
0132 001B 0000      nop
0133 001C 0000      nop
0134 001D 0000      nop
0135 001E 0000      nop
0136 001F 04C6      bcf      port_b,sclk  ; timing adjustment
0137 0020 0000      nop
0138 0021 0000      nop
0139 0022 0800      retlw   0
0140                                ;
0141                                ;*****
0142                                ;      Stop Bit Subroutine
0143                                ;      This routine generates a stop bit
0144                                ;      (High going data line while clock is high)
0145                                ;*****
0146                                BSTOP
0147 0023 0C3F      movlw   b'00111111' ;
0148 0024 0006      tris    port_b   ; set data/clock lines as outputs
0149 0025 04E6      bcf      port_b,sdata ; make sure data line is low
0150 0026 0000      nop
0151 0027 0000      nop
0152 0028 0000      nop
0153 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
0154  002A 0000      nop
0155  002B 0000      nop
0156  002C 0000      nop
0157  002D 05E6      bsf      port_b,sdata ; data goes high while clock high
0158                                ; for stop bit
0159  002E 0000      nop
0160  002F 0000      nop
0161  0030 04C6      bcf      port_b,sclk  ; set clock low again
0162  0031 0000      nop
0163  0032 0000      nop
0164  0033 0000      nop
0165  0034 0800      retlw   0
0166                                ;
0167                                ;*****
0168                                ;      BITOUT routine takes one bit of data in 'do' and
0169                                ;      transmits it to the serial EE device
0170                                ;*****
0171                                BITOUT
0172  0035 0C3F      movlw   b'00111111' ; set data,clock as outputs
0173  0036 0006      tris    port_b
0174  0037 07CA      btfs    eeprom,do   ; check for state of data bit to xmit
0175  0038 0A3B      goto    bitlow      ;
0176  0039 05E6      bsf      port_b,sdata ; set data line high
0177  003A 0A3C      goto    clkout      ; go toggle the clock
0178
0179  003B 04E6      bitlow  bcf      port_b,sdata ; output a low bit
0180  003C 05C6      clkout  bsf      port_b,sclk  ; set clock line high
0181  003D 0000      nop
0182  003E 0000      nop
0183  003F 0000      nop
0184  0040 0000      nop
0185  0041 04C6      bcf      port_b,sclk  ; return clock line low
0186  0042 0800      retlw   0
0187                                ;
0188                                ;*****
```

Using the 24XX65 and 24XX32

```
0189 ; BITIN routine reads one bit of data from the
0190 ; serial EE device and stores it in 'di'
0191 ;*****
0192 BITIN
0193 0043 05EA bsf eeprom,di ; assume input bit is high
0194 0044 0CBF movlw b'10111111' ; make sdata an input line
0195 0045 0006 tris port_b
0196 0046 05E6 bsf port_b,sdata ; set sdata line for input
0197 0047 05C6 bsf port_b,sclk ; set clock line high
0198 0048 0000 nop ; just sit here a sec
0199 0049 0000 nop
0200 004A 0000 nop
0201 004B 0000 nop
0202 004C 0000 nop ;
0203 004D 07E6 btfs port_b,sdata ; read the data bit
0204 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
0205 004F 04C6 bcf port_b,sclk ; set clock line low
0206 ;
0207 0050 0800 retlw 0 ;
0208 ;
0209 ;*****
0210 ; Transmit Data Subroutine
0211 ; This routine takes the byte of data stored in the
0212 ; 'datao' register and transmits it to the serial EE device.
0213 ; It will then send 1 more clock to the serial EE for the
0214 ; acknowledge bit. If the ack bit from the part was low
0215 ; then the transmission was successful. If it is high, then
0216 ; the device did not send a proper ack bit and the ack
0217 ; fail LED will be turned on.
0218 ;*****
0219 TX
0220 0051 0C08 movlw .8
0221 0052 0031 movwf count ; set the #bits to 8
0222 ;
0223 TXLP
0224 0053 04CA bcf eeprom,do ; assume bit out is low
0225 0054 06F0 btfs txbuf,7 ; is bit out really low?
0226 0055 05CA bsf eeprom,do ; otherwise data bit =1
0227 0056 0935 call BITOUT ; serial data out
0228 0057 0370 rlf txbuf ; rotate txbuf left
0229 0058 02F1 decfsz count ; 8 bits done?
0230 0059 0A53 goto TXLP ; no - go again
0231 005A 0943 call BITIN ; read ack bit
0232 005B 06EA btfs eeprom,di ; check ack bit
0233 005C 0525 bsf port_a,ackf ; set acknowledge fail LED if the
0234 ;
0235 005D 0800 retlw 0
0236 ;
0237 ;*****
0238 ; Power up routine
0239 ; This is the program entry point, which in this case simply
0240 ; sets the port_a I/O lines and directs control to the
0241 ; byte write routine.
0242 ;*****
0243 PWRUP
0244 005E 0C00 movlw b'00000000'
0245 005F 0005 tris port_a ; set port A as all output
0246 0060 0065 clrf port_a ; all output lines low
0247 0061 0A62 goto pwrite ; go do the page write
0248 ;
0249 ;*****
0250 ; Page Write Routine
0251 ; This routine writes the data in "datao" to 8 consecutive
0252 ; bytes using page write mode starting at address 00.
```

Using the 24XX65 and 24XX32

```
0253          ;      This routine waits 10mS after every page to give the device
0254          ;      time to do the write.  This program repeats forever.
0255          ;*****
```

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

```
Line  PC   Opcode
0256          ;
0257          pwrite
0258          ;
0259 0062 0065      clrfs   port_a      ; clear all LEDs
0260 0063 0CA0      movlwf b'10100000' ; set slave address and write mode
0261 0064 002F      movwfs slave
0262 0065 0C55      movlwf b'01010101' ; set data to write as 55h
0263 0066 002E      movwfs datao
0264          ;
0265 0067 0C08      movlwf .8          ; set number of bytes
0266 0068 0032      movwfs bcount     ; to write as to 8
0267 0069 0077      clrfs   addr1     ; set high address byte to 00
0268 006A 006C      clrfs   addr      ; set low address byte to 00
0269          ;
0270 006B 090E      call   BSTART     ; generate start bit
0271 006C 020F      movf   slave,w    ; move slave address
0272 006D 0030      movwfs txbuf      ; into transmit buffer
0273 006E 0951      call   TX         ; and send it
0274 006F 0217      movf   addr1,w    ; move word 1 address
0275 0070 0030      movwfs txbuf      ; into transmit buffer
0276 0071 0951      call   TX         ; and send it
0277 0072 020C      movf   addr,w    ; move word 0 address
0278 0073 0030      movwfs txbuf      ; into transmit buffer
0279 0074 0951      call   TX         ; and send it
0280          ;
0281 0075 020E      movf   datao,w    ; move data byte
0282 0076 0030      movwfs txbuf      ; to transmit buffer
0283 0077 0951      call   TX         ; and transmit it
0284 0078 02F2      decfsz bcount     ; all 8 bytes written?
0285 0079 0A75      goto  byte        ; no, do another byte
0286          ;
0287 007A 0923      call   BSTOP     ; generate stop bit
0288 007B 0C0A      movlwf .10
0289 007C 0035      movwfs loops     ; set delay time to give
0290 007D 0901      call   WAIT      ; 10 ms wait after every byte
0291 007E 0A62      goto  pwrite     ; start over
0292          ;
0293          ;
0294          0000  END
```

Using the 24XX65 and 24XX32

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

```
Line   PC   Opcode

0001                LIST P=16C54,c=132
0002                ;*****
0003                ;       64K Sequential Read Program (142 bytes)
0004                ;
0005                ;       This program demonstrates how to interface a
0006                ;       Microchip PIC16C54 to a 24XX32/65 Serial EE device
0007                ;       and perform a sequential read operation. This program
0008                ;       will read 8 consecutive addresses in the 'sequential
0009                ;       read' mode. This entails setting the address pointer
0010                ;       for the first address only and then clocking out as many
0011                ;       bytes of data as needed. The device will automatically
0012                ;       increment the address.
0013                ;
0014                ;       Timing is based on using the PIC16C54 in 'XT' mode
0015                ;       using a 4Mhz crystal. Clock speeds to the serial EE
0016                ;       will be approximately 60 kHz for this setup.
0017                ;
0018                ;       As an option, the user may connect a LED to pin 18
0019                ;       on the PIC16C54 (use about a 1K resistor in series) as an
0020                ;       acknowledge fail indicator. This LED will come on if
0021                ;       the serial EE fails to acknowledge correctly.
0022                ;
0023                ;       PIC16C54 to Serial EE Connections:
0024                ;
0025                ;       PIC16C54           Serial EE
0026                ;       _____
0027                ;       Pin 12 (RB6) ->  SCLK
0028                ;       Pin 13 (RB7) ->  SDATA
0029                ;
0030                ;       PIN 18 (RA1) ->  Acknowledge fail LED (Optional)
0031                ;
0032                ;
0033                ;*****
0034                ;       Register Definitions
0035                ;*****
0036                0003 status equ 03h    ; status register
0037                0005 port_a equ 05h    ; port 5 (port a) used for LED display
0038                0006 port_b equ 06h    ; port 6 (port b) used for data and
0039                ; clock lines
0040                000A eeprom equ 0ah    ; bit buffer
0041                000B bycnt equ 0bh    ; byte counter for read mode
0042                000C addr  equ 0ch    ; word 0 address counter
0043                000D datai equ 0dh    ; data input register
0044                000E datao equ 0eh    ; data output register
0045                000F slave equ 0fh    ; device address 1010xxx0)
0046                0010 txbuf equ 10h   ; transmit buffer
0047                0011 count equ 11h   ; bit counter
0048                0012 bcount equ 12h  ; byte counter
0049                0015 loops equ 15h   ; delay loop counter
0050                0016 loops2 equ 16h  ; delay loop counter
0051                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

0052                ;*****
0053                ;       Bit Definitions
0054                ;*****
0055                0007 di      equ 7     ; eeprom input bit
0056                0006 do      equ 6     ; eeprom output bit
0057                0007 sdata  equ 7     ; serial EE data line (port_b,pin 13)
0058                0006 sclk   equ 6     ; serial EE clock line (port_b,pin 12)
0059                0001 ackf   equ 1     ; acknowledge fail LED (port_a)
0060                ;*****
```

Using the 24XX65 and 24XX32

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

Using the 24XX65 and 24XX32

```
0125          BSTOP
0126 0023 04E6      bcf    port_b,sdata ; make sure data line is low
0127 0024 0C3F      movlw b'00111111' ;
0128 0025 0006      tris   port_b      ; set data/clock lines as outputs
0129 0026 04E6      bcf    port_b,sdata ; make sure data line is low
0130 0027 0000      nop
0131 0028 0000      nop
0132 0029 0000      nop
0133 002A 05C6      bsf    port_b,sclk  ; set clock high
0134 002B 0000      nop
0135 002C 0000      nop
0136 002D 0000      nop
0137 002E 05E6      bsf    port_b,sdata ; data goes high while clock high
0138                                ; for stop bit
0139 002F 0000      nop
0140 0030 0000      nop
0141 0031 04C6      bcf    port_b,sclk  ; set clock low again
0142 0032 0000      nop
0143 0033 0000      nop
0144 0034 0000      nop
0145 0035 0800      retlw  0
0146 ;
0147 ;*****
0148 ;      BITOUT routine takes the bit of data in 'do' and
0149 ;      transmits it to the serial EE device
0150 ;*****
0151 BITOUT
0152 0036 0C3F      movlw  b'00111111' ; set data,clock as outputs
0153 0037 0006      tris   port_b
```

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

```
Line  PC  Opcode
0154 0038 07CA      btfss  eeprom,do   ; check for state of data bit to xmit
0155 0039 0A3C      goto   bitlow      ; low? go set data line low
0156 003A 05E6      bsf    port_b,sdata ; high? set data line high
0157 003B 0A3D      goto   clkout      ; go toggle the clock
0158
0159 003C 04E6      bitlow bcf    port_b,sdata ; output a low bit
0160 003D 05C6      clkout bsf    port_b,sclk  ; set clock line high
0161 003E 0000      nop
0162 003F 0000      nop
0163 0040 0000      nop
0164 0041 0000      nop
0165 0042 04C6      bcf    port_b,sclk  ; return clock line low
0166 0043 0800      retlw  0
0167 ;
0168 ;*****
0169 ;      BITIN routine reads one bit of data from the
0170 ;      serial EE device and stores it in the bit 'di'
0171 ;*****
0172 BITIN
0173 0044 05EA      bsf    eeprom,di   ; assume input bit is high
0174 0045 0CBF      movlw  b'10111111' ; make sdata an input line
0175 0046 0006      tris   port_b
0176 0047 05C6      bsf    port_b,sclk ; set clock line high
0177 0048 0000      nop          ; just sit here a sec
0178 0049 0000      nop
0179 004A 0000      nop
0180 004B 0000      nop
0181 004C 0000      nop
0182 004D 07E6      btfss  port_b,sdata ; read the data bit
0183 004E 04EA      bcf    eeprom,di   ; input bit was low, set 'di' accordingly
0184 004F 04C6      bcf    port_b,sclk ; set clock line low
0185 ;
0186 0050 0800      retlw  0
0187 ;
0188 ;*****
```

Using the 24X65 and 24X32

```
0189          ;      Transmit Data Subroutine
0190          ;      This routine takes the byte of data stored in the
0191          ;      'datao' register and transmits it to the serial EE device.
0192          ;      It will then send 1 more clock to the serial EE for the
0193          ;      acknowledge bit.  If the ack bit from the part was low
0194          ;      then the transmission was successful.  If it is high, then
0195          ;      the device did not send a proper ack bit and the ack
0196          ;      fail LED will be turned on.
0197          ;*****
0198          TX
0199          0051 0C08          movlw   .8
0200          0052 0031          movwf  count          ; set the #bits to 8
0201          ;
0202          TXLP
0203          0053 04CA          bcf    eeprom,do      ; assume bit out is low
0204          0054 06F0          btfsz  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
0205  0055  05CA          bsf    eeprom,do      ; no, set it high
0206  0056  0936          call  BITOUT          ; send the bit to serial EE
0207  0057  0370          rlf   txbuf          ; rotate txbuf left
0208  0058  02F1          decfsz count         ; 8 bits done?
0209  0059  0A53          goto  TXLP           ; no - go again
0210  005A  0944          call  BITIN          ; read ack bit
0211  005B  06EA          btfsz  eeprom,di     ; check ack bit
0212  005C  0525          bsf    port_a,ackf   ; set acknowledge fail LED if the
0213          ;               ; device did not pull data low
0214          ;
0215  005D  0800          retlw  0
0216          ;
0217          ;*****
0218          ;      Receive data routine
0219          ;      This routine reads one byte of data from the part
0220          ;      into the 'datai' register.  It then sends a high
0221          ;      ack bit to indicate that no more data is to be read
0222          ;*****
0223          RX
0224  005E  006D          clrf  datai          ; clear input buffer
0225  005F  0C08          movlw  .8            ; set # bits to 8
0226  0060  0031          movwf  count
0227  0061  0403          bcf   status,0       ; make sure carry bit is low
0228  0062  036D          RXLP  rlf   datai     ; rotate datai 1 bit left
0229  0063  0944          call  BITIN          ; read a bit
0230  0064  06EA          btfsz  eeprom,di
0231  0065  050D          bsf   datai,0        ; set bit 0 if necessary
0232  0066  02F1          decfsz count         ; 8 bits done?
0233  0067  0A62          goto  RXLP           ; no, do another
0234  0068  0800          retlw  0
0235          ;
0236          ;*****
0237          ;      Power up routine
0238          ;      This is the program entry point.  I/O line status is
0239          ;      set for port A here.
0240          ;*****
0241          PWRUP
0242  0069  0C00          movlw  b'00000000'
0243  006A  0005          tris  port_a         ; set port A as all output
0244  006B  0065          clrf  port_a         ; all output lines low
0245  006C  0A6D          goto  READ
0246          ;
0247          ;*****
0248          ;      READ (read routine)
0249          ;      This routine reads 8 consecutive addresses of the
0250          ;      serial EE device starting at address 00 in the
0251          ;      sequential read mode.  Reading in this mode is more
0252          ;      efficient than the random read mode as the control byte
```

Using the 24XX65 and 24XX32

0253 ; and address have to be sent only once at the beginning
0254 ; of the sequence. As many consecutive addresses as
0255 ; 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
0256 ; sent. In the read mode, the 16C54 must send the acknowledge
0257 ; bit after every 8 data bits from the device. When the
0258 ; last byte needed has been read, then the controller will
0259 ; send a high acknowledge bit and then a stop bit to halt
0260 ; transmission from the device.
0261 ;*****
0262 READ
0263 ;
0264 006D 0425 bcf port_a,ackf ; clear the ack fail LED if on
0265 006E 0C08 movlw .8
0266 006F 0032 movwf bcount ; set number of bytes to read as 8
0267 0070 0CA0 movlw b'10100000' ; set slave address and write mode
0268 0071 002F movwf slave
0269 0072 0077 clrf addr1 ; set starting high address to 00
0270 0073 006C clrf addr ; set starting low address to 00
0271 ;
0272 0074 090E call BSTART ; generate start bit
0273 0075 020F movf slave,w ; get slave address
0274 0076 0030 movwf txbuf ; into transmit buffer
0275 0077 0951 call TX ; and send it
0276 0078 0217 movf addr1,w ; get word 1 address
0277 0079 0030 movwf txbuf ; into transmit buffer
0278 007A 0951 call TX ; and send it
0279 007B 020C movf addr,w ; get word 0 address
0280 007C 0030 movwf txbuf ; into transmit buffer
0281 007D 0951 call TX ; and send it
0282 007E 090E call BSTART ; generate start bit
0283 007F 0CA1 movlw b'10100001' ; get slave address and read mode
0284 0080 0030 movwf txbuf ; into transmit buffer
0285 0081 0951 call TX ; and transmit it
0286 ;
0287 0082 095E rbyte call RX ; read 1 byte from device
0288 0083 02F2 decfsz bcount ; are all 8 bytes read?
0289 0084 0A8C goto lowack ; no, send low ack and do another
0290 0085 05CA bsf eeprom,do ; yes, send high ack bit
0291 0086 0936 call BITOUT ; to stop transmission
0292 0087 0923 call BSTOP ; and send a stop bit
0293 ;
0294 0088 0CFF movlw .255 ; long delay for scope
0295 0089 0035 movwf loops ; trigger purposes only
0296 008A 0901 call wait
0297 008B 0A6D goto READ ; start all over
0298 ;
0299 008C 04CA lowack bcf eeprom,do ; send low ack bit
0300 008D 0936 call BITOUT ; to continue transmission
0301 008E 0A82 goto rbyte ; and read another byte
0302 ;
0303 ;
0304 0000 END
```

Using the 24XX65 and 24XX32

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

```
Line  PC  Opcode
0001          LIST P=16C54,c=132
0002          ;*****
0003          ; Program to set the high endurance block on the 64K
0004          ; device (24XX65). This routine does not apply to the 24XX32.(122 bytes)
0005          ;
0006          ; The 24LC65 is a 64K device, 4K of which is error
0007          ; corrected to increase the endurance of that portion
0008          ; of the array. The location of this high endurance
0009          ; 'block' can be set by the user. When the device
0010          ; comes from the factory, the high endurance block
0011          ; will be set as the uppermost block in the array.
0012          ;
0013          ; This program sets the high endurance block as the
0014          ; first block in the array (address range 00h to 1FFh).
0015          ;
0016          ; The high endurance block is set by sending the following
0017          ; sequence to the device:
0018          ;
0019          ; SB 10100000 1XXAAAX XXXXXXXX 00XX0000 STB
0020          ;
0021          ; Where SB=start bit
0022          ;         1=data high
0023          ;         0=data low
0024          ;         X=don't care
0025          ;         AAAA=4 bit high endurance block address
0026          ;         STB=stop bit
0027          ;
0028          ; As an option, the user may connect a LED to pin 18
0029          ; on the PIC16C54 (use about a 1K resistor in series) as an
0030          ; acknowledge fail indicator. This LED will come on if
0031          ; the serial EE fails to acknowledge correctly.
0032          ;
0033          ; Timing is based on using the PIC16C54 in 'XT' mode
0034          ; using a 4MHz crystal. Clock speeds to the serial EE
0035          ; will be approximately 60 kHz for this setup.
0036          ;
0037          ; PIC16C54 to Serial EE Connections:
0038          ;
0039          ; PIC16C54      Serial EE
0040          ; _____
0041          ; Pin 12 (RB6) -> SCLK
0042          ; Pin 13 (RB7) -> SDATA
0043          ;
0044          ; PIN 18 (RA1) -> Acknowledge fail LED (Optional);
0045          ;*****
0046          ; Register Definitions
0047          ;*****
0048          0005 port_a equ 5h ; port 5 (port_a) used for LEDs
0049          0006 port_b equ 6h ; port 6 (port b) used for data and
0050          ; clock lines
0051          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
0052          000B bycnt equ 0bh ; byte counter for read mode
0053          000C addr equ 0ch ; word 0 address counter
0054          000D datai equ 0dh ; data input register
0055          000E datao equ 0eh ; data output register
0056          000F slave equ 0fh ; device address (1010xxx0)
0057          0010 txbuf equ 10h ; transmit buffer
0058          0011 count equ 11h ; bit counter
0059          0012 bcount equ 12h ; byte counter
0060          0015 loops equ 15h ; delay loop counter
```

Using the 24XX65 and 24XX32

```
0061      0016 loops2 equ 16h      ; delay loop counter
0062      0017 addr1 equ 17h      ; word 1 address counter
0063      0018 he_blk equ 18h      ; high endurance block address
0064      ;*****
0065      ;          Bit Definitions
0066      ;*****
0067      0007 di      equ 7        ; eeprom input bit
0068      0006 do      equ 6        ; eeprom output bit
0069      0007 sdata equ 7        ; serial EE data line (port_b,pin 13)
0070      0006 sclk   equ 6        ; serial EE clock line (port_b,pin 12)
0071      0001 ackf   equ 1        ; acknowledge fail LED (port_a,pin 18)
0072      ;*****
0073      ;
0074      ;
0075      0000          org 01fffh  ; set reset vector
0076      01FF 0A5E    goto PWRUP
0077      0000          org 000h    ;
0078      0000 0A5E    goto PWRUP
0079      ;
0080      ;*****
0081      ;          DELAY ROUTINE
0082      ;          This routine takes the value in 'loops'
0083      ;          and multiplies it times 1 millisecond to
0084      ;          determine delay time.
0085      ;*****
0086      WAIT
0087      ;
0088      0001 0C6E    top      movlw .110      ; timing adjustment variable
0089      0002 0036          movwf loops2
0090      0003 0000    top2     nop          ; sit and wait
0091      0004 0000          nop
0092      0005 0000          nop
0093      0006 0000          nop
0094      0007 0000          nop
0095      0008 0000          nop
0096      0009 02F6          decfsz loops2    ; inner loops complete?
0097      000A 0A03          goto top2      ; no, go again
0098      ;
0099      000B 02F5          decfsz loops    ; outer loops complete?
0100      000C 0A01          goto top      ; no, go again
0101      000D 0800          retlw 0        ; yes, return from sub
0102      ;
```

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

```
Line   PC   Opcode
-----
0103      ;*****
0104      ;          Start Bit Subroutine
0105      ;          this routine generates a start bit
0106      ;          (Low going data line while clock is high)
0107      ;*****
0108      ;
0109      BSTART
0110      000E 05E6          bsf port_b,sdata ; make sure data is high
0111      000F 0C3F          movlw b'00111111'
0112      0010 0006          tris port_b      ; set data and clock lines for output
0113      0011 04C6          bcf port_b,sclk  ; make sure clock is low
0114      0012 0000          nop
0115      0013 05C6          bsf port_b,sclk  ; set clock high
0116      0014 0000          nop
0117      0015 0000          nop
0118      0016 0000          nop
0119      0017 0000          nop
0120      0018 0000          nop
0121      0019 04E6          bcf port_b,sdata ; data line goes low during
0122      ;          ; high clock for start bit
0123      001A 0000          nop
0124      001B 0000          nop
```

Using the 24X65 and 24X32

```
0125 001C 0000      nop
0126 001D 0000      nop
0127 001E 0000      nop                ; timing adjustment
0128 001F 04C6      bcf      port_b,sclk  ; start clock train
0129 0020 0000      nop
0130 0021 0000      nop
0131 0022 0800      retlw   0
0132                ;
0133                ;*****
0134                ;      Stop Bit Subroutine
0135                ;      This routine generates a stop bit
0136                ;      (High going data line while clock is high)
0137                ;*****
0138                BSTOP
0139 0023 0C3F      movlw   b'00111111'  ;
0140 0024 0006      tris    port_b      ; set data/clock lines as outputs
0141 0025 04E6      bcf    port_b,sdata  ; make sure data line is low
0142 0026 0000      nop
0143 0027 0000      nop
0144 0028 0000      nop
0145 0029 05C6      bsf    port_b,sclk  ; set clock high
0146 002A 0000      nop
0147 002B 0000      nop
0148 002C 0000      nop
0149 002D 05E6      bsf    port_b,sdata  ; data goes high while clock high
0150                ; for stop bit
0151 002E 0000      nop
0152 002F 0000      nop
0153 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
0154  0031  0000      nop
0155  0032  0000      nop
0156  0033  0000      nop
0157  0034  0800      retlw   0
0158                ;
0159                ;*****
0160                ;      BITOUT routine takes one bit of data in 'do' and
0161                ;      transmits it to the serial EE device
0162                ;*****
0163                BITOUT
0164  0035  0C3F      movlw   b'00111111'  ; set data,clock as outputs
0165  0036  0006      tris    port_b
0166  0037  07CA      btfss  eeprom,do    ; check for state of data bit to xmit
0167  0038  0A3B      goto   bitlow      ;
0168  0039  05E6      bsf    port_b,sdata  ; set data line high
0169  003A  0A3C      goto   clkout      ; go toggle the clock
0170
0171  003B  04E6  bitlow bcf    port_b,sdata  ; output a low bit
0172  003C  05C6  clkout bsf    port_b,sclk  ; set clock line high
0173  003D  0000      nop
0174  003E  0000      nop
0175  003F  0000      nop
0176  0040  0000      nop
0177  0041  04C6      bcf    port_b,sclk  ; return clock line low
0178  0042  0800      retlw   0
0179                ;
0180                ;*****
0181                ;      BITIN routine reads one bit of data from the
0182                ;      serial EE device and stores it in 'di'
0183                ;*****
0184                BITIN
0185  0043  05EA      bsf    eeprom,di    ; assume input bit is high
0186  0044  0CBF      movlw  b'10111111'  ; make sdata an input line
0187  0045  0006      tris    port_b
0188  0046  05E6      bsf    port_b,sdata  ; set sdata line for input
```

Using the 24XX65 and 24XX32

```
0189 0047 05C6      bsf    port_b,sclk    ; set clock line high
0190 0048 0000      nop                                ; just sit here a sec
0191 0049 0000      nop
0192 004A 0000      nop
0193 004B 0000      nop
0194 004C 0000      nop                                ;
0195 004D 07E6      btfs   port_b,sdata  ; read the data bit
0196 004E 04EA      bcf    eeprom,di     ; input bit was low
0197 004F 04C6      bcf    port_b,sclk   ; set clock line low
0198                                     ;
0199 0050 0800      retlw  0              ;
0200                                     ;
0201      ;*****
0202      ; Transmit Data Subroutine
0203      ; This routine takes the byte of data stored in the
0204      ; 'data0' 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

0205      ; It will then send 1 more clock to the serial EE for the
0206      ; acknowledge bit. If the ack bit from the part was low
0207      ; then the transmission was successful. If it is high, then
0208      ; the device did not send a proper ack bit and the ack
0209      ; fail LED will be turned on.
0210      ;*****
0211      TX
0212 0051 0C08      movlw  .8
0213 0052 0031      movwf  count          ; set the #bits to 8
0214                                     ;
0215      TXLP
0216 0053 04CA      bcf    eeprom,do     ; assume bit out is low
0217 0054 06F0      btfs   txbuf,7       ; is bit out really low?
0218 0055 05CA      bsf    eeprom,do     ; otherwise data bit =1
0219 0056 0935      call  BITOUT         ; serial data out
0220 0057 0370      rlf   txbuf          ; rotate txbuf left
0221 0058 02F1      decfsz count         ; 8 bits done?
0222 0059 0A53      goto  TXLP          ; no - go again
0223 005A 0943      call  BITIN         ; read ack bit
0224 005B 06EA      btfs   eeprom,di    ; check ack bit
0225 005C 0525      bsf    port_a,ackf   ; set acknowledge fail LED if the
0226                                     ;
0227 005D 0800      retlw  0
0228      ;
0229      ;*****
0230      ; Power up routine
0231      ; This is the program entry point, which in this case simply
0232      ; sets the port_a I/O lines and directs control to the
0233      ; byte write routine.
0234      ;*****
0235      PWRUP
0236 005E 0C00      movlw  b'00000000'
0237 005F 0005      tris  port_a         ; set port A as all output
0238 0060 0065      clrf  port_a         ; all output lines low
0239 0061 0A62      goto  setheblk       ; set go set the high endurance block
0240      ;
0241      ;*****
0242      ; Set High Endurance Block Routine
0243      ; This routine sets the high endurance block to the first
0244      ; block in the array and then delays about a half second. This
0245      ; routine will repeat forever.
0246      ;*****
0247      ;
0248      setheblk
0249      ;
0250 0062 0065      clrf  port_a         ; clear all LEDs
0251 0063 002F      movwf  slave
0252 0064 0C00      movlw  .0
```

Using the 24XX65 and 24XX32

```
0253 0065 0038      movwf  he_blk      ; set the endurance block as first
0254                ; block in array (block 0)
0255 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
0256
0257 0067 05F8      bsf    he_blk,7    ; to arrange it correctly
0258 0068 090E      call  BSTART      ; set msb bit in block address byte
0259 0069 0CA0      movlw b'10100000' ; generate start bit
0260 006A 0030      movwf txbuf       ; get slave address and write mode
0261 006B 0951      call  TX          ; into transmit buffer
0262 006C 0218      movf  he_blk,w    ; and send it
0263 006D 0030      movwf txbuf       ; get the block address byte
0264 006E 0951      call  TX          ; into transmit buffer
0265 006F 0070      clrf  txbuf       ; and send it
0266 0070 0951      call  TX          ; clear buffer
0267 0071 0070      clrf  txbuf       ; send 8 don't care bits
0268 0072 0951      call  TX          ; send config byte(all 0's)
0269 0073 0923      call  BSTOP       ; send stop bit
0270
0271
0272 0074 0CFF      movlw .255        ; long delay
0273 0075 0035      movwf loops
0274 0076 0901      call  wait
0275 0077 0CFF      movlw .255
0276 0078 0035      movwf loops
0277 0079 0901      call  wait
0278
0279 007A 0A62      goto  setheblk    ; go again
0280                ;
0281                ;
0282                0000  END
```

Using the 24XX65 and 24XX32

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

Line PC Opcode

```
0001 ;***** LIST P=16C54,c=132 0002
;***** 0003 ; Program to
set the security option (write protect)
0004 ; on the 64K device (24LC65). (125 bytes)
0005 ;
0006 ; The security option (or write protect option) allows
0007 ; the user to write protect any number of consecutive
0008 ; 4K blocks in the device.
0009 ;
0010 ; This program sets the security option to protect the
0011 ; lower 4 blocks (2K bytes) in the array.
0012 ;
0013 ; ***** CAUTIONS TO THE USER !!!! *****
0014 ;
0015 ; 1) THE SECURITY OPTION CAN ONLY BE SET ONCE!!
0016 ; 2) THE HIGH ENDURANCE BLOCK CANNOT BE CHANGED
0017 ; AFTER THE SECURITY OPTION IS SET.
0018 ; 3) THE HIGH ENDURANCE BLOCK CANNOT BE PROTECTED.
0019 ;
0020 ; The security option is set by sending the following
0021 ; sequence to the device:
0022 ;
0023 ; SB 10100000 1XXAAAX XXXXXXXX 10XXNNNN STB
0024 ;
0025 ; Where SB=start bit
0026 ; 1=data high
0027 ; 0=data low
0028 ; X=don't care
0029 ; AAAA=4 bit number to indicate first secure block (0-15)
0030 ; NNNN=4 bit number to indicate how many secure blocks(1-15)
0031 ; STB=stop bit
0032 ;
0033 ; As an option, the user may connect a LED to pin 18
0034 ; on the PIC 16C54 (use about a 1K resistor in series) as an
0035 ; acknowledge fail indicator. This LED will come on if
0036 ; the serial EE fails to acknowledge correctly.
0037 ;
0038 ; Timing is based on using the PIC 16C54 in 'XT' mode
0039 ; using a 4Mhz crystal. Clock speeds to the serial EE
0040 ; will be approximately 60 Khz for this setup.
0041 ;
0042 ; PIC 16C54 to Serial EE Connections:
0043 ;
0044 ; PIC 16C54 Serial EE
0045 ;
0046 ; Pin 12 (RB6) -> SCLK
0047 ; Pin 13 (RB7) -> SDATA
0048 ;
0049 ; PIN 18 (RA1) -> Acknowledge fail LED (Optional);
0050 ;*****
0051 ; Register Definitions
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:50:03 1994 Page 2

Line PC Opcode

```
0052 ;***** 0053 0005
port_a equ 5h ; port 5 (port_a) used for LEDs
0054 0006 port_b equ 6h ; port 6 (port b) used for data and
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
```

Using the 24XX65 and 24XX32

```
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      ; serial 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 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
```

Using the 24XX65 and 24XX32

```
0125 0017 0000      nop
0126 0018 0000      nop
0127 0019 04E6      bcf      port_b,sdata    ; data line goes low during
0128                                     ; high clock for start bit
0129 001A 0000      nop
0130 001B 0000      nop
0131 001C 0000      nop
0132 001D 0000      nop
0133 001E 0000      nop
0134 001F 04C6      bcf      port_b,sclk    ; timing adjustment
0135 0020 0000      nop
0136 0021 0000      nop
0137 0022 0800      retlw   0
0138
0139      ;
0140      ;*****
0141      ;      Stop Bit Subroutine
0142      ;      This routine generates a stop bit
0143      ;      (High going data line while clock is high)
0144      ;*****
0144      BSTOP 0145 0023 0C3F      movlw   b'00111111'    ;
0146 0024 0006      tris      port_b      ; set data/clock lines as outputs
0147 0025 04E6      bcf      port_b,sdata    ; make sure data line is low
0148 0026 0000      nop
0149 0027 0000      nop
0150 0028 0000      nop
0151 0029 05C6      bsf      port_b,sclk    ; set clock high
0152 002A 0000      nop
0153 002B 0000      nop
16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:50:03 1994 Page 4
Line PC Opcode
0154 002C 0000      nop
0155 002D 05E6      bsf      port_b,sdata    ; data goes high while clock high
0156                                     ; for stop bit
0157 002E 0000      nop
0158 002F 0000      nop
0159 0030 04C6      bcf      port_b,sclk    ; set clock low again
0160 0031 0000      nop
0161 0032 0000      nop
0162 0033 0000      nop
0163 0034 0800      retlw   0
0164
0165      ;***** 0166 ;
BITOUT routine takes one bit of data in 'do' and
0167      ;      transmits it to the serial EE device
0168      ;*****
0169      BITOUT 0170 0035 0C3F      movlw   b'00111111'    ; set data,clock as outputs
0171 0036 0006      tris      port_b
0172 0037 07CA      btfs    eeprom,do      ; check for state of data bit to xmit
0173 0038 0A3B      goto    bitlow        ;
0174 0039 05E6      bsf      port_b,sdata    ; set data line high
0175 003A 0A3C      goto    clkout        ; go toggle the clock
0176
0177 003B 04E6      bitlow   bcf      port_b,sdata    ; output a low bit
0178 003C 05C6      clkout   bsf      port_b,sclk    ; set clock line high
0179 003D 0000      nop
0180 003E 0000      nop
0181 003F 0000      nop
0182 0040 0000      nop
0183 0041 04C6      bcf      port_b,sclk    ; return clock line low
0184 0042 0800      retlw   0
0185
0186      ;***** 0187 ;
BITIN routine reads one bit of data from the 0188 ;      serial EE device and stores
it in 'di'
0189      ;*****
0190      BITIN 0191 0043 05EA      bsf      eeprom,di      ; assume input bit is high
0192 0044 0CBF      movlw   b'10111111'    ; make sdata an input line
0193 0045 0006      tris      port_b
```

Using the 24X65 and 24X32

```

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                                ; 'data0' 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 successful. 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                ;

```

7x Cross-Assembler V4.12 Released Mon Jun 06 10:50:03 1994 Page 6

16c5x/

```

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

```

Using the 24XX65 and 24XX32

```
0262                                     ; as 4 (2048 bytes)
0263 0067 0378      rlf      strt_blk      ; rotate starting block left 1 bit
0264                                     ; to arrange it correctly
0265 0068 05F8      bsf      strt_blk,7    ; set msb bit in starting block address
0266 0069 05F9      bsf      sec_blks,7    ; set msb bit block count byte
0267 006A 090E      call     BSTART        ; generate start bit
0268 006B 0CA0      movlw   b'10100000'    ; get slave address and write mode
0269 006C 0030      movwf   txbuf        ; into transmit buffer
0270 006D 0951      call     TX                ; and send it
0271
0272 006E 0218      movf    strt_blk,w      ; get the starting block address
0273 006F 0030      movwf   txbuf        ; into transmit buffer
0274 0070 0951      call     TX                ; and send it
0275 0071 0070      clrf   txbuf        ; clear buffer
0276 0072 0951      call     TX                ; send 8 don't care bits
0277
0278 0073 0219      movf    sec_blks,w    ; get secure blocks byte
0279 0074 0030      movwf   txbuf        ; into transmit buffer
0280 0075 0951      call     TX                ; and send it
0281 0076 0923      call     BSTOP        ; send stop bit
0282
0283                0284 0077 0CFF      movlw   .255          ; long delay
0285 0078 0035      movwf   loops
0286 0079 0901      call    wait
0287 007A 0CFF      movlw   .255
0288 007B 0035      movwf   loops
0289 007C 0901      call    wait
0290
0291 007D 0A62      goto   set_sec        ; go again
0292                ;
0293                ; 0294      0000  END          16c5x/7x Cross-Assembler V4.12 Released Mon
Jun 06 10:50:03 1994 Page 7
```

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

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable”.
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

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

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Trademarks


The Microchip name and logo, the Microchip logo, FilterLab, KEELOQ, microID, MPLAB, PIC, PICmicro, PICMASTER, PICSTART, PRO MATE, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

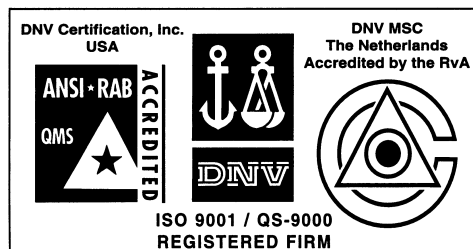
dsPIC, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, MXDEV, PICC, PICDEM, PICDEM.net, rPIC, Select Mode and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A.

Serialized Quick Turn Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2002, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.



Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.



MICROCHIP

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office

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

Rocky Mountain

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

Atlanta

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

Boston

2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848 Fax: 978-692-3821

Chicago

333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas

4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

Detroit

Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Kokomo

2767 S. Albright Road
Kokomo, Indiana 46902
Tel: 765-864-8360 Fax: 765-864-8387

Los Angeles

18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

New York

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

San Jose

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

Toronto

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

ASIA/PACIFIC

Australia

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

China - Beijing

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

China - Chengdu

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

China - Fuzhou

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

China - Shanghai

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

China - Shenzhen

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

Hong Kong

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

India

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

Japan

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

Korea

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

Singapore

Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan

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

EUROPE

Denmark

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

France

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

Germany

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

Italy

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

United Kingdom

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

01/18/02