

Interfacing the 24LCXXB Serial EEPROMs

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:23 1994 Page 1

```
Line  PC  Opcode

0001          LIST P=16C54,c=132
0002          ;*****
0003          ; 2-Wire Byte Read Program (118 bytes)
0004          ;
0005          ; This program demonstrates how to interface a
0006          ; Microchip PIC16C54 to a 24LCXX 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 read command for every 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 '2wseqr.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 3h      ; status register
0043          0005 port_a equ 5h      ; port 5 (port a) used for LED display
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      ; 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 10:48:23 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
0058          ;*****
0059          ; Bit Definitions
0060          ;*****
```

Interfacing the 24LCXXB Serial EEPROMs

```
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 0A5E          goto   PWRUP
0070      0000          org    000h    ;
0071      0000 0A5E          goto   PWRUP
0072      ;
0073      ;*****
0074      ;      Start Bit Subroutine
0075      ;      this routine generates a start bit
0076      ;      (Low going data line while clock is high)
0077      ;*****
0078      ;
0079      BSTART
0080      0001 05E6          bsf    port_b,sdata ; make sure data is high
0081      0002 0C3F          movlw  b'00111111'
0082      0003 0006          tris   port_b      ; set data and clock lines for output
0083      0004 04C6          bcf    port_b,sclk  ; make sure clock is low
0084      0005 0000          nop
0085      0006 05C6          bsf    port_b,sclk  ; set clock high
0086      0007 0000          nop
0087      0008 0000          nop
0088      0009 0000          nop
0089      000A 0000          nop
0090      000B 0000          nop
0091      000C 04E6          bcf    port_b,sdata ; data line goes low during
0092      ;                               ; high clock for start bit
0093      000D 0000          nop
0094      000E 0000          nop
0095      000F 0000          nop
0096      0010 0000          nop
0097      0011 0000          nop ; timing adjustment
0098      0012 04C6          bcf    port_b,sclk  ; start clock train
0099      0013 0000          nop
0100      0014 0000          nop
0101      0015 0800          retlw  0
0102      ;
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:23 1994 Page 3

```
Line  PC  Opcode
0103      ;*****
0104      ;      Stop Bit Subroutine
0105      ;      This routine generates a stop bit
0106      ;      (High going data line while clock is high)
0107      ;*****
0108      BSTOP
0109      0016 04E6          bcf    port_b,sdata ; make sure data line is low
0110      0017 0C3F          movlw  b'00111111'
0111      0018 0006          tris   port_b      ; set data/clock lines as outputs
0112      0019 04E6          bcf    port_b,sdata ; make sure data line is low
0113      001A 0000          nop
0114      001B 0000          nop
0115      001C 0000          nop
0116      001D 05C6          bsf    port_b,sclk  ; set clock high
0117      001E 0000          nop
0118      001F 0000          nop
0119      0020 0000          nop
0120      0021 05E6          bsf    port_b,sdata ; data goes high while clock high
0121      ;                               ; for stop bit
```

Interfacing the 24LCXXB Serial EEPROMs

```
0122 0022 0000      nop
0123 0023 0000      nop
0124 0024 04C6      bcf      port_b,sclk    ; set clock low again
0125 0025 0000      nop
0126 0026 0000      nop
0127 0027 0000      nop
0128 0028 0800      retlw   0
0129                ;
0130                ;*****
0131                ;      BITOUT routine takes the bit of data in 'do' and
0132                ;      transmits it to the serial EE device
0133                ;*****
0134 BITOUT
0135 0029 0C3F      movlw   b'00111111'    ; set data,clock as outputs
0136 002A 0006      tris    port_b
0137 002B 07CA      btfss   eeprom,do      ; check for state of data bit to xmit
0138 002C 0A2F      goto    bitlow        ; low? go set data line low
0139 002D 05E6      bsf     port_b,sdata   ; high? set data line high
0140 002E 0A30      goto    clkout        ; go toggle the clock
0141
0142 002F 04E6      bitlow   bcf     port_b,sdata ; output a low bit
0143 0030 05C6      clkout   bsf     port_b,sclk ; set clock line high
0144 0031 0000      nop
0145 0032 0000      nop
0146 0033 0000      nop
0147 0034 0000      nop
0148 0035 04C6      bcf     port_b,sclk    ; return clock line low
0149 0036 0800      retlw   0
0150                ;
0151                ;*****
0152                ;      BITIN routine reads one bit of data from the
0153                ;      serial EE device and stores it in the bit 'di'
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:23 1994 Page 4

```
Line  PC  Opcode
0154                ;*****
0155 BITIN
0156 0037 05EA      bsf     eeprom,di      ; assume input bit is high
0157 0038 0CBF      movlw   b'10111111'    ; make sdata an input line
0158 0039 0006      tris    port_b
0159 003A 05C6      bsf     port_b,sclk    ; set clock line high
0160 003B 0000      nop
0161 003C 0000      nop
0162 003D 0000      nop
0163 003E 0000      nop
0164 003F 0000      nop
0165 0040 07E6      btfss   port_b,sdata   ; read the data bit
0166 0041 04EA      bcf     eeprom,di      ; input bit was low, set 'di' accordingly
0167 0042 04C6      bcf     port_b,sclk    ; set clock line low
0168                ;
0169 0043 0800      retlw   0
0170                ;
0171                ;*****
0172                ;      Transmit Data Subroutine
0173                ;      This routine takes the byte of data stored in the
0174                ;      'datao' register and transmits it to the serial EE device.
0175                ;      It will then send 1 more clock to the serial EE for the
0176                ;      acknowledge bit. If the ack bit from the part was low
0177                ;      then the transmission was successful. If it is high, then
0178                ;      the device did not send a proper ack bit and the ack
0179                ;      fail LED will be turned on.
0180                ;*****
0181 TX
0182 0044 0C08      movlw   .8
0183 0045 0031      movwf   count          ; set the #bits to 8
0184                ;
0185 TXLP
```

Interfacing the 24LCXXB Serial EEPROMs

```
0186 0046 04CA      bcf    eeprom,do      ; assume bit out is low
0187 0047 06F0      btfscl txbuf,7        ; is bit out really low?
0188 0048 05CA      bsf    eeprom,do      ; no, set it high
0189 0049 0929      call   BITOUT         ; send the bit to serial EE
0190 004A 0370      rlf    txbuf          ; rotate txbuf left
0191 004B 02F1      decfsz count          ; 8 bits done?
0192 004C 0A46      goto   TXLP           ; no - go again
0193 004D 0937      call   BITIN          ; read ack bit
0194 004E 06EA      btfscl eeprom,di     ; check ack bit
0195 004F 0525      bsf    port_a,ackf    ; set acknowledge fail LED if the
0196                          ; device did not pull data low
0197                          ;
0198 0050 0800      retlw  0
0199                          ;
0200                          ;*****
0201                          ;   Receive data routine
0202                          ;   This routine reads one byte of data from the part
0203                          ;   into the 'datai' register.  It then sends a high
0204                          ;   ack bit to indicate that no more data is to be read
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:23 1994 Page 5

```
Line  PC  Opcode
0205                          ;*****
0206                          RX
0207 0051 006D      clrf   datai          ; clear input buffer
0208 0052 0C08      movlw  .8             ; set # bits to 8
0209 0053 0031      movwf  count          ;
0210 0054 0403      bcf    status,0       ; make sure carry bit is low
0211 0055 036D      RXLP   rlf    datai    ; rotate datai 1 bit left
0212 0056 0937      call   BITIN          ; read a bit
0213 0057 06EA      btfscl eeprom,di     ;
0214 0058 050D      bsf    datai,0        ; set bit 0 if necessary
0215 0059 02F1      decfsz count          ; 8 bits done?
0216 005A 0A55      goto   RXLP           ; no, do another
0217 005B 05CA      bsf    eeprom,do      ; set ack bit = 1
0218 005C 0929      call   BITOUT         ; to finish transmission
0219 005D 0800      retlw  0
0220                          ;
0221                          ;*****
0222 0000 0000  END
```

Interfacing the 24LCXXB Serial EEPROMs

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:29 1994 Page 1

```
Line   PC   Opcode

0001                                LIST P=16C54,c=132
0002                                ;*****
0003                                ; 2-Wire Byte Write Program (123 bytes)
0004                                ;
0005                                ; This program demonstrates how to interface a
0006                                ; Microchip PIC16C54 to a 24LCXX Serial EE device
0007                                ; and perform a byte write operation on 8 consecutive
0008                                ; addresses.
0009                                ;
0010                                ; This routine waits approximately 10mS for the write
0011                                ; cycle time, which is enough time for any of the
0012                                ; 24LCxx devices to complete a write. A more efficient
0013                                ; method of determining when the write cycle is complete
0014                                ; is called "data polling." The data polling method is
0015                                ; explained in the program "2wdpoll.asm." That
0016                                ; particular program uses data polling in a byte write
0017                                ; mode but it can be used in exactly the same way for
0018                                ; a page mode write.
0019                                ;
0020                                ; As an option, the user may connect a LED to pin 18
0021                                ; on the PIC16C54 (use about a 1K resistor in series) as an
0022                                ; acknowledge fail indicator. This LED will come on if
0023                                ; the serial EE fails to acknowledge correctly.
0024                                ;
0025                                ; Timing is based on using the PIC16C54 in 'XT' mode
0026                                ; using a 4Mhz crystal. Clock speeds to the serial EE
0027                                ; will be approximately 60 kHz for this setup.
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                                ; Register Definitions
0040                                ;*****
0041      0005 port_a equ 5h ; port 5 (port_a) used for LEDs
0042      0006 port_b equ 6h ; port 6 (port b) used for data and
0043                                ; clock lines
0044      000A eeprom equ 0ah ; bit buffer
0045      000B bycnt equ 0bh ; byte counter for read mode
0046      000C addr equ 0ch ; address counter
0047      000D datai equ 0dh ; data input register
0048      000E datao equ 0eh ; data output register
0049      000F slave equ 0fh ; device address
0050                                ; (1010xxx0)
0051      0010 txbuf equ 10h ; transmit buffer
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:29 1994 Page 2

```
Line   PC   Opcode

0052      0011 count equ 11h ; bit counter
0053      0012 bcount equ 12h ; byte counter
0054      0015 loops equ 15h ; delay loop counter
0055      0016 loops2 equ 16h ; delay loop counter
0056                                ;*****
0057                                ; Bit Definitions
0058                                ;*****
0059      0007 di equ 7 ; eeprom input bit
```

Interfacing the 24LCXXB Serial EEPROMs

```

0060      0006 do      equ    6      ; eeprom output bit
0061      0007 sdata  equ    7      ; serial EE data line (port_b,pin 13)
0062      0006 sclk   equ    6      ; serial EE clock line (port_b,pin 12)
0063      0001 ackf   equ    1      ; acknowledge fail LED (port_a,pin 18)
0064      ;*****
0065      ;
0066      ;
0067      0000      org    01ffh ; set reset vector
0068  01FF  0A5E      goto   PWRUP
0069      0000      org    000h ;
0070  0000  0A5E      goto   PWRUP
0071      ;
0072      ;*****
0073      ;      DELAY ROUTINE
0074      ;      This routine takes the value in 'loops'
0075      ;      and multiplies it times 1 millisecond to
0076      ;      determine delay time.
0077      ;*****
0078      WAIT
0079      ;
0080  0001  0C6E  top    movlw  .110      ; timing adjustment variable
0081  0002  0036      movwf  loops2
0082  0003  0000  top2  nop                ; sit and wait
0083  0004  0000      nop
0084  0005  0000      nop
0085  0006  0000      nop
0086  0007  0000      nop
0087  0008  0000      nop
0088  0009  02F6      decfsz loops2      ; inner loops complete?
0089  000A  0A03      goto   top2        ; no, go again
0090      ;
0091  000B  02F5      decfsz loops      ; outer loops complete?
0092  000C  0A01      goto   top         ; no, go again
0093  000D  0800      retlw  0          ; yes, return from sub
0094      ;
0095      ;*****
0096      ;      Start Bit Subroutine
0097      ;      this routine generates a start bit
0098      ;      (Low going data line while clock is high)
0099      ;*****
0100      ;
0101      BSTART
0102  000E  05E6      bsf    port_b,sdata ; make sure data is high

```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:29 1994 Page 3

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

Interfacing the 24LCXXB Serial EEPROMs

```
0123 0022 0800      retlw  0
0124                ;
0125                ;      End of Subroutine
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 0C3F      movlw  b'00111111'  ;
0133 0024 0006      tris   port_b      ; set data/clock lines as outputs
0134 0025 04E6      bcf   port_b,sdata ; make sure data line is low
0135 0026 0000      nop
0136 0027 0000      nop
0137 0028 0000      nop
0138 0029 05C6      bsf   port_b,sclk  ; set clock high
0139 002A 0000      nop
0140 002B 0000      nop
0141 002C 0000      nop
0142 002D 05E6      bsf   port_b,sdata ; data goes high while clock high
0143                ; for stop bit
0144 002E 0000      nop
0145 002F 0000      nop
0146 0030 04C6      bcf   port_b,sclk  ; set clock low again
0147 0031 0000      nop
0148 0032 0000      nop
0149 0033 0000      nop
0150 0034 0800      retlw  0
0151                ;
0152                ;      End of Subroutine
0153                ;*****
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:29 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      btfs  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                ;      End of Subroutine
0175                ;
0176                ;*****
0177                ;      BITIN routine reads one bit of data from the
0178                ;      serial EE device and stores it in 'di'
0179                ;*****
0180                BITIN
0181 0043 05EA      bsf   eeprom,di    ; assume input bit is high
0182 0044 0CFB      movlw  b'10111111'  ; make sdata an input line
0183 0045 0006      tris   port_b
0184 0046 05E6      bsf   port_b,sdata ; set sdata line for input
0185 0047 05C6      bsf   port_b,sclk  ; set clock line high
```

Interfacing the 24LCXXB Serial EEPROMs

```

0186 0048 0000      nop                ; just sit here a sec
0187 0049 0000      nop
0188 004A 0000      nop
0189 004B 0000      nop
0190 004C 0000      nop
0191 004D 07E6      btfss   port_b,sdata ; read the data bit
0192 004E 04EA      bcf     eeeprom,di   ; input bit was low
0193 004F 04C6      bcf     port_b,sclk  ; set clock line low
0194
0195 0050 0800      retlw   0            ;
0196 ;
0197 ;*****
0198 ;      Transmit Data Subroutine
0199 ;      This routine takes the byte of data stored in the
0200 ;      'data0' register and transmits it to the serial EE device.
0201 ;      It will then send 1 more clock to the serial EE for the
0202 ;      acknowledge bit.  If the ack bit from the part was low
0203 ;      then the transmission was successful.  If it is high, then
0204 ;      the device did not send a proper ack bit and the ack

```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:29 1994 Page 5

```

Line  PC   Opcode
-----
0205      ;      fail LED will be turned on.
0206      ;*****
0207      TX
0208 0051 0C08      movlw   .8
0209 0052 0031      movwf   count      ; set the #bits to 8
0210      ;
0211      TXLP
0212 0053 04CA      bcf     eeeprom,do ; assume bit out is low
0213 0054 06F0      btfsc   txbuf,7    ; is bit out really low?
0214 0055 05CA      bsf     eeeprom,do ; otherwise data bit =1
0215 0056 0935      call    BITOUT     ; serial data out
0216 0057 0370      rlf     txbuf      ; rotate txbuf left
0217 0058 02F1      decfsz  count      ; 8 bits done?
0218 0059 0A53      goto   TXLP       ; no - go again
0219 005A 0943      call    BITIN     ; read ack bit
0220 005B 06EA      btfsc   eeeprom,di ; check ack bit
0221 005C 0525      bsf     port_a,ackf ; set acknowledge fail LED if the
0222      ;
0223 005D 0800      retlw   0
0224 ;
0225 ;*****
0226 ;      Power up routine
0227 ;      This is the program entry point, which in this case simply
0228 ;      sets the port_a I/O lines and directs control to the
0229 ;      byte write routine.
0230 ;*****
0231      PWRUP
0232 005E 0C00      movlw   b'00000000'
0233 005F 0005      tris   port_a      ; set port A as all output
0234 0060 0065      clrf   port_a      ; all output lines low
0235 0061 0A62      goto   WRBYTE
0236
0237 ;*****
0238 ;      Byte Write Routine
0239 ;      This routine writes the data in "data0" to
0240 ;      8 consecutive bytes in the serial EE device starting
0241 ;      at address 00.  This routine waits 10mS after every
0242 ;      byte to give the device time to do the write.  This
0243 ;      program repeats forever.
0244 ;*****
0245 ;
0246      WRBYTE
0247 ;
0248 0062 0065      clrf   port_a      ; clear all LEDs

```

Interfacing the 24LCXXB Serial EEPROMs

```
0249 0063 0CA0      movlw  b'10100000' ; set slave address and write mode
0250 0064 002F      movwf  slave
0251 0065 0C55      movlw  b'01010101' ; set data to write as 55h
0252 0066 002E      movwf  data0
0253                ;
0254 0067 0C08      movlw  .8           ; set number of bytes
0255 0068 0032      movwf  bcount      ; to write as to 8
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:29 1994 Page 6

```
Line  PC  Opcode
0256  0069  006C      clrfs  addr        ; set starting address to 00
0257                ;
0258  006A  090E  byte    call   BSTART      ; generate start bit
0259  006B  020F      movf   slave,w     ; move slave address
0260  006C  0030      movwf  txbuf       ; into transmit buffer
0261  006D  0951      call   TX          ; and send it
0262  006E  020C      movf   addr,w     ; move word address
0263  006F  0030      movwf  txbuf       ; into transmit buffer
0264  0070  0951      call   TX          ; and send it
0265  0071  020E      movf   data0,w    ; move data byte
0266  0072  0030      movwf  txbuf       ; to transmit buffer
0267  0073  0951      call   TX          ; and transmit it
0268  0074  0923      call   BSTOP      ; generate stop bit
0269                ;
0270  0075  0C0A      movlw  .10
0271  0076  0035      movwf  loops       ; set delay time to give
0272  0077  0901      call   WAIT        ; 10 ms wait after every byte
0273  0078  02AC      incf   addr        ; add 1 to address counter
0274  0079  02F2      decfsz bcount     ; all 8 bytes written?
0275  007A  0A6A      goto   byte        ; no, do another byte
0276                ;
0277  007B  0A62      goto   wrbyte      ; start over
0278                ;
0279                ;
0280                0000  END
```

Interfacing the 24LCXXB Serial EEPROMs

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:50 1994 Page 1

```
Line  PC  Opcode
0001          LIST P=16C54,c=132
0002          ;*****
0003          ; 2-Wire Byte Write With Data Polling Program (129 bytes)
0004          ;
0005          ; This program demonstrates how to interface a
0006          ; Microchip PIC16C54 to a 24LCXX 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 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 10:48:50 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
```

Interfacing the 24LCXXB Serial EEPROMs

```
0061      0015 loops equ 15h ; delay loop counter
0062      0016 loops2 equ 16h ; delay loop counter
0063      0017 pollent equ 17h ; data polling counter
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      0002 timeout equ 2 ; write cycle timeout fail LED, port_a (pin 1)
0072      0001 ackf equ 1 ; acknowledge fail LED, port_a (pin 18)
0073      ;*****
0074      ;
0075      0000 org 01ffh ; set reset vector
0076      01FF 0A5C goto PWRUP
0077      0000 org 000h ;
0078      0000 0A5C 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 10:48:50 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
```

Interfacing the 24LCXXB Serial EEPROMs

```
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 10:48:50 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                ;      End of Subroutine
0181                ;
0182                ;*****
0183                ;      BITIN routine reads one bit of data from the
0184                ;      serial EE device and stores it in 'di'
0185                ;*****
0186                BITIN
0187  0043  05EA      bsf    eeprom,di    ; assume input bit is high
0188  0044  0CBF      movlw   b'10111111'  ; make sdata an input line
```

Interfacing the 24LCXXB Serial EEPROMs

```
0189 0045 0006      tris    port_b
0190 0046 05E6      bsf    port_b,sdata ; set sdata line for input
0191 0047 05C6      bsf    port_b,sclk  ; set clock line high
0192 0048 0000      nop
0193 0049 0000      nop
0194 004A 0000      nop
0195 004B 0000      nop
0196 004C 0000      nop
0197 004D 07E6      btfsz port_b,sdata ; read the data bit
0198 004E 04EA      bcf    eeprom,di   ; input bit was low
0199 004F 04C6      bcf    port_b,sclk ; set clock line low
0200
0201 0050 0800      retlw  0
0202
0203 ;
0204 ;          Transmit Data Subroutine
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:50 1994 Page 5

```
Line  PC  Opcode

0205      ;          This routine takes the byte of data stored in the
0206      ;          'datao' register and transmits it to the serial EE device.
0207      ;          It will then send 1 more clock to the serial EE for the
0208      ;          acknowledge bit.  If the ack bit from the part was low
0209      ;          then the transmission was successful.  If it is high, then
0210      ;          the device did not send a proper ack bit and the ack
0211      ;          fail LED will be turned on.
0212      ;*****
0213      TX
0214 0051 0C08      movlw  .8
0215 0052 0031      movwf  count      ; set the #bits to 8
0216      ;
0217      TXLP
0218 0053 04CA      bcf    eeprom,do  ; assume bit out is low
0219 0054 06F0      btfsz  txbuf,7    ; is bit out really low?
0220 0055 05CA      bsf    eeprom,do  ; otherwise data bit =1
0221 0056 0935      call  BITOUT      ; serial data out
0222 0057 0370      rlf    txbuf      ; rotate txbuf left
0223 0058 02F1      decfsz count     ; 8 bits done?
0224 0059 0A53      goto  TXLP        ; no - go again
0225 005A 0943      call  BITIN       ; read ack bit
0226      ;
0227 005B 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      ;          write routine.
0234      ;*****
0235      PWRUP
0236 005C 0C00      movlw  b'00000000'
0237 005D 0005      tris  port_a      ; set port A as all output
0238 005E 0065      clrf  port_a      ; all output lines low
0239 005F 0A60      goto  WRBYTE
0240
0241      ;*****
0242      ;          Byte Write Routine with data polling technique
0243      ;
0244      ;          This routine writes the data in "datao" to
0245      ;          8 consecutive bytes in the serial EE device starting
0246      ;          at address 00.  To determine when the write cycle time
0247      ;          of the device, the 'data polling' method is used.  This
0248      ;          involves sending a start bit and control byte to the part
0249      ;          and checking for an acknowledge.  If the ack bit is low, then
0250      ;          the device is through writing, otherwise the the sequence
0251      ;          is repeated.  If no low acknowledge is found within 40
0252      ;          attempts (about 10 milliseconds) then the routine times
```

Interfacing the 24LCXXB Serial EEPROMs

```
0253 ; out and sets the timeout LED (pin 18) high. This program
0254 ; will repeat forever.
0255 ;
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:50 1994 Page 6

```
Line PC Opcode
0256 ;*****
0257 ;
0258 WRBYTE
0259 ;
0260 0060 0065 clrf port_a ; all LEDs off
0261 0061 0CA0 movlw b'10100000' ; set slave address and write mode
0262 0062 002F movwf slave
0263 0063 0CAA movlw b'10101010' ; set data to write as AAh
0264 0064 002E movwf data0
0265 0065 0C08 movlw .8 ; set number of bytes
0266 0066 0032 movwf bcount ; to write as to 8
0267 0067 006C clrf addr ; set starting address to 00
0268 ;
0269 0068 090E byte call BSTART ; generate start bit
0270 0069 020F movf slave,w ; move slave address
0271 006A 0030 movwf txbuf ; into transmit buffer
0272 006B 0951 call TX ; and send it
0273 006C 020C movf addr,w ; move word address
0274 006D 0030 movwf txbuf ; into transmit buffer
0275 006E 0951 call TX ; and send it
0276 006F 020E movf data0,w ; move data byte
0277 0070 0030 movwf txbuf ; to transmit buffer
0278 0071 0951 call TX ; and transmit it
0279 0072 0923 call BSTOP ; generate stop bit
0280 ;
0281 ; now start polling for a low ack bit
0282 ;
0283 0073 0C28 movlw .40 ;
0284 0074 0037 movwf pollcnt ; set max number of times to poll as 40
0285 0075 090E poll call BSTART ; generate start bit
0286 0076 0CA0 movlw b'10100000' ; move slave address (write mode)
0287 0077 0030 movwf txbuf ; into transmit buffer
0288 0078 0951 call TX ; and send it
0289 0079 07EA btfsz eeprom,di ; was the ack bit low?
0290 007A 0A7E goto exitpoll ; yes, do another byte
0291 007B 02F7 decfsz pollcnt ; is poll counter down to zero?
0292 007C 0A75 goto poll ; no, poll again. Other wise the part is
0293 007D 0545 bsf port_a,timeout ; not responding in time so set timeout
0294 ; LED and continue on
0295 ;
0296 007E 02AC exitpoll incf addr ; add 1 to address counter
0297 007F 02F2 decfsz bcount ; all 8 bytes written?
0298 0080 0A68 goto byte ; no, do another byte
0299 0081 0A60 goto WRBYTE ; yes, start over
0300 ;
0301 ;
0302 0000 END
```

Interfacing the 24LCXXB Serial EEPROMs

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:36 1994 Page 1

```
Line   PC   Opcode
0001                LIST P=16C54,c=132
0002                ;*****
0003                ;   2-Wire Page Write Program (122 bytes)
0004                ;
0005                ;   This program demonstrates how to interface a
0006                ;   Microchip PIC16C54 to a 24LCXX Serial EE device
0007                ;   and perform a page write operation on 8 consecutive
0008                ;   addresses.
0009                ;
0010                ;   All of Microchip's 2-wire serial ES devices have a
0011                ;   buffer or 'page' that can be used as a more efficient
0012                ;   method of writing to consecutive addresses. Some
0013                ;   devices have page lengths of 2 bytes, others have
0014                ;   page lengths of 4,8,16 or 64 bytes. Please consult
0015                ;   the databook for the page length of the device you
0016                ;   are using. THIS ROUTINE IS WRITTEN FOR A DEVICE
0017                ;   WITH A PAGE LENGTH OF 8 OR MORE BYTES.
0018                ;
0019                ;   When using page mode, the control byte and address
0020                ;   are sent for the first address only. After the data
0021                ;   byte for the first address is sent, the data for the
0022                ;   next consecutive address is clocked in. This is
0023                ;   repeated as many times as needed (as long as the page
0024                ;   length is not exceeded) and then a stop bit is sent.
0025                ;   The device will still acknowledge between every byte of
0026                ;   data. After the stop bit is sent, the part will
0027                ;   initiate the self timed write cycle. For all of
0028                ;   Microchip's 24LCxx devices, the cycle time for
0029                ;   a byte write and a page write is the same. Therefore,
0030                ;   writing 8 bytes in byte mode with a typical 5ms write
0031                ;   cycle consumes 40ms of wait time, while the same data
0032                ;   written in page mode would consume only 5ms of wait time.
0033                ;
0034                ;   This routine waits approximately 10mS for the write
0035                ;   cycle time, which is enough time for any of the
0036                ;   24LCxx devices to complete a write. A more efficient
0037                ;   method of determining when the write cycle is complete
0038                ;   is called "data polling." The data polling method is
0039                ;   explained in the program "2wdpoll.asm." That
0040                ;   particular program uses data polling in a byte write
0041                ;   mode but it can be used in exactly the same way for
0042                ;   a page mode write.
0043                ;
0044                ;   As an option, the user can connect a LED to pin 18
0045                ;   on the PIC16C54 (use about a 1K resistor in series) as a
0046                ;   acknowledge fail indicator. This LED will come on
0047                ;   if the device being programmed does not send a low
0048                ;   acknowledge bit at the proper times. This can be
0049                ;   tested by simply running the program with no part
0050                ;   in the socket.
0051                ;
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:36 1994 Page 2

```
Line   PC   Opcode
0052                ;   Timing is based on using the PIC16C5x in 'XT' mode
```

Interfacing the 24LCXXB Serial EEPROMs

```

0053 ;      using a 4MHz crystal.  Clock speeds to the serial EE
0054 ;      will be approximately 40 kHz for this setup.
0055 ;
0056 ;
0057 ;      PIC16C54 to Serial EE Connections:
0058 ;
0059 ;      PIC16C54      Serial EE
0060 ;      _____  _____
0061 ;      Pin 12 (RB6) -> SCLK
0062 ;      Pin 13 (RB7) -> SDATA
0063 ;
0064 ;      PIN 18 (RA1) -> Acknowledge fail LED  (Optional)
0065 ;
0066 ;
0067 ;*****
0068 ;      Register Definitions
0069 ;*****
0070 0005 port_a equ 5h ; port 5 (port a) used for LED outputs
0071 0006 port_b equ 6h ; port 6 (port b) used for data and
0072 ;      clock lines
0073 000A eeprom equ 0ah ; bit buffer
0074 000C addr equ 0ch ; address counter
0075 000E datao equ 0eh ; data output register
0076 000F slave equ 0fh ; device address (1010xxx0)
0077 0010 txbuf equ 10h ; transmit buffer
0078 0011 count equ 11h ; bit counter
0079 0012 bcount equ 12h ; byte counter
0080 0015 loops equ 15h ; delay loop counter
0081 0016 loops2 equ 16h ; delay loop counter
0082 ;*****
0083 ;      Bit Definitions
0084 ;*****
0085 0007 di equ 7 ; eeprom input bit
0086 0006 do equ 6 ; eeprom output bit
0087 0007 sdata equ 7 ; serial EE data line (port_b,pin 13)
0088 0006 sclk equ 6 ; serial EE clock line (port_b,pin 12)
0089 0001 ackf equ 1 ; acknowledge fail LED, port_a (pin 18)
0090 ;*****
0091 ;
0092 ;
0093 0000 org 01ffh ; set reset vector
0094 01FF 0A5E goto PWRUP
0095 0000 org 000h ;
0096 0000 0A5E goto PWRUP
0097 ;
0098 ;*****
0099 ;      DELAY ROUTINE
0100 ;      This routine takes the value in 'loops'
0101 ;      and multiplies it times 1 millisecond to
0102 ;      determine delay time.

```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:36 1994 Page 3

```

Line  PC  Opcode
0103 ;*****
0104 WAIT
0105 ;
0106 0001 0C6E top movlw .110 ; timing adjustment variable
0107 0002 0036 movwf loops2
0108 0003 0000 top2 nop ; sit and wait

```

Interfacing the 24LCXXB Serial EEPROMs

```
0109 0004 0000      nop
0110 0005 0000      nop
0111 0006 0000      nop
0112 0007 0000      nop
0113 0008 0000      nop
0114 0009 02F6      decfsz  loops2      ; inner loops complete?
0115 000A 0A03      goto   top2        ; no, go again
0116                ;
0117 000B 02F5      decfsz  loops      ; outer loops complete?
0118 000C 0A01      goto   top        ; no, go again
0119 000D 0800      retlw  0          ; yes, return from sub
0120                ;
0121                ;*****
0122                ;      Start Bit Subroutine
0123                ;      this routine generates a start bit
0124                ;      (Low going data line while clock is high)
0125                ;*****
0126                ;
0127                BSTART
0128 000E 05E6      bsf    port_b,sdata ; make sure data is high
0129 000F 0C3F      movlw  b'00111111'
0130 0010 0006      tris   port_b      ; set data and clock lines for
output
0131 0011 04C6      bcf    port_b,sclk  ; make sure clock is low
0132 0012 0000      nop
0133 0013 05C6      bsf    port_b,sclk  ; set clock high
0134 0014 0000      nop
0135 0015 0000      nop
0136 0016 0000      nop
0137 0017 0000      nop
0138 0018 0000      nop
0139 0019 04E6      bcf    port_b,sdata ; data line goes low during
0140                ; high clock for start bit
0141 001A 0000      nop
0142 001B 0000      nop
0143 001C 0000      nop
0144 001D 0000      nop
0145 001E 0000      nop                ; timing adjustment
0146 001F 04C6      bcf    port_b,sclk  ; start clock train
0147 0020 0000      nop
0148 0021 0000      nop
0149 0022 0800      retlw  0
0150                ;
0151                ;      End of Subroutine
0152                ;*****
0153                ;      Stop Bit Subroutine
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:36 1994 Page 4

```
Line  PC      Opcode

0154                ;      This routine generates a stop bit
0155                ;      (High going data line while clock is high)
0156                ;*****
0157                BSTOP
0158 0023 0C3F      movlw  b'00111111' ;
0159 0024 0006      tris   port_b      ; set data/clock lines as outputs
0160 0025 04E6      bcf    port_b,sdata ; make sure data line is low
0161 0026 0000      nop
0162 0027 0000      nop
0163 0028 0000      nop
```

Interfacing the 24LCXXB Serial EEPROMs

```
0164 0029 05C6      bsf     port_b,sclk   ; set clock high
0165 002A 0000      nop
0166 002B 0000      nop
0167 002C 0000      nop
0168 002D 05E6      bsf     port_b,sdata   ; data goes high while clock high
0169                      ; for stop bit
0170 002E 0000      nop
0171 002F 0000      nop
0172 0030 04C6      bcf     port_b,sclk   ; set clock low again
0173 0031 0000      nop
0174 0032 0000      nop
0175 0033 0000      nop
0176 0034 0800      retlw  0
0177                      ;
0178                      ;           End of Subroutine
0179                      ;*****
0180                      ;           BITOUT routine
0181                      ;           This routine takes one bit of data in 'do' and
0182                      ;           transmits it to the serial EE device
0183                      ;*****
0184 BITOUT
0185 0035 0C3F      movlw  b'00111111'   ; set data,clock as outputs
0186 0036 0006      tris  port_b
0187 0037 07CA      btfs   eeprom,do    ; check for state of data bit to
xmit
0188 0038 0A3B      goto  bitlow        ;
0189 0039 05E6      bsf     port_b,sdata   ; set data line high
0190 003A 0A3C      goto  clkout        ; go toggle the clock
0191
0192 003B 04E6 bitlow bcf     port_b,sdata   ; output a low bit
0193 003C 05C6 clkout bsf     port_b,sclk   ; set clock line high
0194 003D 0000      nop
0195 003E 0000      nop
0196 003F 0000      nop
0197 0040 0000      nop
0198 0041 04C6      bcf     port_b,sclk   ; return clock line low
0199 0042 0800      retlw  0
0200                      ;
0201                      ;           End of Subroutine
0202                      ;
0203                      ;*****
0204                      ;           BITIN routine reads one bit of data from the
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:36 1994 Page 5

```
Line  PC  Opcode
0205                      ;           serial EE device and stores it in 'di'
0206                      ;*****
0207 BITIN
0208 0043 05EA      bsf     eeprom,di    ; assume input bit is high
0209 0044 0CBF      movlw  b'10111111'   ; make sdata an input line
0210 0045 0006      tris  port_b
0211 0046 05E6      bsf     port_b,sdata   ; set sdata line for input
0212 0047 05C6      bsf     port_b,sclk   ; set clock line high
0213 0048 0000      nop                      ; just sit here a sec
0214 0049 0000      nop
0215 004A 0000      nop
0216 004B 0000      nop
0217 004C 0000      nop                      ;
0218 004D 07E6      btfs   port_b,sdata   ; read the data bit
```

Interfacing the 24LCXXB Serial EEPROMs

```
0219 004E 04EA      bcf    eeprom,di      ; input bit was low
0220 004F 04C6      bcf    port_b,sclk    ; set clock line low
0221                ;
0222 0050 0800      retlw  0              ;
0223                ;
0224                ;*****
0225                ; Transmit Data Subroutine
0226                ; This routine takes the byte of data stored in the
0227                ; 'data0' register and transmits it to the serial EE device.
0228                ; It will then send 1 more clock to the serial EE for the
0229                ; acknowledge bit. If the ack bit from the part was low
0230                ; then the transmission was successful. If it is high, then
0231                ; the device did not send a proper ack bit and the ack
0232                ; fail LED will be turned on.
0233                ;*****
0234                TX
0235 0051 0C08      movlw  .8
0236 0052 0031      movwf  count          ; set the #bits to 8
0237                ;
0238                TXLP
0239 0053 04CA      bcf    eeprom,do      ; assume bit out is low
0240 0054 06F0      btfsz  txbuf,7        ; is bit out really low?
0241 0055 05CA      bsf    eeprom,do      ; otherwise data bit =1
0242 0056 0935      call  BITOUT          ; serial data out
0243 0057 0370      rlf    txbuf          ; rotate txbuf left
0244 0058 02F1      decfsz count         ; 8 bits done?
0245 0059 0A53      goto  TXLP           ; no - go again
0246 005A 0943      call  BITIN          ; read ack bit
0247 005B 06EA      btfsz  eeprom,di     ; check ack bit
0248 005C 0525      bsf    port_a,ackf    ; set acknowledge fail LED if the
0249                ; device did not pull data low
0250                ;
0251 005D 0800      retlw  0
0252                ;
0253                ;*****
0254                ; Power up routine
0255                ; This is the program entry point, which in this case simply
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:36 1994 Page 6

```
Line  PC  Opcode
0256                ; sets the port_a I/O lines and directs control to the Page
0257                ; write routine.
0258                ;*****
0259                PWRUP
0260 005E 0C00      movlw  b'00000000'
0261 005F 0005      tris  port_a          ; set port A as all output
0262 0060 0065      clrf  port_a          ; all output lines low
0263 0061 0A62      goto  WRPAGE
0264                ;
0265                ;*****
0266                ; Page Write Routine
0267                ;
0268                ; This routine uses page mode to write the data in "data0" to
0269                ; 8 consecutive bytes in the serial EE device starting
0270                ; at address 00. This routine waits 10mS after every
0271                ; page to give the device time to do the write. This
0272                ; routine executes forever
0273                ;*****
0274                ;
```

Interfacing the 24LCXXB Serial EEPROMs

```
0275                WRPAGE
0276                ;
0277 0062 0065      clr    port_a      ; clear all LEDs
0278 0063 0CA0     movlw b'10100000'  ; set slave address and write mode
0279 0064 002F     movwf slave
0280 0065 0C55     movlw b'01010101'  ; set data to write as 55h
0281 0066 002E     movwf datao
0282                ;
0283 0067 0C08     movlw .8            ; set number of bytes
0284 0068 0032     movwf bcount       ; to write as to 8
0285 0069 006C     clr    addr        ; set starting address to 00
0286                ;
0287 006A 090E     call   BSTART      ; generate start bit
0288 006B 020F     movf   slave,w      ; move slave address
0289 006C 0030     movwf txbuf       ; into transmit buffer
0290 006D 0951     call   TX          ; and send it
0291                ;
0292 006E 020C     movf   addr,w      ; move word address
0293 006F 0030     movwf txbuf       ; into transmit buffer
0294 0070 0951     call   TX          ; and send it
0295                ;
0296 0071 020E     movf   datao,w     ; move data byte
0297 0072 0030     movwf txbuf       ; to transmit buffer
0298 0073 0951     call   TX          ; and transmit it
0299 0074 02F2     decfsz bcount     ; all 8 bytes written?
0300 0075 0A71     goto  byte      ; no, do another
0301 0076 0923     call   BSTOP     ; yes,generate stop bit
0302                ;
0303 0077 0C0A     movlw .10       ; set delay time to give
0304 0078 0035     movwf loops     ; set delay time to give
0305 0079 0901     call   WAIT     ; 10 ms wait after every byte
0306                ;
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:36 1994 Page 7

```
Line  PC  Opcode
0307 007A 0A62      goto  wrpage      ; start over
0308                ;
0309                ;
0310        0000  END
```

Interfacing the 24LCXXB Serial EEPROMs

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:46 1994 Page 1

```
Line   PC   Opcode

0001                               LIST P=16C54,c=132
0002                               ;*****
0003                               ;       2-Wire Sequential Read Program (120 bytes)
0004                               ;
0005                               ;       This program demonstrates how to interface a
0006                               ;       Microchip PIC16C54 to a 24LCXX Serial EE device
0007                               ;       and perform a sequential read operation.  A sequential
0008                               ;       read involves setting the address pointer once and then
0009                               ;       using the auto-increment ability of the part to read
0010                               ;       consecutive addresses by simply providing more clocks.
0011                               ;
0012                               ;       Timing is based on using the PIC16C54 in 'XT' mode
0013                               ;       using a 4MHz crystal.  Clock speeds to the serial EE
0014                               ;       will be approximately 60 kHz for this setup.
0015                               ;
0016                               ;       As an option, the user may connect a LED to pin 18
0017                               ;       on the PIC16C54 (use about a 1K resistor in series) as an
0018                               ;       acknowledge fail indicator.  This LED will come on if
0019                               ;       the serial EE fails to acknowledge correctly.
0020                               ;
0021                               ;       PIC16C54 to Serial EE Connections:
0022                               ;
0023                               ;       PIC16C54           Serial EE
0024                               ;       _____
0025                               ;       Pin 12 (RB6) ->  SCLK
0026                               ;       Pin 13 (RB7) ->  SDATA
0027                               ;
0028                               ;       PIN 18 (RA1) ->  Acknowledge fail LED  (Optional)
0029                               ;
0030                               ;*****
0031                               ;       Register Definitions
0032                               ;*****
0033      0003  status  equ   3h      ; status register
0034      0005  port_a  equ   5h      ; port 5 (port_a) used for LEDs
0035      0006  port_b  equ   6h      ; port 6 (port b) used for data and
0036                               ; clock lines
0037      000A  eeprom  equ   0ah     ; bit buffer
0038      000B  bycnt  equ   0bh     ; byte counter for read mode
0039      000C  addr   equ   0ch     ; address counter
0040      000D  datai  equ   0dh     ; data input register
0041      000E  datao  equ   0eh     ; data output register
0042      000F  slave  equ   0fh     ; device address (1010xxx0)
0043      0010  txbuf  equ   10h     ; transmit buffer
0044      0011  count  equ   11h     ; bit counter
0045      0012  bcount equ   12h     ; byte counter
0046      0015  loops  equ   15h     ; delay loop counter
0047      0016  loops2 equ   16h     ; delay loop counter
0048                               ;*****
0049                               ;       Bit Definitions
0050                               ;*****
0051      0007  di     equ    7       ; eeprom input bit
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:46 1994 Page 2

```
Line   PC   Opcode

0052      0006  do     equ    6       ; eeprom output bit
0053      0007  sdata  equ    7       ; serial EE data line (port_b,pin 13)
0054      0006  sclk   equ    6       ; serial EE clock line (port_b,pin 12)
0055      0001  ackf   equ    1       ; acknowledge fail LED, port_a
0056                               ;*****
0057                               ;
0058                               ;
0059      0000      org   01ffh      ; set reset vector
0060      01FF  0A5B    goto  PWRUP
```

Interfacing the 24LCXXB Serial EEPROMs

```

0061          0000          org    000h   ;
0062 0000 0A5B          goto   PWRUP
0063          ;
0064          ;*****
0065          ;      Start Bit Subroutine
0066          ;      this routine generates a start bit
0067          ;      (Low going data line while clock is high)
0068          ;*****
0069          ;
0070          BSTART
0071 0001 05E6          bsf    port_b,sdata ; make sure data is high
0072 0002 0C3F          movlw  b'00111111'
0073 0003 0006          tris   port_b    ; set data and clock lines for output
0074 0004 04C6          bcf    port_b,sclk ; make sure clock is low
0075 0005 0000          nop
0076 0006 05C6          bsf    port_b,sclk ; set clock high
0077 0007 0000          nop
0078 0008 0000          nop
0079 0009 0000          nop
0080 000A 0000          nop
0081 000B 0000          nop
0082 000C 04E6          bcf    port_b,sdata ; data line goes low during
0083          ; high clock for start bit
0084 000D 0000          nop
0085 000E 0000          nop
0086 000F 0000          nop
0087 0010 0000          nop
0088 0011 0000          nop ; timing adjustment
0089 0012 04C6          bcf    port_b,sclk ; start clock train
0090 0013 0000          nop
0091 0014 0000          nop
0092 0015 0800          retlw  0
0093          ;
0094          ;      End of Subroutine
0095          ;*****
0096          ;      Stop Bit Subroutine
0097          ;      This routine generates a stop bit
0098          ;      (High going data line while clock is high)
0099          ;*****
0100          BSTOP
0101 0016 0C3F          movlw  b'00111111' ;
0102 0017 0006          tris   port_b    ; set data/clock lines as outputs

```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:46 1994 Page 3

Line	PC	Opcode			
0103	0018	04E6	bcf	port_b,sdata	; make sure data line is low
0104	0019	0000	nop		
0105	001A	0000	nop		
0106	001B	0000	nop		
0107	001C	05C6	bsf	port_b,sclk	; set clock high
0108	001D	0000	nop		
0109	001E	0000	nop		
0110	001F	0000	nop		
0111	0020	05E6	bsf	port_b,sdata	; data goes high while clock high
0112					; for stop bit
0113	0021	0000	nop		
0114	0022	0000	nop		
0115	0023	04C6	bcf	port_b,sclk	; set clock low again
0116	0024	0000	nop		
0117	0025	0000	nop		
0118	0026	0000	nop		
0119	0027	0800	retlw	0	
0120					
0121					; End of Subroutine
0122					;*****
0123					; BITOUT routine takes one bit of data in 'do' and
0124					; transmits it to the serial EE device

Interfacing the 24LCXXB Serial EEPROMs

```
0125 ;*****
0126 BITOUT
0127 0028 0C3F movlw b'00111111' ; set data,clock as outputs
0128 0029 0006 tris port_b
0129 002A 07CA btfss eeprom,do ; check for state of data bit to xmit
0130 002B 0A2E goto bitlow ;
0131 002C 05E6 bsf port_b,sdata ; set data line high
0132 002D 0A2F goto clkout ; go toggle the clock
0133
0134 002E 04E6 bitlow bcf port_b,sdata ; output a low bit
0135 002F 05C6 clkout bsf port_b,sclk ; set clock line high
0136 0030 0000 nop
0137 0031 0000 nop
0138 0032 0000 nop
0139 0033 0000 nop
0140 0034 04C6 bcf port_b,sclk ; return clock line low
0141 0035 0800 retlw 0
0142 ;
0143 ; End of Subroutine
0144 ;
0145 ;*****
0146 ; BITIN routine reads one bit of data from the
0147 ; serial EE device and stores it in 'di'
0148 ;*****
0149 BITIN
0150 0036 05EA bsf eeprom,di ; assume input bit is high
0151 0037 0CFB movlw b'10111111' ; make sdata an input line
0152 0038 0006 tris port_b
0153 0039 05C6 bsf port_b,sclk ; set clock line high
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:46 1994 Page 4

```
Line PC Opcode
0154 003A 0000 nop ; just sit here a sec
0155 003B 0000 nop
0156 003C 0000 nop
0157 003D 0000 nop
0158 003E 0000 nop ;
0159 003F 07E6 btfss port_b,sdata ; read the data bit
0160 0040 04EA bcf eeprom,di ; input bit was low, set 'di' accordingly
0161 0041 04C6 bcf port_b,sclk ; set clock line low
0162 ;
0163 0042 0800 retlw 0
0164 ;
0165 ;*****
0166 ; Transmit Data Subroutine
0167 ; This routine takes the byte of data stored in the
0168 ; 'datao' register and transmits it to the serial EE device.
0169 ; It will then send 1 more clock to the serial EE for the
0170 ; acknowledge bit. If the ack bit from the part was low
0171 ; then the transmission was successful. If it is high, then
0172 ; the device did not send a proper ack bit and the ack
0173 ; fail LED will be turned on.
0174 ;*****
0175 TX
0176 0043 0C08 movlw .8
0177 0044 0031 movwf count ; set the #bits to 8
0178 ;
0179 TXLP
0180 0045 04CA bcf eeprom,do ; assume bit out is low
0181 0046 06F0 btfsc txbuf,7 ; is bit out really low?
0182 0047 05CA bsf eeprom,do ; no, set it high
0183 0048 0928 call BITOUT ; send the bit to serial EE
0184 0049 0370 rlf txbuf ; rotate txbuf left
0185 004A 02F1 decfsz count ; 8 bits done?
0186 004B 0A45 goto TXLP ; no - go again
0187 004C 0936 call BITIN ; read ack bit
0188 004D 06EA btfsc eeprom,di ; check ack bit
```

Interfacing the 24LCXXB Serial EEPROMs

```

0189 004E 0525      bsf    port_a,ackf    ; set acknowledge fail LED if the
0190                                     ; device did not pull data low
0191                ;
0192 004F 0800      retlw  0
0193                ;
0194                ;*****
0195                ;   Receive data Routine
0196                ;   This routine reads one byte of data from the part
0197                ;   into the 'datai' register. It then sends a high
0198                ;   ack bit to indicate that no more data is to be read
0199                ;*****
0200                RX
0201 0050 0C08      movlw  .8              ; set # bits to 8
0202 0051 0031      movwf  count
0203 0052 006D      clrf   datai          ; clear input register
0204 0053 0403      bcf    status,0       ; make sure carry bit is low

```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:46 1994 Page 5

```

Line  PC  Opcode
0205 0054 036D RXLP   rlf    datai          ; rotate datai 1 bit left
0206 0055 0936      call  BITIN          ; read a bit
0207 0056 06EA      btfsc eeeprom,di
0208 0057 050D      bsf    datai,0       ; set bit 0 if necessary
0209 0058 02F1      decfsz count         ; 8 bits done?
0210 0059 0A54      goto  RXLP          ; no, do another
0211 005A 0800      retlw  0
0212                ;
0213                ;*****
0214                ;   Power up routine
0215                ;   This is the program entry point, which in this case simply
0216                ;   sets the port_a I/O lines and directs control to the
0217                ;   Read routine.
0218                ;*****
0219                PWRUP
0220 005B 0C00      movlw  b'00000000'
0221 005C 0005      tris  port_a         ; set port A as all output
0222 005D 0065      clrf  port_a         ; all output lines low
0223 005E 0A5F      goto  READ
0224                ;
0225                ;*****
0226                ;   READ (sequential read routine)
0227                ;
0228                ;
0229                ;   This routine reads 8 consecutive addresses of the
0230                ;   serial EE device starting at address 00 in the
0231                ;   sequential read mode. Reading in this mode is more
0232                ;   efficient than the random read mode as the control byte
0233                ;   and address have to be sent only once at the beginning
0234                ;   of the sequence. As many consecutive addresses as
0235                ;   needed can then be read from the part until a stop bit is
0236                ;   sent. In the read mode, the PIC16C54 must send the acknowledge
0237                ;   bit after every 8 data bits from the device. When the
0238                ;   last byte needed has been read, then the controller will
0239                ;   send a high acknowledge bit and then a stop bit to halt
0240                ;   transmission from the device.
0241                ;*****
0242                READ
0243                ;
0244 005F 0425      bcf    port_a,ackf   ; clear the ack fail LED if on
0245 0060 0C08      movlw  .8
0246 0061 0032      movwf  bcount        ; set number of bytes to read as 8
0247 0062 0CA0      movlw  b'10100000'  ; set slave address and write mode
0248 0063 002F      movwf  slave
0249 0064 006C      clrf  addr           ; set starting address to 00
0250                ;
0251 0065 0901      call  BSTART        ; generate start bit
0252 0066 020F      movf  slave,w        ; get slave address

```

Interfacing the 24LCXXB Serial EEPROMs

```
0253 0067 0030      movwf  txbuf      ; into transmit buffer
0254 0068 0943      call   TX         ; and send it
0255 0069 020C      movf   addr,w    ; get word address
```

16c5x/7x Cross-Assembler V4.12 Released Mon Jun 06 10:48:46 1994 Page 6

```
Line  PC  Opcode
-----
0256 006A 0030      movwf  txbuf      ; into transmit buffer
0257 006B 0943      call   TX         ; and send it
0258 006C 0901      call   BSTART     ; generate start bit
0259 006D 0CA1      movlw  b'10100001' ; get slave address and read mode
0260 006E 0030      movwf  txbuf      ; into transmit buffer
0261 006F 0943      call   TX         ; and transmit it
0262                                ;
0263 0070 0950  rbyte    call   RX         ; read 1 byte from device
0264 0071 02F2      decfsz bcount     ; are all 8 bytes read?
0265 0072 0A77      goto  lowack      ; no, send low ack and do another
0266 0073 05CA      bsf    eeprom,do  ; yes, send high ack bit
0267 0074 0928      call   BITOUT     ; to stop transmission
0268 0075 0916      call   BSTOP      ; and send a stop bit
0269 0076 0A5F      goto  READ        ; start all over
0270
0271 0077 04CA  lowack    bcf    eeprom,do  ; send low ack bit
0272 0078 0928      call   BITOUT     ; to continue transmission
0273 0079 0A70      goto  rbyte       ; and read another byte
0274                                ;
0275          0000  END
```

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.

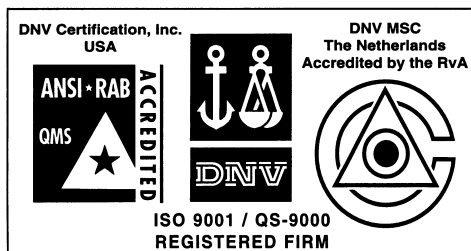
dsPIC, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, MXDEV, PICC, PICDEM, PICDEM.net, rPIC, Select Mode and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A.

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