

Using the 93LC56 and 93LC66

INTRODUCTION

The Microchip Technology Inc. 93LC56/66 are low-power 3-wire non-volatile memories and are suitable for many embedded system code and data storage applications. These devices are easily interfaced to most microcontrollers in today's market place, but Microchip's 8-bit RISC series PIC16CXX offers the best code density of any microcontroller on the market today. Using the PIC16C54, the assembly programs contained in this application note have been fully tested and provide the correct timing and 3-wire sequences to fully operate the 93LC56/66 in a PIC16CXX-based embedded application. The PIC16C54 was clocked at a 10MHz frequency. This application note is intended to provide the engineer with readily available stand-alone code modules to accomplish all of the necessary functions to utilize these devices in a low power application using the efficient PIC16C54 microcontroller.

The 93 series of devices have essentially four I/O pins:

CS	Chip Select
CLK	Clock
DI	Data In
DO	Data Out

This series of devices use a series of commands to accomplish the normal memory functions. These are READ, WRITE, EWEN, ERASE, ERAL, WRAL, EWDS. For a more detailed discussion of the function of these devices reference the appropriate data sheet and AN536, also published by Microchip Technology.

The following programs are included in this application note and are fully functional stand-alone modules. They are intended for use by those who are not already familiar with interfacing a PIC16CXX microcontroller to a 93 series device. For those with more experience, please refer to application note AN530.

3-Wire Byte Read Program

- Start Bit Routine
- Receive Data Routine
- Bit Out Routine
- Transmit Data Routine
- Power-up Routine
- Read Routine

3-Wire Byte Write Program

- Delay Routine
- Start Bit Routine
- Bit Out Routine
- Transmit Data Routine
- Power-up Routine
- Erase/Write Enable Routine (EWEN)
- Byte Write Routine
- Erase/Write Disable Routine (EWDS)

3-Wire Byte Write with Data Polling Program

- Data Polling Delay Routine
- Start Bit Routine
- Bit Out Routine
- Transmit Data Routine
- Power-up Routine
- Erase/Write Enable Routine
- Write Routine
- Erase/Write Disable Routine (EWDS)

3-Wire Sequential Read Program

- Delay Routine
- Start Bit Routine
- Bit In Routine
- Receive Data Routine
- Bit Out Routine
- Transmit Data Routine
- Power-up Routine
- Read Routine

*Author: Bruce Negley
Memory Products Division*

Using the 93LC56 and 93LC66

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

```
Line   PC   Opcode

0001                                     LIST P=16C54,c=132
0002                                     ;*****
0003                                     ;       3-Wire Byte Read Program (80 bytes)
0004                                     ;
0005                                     ;       This program demonstrates how to interface a
0006                                     ;       Microchip PIC16C54 to a 93LC56 or 93LC66 Serial EE
0007                                     ;       device. This program will read 8 consecutive addresses
0008                                     ;       in the 'random read' mode. This means that the opcode
0009                                     ;       and address for each byte will be sent to the device.
0010                                     ;       This program will repeat forever.
0011                                     ;
0012                                     ;       Another, more efficient method of reading consecutive
0013                                     ;       addresses is called the 'sequential read' mode. This
0014                                     ;       involves sending the opcode and address for the first
0015                                     ;       byte to read, then continuing to provide clocks for the
0016                                     ;       next addresses. The device will automatically increment
0017                                     ;       the address. An example of the sequential read mode is
0018                                     ;       provided in the '3wseqr.asm' file.
0019                                     ;
0020                                     ;       This program communicates to the serial EE in the
0021                                     ;       x16 mode, and ASSUMES THE USER HAS SET THE ORG PIN
0022                                     ;       ON THE DEVICE TO Vcc.
0023                                     ;
0024                                     ;       Timing is based on using the PIC16C54 in 'XT' mode
0025                                     ;       using a 4Mhz crystal. Clock speeds to the serial EE
0026                                     ;       will be approximately 50 kHz for this setup.
0027                                     ;
0028                                     ;       PIC16C54 to Serial EE Connections:
0029                                     ;
0030                                     ;       PIC16C54           Serial EE
0031                                     ;
0032                                     ;       Pin 10 (RB4) -> Chip Select
0033                                     ;       Pin 11 (RB5) -> Clock
0034                                     ;       Pin 12 (RB6) -> Data In
0035                                     ;       Pin 13 (RB7) -> Data Out
0036                                     ;
0037                                     ;
0038                                     ;*****
0039                                     ;       Register Assignments
0040                                     ;*****
0041         0003 status equ 3h       ; status register
0042         0005 port_a equ 5h       ; port 5 (port_a)
0043         0006 port_b equ 6h       ; port 6 (port b) comm lines to serial EE
0044         000A eeprom equ 0ah      ; bit buffer
0045         000C addr equ 0ch        ; address register
0046         000D datai equ 0dh       ; stored data input reg.
0047         000E datao equ 0eh       ; stored data output reg.
0048         0010 txbuf equ 10h      ; transmit buffer
0049         0011 count equ 11h      ; bits transmitted so far
0050         0012 bits equ 12h       ; bits to transmit
0051         0013 bytcnt equ 13h     ; byte counter for read routine
```

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

```
Line   PC   Opcode

0052         0015 loops equ 15h     ; delay loop counter
0053         0016 loops2 equ 16h     ; delay loop counter
0054         0017 hbyte equ 17h      ; high byte for input data
0055         0018 lbyte equ 18h      ; low byte for input data
0056                                     ;*****
0057                                     ;       Bit Assignments
0058                                     ;*****
0059         0007 di equ 7           ; eeprom input
0060         0006 do equ 6           ; eeprom output
```

Using the 93LC56 and 93LC66

```

0061      0007  datout  equ   7      ; data out line (port_b)
0062      0006  datin   equ   6      ; data in line (port_b)
0063      0005  sclk    equ   5      ; clock line (port_b)
0064      0004  chpsel  equ   4      ; chip select line (port_b)
0065      ;
0066      ;*****
0067      0000          org   01ffh
0068  01FF  0A33  begin   goto   PWRUP  ; set the reset vector
0069      0000          org   000h
0070  0000  0A33          goto   PWRUP
0071      ;
0072      ;*****
0073      ;      Start Bit Subroutine
0074      ;      this routine generates a start bit
0075      ;      (Chip select and DI high when clock goes high)
0076      ;*****
0077      BSTART
0078  0001  04C6          bcf    port_b,datin  ; set datain and chipselect lines
0079  0002  0486          bcf    port_b,chpsel ; low just to check operation
0080  0003  04A6          bcf    port_b,sclk   ; make sure clock starts low too.
0081  0004  0000          nop
0082      ;
0083  0005  0586          bsf    port_b,chpsel ; set chip select line high
0084  0006  05C6          bsf    port_b,datin  ; set data in line high
0085  0007  0000          nop
0086  0008  05A6          bsf    port_b,sclk   ; set the clock line high to
0087      ; generate the start bit
0088  0009  0000          nop
0089  000A  0000          nop
0090  000B  04A6          bcf    port_b,sclk   ; set clock low again
0091  000C  0800          retlw  0
0092      ;
0093      ;*****
0094      ;      BITIN routine reads one bit of data from the
0095      ;      serial EE device and stores it in 'di'
0096      ;*****
0097      BITIN
0098  000D  05EA          bsf    eeprom,di    ; assume input bit is high
0099  000E  05A6          bsf    port_b,sclk   ; set clock line high
0100  000F  0000          nop
0101  0010  07E6          btfsz  port_b,dout   ; read the data bit
0102  0011  04EA          bcf    eeprom,di    ; input bit was low

```

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

```

Line  PC  Opcode
-----
0103  0012  04A6          bcf    port_b,sclk   ; set clock line low
0104      ;
0105  0013  0800          retlw  0
0106      ;
0107      ;*****
0108      ;      Receive data routine
0109      ;      This routine reads one byte of data from the part
0110      ;      into the 'datai' register.
0111      ;*****
0112      RX
0113  0014  006D          clrf   datai        ; clear input buffer
0114  0015  0C08          movlw  .8           ; set # bits to 8
0115  0016  0031          movwf  count
0116  0017  0403          bcf    status,0     ; make sure carry bit is low
0117  0018  036D  RXLP    rlf    datai        ; rotate the buffer left 1 bit
0118  0019  090D          call  BITIN        ; read 1 bit
0119  001A  040D          bcf    datai,0     ; assume the input bit was low
0120  001B  06EA          btfsz  eeprom,di    ; check the bit
0121  001C  050D          bsf    datai,0     ; set high if necessary
0122  001D  02F1          decfsz count       ; 8 bits done?
0123  001E  0A18          goto  RXLP        ; no, do another
0124  001F  0800          retlw  0

```

Using the 93LC56 and 93LC66

```
0125 ;
0126 ;*****
0127 ; BITOUT routine
0128 ; This routine takes one bit of data in 'do' and
0129 ; transmits it to the serial EE device
0130 ;*****
0131 BITOUT
0132 0020 07CA btfss eeprom,do ; check state of data bit
0133 0021 0A24 goto bitlow ; low, goto bitlow
0134 0022 05C6 bsf port_b,datin ; high, set datain high
0135 0023 0A25 goto clkout ; and clock it
0136 ;
0137 0024 04C6 bitlow bcf port_b,datin ; output a logic low
0138 0025 05A6 clkout bsf port_b,sclk ; set clock line high
0139 0026 0000 nop
0140 0027 04A6 bcf port_b,sclk ; return clock line low
0141 0028 0800 retlw 0
0142 ;
0143 ;*****
0144 ; Transmit Data Subroutine
0145 ; This routine takes the byte of data stored in the
0146 ; 'data0' register and transmits it to the serial EE device.
0147 ;*****
0148 TX
0149 0029 0212 movf bits,w ; set the number of bits to xmit
0150 002A 0031 movwf count
0151 ;
0152 TXLP
0153 002B 04CA bcf eeprom,do ; assume bit 7 is low
```

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

```
Line PC Opcode
0154 002C 06F0 btfsc txbuf,7 ; is bit 7 clear?
0155 002D 05CA bsf eeprom,do ; no, set data bit =1
0156 002E 0920 call BITOUT ; transmit 1 bit to serial EE
0157 002F 0370 rlf txbuf ; rotate txbuf left
0158 0030 02F1 decfsz count ; all bits done?
0159 0031 0A2B goto TXLP ; no, do another bit
0160 0032 0800 retlw 0 ; yes, jump out
0161 ;
0162 ;*****
0163 ; POWER-UP ROUTINE
0164 ; This is the program entry point, which in this case simply
0165 ; sets the port_a I/O lines and directs control to the
0166 ; read routine.
0167 ;*****
0168 PWRUP
0169 ;
0170 0033 0C00 movlw b'00000000'
0171 0034 0005 tris port_a ; set port_a as all output
0172 0035 0065 clrfs port_a ; all lines low
0173 ;
0174 0036 0C80 movlw b'10000000'
0175 0037 0006 tris port_b ; set RB7 as input, rest output;
0176 ;
0177 ; Fall through and do the read
0178 ;
0179 ;*****
0180 ; READ ROUTINE
0181 ; This routine reads 8 consecutive addresses in
0182 ; random mode starting at address 0. This is done in
0183 ; x16 mode and will repeat forever.
0184 ;*****
0185 READ
0186 ;
0187 0038 0C00 movlw .0 ; set starting address to 00
0188 0039 002C movwf addr ;
```

Using the 93LC56 and 93LC66

```
0189 003A 0C08      movlw  .8           ; set number of addresses to
0190 003B 0033      movwf  bytcnt      ; read as 8
0191                                     ;
0192 003C 0901 rbyte call  BSTART      ; generate the start bit
0193 003D 0C02      movlw  .2           ; set # bits to 2
0194 003E 0032      movwf  bits        ;
0195 003F 0C80      movlw  b'10000000' ; get opcode (10b)
0196 0040 0030      movwf  txbuf       ; into output buffer
0197 0041 0929      call  TX           ; and transmit it
0198 0042 0C08      movlw  .8           ;
0199 0043 0032      movwf  bits        ; set number of bits to 8
0200 0044 020C      movf   addr,w      ; get the address
0201 0045 0030      movwf  txbuf       ; into the output buffer
0202 0046 0929      call  TX           ; and transmit it
0203
0204 0047 0914      call  RX           ; read the high byte
```

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

```
Line  PC  Opcode
0205  0048 020D      movf   datai,w     ; move input data to w
0206  0049 0037      movwf  hbyte       ; xfer it to high byte
0207
0208  004A 0914      call  RX           ; read the low byte
0209  004B 020D      movf   datai,w     ; move input data to w
0210  004C 0037      movwf  hbyte       ; xfer it to low byte
0211
0212  004D 0486      bcf    port_b,chpsel ; clear the chip select line
0213
0214  004E 02AC      incf   addr        ; add 1 to the address
0215  004F 02F3      decfsz bytcnt      ; have all bytes been read?
0216  0050 0A3C      goto  rbyte        ; no, read another byte
0217  0051 0A38      goto  READ         ; yes, start over
0218                                     END
```

Using the 93LC56 and 93LC66

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

```
Line   PC   Opcode
0001                               LIST P=16C54,c=132
0002   ;*****
0003   ;       3-Wire Byte Write Program (106 bytes)
0004   ;
0005   ;       This program demonstrates how to interface a
0006   ;       Microchip PIC16C54 to a 93LC56 or 93LC66 Serial EE
0007   ;       device. This program will execute the erase/write enable
0008   ;       command, write to 8 consecutive addresses, and then
0009   ;       execute the erase/write disable command. This
0010   ;       sequence will repeat forever.
0011   ;
0012   ;       After each byte is written, time must be given to the
0013   ;       device for it to complete the write cycle before
0014   ;       the next command can be sent. The easiest solution
0015   ;       is to consult the data book for the maximum write
0016   ;       cycle time and just wait that long before the next
0017   ;       command is sent. This program demonstrates that
0018   ;       solution.
0019   ;
0020   ;       Another, more efficient method of determining when the
0021   ;       write cycle is complete is called 'data polling.' This
0022   ;       method is demonstrated in the program "3wdpoll."
0023   ;
0024   ;       This program communicates to the serial EE in the
0025   ;       x16 mode, and ASSUMES THE USER HAS SET THE ORG PIN
0026   ;       ON THE DEVICE TO Vcc.
0027   ;
0028   ;       Timing is based on using the PIC16C54 in 'XT' mode
0029   ;       using a 4Mhz crystal. Clock speeds to the serial EE
0030   ;       will be approximately 40 kHz for this setup.
0031   ;
0032   ;       PIC16C54 to Serial EE Connections:
0033   ;
0034   ;       PIC16C54       Serial EE
0035   ;       -----
0036   ;       Pin 10 (RB4) -> Chip Select
0037   ;       Pin 11 (RB5) -> Clock
0038   ;       Pin 12 (RB6) -> Data In
0039   ;       Pin 13 (RB7) -> Data Out
0040   ;                       ORG=Vcc
0041   ;
0042   ;*****
0043   ;       Register Assignments
0044   ;*****
0045   0005 port_a equ 5h       ; port 5 (port_a)
0046   0006 port_b equ 6h       ; port 6 (port b) comm lines to serial EE
0047   000A eeprom equ 0ah      ; bit buffer
0048   000C addr  equ 0ch       ; address register
0049   000D datai equ 0dh       ; stored data input reg.
0050   000E datao equ 0eh       ; stored data output reg.
0051   0010 txbuf equ 10h      ; transmit buffer
```

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

```
Line   PC   Opcode
0052   0011 count equ 11h      ; bits transmitted so far
0053   0012 bits  equ 12h      ; bits to transmit
0054   0013 bytcnt equ 13h      ; byte counter for write routine
0055   0015 loops equ 15h      ; delay loop counter
0056   0016 loops2 equ 16h     ; delay loop counter ;
0057   ;*****
0058   ;       Bit Assignments
0059   ;*****
0060   0007 di    equ 7        ; eeprom input
```

Using the 93LC56 and 93LC66

```
0061      0006 do      equ    6      ; eeprom output
0062      0007 datout  equ    7      ; data out line (port_b)
0063      0006 datin   equ    6      ; data in line (port_b)
0064      0005 sclk    equ    5      ; clock line (port_b)
0065      0004 chpsel  equ    4      ; chip select line (port_b)
0066      ;
0067      ;*****
0068      0000          org    01ffh
0069  01FF  0A2F begin  goto    PWRUP  ; set the reset vector
0070      0000          org    000h
0071  0000  0A2F          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                ; sit and wait
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      ;      (Chip select and DI high when clock goes high)
0100      ;*****
0101      BSTART
0102  000E  0C8F          movlw   b'10001111'
```

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

```
Line  PC  Opcode
0103  000F  0006      tris   port_b      ; set port b for output
0104      ;
0105  0010  04C6      bcf   port_b,datin ; set datain and chipselect lines
0106  0011  0486      bcf   port_b,chpsel ; low just to check operation
0107  0012  04A6      bcf   port_b,sclk  ; make sure clock starts low too.
0108  0013  0000      nop
0109      ;
0110  0014  0586      bsf   port_b,chpsel ; set chip select line high
0111  0015  05C6      bsf   port_b,datin  ; set data in line high
0112  0016  0000      nop
0113  0017  05A6      bsf   port_b,sclk  ; set the clock line high to
0114      ; generate the start bit
0115  0018  0000      nop
0116  0019  0000      nop
0117  001A  04A6      bcf   port_b,sclk  ; set clock low again
0118  001B  0800      retlw  0
0119      ;
0120      ;*****
0121      ;      BITOUT routine
0122      ;      This routine takes one bit of data in 'do' and
0123      ;      transmits it to the serial EE device
0124      ;*****
```

Using the 93LC56 and 93LC66

```
0125          BITOUT
0126 001C 07CA      btfss  eeprom,do      ; check state of data bit
0127 001D 0A20      goto   bitlow      ; low, goto bitlow
0128 001E 05C6      bsf    port_b,datin  ; high, set datain high
0129 001F 0A21      goto   clkout      ; and clock it
0130
0131 0020 04C6      bitlow  bcf    port_b,datin  ; output a logic low
0132 0021 05A6      clkout  bsf    port_b,sclk   ; set clock line high
0133 0022 0000      nop
0134 0023 04A6      bcf    port_b,sclk   ; return clock line low
0135 0024 0800      retlw  0
0136
0137          ;
0138          ; *****
0139          ; Transmit Data Subroutine
0140          ; This routine takes the byte of data stored in the
0141          ; 'datao' register and transmits it to the serial EE device.
0142          ; *****
0142          TX
0143 0025 0212      movf   bits,w        ; set the number of bits to xmit
0144 0026 0031      movwf  count
0145
0146          ;
0146          TXLP
0147 0027 04CA      bcf    eeprom,do     ; assume bit 7 is low
0148 0028 06F0      btfsc  txbuf,7      ; is bit 7 clear?
0149 0029 05CA      bsf    eeprom,do     ; no, set data bit =1
0150 002A 091C      call   BITOUT        ; transmit 1 bit to serial EE
0151 002B 0370      rlf    txbuf        ; rotate txbuf left
0152 002C 02F1      decfsw count        ; all bits done?
0153 002D 0A27      goto   TXLP         ; no, do another bit
```

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

```
Line  PC  Opcode
0154 002E 0800      retlw  0            ; yes, jump out
0155
0156          ;
0156          ; *****
0157          ; POWER-UP ROUTINE
0158          ; This is the program entry point, which in this case simply
0159          ; sets the port_a I/O lines and directs control to the
0160          ; erase/write enable routine.
0161          ; *****
0162          PWRUP
0163          ;
0164 002F 0C00      movlw  b'00000000'
0165 0030 0005      tris  port_a        ; set port_a as all output
0166 0031 0065      clrf  port_a        ; all lines low
0167
0168 0032 0C80      movlw  b'10000000'
0169 0033 0006      tris  port_b        ; set RB7 as input, rest output;
0170
0171          ; Fall through and do erase/write enable
0172          ;
0173          ; *****
0174          ; EWEN (Erase/Write ENable Routine)
0175          ; this routine enables the dut for erasing and
0176          ; writing. This must be done prior to any erase,write
0177          ; eral,wral instructions.
0178          ; *****
0179          EWEN
0180          ;
0181 0034 090E      call   BSTART       ; generate a start bit
0182
0183 0035 0C02      movlw  .2            ; set # bits to 2
0184 0036 0032      movwf  bits         ;
0185 0037 0C00      movlw  b'00000000'  ; get the opcode (00b)
0186 0038 0030      movwf  txbuf        ; into the output buffer
0187 0039 0925      call   TX           ; and transmit it
0188 003A 0C08      movlw  .8            ; set # bits to 8
```


Using the 93LC56 and 93LC66

```
0189 003B 0032      movwf  bits      ;
0190 003C 0CC0      movlw  b'11000000' ; get opcode and address
0191                ;      (11XXXXXX)
0192 003D 0030      movwf  txbuf     ; into output buffer
0193 003E 0925      call  TX         ; and transmit it
0194 003F 0486      bcf   port_b,chpsel ; set chip select line low
0195 0040 0000      nop
0196                ;
0197                ;      Now continue on to the write command
0198                ;
0199                ;*****
0200                ;      Byte Write Routine
0201                ;      This routine writes an AA55h pattern into
0202                ;      8 consecutive addresses starting at address 00.
0203                ;      A delay of about 10ms is given after each byte
0204                ;      for the write cycle to complete.
```

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

```
Line  PC  Opcode
0205                ;      The write is done in the x16 mode: the user must
0206                ;      have the ORG pin tied to Vcc on the device
0207                ;*****
0208                WRITE
0209                ;
0210 0041 0C00      movlw  .0        ; set starting address to 00
0211 0042 002C      movwf  addr     ;
0212 0043 0C08      movlw  .8        ; set number of bytes to write as 8
0213 0044 0033      movwf  bytcnt   ;
0214                ;
0215 0045 090E topwr  call  BSTART     ; generate the start bit
0216                ;
0217 0046 0C02      movlw  .2        ; set # bits to 2 for the opcode
0218 0047 0032      movwf  bits     ;
0219 0048 0C40      movlw  b'01000000' ; get opcode (01b)
0220 0049 0030      movwf  txbuf    ; into the transmit buffer
0221 004A 0925      call  TX        ; and send it
0222                ;
0223 004B 0C08      movlw  .8        ; set # of bits to 8 for the
0224 004C 0032      movwf  bits     ; address
0225 004D 020C      movf   addr,w   ; get address counter
0226 004E 0030      movwf  txbuf    ; into output buffer
0227 004F 0925      call  TX        ; and send it
0228 0050 0CAA      movlw  b'10101010' ; get upper byte of data (AAh)
0229 0051 0030      movwf  txbuf    ; into the transmit buffer
0230 0052 0925      call  TX        ; and send it
0231 0053 0C55      movlw  b'01010101' ; get lower byte of data (55h)
0232 0054 0030      movwf  txbuf    ; into transmit buffer
0233 0055 0925      call  TX        ; and send it
0234                ;
0235 0056 0486      bcf   port_b,chpsel ; clear the chip select line
0236                ;      to initiate write cycle
0237                ;
0238 0057 0C0A      movlw  .10       ;
0239 0058 0035      movwf  loops    ; set delay time to 10mS
0240 0059 0901      call  WAIT      ; and wait
0241                ;
0242 005A 02AC      incf  addr      ; increment address counter
0243 005B 02F3      decfsz bytcnt   ; all bytes written?
0244 005C 0A45      goto  topwr     ; no, do another
0245                ;      yes, go on
0246                ;
0247                ;      Now continue on to the erase/write disable command
0248                ;
0249                ;*****
0250                ;      EWDS (Erase/Write Disable Routine)
0251                ;      This routine executes the erase/write disable command
0252                ;      which prevents the contents of the array from being
```

Using the 93LC56 and 93LC66

```
0253           ;      written to.
0254           ;
0255           ;*****
```

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

```
Line  PC   Opcode
0256           EWDS
0257           ;
0258 005D 090E      call   BSTART      ; generate a start bit
0259           ;
0260 005E 0C02      movlw  .2          ; set # bits to 2
0261 005F 0032      movwf  bits        ;
0262 0060 0C00      movlw  b'00000000' ; get the opcode (00b)
0263 0061 0030      movwf  txbuf       ; into the output buffer
0264 0062 0925      call   TX          ; and transmit it
0265 0063 0C08      movlw  .8          ; set # bits to 8
0266 0064 0032      movwf  bits        ;
0267 0065 0C00      movlw  b'00000000' ; get opcode and address
0268           ;      (00XXXXXX)
0269 0066 0030      movwf  txbuf       ; into output buffer
0270 0067 0925      call   TX          ; and transmit it
0271 0068 0486      bcf   port_b,chpsel ; set chip select line low
0272 0069 0000      nop
0273 006A 0A34      goto  EWEN        ; start all over
0274           ;
0275           0000  END
```

Using the 93LC56 and 93LC66

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:49:00 1994 Page 1

```
Line  PC  Opcode
0001          LIST P=16C54,c=132
0002          ;*****
0003          ; 3-Wire Byte Write with Data Poll Program (107 bytes)
0004          ;
0005          ; This program demonstrates how to interface a
0006          ; Microchip PIC16C54 to a 93LC56 or 93LC66 Serial EE
0007          ; device. This program will execute the erase/write enable
0008          ; command, write to 8 consecutive addresses, and then
0009          ; use the 'data polling' method to determine when the
0010          ; write cycle is complete. The program then executes
0011          ; the erase/write disable command. This sequence will
0012          ; repeat forever.
0013          ;
0014          ; When writing to a 3-wire serial EE device, the
0015          ; internally timed write cycle will begin after
0016          ; the opcode, address and data are sent to the
0017          ; device. There are two ways to make sure that the
0018          ; cycle is complete before you send it another command.
0019          ; The simplest method is to simply wait for the maximum
0020          ; write cycle time, which can be obtained from the data book.
0021          ; A more efficient method is "Data polling." This is
0022          ; done by toggling the chip select line low and then
0023          ; back high after the opcode, address and data are sent.
0024          ; The chip select line must be low for at least 250ns
0025          ; before bringing it high again. The device will pull
0026          ; the data out line low at that point, and set it high
0027          ; when the write cycle is complete. The user can then check
0028          ; or "poll" the dataout line until it goes high before
0029          ; sending the next command.
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          ; timeout indicator. This LED will come on if the
0034          ; data polling is unsuccessful and the device being
0035          ; programmed does not respond. This can be tested by
0036          ; simply running the program with no part in the socket.
0037          ;
0038          ; This program communicates to the serial EE in the
0039          ; x16 mode, and ASSUMES THE USER HAS SET THE ORG PIN
0040          ; ON THE DEVICE TO Vcc.
0041          ;
0042          ; Timing is based on using the PIC16C54 in 'XT' mode
0043          ; using a 4Mhz crystal. Clock speeds to the serial EE
0044          ; will be approximately 50 kHz for this setup.
0045          ;
0046          ; PIC16C54 to Serial EE Connections:
0047          ;
0048          ; PIC16C54      Serial EE
0049          ; _____
0050          ; Pin 10 (RB4) -> Chip Select
0051          ; Pin 11 (RB5) -> Clock
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:49:00 1994 Page 2

```
Line  PC  Opcode
0052          ; Pin 12 (RB6) -> Data In
0053          ; Pin 13 (RB7) -> Data Out
0054          ; ORG=Vcc
0055          ;
0056          ;*****
0057          ; Register Assignments
0058          ;*****
0059          0005 port_a equ 5h ; port 5 (port_a)
0060          0006 port_b equ 6h ; port 6 (port b) used comm lines to serial EE
```

Using the 93LC56 and 93LC66

```
0061      000A eeprom equ 0ah    ; bit buffer
0062      000C addr  equ 0ch    ; address register
0063      000D datai equ 0dh    ; stored data input reg.
0064      000E datao equ 0eh    ; stored data output reg.
0065      0010 txbuf  equ 10h   ; transmit buffer
0066      0011 count  equ 11h   ; bits transmitted so far
0067      0012 bits   equ 12h   ; bits to transmit
0068      0013 bytcnt equ 13h   ; byte counter for write routine
0069      0015 loops  equ 15h   ; delay loop counter
0070      0016 loops2 equ 16h   ; delay loop counter
0071      ;*****
0072      ;          Bit Assignments
0073      ;*****
0074      0007 di      equ 7      ; eeprom input
0075      0006 do      equ 6      ; eeprom output
0076      0007 datout  equ 7      ; data out line (port_b)
0077      0006 datin  equ 6      ; data in line (port_b)
0078      0005 sclk   equ 5      ; clock line (port_b)
0079      0004 chpsel  equ 4      ; chip select line (port_b)
0080      0001 timeout equ 1      ; write cycle timeout warning (port_a)
0081      ;
0082      ;*****
0083      0000 org     01ffh
0084      01FF 0A26 begin goto PWRUP ; set the reset vector
0085      0000 org     000h
0086      0000 0A26      goto PWRUP
0087      ;
0088      ;*****
0089      ;          DATA POLL DELAY ROUTINE
0090      ;          This routine delays 100 us is
0091      ;          used for delay between polls
0092      ;*****
0093      dpdelay
0094      ;
0095      0001 0C19      movlw .25    ; timing adjustment variable
0096      0002 0036      movwf loops2
0097      0003 0000 top   nop
0098      0004 02F6      decfsz loops2 ; loops complete?
0099      0005 0A03      goto top    ; no, go again
0100      ;
0101      0006 0800      retlw 0    ; yes, return from sub
0102      ;
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:49:00 1994 Page 3

```
Line PC Opcod
0103      ;*****
0104      ;          Start Bit Subroutine
0105      ;          this routine generates a start bit
0106      ;          (Chip select and DI high when clock goes high)
0107      ;*****
0108      BSTART
0109      0007 04C6      bcf port_b,datin ; set datin and chipselect lines
0110      0008 0486      bcf port_b,chpsel ; low just to check operation
0111      0009 04A6      bcf port_b,sclk  ; make sure clock starts low too.
0112      000A 0000      nop
0113      ;
0114      000B 0586      bsf port_b,chpsel ; set chip select line high
0115      000C 05C6      bsf port_b,datin ; set data in line high
0116      000D 0000      nop
0117      000E 05A6      bsf port_b,sclk  ; set the clock line high to
0118      ; generate the start bit
0119      000F 0000      nop
0120      0010 0000      nop
0121      0011 04A6      bcf port_b,sclk  ; set clock low again
0122      0012 0800      retlw 0
0123      ;
0124      ;*****
```

Using the 93LC56 and 93LC66

```
0125      ;          BITOUT routine
0126      ;          This routine takes one bit of data in 'do' and
0127      ;          transmits it to the serial EE device
0128      ;*****
0129      BITOUT
0130 0013 07CA          btfss  eeprom,do      ; check state of data bit
0131 0014 0A17          goto   bitlow       ; low, goto bitlow
0132 0015 05C6          bsf    port_b,datin   ; high, set datain high
0133 0016 0A18          goto   clkout       ; and clock it
0134      ;
0135 0017 04C6 bitlow  bcf    port_b,datin   ; output a logic low
0136 0018 05A6 clkout  bsf    port_b,sclk    ; set clock line high
0137 0019 0000          nop
0138 001A 04A6          bcf    port_b,sclk    ; return clock line low
0139 001B 0800          retlw  0
0140      ;
0141      ;*****
0142      ;          Transmit Data Subroutine
0143      ;          This routine takes the byte of data stored in the
0144      ;          'datao' register and transmits it to the serial EE device.
0145      ;*****
0146      TX
0147 001C 0212          movf   bits,w          ; set the number of bits to xmit
0148 001D 0031          movwf  count
0149      ;
0150      TXLP
0151 001E 04CA          bcf    eeprom,do      ; assume bit 7 is low
0152 001F 06F0          btfsc  txbuf,7        ; is bit 7 clear?
0153 0020 05CA          bsf    eeprom,do      ; no, set data bit =1
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:49:00 1994 Page 4

```
Line  PC  Opcode
0154  0021  0913          call  BITOUT          ; transmit 1 bit to serial EE
0155  0022  0370          rlf   txbuf          ; rotate txbuf left
0156  0023  02F1          decfsz count         ; all bits done?
0157  0024  0A1E          goto  TXLP          ; no, do another bit
0158  0025  0800          retlw 0             ; yes, jump out
0159      ;
0160      ;*****
0161      ;          POWER-UP ROUTINE
0162      ;          This is the program entry point, which in this case simply
0163      ;          sets the port_a I/O lines and directs control to the
0164      ;          erase/write enable routine.
0165      ;*****
0166      PWRUP
0167      ;
0168 0026 0C00          movlw  b'00000000'
0169 0027 0005          tris  port_a        ; set port_a as all output
0170 0028 0065          clrf  port_a        ; all lines low
0171      ;
0172 0029 0C80          movlw  b'10000000'
0173 002A 0006          tris  port_b        ; set RB7 as input, rest output;
0174      ;
0175      ;          Fall through and do erase/write enable
0176      ;
0177      ;*****
0178      ;          EWEN (Erase/Write ENable Routine)
0179      ;          this routine enables the dut for erasing and
0180      ;          writing. This must be done prior to any erase,write
0181      ;          eral,wral instructions.
0182      ;*****
0183      EWEN
0184      ;
0185 002B 0907          call  BSTART         ; generate a start bit
0186      ;
0187 002C 0C02          movlw  .2           ; set # bits to 2
0188 002D 0032          movwf  bits         ;
```

Using the 93LC56 and 93LC66

```
0189 002E 0C00      movlw  b'00000000'   ; get the opcode (00b)
0190 002F 0030      movwf  txbuf         ; into the output buffer
0191 0030 091C      call   TX            ; and transmit it
0192 0031 0C08      movlw  .8            ; set # bits to 8
0193 0032 0032      movwf  bits         ;
0194 0033 0CC0      movlw  b'11000000'   ; get opcode and address
0195                ;      (11XXXXXX)
0196 0034 0030      movwf  txbuf         ; into output buffer
0197 0035 091C      call   TX            ; and transmit it
0198 0036 0486      bcf    port_b,chpsel ; set chip select line low
0199 0037 0000      nop
0200                ;
0201                ;      Now continue on to the write command
0202                ;
0203                ;*****
0204                ;      WRITE
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:49:00 1994 Page 5

```
Line  PC  Opcode
-----
0205                ;      This routine writes a AA55h pattern into
0206                ;      8 consecutive addresses starting at address 00.
0207                ;      It will then 'poll' the data out line to determine
0208                ;      when the write cycle is complete.  If the cycle has
0209                ;      not completed within 20 ms, then it will continue
0210                ;      anyway and turn the timeout fail LED on.
0211                ;      The write is done in the x16 mode: the user must
0212                ;      have the ORG pin tied to Vcc on the device. This
0213                ;      program will repeat forever.
0214                ;*****
0215                ;      WRITE
0216                ;
0217 0038 0C00      movlw  .0            ; set starting address to 00
0218 0039 002C      movwf  addr         ;
0219 003A 0C08      movlw  .8            ; set number of bytes to write as 8
0220 003B 0033      movwf  bytcnt       ;
0221                ;
0222 003C 0907  topwr  call   BSTART        ; generate the start bit
0223                ;
0224 003D 0C02      movlw  .2            ; set # bits to 2 for the opcode
0225 003E 0032      movwf  bits         ;
0226 003F 0C40      movlw  b'01000000'   ; get opcode (01b)
0227 0040 0030      movwf  txbuf         ; into the transmit buffer
0228 0041 091C      call   TX            ; and send it
0229                ;
0230 0042 0C08      movlw  .8            ; set # of bits to 8 for the
0231 0043 0032      movwf  bits         ; address
0232 0044 020C      movf   addr,w        ; get address counter
0233 0045 0030      movwf  txbuf         ; into output buffer
0234 0046 091C      call   TX            ; and send it
0235 0047 0CAA      movlw  b'10101010'   ; get first half of data (AAh)
0236 0048 0030      movwf  txbuf         ; into the transmit buffer
0237 0049 091C      call   TX            ; and send it
0238 004A 0C55      movlw  b'01010101'   ; get second half of data (55h)
0239 004B 0030      movwf  txbuf         ; into transmit buffer
0240 004C 091C      call   TX            ; and send it
0241                ;
0242 004D 0486      bcf    port_b,chpsel ; clear the chip select line
0243                ;      to initiate write cycle
0244 004E 0000      nop
0245 004F 0586      bsf    port_b,chpsel ; set the chip sel line high
0246                ;      to begin data polling
0247 0050 0CC8      movlw  .200         ;
0248 0051 0035      movwf  loops        ; poll 100 times before timeout(about 20ms)
0249 0052 0901  polltop  call   dpdelay       ; wait 100us
0250 0053 06E6      btfsc  port_b,datout ; is the data out line high?
0251 0054 0A59      goto  okpoll        ; yes-cycle complete: do another
0252 0055 02F5      decf   loops        ; has it timed out?
```

Using the 93LC56 and 93LC66

```
0253 0056 0A52      goto    polltop      ; no, check again
0254 0057 0525      bsf     port_a,timeout; yes, set timeout LED and go on
0255 0058 0A5A      goto    badpoll     ;
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:49:00 1994 Page 6

```
Line  PC   Opcode
0256
0257 0059 0425  okpoll  bcf     port_a,timeout; clear the timeout LED
0258 005A 0486  badpoll bcf     port_b,chpsel ; chip sel line back low
0259 005B 02AC      incf   addr         ; increment address counter
0260 005C 02F3      decfsz bytcnt      ; all bytes written?
0261 005D 0A3C      goto   topwr       ; no, do another
0262                          ; yes, go on
0263                          ;
0264                          ;   Now continue on to the erase/write disable command
0265                          ;
0266 ;*****
0267 ;   EWDS (Erase/Write Disable Routine)
0268 ;   This routine executes the erase/write disable command
0269 ;   which prevents the contents of the array from being
0270 ;   written to.
0271 ;
0272 ;*****
0273 EWDS
0274 ;
0275 005E 0907      call   BSTART      ; generate a start bit
0276 ;
0277 005F 0C02      movlw  .2          ; set # bits to 2
0278 0060 0032      movwf  bits        ;
0279 0061 0C00      movlw  b'00000000' ; get the opcode (00b)
0280 0062 0030      movwf  txbuf       ; into the output buffer
0281 0063 091C      call   TX          ; and transmit it
0282 0064 0C08      movlw  .8          ; set # bits to 8
0283 0065 0032      movwf  bits        ;
0284 0066 0C00      movlw  b'00000000' ; get opcode and address
0285                          ;   (00XXXXXX)
0286 0067 0030      movwf  txbuf       ; into output buffer
0287 0068 091C      call   TX          ; and transmit it
0288 0069 0486      bcf     port_b,chpsel ; set chip select line low
0289 006A 0000      nop
0290 006B 0A2B      goto   EWEN        ; start all over
0291 ;
0292 ;
0293
0294      0000  END
```

Using the 93LC56 and 93LC66

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:49:20 1994 Page 1

```
Line  PC  Opcode
0001                                LIST P=16C54,c=132
0002                                ;*****
0003                                ;    3-Wire Sequential Read Program (93 bytes)
0004                                ;
0005                                ;    This program demonstrates how to interface a
0006                                ;    Microchip PIC16C54 to a 93LC56 or 93LC66 Serial EE
0007                                ;    device. This program will read 8 consecutive addresses
0008                                ;    in the 'sequential read' mode. This means that the opcode
0009                                ;    and address are sent for the first address only. After
0010                                ;    the first address is read, the Chip Select line is left
0011                                ;    high and clocks for the remaining 7 bytes are sent to the
0012                                ;    device. The device will automatically increment the address
0013                                ;    pointer in this mode, allowing the user to read as many
0014                                ;    consecutive addresses as needed. This program will repeat
0015                                ;    forever.
0016                                ;
0017                                ;    This program communicates to the serial EE in the
0018                                ;    x16 mode, and ASSUMES THE USER HAS SET THE ORG PIN
0019                                ;    ON THE DEVICE TO Vcc.
0020                                ;
0021                                ;    Timing is based on using the PIC16C54 in 'XT' mode
0022                                ;    using a 4MHz crystal. Clock speeds to the serial EE
0023                                ;    will be approximately 50 kHz for this setup.
0024                                ;
0025                                ;    PIC16C54 to Serial EE Connections:
0026                                ;
0027                                ;    PIC16C54          Serial EE
0028                                ;    _____          _____
0029                                ;    Pin 10 (RB4) -> Chip Select
0030                                ;    Pin 11 (RB5) -> Clock
0031                                ;    Pin 12 (RB6) -> Data In
0032                                ;    Pin 13 (RB7) -> Data Out
0033                                ;                                ORG = Vcc
0034                                ;
0035                                ;*****
0036                                ;    Register Assignments
0037                                ;*****
0038    0003  status  equ   3h    ; status register
0039    0005  port_a  equ   5h    ; port 5 (port_a)
0040    0006  port_b  equ   6h    ; port 6 (port b) used comm lines to serial EE
0041    000A  eeprom equ   0ah   ; bit buffer
0042    000C  addr   equ   0ch   ; address register
0043    000D  datai  equ   0dh   ; stored data input reg.
0044    000E  datao  equ   0eh   ; stored data output reg.
0045    0010  txbuf  equ   10h   ; transmit buffer
0046    0011  count  equ   11h   ; bits transmitted so far
0047    0012  bits   equ   12h   ; bits to transmit
0048    0013  bytcnt equ   13h   ; byte counter for read routine
0049    0015  loops  equ   15h   ; delay loop counter
0050    0016  loops2 equ   16h   ; delay loop counter
0051    0017  hbyte  equ   17h   ; high byte for input data
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:49:20 1994 Page 2

```
Line  PC  Opcode
0052    0018  lbyte  equ   18h   ; low byte for input data
0053                                ;*****
0054                                ;    Bit Assignments
0055                                ;*****
0056    0007  di     equ    7     ; eeprom input
0057    0006  do     equ    6     ; eeprom output
0058    0007  datout equ    7     ; data out line (port_b)
0059    0006  datin  equ    6     ; data in line (port_b)
0060    0005  sclk   equ    5     ; clock line (port_b)
```


Using the 93LC56 and 93LC66

```
0061      0004  chpsel equ   4      ; chip select line (port_b)
0062      ;
0063      ;*****
0064      0000      org   01ffh
0065      01FF  0A40  begin   goto   PWRUP   ; set the reset vector
0066      0000      org   000h
0067      0000  0A40      goto   PWRUP
0068      ;
0069      ;*****
0070      ;      DELAY ROUTINE
0071      ;      This routine takes the value in 'loops'
0072      ;      and multiplies it times 1 millisecond to
0073      ;      determine delay time.
0074      ;*****
0075      WAIT
0076      ;
0077      0001  0C6E  top      movlw   .110      ; timing adjustment variable
0078      0002  0036      movwf   loops2
0079      0003  0000  top2     nop                ; sit and wait
0080      0004  0000      nop
0081      0005  0000      nop
0082      0006  0000      nop
0083      0007  0000      nop
0084      0008  0000      nop
0085      0009  02F6      decfsz loops2      ; inner loops complete?
0086      000A  0A03      goto   top2        ; no, go again
0087      ;
0088      000B  02F5      decfsz loops      ; outer loops complete?
0089      000C  0A01      goto   top         ; no, go again
0090      000D  0800      retlw  0           ; yes, return from sub
0091      ;
0092      ;*****
0093      ;      Start Bit Subroutine
0094      ;      this routine generates a start bit
0095      ;      (Chip select and DI high when clock goes high)
0096      ;*****
0097      BSTART
0098      000E  04C6      bcf    port_b,datin ; set datain and chipselect lines
0099      000F  0486      bcf    port_b,chpsel ; low just to check operation
0100      0010  04A6      bcf    port_b,sclk  ; make sure clock starts low too.
0101      0011  0000      nop
0102      ;
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:49:20 1994 Page 3

```
Line  PC  Opcode
0103  0012  0586      bsf    port_b,chpsel ; set chip select line high
0104  0013  05C6      bsf    port_b,datin  ; set data in line high
0105  0014  0000      nop
0106  0015  05A6      bsf    port_b,sclk   ; set the clock line high to
0107      ; generate the start bit
0108  0016  0000      nop
0109  0017  0000      nop
0110  0018  04A6      bcf    port_b,sclk   ; set clock low again
0111  0019  0800      retlw  0
0112      ;
0113      ;*****
0114      ;      BITIN routine reads one bit of data from the
0115      ;      serial EE device and stores it in 'di'
0116      ;*****
0117      BITIN
0118  001A  05EA      bsf    eeprom,di    ; assume input bit is high
0119  001B  05A6      bsf    port_b,sclk   ; set clock line high
0120  001C  0000      nop
0121  001D  07E6      btfs   port_b,datout ; read the data bit
0122  001E  04EA      bcf    eeprom,di    ; input bit was low
0123  001F  04A6      bcf    port_b,sclk   ; set clock line low
0124      ;
```

Using the 93LC56 and 93LC66

```
0125 0020 0800      retlw  0          ;
0126
0127      ;*****
0128      ;      Receive data routine
0129      ;      This routine reads one byte of data from the part
0130      ;      into the 'data1' register.
0131      ;*****
0132      RX
0133 0021 006D      clrf   data1      ; clear input buffer
0134 0022 0C08      movlw  .8          ; set # bits to 8
0135 0023 0031      movwf  count      ;
0136 0024 0403      bcf   status,0    ; make sure carry bit is low
0137 0025 036D      RXLP   rlf   data1      ; rotate the buffer left 1 bit
0138 0026 091A      call  BITIN       ; read 1 bit
0139 0027 040D      bcf   data1,0     ; assume the input bit was low
0140 0028 06EA      btfsz eeprom,di  ; check the bit
0141 0029 050D      bsf   data1,0     ; set high if necessary
0142 002A 02F1      decfsz count      ; 8 bits done?
0143 002B 0A25      goto  RXLP       ; no, do another
0144 002C 0800      retlw  0          ;
0145
0146      ;*****
0147      ;      BITOUT routine
0148      ;      This routine takes one bit of data in 'do' and
0149      ;      transmits it to the serial EE device
0150      ;*****
0151      BITOUT
0152 002D 07CA      btfss  eeprom,do  ; check state of data bit
0153 002E 0A31      goto  bitlow     ; low, goto bitlow
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:49:20 1994 Page 4

```
Line  PC  Opcode
0154 002F 05C6      bsf   port_b,datin ; high, set datain high
0155 0030 0A32      goto  clkout      ; and clock it
0156
0157 0031 04C6      bitlow bcf   port_b,datin ; output a logic low
0158 0032 05A6      clkout bsf   port_b,sclk  ; set clock line high
0159 0033 0000      nop
0160 0034 04A6      bcf   port_b,sclk  ; return clock line low
0161 0035 0800      retlw  0          ;
0162
0163      ;*****
0164      ;      Transmit Data Subroutine
0165      ;      This routine takes the byte of data stored in the
0166      ;      'data0' register and transmits it to the serial EE device.
0167      ;*****
0168      TX
0169 0036 0212      movf   bits,w     ; set the number of bits to xmit
0170 0037 0031      movwf  count      ;
0171
0172      TXLP
0173 0038 04CA      bcf   eeprom,do   ; assume bit 7 is low
0174 0039 06F0      btfsz txbuf,7    ; is bit 7 clear?
0175 003A 05CA      bsf   eeprom,do   ; no, set data bit =1
0176 003B 092D      call  BITOUT      ; transmit 1 bit to serial EE
0177 003C 0370      rlf   txbuf       ; rotate txbuf left
0178 003D 02F1      decfsz count      ; all bits done?
0179 003E 0A38      goto  TXLP       ; no, do another bit
0180 003F 0800      retlw  0          ; yes, jump out
0181
0182      ;*****
0183      ;      POWER-UP ROUTINE
0184      ;      This is the program entry point, which in this case simply
0185      ;      sets the port_a I/O lines and directs control to the
0186      ;      erase/write enable routine.
0187      ;*****
0188      PWRUP
```

Using the 93LC56 and 93LC66

```
0189          ;
0190 0040 0C11      movlw  b'00010001'
0191 0041 0005      tris   port_a      ; set RA0 as input, rest output
0192 0042 0065      clrfs port_a      ; all lines low
0193 0043 0C80      movlw  b'10000000'
0194 0044 0006      tris   port_b      ; set RB7 as input, rest output;
0195          ;
0196          ;          Fall through and do the read
0197          ;
0198          ;*****
0199          ;          READ ROUTINE (Sequential read mode)
0200          ;          This routine reads 8 consecutive addresses in
0201          ;          sequential mode starting at address 0. This
0202          ;          program will repeat forever
0203          ;*****
0204          READ
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:49:20 1994 Page 5

```
Line  PC  Opcode
0205          ;
0206 0045 0C00      movlw  .0          ; set starting address to 00
0207 0046 002C      movwf  addr      ;
0208 0047 0C08      movlw  .8          ; set number of addresses to
0209 0048 0033      movwf  bytcnt    ; read as 8
0210          ;
0211 0049 090E      call  BSTART     ; generate the start bit
0212 004A 0C02      movlw  .2          ; set # bits to 2
0213 004B 0032      movwf  bits      ;
0214 004C 0C80      movlw  b'10000000' ; get opcode (10b)
0215 004D 0030      movwf  txbuf     ; into output buffer
0216 004E 0936      call  TX         ; and transmit it
0217 004F 0C08      movlw  .8          ;
0218 0050 0032      movwf  bits      ; set number of bits to 8
0219 0051 020C      movf   addr,w    ; get the address
0220 0052 0030      movwf  txbuf     ; into the output buffer
0221 0053 0936      call  TX         ; and transmit it
0222          ;
0223 0054 0921  rbyte call  RX         ; read the high byte
0224 0055 020D      movf   datai,w   ; move input data to w
0225 0056 0037      movwf  hbyte     ; xfer it to high byte
0226          ;
0227 0057 0921      call  RX         ; read the low byte
0228 0058 020D      movf   datai,w   ; move input data to w
0229 0059 0037      movwf  hbyte     ; xfer it to low byte
0230          ;
0231 005A 02AC      incf  addr      ; add 1 to the address
0232 005B 02F3      decfsz bytcnt   ; have all bytes been read?
0233 005C 0A54      goto  rbyte     ; no, read another byte
0234          ;
0235 005D 0486      bcf   port_b,chpsel ; yes,clear the chip select line
0236 005E 0A45      goto  READ     ; and start over
0237 0000 0000  END
```

Using the 93LC56 and 93LC66

NOTES:



WORLDWIDE SALES & SERVICE

AMERICAS

Corporate Office

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

Atlanta

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

Boston

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

Chicago

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

Dallas

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

Dayton

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

Los Angeles

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

New York

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

AMERICAS (continued)

San Jose

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

ASIA/PACIFIC

Hong Kong

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

Korea

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

Singapore

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

Taiwan

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

EUROPE

United Kingdom

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

France

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

Germany

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

Italy

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

JAPAN

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

9/22/95

All rights reserved. © 1995, Microchip Technology Incorporated, USA.

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