

Interfacing PICmicroTM to an LCD Module

*Author: Mark Palmer
Code: Mark Palmer/Scott Fink
Microchip Technology Inc.*

INTRODUCTION

This application note interfaces a micrange PICmicro device to a Hitachi[®] LM032L LCD character display module, with a two line by twenty character display. LCD modules are useful for displaying text information from a system. In large volume applications, the use of custom LCD displays becomes economical. The routines provided should be a good starting point for users whose applications implement a custom LCD. This source code should be compatible with the PIC16C5X devices, after modifications for the special function register initialization, but has not been verified on those devices.

OPERATION

The Hitachi LM032L LCD character display module can operate in one of two modes. The first (and default) mode is the 4-bit data interface mode. The second is the 8-bit data interface mode. When operating in 4-bit mode, two transfers per character / command are required. 8-bit mode, though easier to implement (less program memory) requires four additional I/O lines. The use of 8-bit mode is strictly a program memory size vs. I/O trade-off. The three most common data interfaces from the microcontroller are:

1. An 8-bit interface.
2. A 4-bit interface, with data transfers on the high nibble of the port.
3. A 4-bit interface, with data transfers on the low nibble of the port.

The LCD module also has three control signals, Enable (E), Read/Write (R_W), and Register Select (RS). The function of each control signal is shown in Table 1.

TABLE 1: CONTROL SIGNAL FUNCTIONS

Control Signal	Function
E	Causes data/control state to be latched Rising Edge = Latches control state (RS and R_W) Falling Edge = Latches data
RS	Register Select Control 1 = LCD in data mode 0 = LCD in command mode
R_W	Read / Write control 1 = LCD to write data 0 = LCD to read data

A single source file, with conditional assembly is used to generate each of these three options. This requires two flags. The flags and their results are shown in Table 2.

TABLE 2: CONDITIONAL ASSEMBLY FLAGS

Flags		Result
Four_bit	Data_HI	
1	0	4-bit mode. Data transferred on the low nibble of the port.
1	1	4-bit mode. Data transferred on the high nibble of the port.
0	x	8-bit mode.

AN587

Figure 1, Figure 2, and Figure 3 show the block diagrams for the three different data interfaces. The LCD_CNTL and LCD_DATA lines are user definable to

their port assignment. This is accomplished with EQUate statements in the source code. See Appendices B, C, and D.

FIGURE 1: 8-BIT DATA INTERFACE

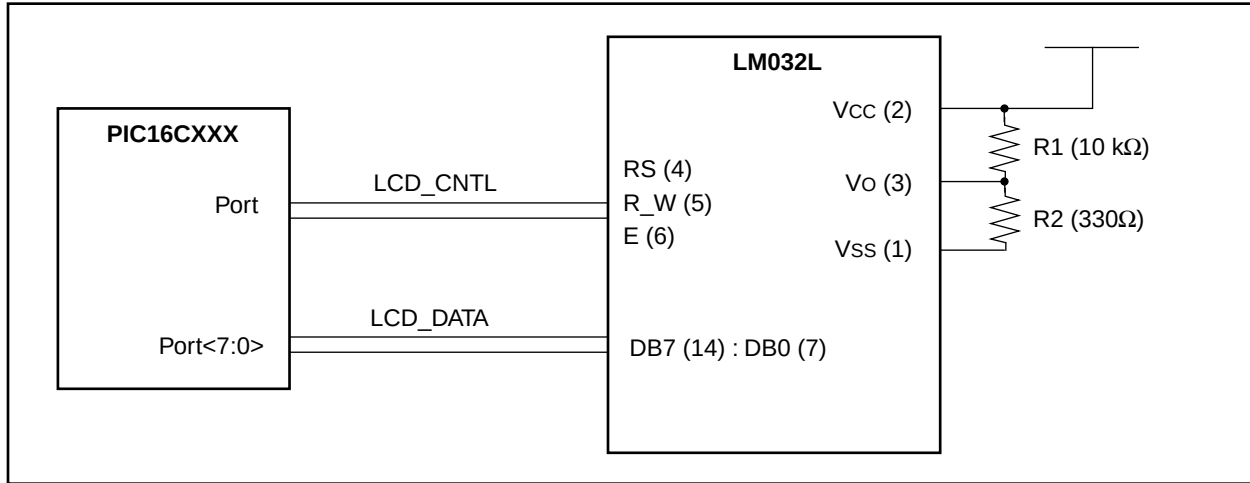


FIGURE 2: 4-BIT MODE; DATA TRANSFERRED ON THE HIGH NIBBLE OF THE PORT

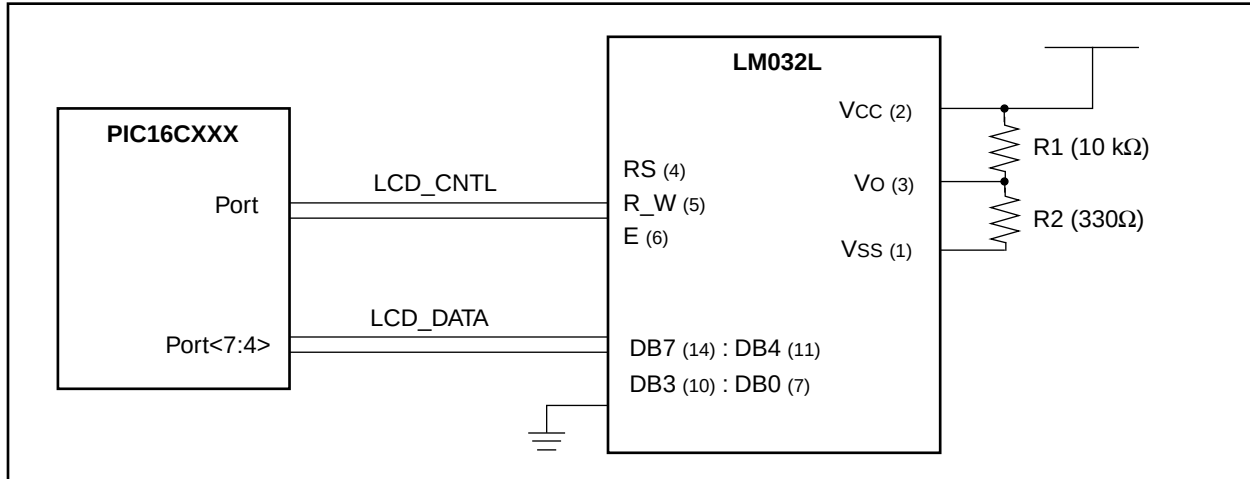
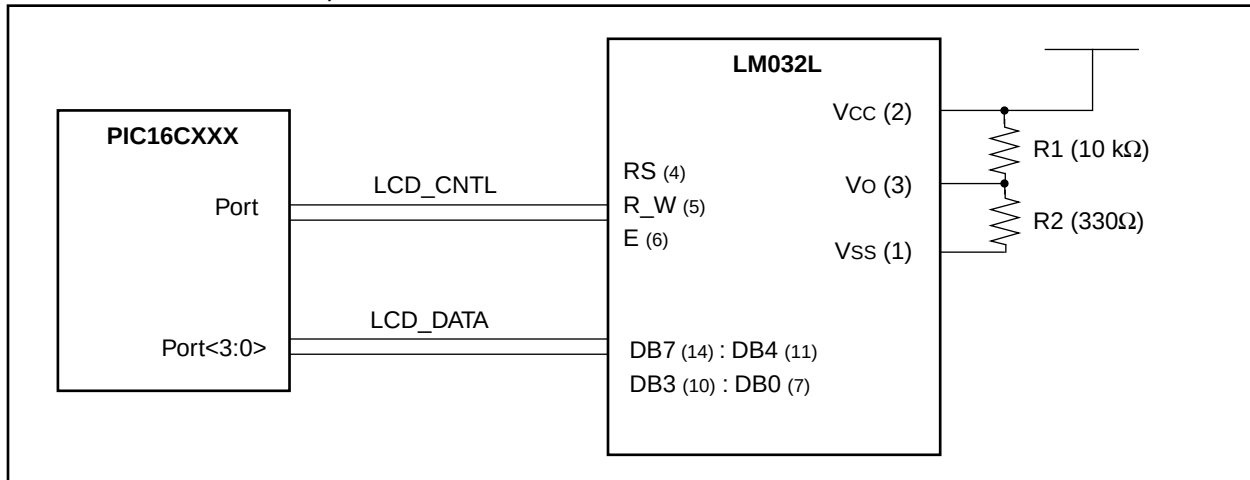


FIGURE 3: 4-BIT MODE; DATA TRANSFERRED ON THE LOW NIBBLE OF THE PORT



LCD's (drivers) are slow devices when compared to microcontrollers. Care must be taken from having communication occur too quickly. The software will need to control communication speed and timing to ensure the slow LCD and fast microcontroller can stay synchronized. The timing requirements of the LM032L are shown in Appendix A. We recommend that the complete specifications of the LM032L be acquired from Hitachi or a Hitachi distributor. The literature numbers are CE-E613Q and M24T013 for a LM032L display driver.

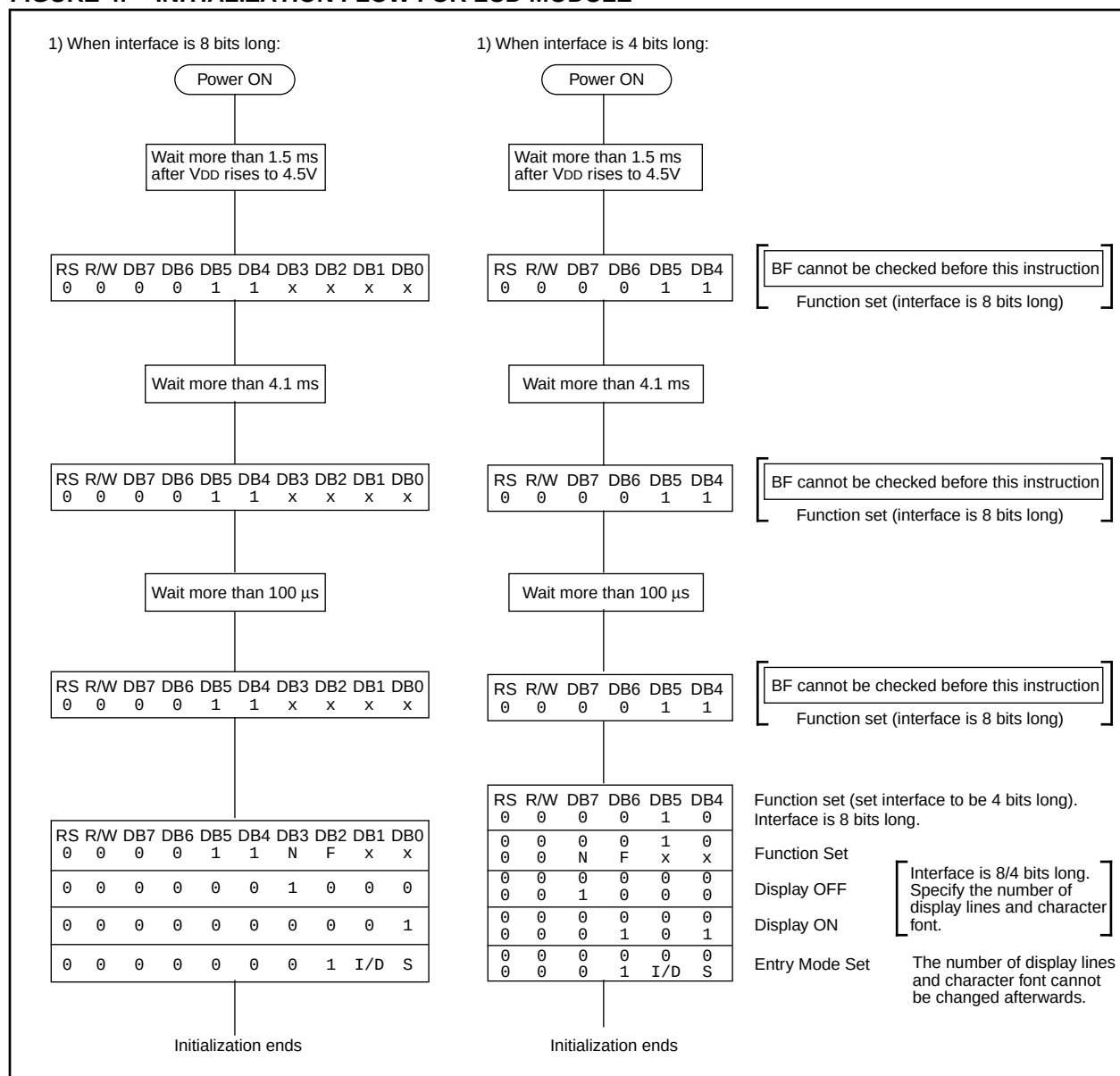
When the module powers up, the default data transfer mode is 8-bit. The initialization sequence only requires commands that are 4-bit in length. The last initialization

command needs to specify the data transfer width (4-or 8-bit). Then a delay of 4.6 ms must be executed before the LCD module can be initialized. Some of the LCD module commands are:

- 1 or 2 lines of characters
- Display on /off
- Clear display
- Increment / do not increment character address pointer after each character
- Load character address pointer

The initialization flow for the module is shown in Figure 4.

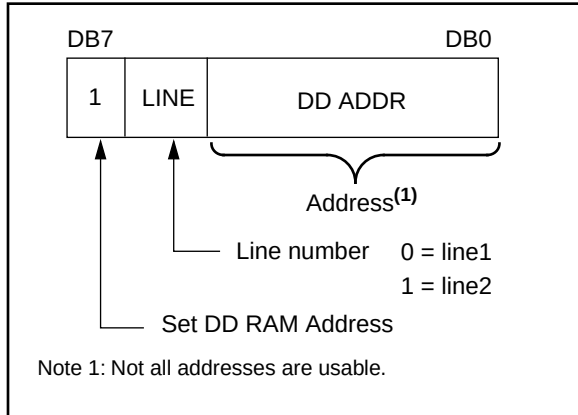
FIGURE 4: INITIALIZATION FLOW FOR LCD MODULE



AN587

After initialization, each character address is individually addressable. Figure 5 shows the structure of the command to specify the character address.

FIGURE 5: CHARACTER ADDRESS COMMAND FORMAT



The Hitachi Display Drive (HD44780A) has 80 bytes of RAM. The LM032L modules only use 40 bytes of the available RAM (2 x 20 characters). It is possible to use the remaining RAM locations for storage of other information.

Figure 6 shows the display data positions supported by the display driver as well as the characters actually displayed by the module (the non-shaded addresses).

The program example implemented here uses the character auto increment feature. This automatically increments the character address pointer after each character is written to the display.

CONCLUSION

The Hitachi LM032L character display module is well suited for displaying information. The selection of 4-bit or 8-bit data transfer mode is strictly a program memory size vs. I/O resource trade-off. The supplied code is easily used in any of three common data interfaces. The source is easily modifiable to a designers specific application needs. Other display modules/drivers may be implemented with the appropriate modifications. Table 3 shows the resource requirements for the three subroutines SEND_CHAR, SEND_COMMAND, and BUSY_CHECK in the various data interface modes.

FIGURE 6: DISPLAY DRIVER (DD) RAM LOCATIONS

digit	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	---	33	34	35	36	37	38	39	40	← Display position
line-1	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	---	20	21	22	23	24	25	26	27	← DD RAM address (Hexadecimal)
line-2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	---	60	61	62	63	64	65	66	67	←

Note: Shaded locations are not displayed on the LM032L display module.

TABLE 3: RESOURCE REQUIREMENTS

Mode	Program Memory	Data Memory	Verified On
8-bit	32	3	PICDEM-2 ⁽¹⁾
4-bit, Data transferred on the high nibble of the port.	53	3	PICDEM-2 ⁽¹⁾
4-bit, Data transferred on the high nibble of the port.	53	3	Low-Power Real-Time Clock Board (AN582)

Note 1: Jumper J6 must be removed.

APPENDIX A: LM032L TIMING REQUIREMENTS

TABLE A-1: TIMING CHARACTERISTICS

Parameter #	Symbol	Characteristics	Min.	Typ.	Max.	Unit
1	TCYC	Enable cycle time	1.0	—	—	μs
2	PWEH	Enable pulse width	450	—	—	μs
3	T_{ER} , T_{EF}	Enable rise / fall time	—	—	25	μs
4	TAS	RS, R/W set-up time	140	—	—	μs
5	TDDR	Data delay time	—	—	320	μs
6	Tdsu	Data setup time	195	—	—	μs
7	TH	Hold time	20	—	—	μs

FIGURE A-1: DATA WRITE INTERFACE TIMING

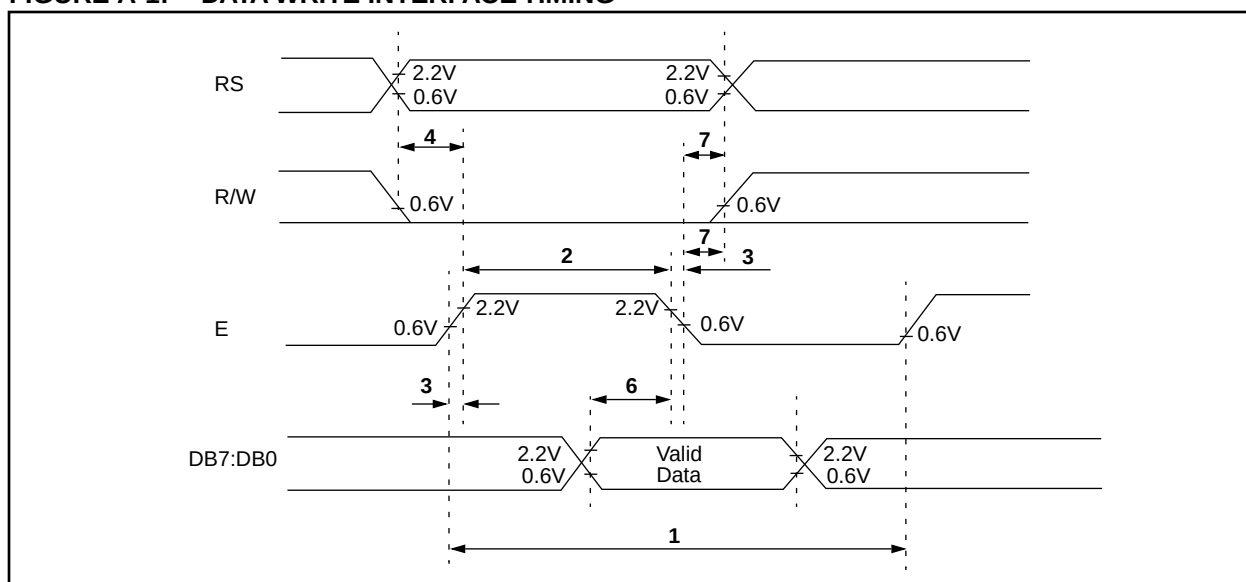
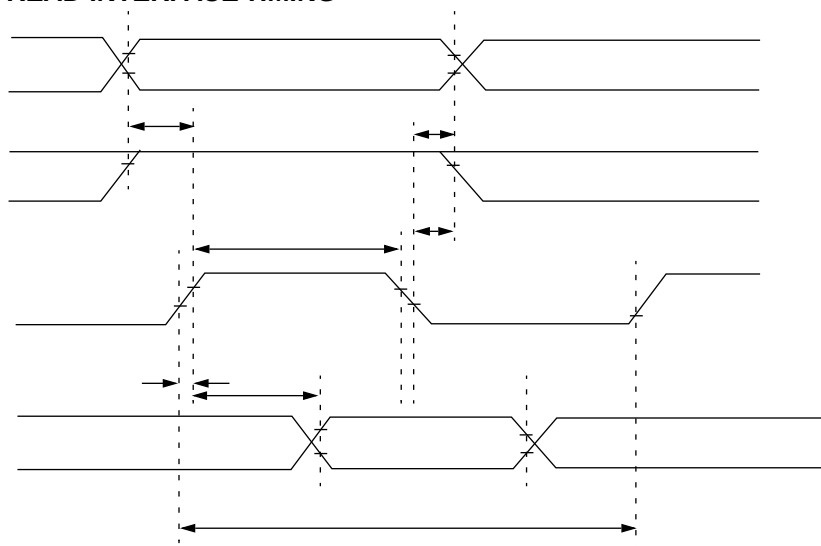


FIGURE A-2: DATA READ INTERFACE TIMING



Note: Refer to Hitachi documentation for the most current timing specifications.

TABLE A-2: LM032L PIN CONNECTION

Pin No.	Symbol	Level	Function	
1	VSS	—	0V	Ground
2	VDD	—	+5V	Power Supply(+)
3	Vo	—	—	Ground
4	RS	H/L	L: Instruction Code Input H: Data Input	
5	R/W	H/L	H: Data Read (LCD module→MPU) L: Data Write (LCD module←MPU)	
6	E	H,H→L	Enable Signal	
7	DB0	H/L	Data Bus Line Note (1), (2)	
8	DB1	H/L		
9	DB2	H/L		
10	DB3	H/L		
11	DB4	H/L		
12	DB5	H/L		
13	DB6	H/L		
14	DB7	H/L		

In the HD44780, the data can be sent in either two 4-bit operations or one 8-bit operation, This flexibility allows an interface to both 4- and 8-bit MPUs.

- Note 1: When interface data is 4-bits long, data is transferred using only 4 lines of DB7:DB4 (DB3:DB0 are not used). Data transfer between the HD44780 and the MPU completes when 4-bits of data is transferred twice. Data of the higher order 4 bits (contents of DB7:DB4 when interface data is 8-bits long) is transferred first and then lower order 4 bits (contents of DB3:DB0 when interface data is 8-bits long).
- 2: When interface data is 8-bits long, data is transferred using 8 data lines of DB7:DB0.

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX B: 8-BIT DATA INTERFACE LISTING

MPASM 01.40.01 Intermediate LM032L.ASM 4-7-1997 9:43:02 PAGE 1

LOC	OBJECT CODE	LINE	SOURCE TEXT
	VALUE		
		00001	LIST P=16C64
		00002	ERRORLEVEL -302
		00003	;
		00004	; This program interfaces to a Hitachi (LM032L) 2 line by 20 character display
		00005	module. The program assembles for either 4-bit or 8-bit data interface, depending
		00006	on the value of the 4bit flag. LCD_DATA is the port which supplies the data to
		00007	the LM032L, while LCD_CNTL is the port that has the control lines (E, RS, RW).
		00008	In 4-bit mode the data is transfer on the high nibble of the port (PORT<7:4>).
		00009	;
		00010	Program = LM032L.ASM
		00011	Revision Date: 5-10-94
		00012	1-22-97 Compatibility with MPASMWIN 1.40
		00013	;
		00014	;
		00015	include <p16c64.inc>
		00001	LIST
		00002	; P16C64.INC Standard Header File, Version 1.01 Microchip Technology, Inc.
		00238	LIST
		00016	
0000009F		00017	ADCON1 EQU 9F
		00018	
00000000		00019	FALSE EQU 0
00000001		00020	TRUE EQU 1
		00021	
		00022	include <lm032l.h>
		00069	list
		00023	;
00000001		00024	Four_bit EQU TRUE ; Selects 4- or 8-bit data transfers
00000000		00025	Data_HI EQU FALSE ; If 4-bit transfers, Hi or Low nibble of PORT
		00026	;
		00027	;
		00028	if (Four_bit && !Data_HI)
		00029	;
00000006		00030	LCD_DATA EQU PORTB
00000086		00031	LCD_DATA_TRIS EQU TRISB

```

00032 ;
00033     else
00034 ;
00035 LCD_DATA      EQU  PORTD
00036 LCD_DATA_TRIS EQU  TRISD
00037 ;
00038     endif
00039 ;
00000005 00040 LCD_CNTL      EQU  PORTA
00041 ;
00042 ;
00043 ;
00044 ; LCD Display Commands and Control Signal names.
00045 ;
00046     if ( Four_bit && !Data_HI )
00047 ;
00000000 00048 E              EQU  0              ; LCD Enable control line
00000001 00049 RW          EQU  1              ; LCD Read/Write control line
00000002 00050 RS          EQU  2              ; LCD Register Select control line
00051 ;
00052     else
00053 ;
00054 E              EQU  3              ; LCD Enable control line
00055 RW          EQU  2              ; LCD Read/Write control line
00056 RS          EQU  1              ; LCD Register Select control line
00057 ;
00058     endif
00059 ;
00060 ;
00000030 00061 TEMP1          EQU  0x030
00062 ;
0000      00063     org      RESET_V          ; RESET vector location
0000 2808 00064 RESET      GOTO      START      ;
00065 ;
00066 ; This is the Periperal Interrupt routine. Should NOT get here
00067 ;
00068     page
0000      00069     org      ISR_V              ; Interrupt vector location
0000      00070 PER_INT_V
0004 1283 00071 ERROR1      BCF  STATUS, RP0      ; Bank 0
0005 1407 00072          BSF  PORTC, 0
0006 1007 00073          BCF  PORTC, 0
0007 2804 00074          GOTO ERROR1
00075 ;
00076 ;
00077 ;
0008      00078 START                      ; POWER_ON Reset (Beginning of program)

```



```

0008 0183      00079          CLRF    STATUS          ; Do initialization (Bank 0)
0009 018B      00080          CLRF    INTCON
000A 018C      00081          CLRF    PIR1
000B 1683      00082          BSF     STATUS, RP0      ; Bank 1
000C 3000      00083          MOVLW   0x00            ; The LCD module does not like to work w/ weak pull-ups
000D 0081      00084          MOVWF   OPTION_REG      ;
000E 018C      00085          CLRF    PIE1            ; Disable all peripheral interrupts
00086 ;***
00087 ;***      If using device with A/D, these two instructions are required.
00088 ;***
00089 ;          MOVLW   0xFF          ;
00090 ;          MOVWF   ADCON1        ; Port A is Digital.
00091 ;
00092 ;
000F 1283      00093          BCF     STATUS, RP0      ; Bank 0
0010 0185      00094          CLRF    PORTA          ; ALL PORT output should output Low.
0011 0186      00095          CLRF    PORTB
0012 0187      00096          CLRF    PORTC
0013 0188      00097          CLRF    PORTD
0014 0189      00098          CLRF    PORTE
0015 1010      00099          BCF     T1CON, TMR1ON    ; Timer 1 is NOT incrementing
00100 ;
0016 1683      00101          BSF     STATUS, RP0      ; Select Bank 1
0017 0185      00102          CLRF    TRISA          ; RA5 - 0 outputs
0018 30F0      00103          MOVLW   0xF0            ;
0019 0086      00104          MOVWF   TRISB          ; RB7 - 4 inputs, RB3 - 0 outputs
001A 0187      00105          CLRF    TRISC          ; RC Port are outputs
001B 1407      00106          BSF     TRISC, T10S0     ; RC0 needs to be input for the oscillator to function
001C 0188      00107          CLRF    TRISD          ; RD Port are outputs
001D 0189      00108          CLRF    TRISE          ; RE Port are outputs
001E 140C      00109          BSF     PIE1, TMR1IE     ; Enable TMR1 Interrupt
001F 1781      00110          BSF     OPTION_REG, NOT_RBPU ; Disable PORTB pull-ups
0020 1283      00111          BCF     STATUS, RP0      ; Select Bank 0
00112 ;
00113      page
00114 ;
00115 ; Initialize the LCD Display Module
00116 ;
0021 0185      00117          CLRF    LCD_CNTL        ; ALL PORT output should output Low.
00118
0022          00119 DISPLAY_INIT
00120          if ( Four_bit && !Data_HI )
0022 3002      00121          MOVLW   0x02            ; Command for 4-bit interface low nibble
00122          endif
00123 ;
00124          if ( Four_bit && Data_HI )
00125          MOVLW   0x020            ; Command for 4-bit interface high nibble

```

```

00126     endif
00127 ;
00128     if ( !Four_bit )
00129         MOVLW    0x038                ; Command for 8-bit interface
00130     endif
00131 ;
0023 0086 00132         MOVWF    LCD_DATA                ;
0024 1405 00133         BSF      LCD_CNTL, E                ;
0025 1005 00134         BCF      LCD_CNTL, E                ;
00135 ;
00136 ; This routine takes the calculated times that the delay loop needs to
00137 ; be executed, based on the LCD_INIT_DELAY EQUate that includes the
00138 ; frequency of operation. These uses registers before they are needed to
00139 ; store the time.
00140 ;
0026 3006 00141 LCD_DELAY  MOVLW    LCD_INIT_DELAY                ;
0027 00B3 00142         MOVWF    MSD                        ; Use MSD and LSD Registers to Initialize LCD
0028 01B4 00143         CLRF     LSD                        ;
0029 0BB4 00144 LOOP2     DECFSZ  LSD, F                    ; Delay time = MSD * ((3 * 256) + 3) * Tcy
002A 2829 00145         GOTO    LOOP2                        ;
002B 0BB3 00146         DECFSZ  MSD, F                    ;
002C      00147 END_LCD_DELAY
002C 2829 00148         GOTO    LOOP2                        ;
00149 ;
00150 ; Command sequence for 2 lines of 5x7 characters
00151 ;
002D      00152 CMD_SEQ
00153 ;
00154         if ( Four_bit )
00155             if ( !Data_HI )
002D 3002 00156                 MOVLW    0x02                ; 4-bit low nibble xfer
00157             else
00158                 MOVLW    0x020                ; 4-bit high nibble xfer
00159             endif
00160         ;
00161         else
00162             MOVLW    0x038                ; 8-bit mode
00163         endif
00164 ;
002E 0086 00165         MOVWF    LCD_DATA                ; This code for both 4-bit and 8-bit modes
002F 1405 00166         BSF      LCD_CNTL, E                ;
0030 1005 00167         BCF      LCD_CNTL, E                ;
00168 ;
00169         if ( Four_bit )
00170             if ( !Data_HI )
0031 3008 00171                 MOVLW    0x08                ; 4-bit low nibble xfer
00172             else

```

```

00173          MOVLW    0x080          ; 4-bit high nibble xfer
00174          endif
0032 0086      00175          MOVWF    LCD_DATA          ;
0033 1405      00176          BSF      LCD_CNTL, E          ;
0034 1005      00177          BCF      LCD_CNTL, E          ;
00178          endif
00179          ;
00180          ; Busy Flag should be valid after this point
00181          ;
0035 300C      00182          MOVLW    DISP_ON          ;
0036 2072      00183          CALL     SEND_CMD          ;
0037 3001      00184          MOVLW    CLR_DISP          ;
0038 2072      00185          CALL     SEND_CMD          ;
0039 3006      00186          MOVLW    ENTRY_INC          ;
003A 2072      00187          CALL     SEND_CMD          ;
003B 3080      00188          MOVLW    DD_RAM_ADDR          ;
003C 2072      00189          CALL     SEND_CMD          ;
00190          ;
00191          page
00192          ;
00193          ;Send a message the hard way
003D 304D      00194          movlw    'M'
003E 2063      00195          call     SEND_CHAR
003F 3069      00196          movlw    'i'
0040 2063      00197          call     SEND_CHAR
0041 3063      00198          movlw    'c'
0042 2063      00199          call     SEND_CHAR
0043 3072      00200          movlw    'r'
0044 2063      00201          call     SEND_CHAR
0045 306F      00202          movlw    'o'
0046 2063      00203          call     SEND_CHAR
0047 3063      00204          movlw    'c'
0048 2063      00205          call     SEND_CHAR
0049 3068      00206          movlw    'h'
004A 2063      00207          call     SEND_CHAR
004B 3069      00208          movlw    'i'
004C 2063      00209          call     SEND_CHAR
004D 3070      00210          movlw    'p'
004E 2063      00211          call     SEND_CHAR
00212
004F 30C0      00213          movlw    B'11000000'          ;Address DDRam first character, second line
0050 2072      00214          call     SEND_CMD
00215
00216          ;Demonstration of the use of a table to output a message
0051 3000      00217          movlw    0          ;Table address of start of message
0052          00218          dispmsg
0052 00B0      00219          movwf    TEMP1          ;TEMP1 holds start of message address

```

```

0053 2099      00220      call    Table
0054 39FF      00221      andlw   0FFh          ;Check if at end of message (zero
0055 1903      00222      btfsc   STATUS,Z      ;returned at end)
0056 285B      00223      goto    out
0057 2063      00224      call    SEND_CHAR      ;Display character
0058 0830      00225      movf    TEMPI,w        ;Point to next character
0059 3E01      00226      addlw   1
005A 2852      00227      goto    dispmsg
005B           00228      out
005B           00229      loop
005B 285B      00230      goto    loop          ;Stay here forever
005B           00231      ;
005B           00232      ;
005C           00233      INIT_DISPLAY
005C 300C      00234      MOVLW   DISP_ON        ; Display On, Cursor On
005D 2072      00235      CALL     SEND_CMD      ; Send This command to the Display Module
005E 3001      00236      MOVLW   CLR_DISP      ; Clear the Display
005F 2072      00237      CALL     SEND_CMD      ; Send This command to the Display Module
0060 3006      00238      MOVLW   ENTRY_INC     ; Set Entry Mode Inc., No shift
0061 2072      00239      CALL     SEND_CMD      ; Send This command to the Display Module
0062 0008      00240      RETURN
0062           00241      ;
0062           00242      page
0062           00243      ;
0062           00244      ;*****
0062           00245      ;* The LCD Module Subroutines *
0062           00246      ;*****
0062           00247      ;
0062           00248      ;      if ( Four_bit )      ; 4-bit Data transfers?
0062           00249      ;
0062           00250      ;      if ( Data_HI )      ; 4-bit transfers on the high nibble of the PORT
0062           00251      ;
0062           00252      ;*****
0062           00253      ;*SendChar - Sends character to LCD *
0062           00254      ;*This routine splits the character into the upper and lower *
0062           00255      ;*nibbles and sends them to the LCD, upper nibble first. *
0062           00256      ;*****
0062           00257      ;
0062           00258      SEND_CHAR
0062           00259      MOVWF   CHAR          ;Character to be sent is in W
0062           00260      CALL     BUSY_CHECK      ;Wait for LCD to be ready
0062           00261      MOVF     CHAR,w
0062           00262      ANDLW   0xF0          ;Get upper nibble
0062           00263      MOVWF   LCD_DATA      ;Send data to LCD
0062           00264      BCF      LCD_CNTL, RW      ;Set LCD to read
0062           00265      BSF      LCD_CNTL, RS      ;Set LCD to data mode
0062           00266      BSF      LCD_CNTL, E      ;toggle E for LCD

```

```

00267          BCF      LCD_CNTL, E
00268          SWAPF    CHAR, W
00269          ANDLW    0xF0          ;Get lower nibble
00270          MOVWF    LCD_DATA      ;Send data to LCD
00271          BSF      LCD_CNTL, E   ;toggle E for LCD
00272          BCF      LCD_CNTL, E
00273          RETURN
00274 ;
00275          else          ; 4-bit transfers on the low nibble of the PORT
00276 ;
00277 ;*****
00278 ;* SEND_CHAR - Sends character to LCD *
00279 ;* This routine splits the character into the upper and lower *
00280 ;* nibbles and sends them to the LCD, upper nibble first. *
00281 ;* The data is transmitted on the PORT<3:0> pins *
00282 ;*****
00283 ;
0063 00284 SEND_CHAR
0063 00B6 00285          MOVWF    CHAR          ; Character to be sent is in W
0064 2081 00286          CALL     BUSY_CHECK      ; Wait for LCD to be ready
0065 0E36 00287          SWAPF    CHAR, W
0066 390F 00288          ANDLW    0x0F          ; Get upper nibble
0067 0086 00289          MOVWF    LCD_DATA      ; Send data to LCD
0068 1085 00290          BCF      LCD_CNTL, RW   ; Set LCD to read
0069 1505 00291          BSF      LCD_CNTL, RS   ; Set LCD to data mode
006A 1405 00292          BSF      LCD_CNTL, E   ; toggle E for LCD
006B 1005 00293          BCF      LCD_CNTL, E
006C 0836 00294          MOVF     CHAR, W
006D 390F 00295          ANDLW    0x0F          ; Get lower nibble
006E 0086 00296          MOVWF    LCD_DATA      ; Send data to LCD
006F 1405 00297          BSF      LCD_CNTL, E   ; toggle E for LCD
0070 1005 00298          BCF      LCD_CNTL, E
0071 0008 00299          RETURN
00300 ;
00301          endif
00302          else
00303 ;
00304 ;*****
00305 ;* SEND_CHAR - Sends character contained in register W to LCD *
00306 ;* This routine sends the entire character to the PORT *
00307 ;* The data is transmitted on the PORT<7:0> pins *
00308 ;*****
00309 ;
00310 SEND_CHAR
00311          MOVWF    CHAR          ; Character to be sent is in W
00312          CALL     BUSY_CHECK      ; Wait for LCD to be ready
00313          MOVF     CHAR, W

```

```

00314          MOVWF    LCD_DATA      ; Send data to LCD
00315          BCF      LCD_CNTL, RW  ; Set LCD in read mode
00316          BSF      LCD_CNTL, RS   ; Set LCD in data mode
00317          BSF      LCD_CNTL, E     ; toggle E for LCD
00318          BCF      LCD_CNTL, E
00319          RETURN
00320 ;
00321     endif
00322 ;
00323     page
00324 ;
00325 ;*****
00326 ;* SendCmd - Sends command to LCD *
00327 ;* This routine splits the command into the upper and lower *
00328 ;* nibbles and sends them to the LCD, upper nibble first. *
00329 ;* The data is transmitted on the PORT<3:0> pins *
00330 ;*****
00331 ;
00332     if ( Four_bit )      ; 4-bit Data transfers?
00333 ;
00334         if ( Data_HI )   ; 4-bit transfers on the high nibble of the PORT
00335 ;
00336 ;*****
00337 ;* SEND_CMD - Sends command to LCD *
00338 ;* This routine splits the command into the upper and lower *
00339 ;* nibbles and sends them to the LCD, upper nibble first. *
00340 ;*****
00341 ;
00342 SEND_CMD
00343     MOVWF    CHAR          ; Character to be sent is in W
00344     CALL     BUSY_CHECK    ; Wait for LCD to be ready
00345     MOVF     CHAR,w
00346     ANDLW    0xF0          ; Get upper nibble
00347     MOVWF    LCD_DATA      ; Send data to LCD
00348     BCF      LCD_CNTL,RW   ; Set LCD to read
00349     BCF      LCD_CNTL,RS   ; Set LCD to command mode
00350     BSF      LCD_CNTL,E     ; toggle E for LCD
00351     BCF      LCD_CNTL,E
00352     SWAPF    CHAR,w
00353     ANDLW    0xF0          ; Get lower nibble
00354     MOVWF    LCD_DATA      ; Send data to LCD
00355     BSF      LCD_CNTL,E     ; toggle E for LCD
00356     BCF      LCD_CNTL,E
00357     RETURN
00358 ;
00359     else                ; 4-bit transfers on the low nibble of the PORT
00360 ;

```

```

0072      00361 SEND_CMD
0072 00B6      00362      MOVWF  CHAR      ; Character to be sent is in W
0073 2081      00363      CALL   BUSY_CHECK ; Wait for LCD to be ready
0074 0E36      00364      SWAPF  CHAR, W
0075 390F      00365      ANDLW  0x0F      ; Get upper nibble
0076 0086      00366      MOVWF  LCD_DATA ; Send data to LCD
0077 1085      00367      BCF    LCD_CNTL, RW ; Set LCD to read
0078 1105      00368      BCF    LCD_CNTL, RS ; Set LCD to command mode
0079 1405      00369      BSF    LCD_CNTL, E ; toggle E for LCD
007A 1005      00370      BCF    LCD_CNTL, E
007B 0836      00371      MOVF   CHAR, W
007C 390F      00372      ANDLW  0x0F      ; Get lower nibble
007D 0086      00373      MOVWF  LCD_DATA ; Send data to LCD
007E 1405      00374      BSF    LCD_CNTL, E ; toggle E for LCD
007F 1005      00375      BCF    LCD_CNTL, E
0080 0008      00376      RETURN
00377 ;
00378      endif
00379      else
00380 ;
00381 ;*****
00382 ;* SEND_CND - Sends command contained in register W to LCD *
00383 ;* This routine sends the entire character to the PORT *
00384 ;* The data is transmitted on the PORT<7:0> pins *
00385 ;*****
00386
00387 SEND_CMD
00388      MOVWF  CHAR      ; Command to be sent is in W
00389      CALL   BUSY_CHECK ; Wait for LCD to be ready
00390      MOVF   CHAR, W
00391      MOVWF  LCD_DATA ; Send data to LCD
00392      BCF    LCD_CNTL, RW ; Set LCD in read mode
00393      BCF    LCD_CNTL, RS ; Set LCD in command mode
00394      BSF    LCD_CNTL, E ; toggle E for LCD
00395      BCF    LCD_CNTL, E
00396      RETURN
00397 ;
00398      endif
00399 ;
00400      page
00401 ;
00402      if ( Four_bit ) ; 4-bit Data transfers?
00403 ;
00404          if ( Data_HI ) ; 4-bit transfers on the high nibble of the PORT
00405 ;
00406 ;*****
00407 ;* This routine checks the busy flag, returns when not busy *

```

```

00408 ;* Affects:
00409 ;* TEMP - Returned with busy/address
00410 ;*****
00411 ;
00412 BUSY_CHECK
00413 BSF STATUS, RP0 ; Select Register Bank1
00414 MOVLW 0xFF ; Set Port_D for input
00415 MOVWF LCD_DATA_TRIS
00416 BCF STATUS, RP0 ; Select Register Bank0
00417 BCF LCD_CNTL, RS ; Set LCD for Command mode
00418 BSF LCD_CNTL, RW ; Setup to read busy flag
00419 BSF LCD_CNTL, E ; Set E high
00420 BCF LCD_CNTL, E ; Set E low
00421 MOVF LCD_DATA, W ; Read upper nibble busy flag, DDRam address
00422 ANDLW 0xF0 ; Mask out lower nibble
00423 MOVWF TEMP
00424 BSF LCD_CNTL, E ; Toggle E to get lower nibble
00425 BCF LCD_CNTL, E
00426 SWAPF LCD_DATA, W ; Read lower nibble busy flag, DDRam address
00427 ANDLW 0x0F ; Mask out upper nibble
00428 IORWF TEMP ; Combine nibbles
00429 BTFSF TEMP, 7 ; Check busy flag, high = busy
00430 GOTO BUSY_CHECK ; If busy, check again
00431 BCF LCD_CNTL, RW
00432 BSF STATUS, RP0 ; Select Register Bank1
00433 MOVLW 0x0F
00434 MOVWF LCD_DATA_TRIS ; Set Port_D for output
00435 BCF STATUS, RP0 ; Select Register Bank0
00436 RETURN
00437 ;
00438 els 1 0 ; 4-bit transfers on the low nibble of the PORT
00439 ;
00440 ;*****
00441 ;* This routine checks the busy flag, returns when not busy
00442 ;* Affects: 0 0
00443 ;* TEMP - Returned with busy/address 0
00444 ;*****
00445 ;
00446 BUSY_CHECK
00447 BSF STATUS, RP0 ; Bank 1
00448 MOVLW 0xFF ; Set PortB for input
00449 MOVWF LCD_DATA_TRIS
00450 BCF STATUS, RP0 ; Bank 0
00451 BCF LCD_CNTL, RS ; Set LCD for Command mode
00452 BSF LCD_CNTL, RW ; Setup to read busy flag
00453 BSF LCD_CNTL, E ; Set E high
00454 BCF LCD_CNTL, E ; Set E low

```



```

0089 0E06      00455      SWAPF    LCD_DATA, W      ; Read upper nibble busy flag, DDRam address
008A 39F0      00456      ANDLW    0xF0           ; Mask out lower nibble
008B 00B5      00457      MOVWF    TEMP           ;
008C 1405      00458      BSF      LCD_CNTL, E      ; Toggle E to get lower nibble
008D 1005      00459      BCF      LCD_CNTL, E
008E 0806      00460      MOVF     LCD_DATA, W      ; Read lower nibble busy flag, DDRam address
008F 390F      00461      ANDLW    0x0F           ; Mask out upper nibble
0090 04B5      00462      IORWF    TEMP, F         ; Combine nibbles
0091 1BB5      00463      BTFSC    TEMP, 7         ; Check busy flag, high = busy
0092 2881      00464      GOTO     BUSY_CHECK      ; If busy, check again
0093 1085      00465      BCF      LCD_CNTL, RW
0094 1683      00466      BSF      STATUS, RP0      ; Bank 1
0095 30F0      00467      MOVLW    0xF0           ;
0096 0086      00468      MOVWF    LCD_DATA_TRIS    ; RB7 - 4 = inputs, RB3 - 0 = output
0097 1283      00469      BCF      STATUS, RP0      ; Bank 0
0098 0008      00470      RETURN
00471 ;
00472      endif
00473      else
00474 ;
00475 ;*****
00476 ;* This routine checks the busy flag, returns when not busy *
00477 ;* Affects: *
00478 ;* TEMP - Returned with busy/address *
00479 ;*****
00480 ;
00481 BUSY_CHECK
00482      BSF      STATUS, RP0      ; Select Register Bank1
00483      MOVLW    0xFF           ; Set port_D for input
00484      MOVWF    LCD_DATA_TRIS
00485      BCF      STATUS, RP0      ; Select Register Bank0
00486      BCF      LCD_CNTL, RS      ; Set LCD for command mode
00487      BSF      LCD_CNTL, RW      ; Setup to read busy flag
00488      BSF      LCD_CNTL, E      ; Set E high
00489      BCF      LCD_CNTL, E      ; Set E low
00490      MOVF     LCD_DATA, w      ; Read busy flag, DDRam address
00491      MOVWF    TEMP
00492      BTFSC    TEMP, 7         ; Check busy flag, high=busy
00493      GOTO     BUSY_CHECK
00494      BCF      LCD_CNTL, RW
00495      BSF      STATUS, RP0      ; Select Register Bank1
00496      MOVLW    0x00           ;
00497      MOVWF    LCD_DATA_TRIS    ; Set port_D for output
00498      BCF      STATUS, RP0      ; Select Register Bank0
00499      RETURN
00500 ;
00501      endif

```

```

00502     page
00503 ;
0099     00504 Table
0099 0782     00505         addwf PCL, F           ;Jump to char pointed to in W reg
009A 344D     00506         retlw 'M'
009B 3469     00507         retlw 'i'
009C 3463     00508         retlw 'c'
009D 3472     00509         retlw 'r'
009E 346F     00510         retlw 'o'
009F 3463     00511         retlw 'c'
00A0 3468     00512         retlw 'h'
00A1 3469     00513         retlw 'i'
00A2 3470     00514         retlw 'p'
00A3 3420     00515         retlw ' '
00A4 3454     00516         retlw 'T'
00A5 3465     00517         retlw 'e'
00A6 3463     00518         retlw 'c'
00A7 3468     00519         retlw 'h'
00A8 346E     00520         retlw 'n'
00A9 346F     00521         retlw 'o'
00AA 346C     00522         retlw 'l'
00AB 346F     00523         retlw 'o'
00AC 3467     00524         retlw 'g'
00AD 3479     00525         retlw 'y'
00AE         00526 Table_End
00AE 3400     00527         retlw 0
00528 ;
00529         if ( (Table & 0x0FF) >= (Table_End & 0x0FF) )
00530             MESSG "Warning - User Defined: Table Table crosses page boundry in computed jump"
00531         endif
00532 ;
00533
00534
00535
00536     end
MEMORY USAGE MAP ('X' = Used, '-' = Unused)

```

```

0000 : X--XXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX- -----

```

All other memory blocks unused.

```

Program Memory Words Used: 172
Program Memory Words Free: 1876

```

Errors : 0
Warnings : 0 reported, 0 suppressed
Messages : 0 reported, 12 suppressed

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX C: 4-BIT DATA INTERFACE, HIGH NIBBLE LISTING

MPASM 01.40.01 Intermediate LM032L.ASM 4-7-1997 9:50:32 PAGE 1

LOC	OBJECT CODE	LINE	SOURCE TEXT
	VALUE		
		00001	LIST P=16C64
		00002	ERRORLEVEL -302
		00003	;
		00004	; This program interfaces to a Hitachi (LM032L) 2 line by 20 character display
		00005	; module. The program assembles for either 4-bit or 8-bit data interface, depending
		00006	; on the value of the 4bit flag. LCD_DATA is the port which supplies the data to
		00007	; the LM032L, while LCD_CNTL is the port that has the control lines (E, RS, RW).
		00008	; In 4-bit mode the data is transfer on the high nibble of the port (PORT<7:4>).
		00009	;
		00010	; Program = LM032L.ASM
		00011	; Revision Date: 5-10-94
		00012	; 1-22-97 Compatibility with MPASMWIN 1.40
		00013	;
		00014	;
		00015	include <p16c64.inc>
		00001	LIST
		00002	; P16C64.INC Standard Header File, Version 1.01 Microchip Technology, Inc.
		00238	LIST
		00016	
0000009F		00017	ADCON1 EQU 9F
		00018	
00000000		00019	FALSE EQU 0
00000001		00020	TRUE EQU 1
		00021	
		00022	include <lm032l.h>
		00069	list
		00023	;
00000000		00024	Four_bit EQU FALSE ; Selects 4- or 8-bit data transfers
00000001		00025	Data_HI EQU TRUE ; If 4-bit transfers, Hi or Low nibble of PORT
		00026	;
		00027	;
		00028	if (Four_bit && !Data_HI)
		00029	;
		00030	LCD_DATA EQU PORTB
		00031	LCD_DATA_TRIS EQU TRISB

```

00032 ;
00033     else
00034 ;
00000008 00035 LCD_DATA      EQU      PORTD
00000088 00036 LCD_DATA_TRIS EQU      TRISD
00037 ;
00038     endif
00039 ;
00000005 00040 LCD_CNTL      EQU      PORTA
00041 ;
00042 ;
00043 ;
00044 ; LCD Display Commands and Control Signal names.
00045 ;
00046     if ( Four_bit && !Data_HI )
00047 ;
00048 E            EQU      0            ; LCD Enable control line
00049 RW          EQU      1            ; LCD Read/Write control line
00050 RS          EQU      2            ; LCD Register Select control line
00051 ;
00052     else
00053 ;
00000003 00054 E            EQU      3            ; LCD Enable control line
00000002 00055 RW          EQU      2            ; LCD Read/Write control line
00000001 00056 RS          EQU      1            ; LCD Register Select control line
00057 ;
00058     endif
00059 ;
00060 ;
00000030 00061 TEMP1      EQU      0x030
00062 ;
0000      00063         org      RESET_V            ; RESET vector location
0000 2808 00064 RESET      GOTO    START            ;
00065 ;
00066 ; This is the Peripheral Interrupt routine. Should NOT get here
00067 ;
00068     page
00004      00069         org      ISR_V            ; Interrupt vector location
00004      00070 PER_INT_V
00004 1283 00071 ERROR1      BCF      STATUS, RP0        ; Bank 0
00005 1407 00072             BSF      PORTC, 0
00006 1007 00073             BCF      PORTC, 0
00007 2804 00074             GOTO    ERROR1
00075 ;
00076 ;
00077 ;
0008      00078 START            ; POWER_ON Reset (Beginning of program)

```

```

0008 0183      00079          CLRF    STATUS          ; Do initialization (Bank 0)
0009 018B      00080          CLRF    INTCON
000A 018C      00081          CLRF    PIR1
000B 1683      00082          BSF     STATUS, RP0      ; Bank 1
000C 3000      00083          MOVLW   0x00           ; The LCD module does not like to work w/ weak pull-ups
000D 0081      00084          MOVWF   OPTION_REG      ;
000E 018C      00085          CLRF    PIE1           ; Disable all peripheral interrupts
00086 ;***
00087 ;***      If using device with A/D, these two instructions are required.
00088 ;***
00089 ;          MOVLW   0xFF           ;
00090 ;          MOVWF   ADCON1         ; Port A is Digital.
00091 ;
00092 ;
000F 1283      00093          BCF     STATUS, RP0      ; Bank 0
0010 0185      00094          CLRF    PORTA          ; ALL PORT output should output Low.
0011 0186      00095          CLRF    PORTB
0012 0187      00096          CLRF    PORTC
0013 0188      00097          CLRF    PORTD
0014 0189      00098          CLRF    PORTE
0015 1010      00099          BCF     T1CON, TMR10N    ; Timer 1 is NOT incrementing
00100 ;
0016 1683      00101          BSF     STATUS, RP0      ; Select Bank 1
0017 0185      00102          CLRF    TRISA          ; RA5 - 0 outputs
0018 30F0      00103          MOVLW   0xF0           ;
0019 0086      00104          MOVWF   TRISB          ; RB7 - 4 inputs, RB3 - 0 outputs
001A 0187      00105          CLRF    TRISC          ; RC Port are outputs
001B 1407      00106          BSF     TRISC, T10S0     ; RC0 needs to be e e

```

```

00126     endif
00127 ;
00128     if ( !Four_bit )
00129         MOVLW    0x038                ; Command for 8-bit interface
00130     endif
00131 ;
00132         MOVWF    LCD_DATA            ;
00133         BSF      LCD_CNTL, E          ;
00134         BCF      LCD_CNTL, E          ;
00135 ;
00136 ; This routine takes the calculated times that the delay loop needs to
00137 ; be executed, based on the LCD_INIT_DELAY EQUate that includes the
00138 ; frequency of operation. These use registers before they are needed to
00139 ; store the time.
00140 ;
00141 LCD_DELAY    MOVLW    LCD_INIT_DELAY    ;
00142             MOVWF    MSD                ; Use MSD and LSD Registers to Initilize LCD
00143             CLRF     LSD                ;
00144 LOOP2        DECFSZ   LSD, F            ; Delay time = MSD * ((3 * 256) + 3) * Tcy
00145             GOTO     LOOP2            ;
00146             DECFSZ   MSD, F            ;
00147 END_LCD_DELAY
00148             GOTO     LOOP2            ;
00149 ;
00150 ; Command sequence for 2 lines of 5x7 characters
00151 ;
00152 CMD_SEQ
00153 ;
00154     if ( Four_bit )
00155         if ( !Data_HI )
00156             MOVLW    0x02                ; 4-bit low nibble xfer
00157         else
00158             MOVLW    0x020              ; 4-bit high nibble xfer
00159         endif
00160 ;
00161     else
00162         MOVLW    0x038                ; 8-bit mode
00163     endif
00164 ;
00165         MOVWF    LCD_DATA            ; This code for both 4-bit and 8-bit modes
00166         BSF      LCD_CNTL, E          ;
00167         BCF      LCD_CNTL, E          ;
00168 ;
00169         if ( Four_bit )                ; This code for only 4-bit mode (2nd xfer)
00170             if ( !Data_HI )
00171                 MOVLW    0x08            ; 4-bit low nibble xfer
00172             else

```

```

00173          MOVLW    0x080          ; 4-bit high nibble xfer
00174      endif
00175          MOVWF     LCD_DATA          ;
00176          BSF      LCD_CNTL, E        ;
00177          BCF      LCD_CNTL, E        ;
00178      endif
00179 ;
00180 ; Busy Flag should be valid after this point
00181 ;
0031 300C      00182          MOVLW     DISP_ON          ;
0032 2068      00183          CALL      SEND_CMD          ;
0033 3001      00184          MOVLW     CLR_DISP          ;
0034 2068      00185          CALL      SEND_CMD          ;
0035 3006      00186          MOVLW     ENTRY_INC          ;
0036 2068      00187          CALL      SEND_CMD          ;
0037 3080      00188          MOVLW     DD_RAM_ADDR          ;
0038 2068      00189          CALL      SEND_CMD          ;
00190 ;
00191      page
00192 ;
00193 ;Send a message the hard way
0039 304D      00194          movlw     'M'
003A 205F      00195          call      SEND_CHAR
003B 3069      00196          movlw     'i'
003C 205F      00197          call      SEND_CHAR
003D 3063      00198          movlw     'c'
003E 205F      00199          call      SEND_CHAR
003F 3072      00200          movlw     'r'
0040 205F      00201          call      SEND_CHAR
0041 306F      00202          movlw     'o'
0042 205F      00203          call      SEND_CHAR
0043 3063      00204          movlw     'c'
0044 205F      00205          call      SEND_CHAR
0045 3068      00206          movlw     'h'
0046 205F      00207          call      SEND_CHAR
0047 3069      00208          movlw     'i'
0048 205F      00209          call      SEND_CHAR
0049 3070      00210          movlw     'p'
004A 205F      00211          call      SEND_CHAR
00212
004B 30C0      00213          movlw     B'11000000'          ;Address DDRAM first character, second line
004C 2068      00214          call      SEND_CMD
00215
00216          ;Demonstration of the use of a table to output a message
004D 3000      00217          movlw     0          ;Table address of start of message
004E          00218      dispmsg
004E 00B0      00219          movwf     TEMP1          ;TEMP1 holds start of message address

```



```

004F 2083      00220      call    Table
0050 39FF      00221      andlw   0FFh           ;Check if at end of message (zero
0051 1903      00222      btfsc   STATUS,Z       ;returned at end)
0052 2857      00223      goto    out
0053 205F      00224      call    SEND_CHAR       ;Display character
0054 0830      00225      movf    TEMP1,w         ;Point to next character
0055 3E01      00226      addlw   1
0056 284E      00227      goto    dispmsg
0057           00228      out
0057           00229      loop
0057 2857      00230      goto    loop           ;Stay here forever
00231      ;
00232      ;
0058           00233      INIT_DISPLAY
0058 300C      00234      MOVLW   DISP_ON         ; Display On, Cursor On
0059 2068      00235      CALL     SEND_CMD        ; Send This command to the Display Module
005A 3001      00236      MOVLW   CLR_DISP        ; Clear the Display
005B 2068      00237      CALL     SEND_CMD        ; Send This command to the Display Module
005C 3006      00238      MOVLW   ENTRY_INC       ; Set Entry Mode Inc., No shift
005D 2068      00239      CALL     SEND_CMD        ; Send This command to the Display Module
005E 0008      00240      RETURN
00241      ;
00242      page
00243      ;
00244      ;*****
00245      ;* The LCD Module Subroutines *
00246      ;*****
00247      ;
00248      if ( Four_bit )      ; 4-bit Data transfers?
00249      ;
00250      if ( Data_HI )      ; 4-bit transfers on the high nibble of the PORT
00251      ;
00252      ;*****
00253      ;*SendChar - Sends character to LCD *
00254      ;*This routine splits the character into the upper and lower *
00255      ;*nibbles and sends them to the LCD, upper nibble first. *
00256      ;*****
00257      ;
00258      SEND_CHAR
00259      MOVWF   CHAR           ;Character to be sent is in W
00260      CALL    BUSY_CHECK     ;Wait for LCD to be ready
00261      MOVF    CHAR, w
00262      ANDLW   0xF0           ;Get upper nibble
00263      MOVWF   LCD_DATA       ;Send data to LCD
00264      BCF     LCD_CNTL, RW    ;Set LCD to read
00265      BSF     LCD_CNTL, RS    ;Set LCD to data mode
00266      BSF     LCD_CNTL, E     ;toggle E for LCD

```

```

00267         BCF      LCD_CNTL, E
00268         SWAPF    CHAR, W
00269         ANDLW    0xF0          ; Get lower nibble
00270         MOVWF    LCD_DATA      ; Send data to LCD
00271         BSF      LCD_CNTL, E   ; toggle E for LCD
00272         BCF      LCD_CNTL, E
00273         RETURN
00274 ;
00275         else                      ; 4-bit transfers on the low nibble of the PORT
00276 ;
00277 ;*****
00278 ;* SEND_CHAR - Sends character to LCD *
00279 ;* This routine splits the character into the upper and lower *
00280 ;* nibbles and sends them to the LCD, upper nibble first. *
00281 ;* The data is transmitted on the PORT<3:0> pins *
00282 ;*****
00283 ;
00284 SEND_CHAR
00285         MOVWF    CHAR          ; Character to be sent is in W
00286         CALL     BUSY_CHECK     ; Wait for LCD to be ready
00287         SWAPF    CHAR, W
00288         ANDLW    0x0F          ; Get upper nibble
00289         MOVWF    LCD_DATA      ; Send data to LCD
00290         BCF      LCD_CNTL, RW   ; Set LCD to read
00291         BSF      LCD_CNTL, RS   ; Set LCD to data mode
00292         BSF      LCD_CNTL, E   ; toggle E for LCD
00293         BCF      LCD_CNTL, E
00294         MOVF     CHAR, W
00295         ANDLW    0x0F          ; Get lower nibble
00296         MOVWF    LCD_DATA      ; Send data to LCD
00297         BSF      LCD_CNTL, E   ; toggle E for LCD
00298         BCF      LCD_CNTL, E
00299         RETURN
00300 ;
00301         endif
00302         else
00303 ;
00304 ;*****
00305 ;* SEND_CHAR - Sends character contained in register W to LCD *
00306 ;* This routine sends the entire character to the PORT *
00307 ;* The data is transmitted on the PORT<7:0> pins *
00308 ;*****
00309 ;
00310 SEND_CHAR
00311         MOVWF    CHAR          ; Character to be sent is in W
00312         CALL     BUSY_CHECK     ; Wait for LCD to be ready
00313         MOVF     CHAR, W

```

```

0062 0088      00314      MOVWF    LCD_DATA      ; Send data to LCD
0063 1105      00315      BCF      LCD_CNTL, RW      ; Set LCD in read mode
0064 1485      00316      BSF      LCD_CNTL, RS      ; Set LCD in data mode
0065 1585      00317      BSF      LCD_CNTL, E      ; toggle E for LCD
0066 1185      00318      BCF      LCD_CNTL, E
0067 0008      00319      RETURN
00320 ;
00321      endif
00322 ;
00323      page
00324 ;
00325 ;*****
00326 ;* SendCmd - Sends command to LCD *
00327 ;* This routine splits the command into the upper and lower *
00328 ;* nibbles and sends them to the LCD, upper nibble first. *
00329 ;* The data is transmitted on the PORT<3:0> pins *
00330 ;*****
00331 ;
00332      if ( Four_bit )      ; 4-bit Data transfers?
00333 ;
00334      if ( Data_HI )      ; 4-bit transfers on the high nibble of the PORT
00335 ;
00336 ;*****
00337 ;* SEND_CMD - Sends command to LCD *
00338 ;* This routine splits the command into the upper and lower *
00339 ;* nibbles and sends them to the LCD, upper nibble first. *
00340 ;*****
00341
00342 SEND_CMD
00343      MOVWF    CHAR      ; Character to be sent is in W
00344      CALL     BUSY_CHECK ; Wait for LCD to be ready
00345      MOVF     CHAR,w
00346      ANDLW    0xF0      ; Get upper nibble
00347      MOVWF    LCD_DATA ; Send data to LCD
00348      BCF      LCD_CNTL,RW ; Set LCD to read
00349      BCF      LCD_CNTL,RS ; Set LCD to command mode
00350      BSF      LCD_CNTL,E ; toggle E for LCD
00351      BCF      LCD_CNTL,E
00352      SWAPF    CHAR,w
00353      ANDLW    0xF0      ; Get lower nibble
00354      MOVWF    LCD_DATA ; Send data to LCD
00355      BSF      LCD_CNTL,E ; toggle E for LCD
00356      BCF      LCD_CNTL,E
00357      RETURN
00358 ;
00359      else      ; 4-bit transfers on the low nibble of the PORT
00360 ;

```

```

00361 SEND_CMD
00362     MOVWF    CHAR           ; Character to be sent is in W
00363     CALL     BUSY_CHECK     ; Wait for LCD to be ready
00364     SWAPF    CHAR, W
00365     ANDLW    0x0F           ; Get upper nibble
00366     MOVWF    LCD_DATA       ; Send data to LCD
00367     BCF      LCD_CNTL, RW    ; Set LCD to read
00368     BCF      LCD_CNTL, RS    ; Set LCD to command mode
00369     BSF      LCD_CNTL, E     ; toggle E for LCD
00370     BCF      LCD_CNTL, E
00371     MOVF     CHAR, W
00372     ANDLW    0x0F           ; Get lower nibble
00373     MOVWF    LCD_DATA       ; Send data to LCD
00374     BSF      LCD_CNTL, E     ; toggle E for LCD
00375     BCF      LCD_CNTL, E
00376     RETURN
00377 ;
00378     endif
00379     else
00380 ;
00381 ;*****
00382 ;* SEND_CND - Sends command contained in register W to LCD      *
00383 ;* This routine sends the entire character to the PORT          *
00384 ;* The data is transmitted on the PORT<7:0> pins                *
00385 ;*****
00386
0068 00387 SEND_CMD
0068 00B6 00388     MOVWF    CHAR           ; Command to be sent is in W
0069 2071 00389     CALL     BUSY_CHECK     ; Wait for LCD to be ready
006A 0836 00390     MOVF     CHAR, W
006B 0088 00391     MOVWF    LCD_DATA       ; Send data to LCD
006C 1105 00392     BCF      LCD_CNTL, RW    ; Set LCD in read mode
006D 1085 00393     BCF      LCD_CNTL, RS    ; Set LCD in command mode
006E 1585 00394     BSF      LCD_CNTL, E     ; toggle E for LCD
006F 1185 00395     BCF      LCD_CNTL, E
0070 0008 00396     RETURN
00397 ;
00398     endif
00399 ;
00400     page
00401 ;
00402     if ( Four_bit )           ; 4-bit Data transfers?
00403 ;
00404         if ( Data_HI )       ; 4-bit transfers on the high nibble of the PORT
00405 ;
00406 ;*****
00407 ;* This routine checks the busy flag, returns when not busy      *

```

```

00408 ;* Affects:
00409 ;* TEMP - Returned with busy/address
00410 ;*****
00411 ;
00412 BUSY_CHECK
00413     BSF     STATUS, RP0      ; Select Register Bank1
00414     MOVLW   0xFF            ; Set Port_D for input
00415     MOVWF   LCD_DATA_TRIS
00416     BCF     STATUS, RP0      ; Select Register Bank0
00417     BCF     LCD_CNTL, RS     ; Set LCD for Command mode
00418     BSF     LCD_CNTL, RW     ; Setup to read busy flag
00419     BSF     LCD_CNTL, E      ; Set E high
00420     BCF     LCD_CNTL, E      ; Set E low
00421     MOVF    LCD_DATA, W      ; Read upper nibble busy flag, DDRam address
00422     ANDLW   0xF0            ; Mask out lower nibble
00423     MOVWF   TEMP
00424     BSF     LCD_CNTL, E      ; Toggle E to get lower nibble
00425     BCF     LCD_CNTL, E
00426     SWAPF   LCD_DATA, W      ; Read lower nibble busy flag, DDRam address
00427     ANDLW   0x0F            ; Mask out upper nibble
00428     IORWF   TEMP, F          ; Combine nibbles
00429     BTFSC   TEMP, 7          ; Check busy flag, high = busy
00430     GOTO    BUSY_CHECK      ; If busy, check again
00431     BCF     LCD_CNTL, RW
00432     BSF     STATUS, RP0      ; Select Register Bank1
00433     MOVLW   0x0F
00434     MOVWF   LCD_DATA_TRIS    ; Set Port_D for output
00435     BCF     STATUS, RP0      ; Select Register Bank0
00436     RETURN
00437 ;
00438     else                    ; 4-bit transfers on the low nibble of the PORT
00439 ;
00440 ;*****
00441 ;* This routine checks the busy flag, returns when not busy
00442 ;* Affects:
00443 ;* TEMP - Returned with busy/address
00444 ;*****
00445 ;
00446 BUSY_CHECK
00447     BSF     STATUS, RP0      ; Bank 1
00448     MOVLW   0xFF            ; Set PortB for input
00449     MOVWF   LCD_DATA_TRIS
00450     BCF     STATUS, RP0      ; Bank 0
00451     BCF     LCD_CNTL, RS     ; Set LCD for Command mode
00452     BSF     LCD_CNTL, RW     ; Setup to read busy flag
00453     BSF     LCD_CNTL, E      ; Set E high
00454     BCF     LCD_CNTL, E      ; Set E low

```



```

00502     page
00503 ;
0083     00504 Table
0083 0782     00505     addwf    PCL, F           ; Jump to char pointed to in W reg
0084 344D     00506     retlw    'M'
0085 3469     00507     retlw    'i'
0086 3463     00508     retlw    'c'
0087 3472     00509     retlw    'r'
0088 346F     00510     retlw    'o'
0089 3463     00511     retlw    'c'
008A 3468     00512     retlw    'h'
008B 3469     00513     retlw    'i'
008C 3470     00514     retlw    'p'
008D 3420     00515     retlw    ' '
008E 3454     00516     retlw    'T'
008F 3465     00517     retlw    'e'
0090 3463     00518     retlw    'c'
0091 3468     00519     retlw    'h'
0092 346E     00520     retlw    'n'
0093 346F     00521     retlw    'o'
0094 346C     00522     retlw    'l'
0095 346F     00523     retlw    'o'
0096 3467     00524     retlw    'g'
0097 3479     00525     retlw    'y'
0098     00526 Table_End
0098 3400     00527     retlw    0
00528 ;
00529     if ( (Table & 0x0FF) >= (Table_End & 0x0FF) )
00530         MESSG    "Warning - User Defined: Table Table crosses page boundry in computed jump"
00531     endif
00532 ;
00533
00534
00535
00536     end

```

MEMORY USAGE MAP ('X' = Used, '-' = Unused)

```
0000 : X--XXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXX XXXXXXXX----- -----
```

All other memory blocks unused.

Program Memory Words Used: 150
Program Memory Words Free: 1898

Errors : 0
Warnings : 0 reported, 0 suppressed
Messages : 0 reported, 12 suppressed

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.microchip.com>

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: 630-285-0071 Fax: 630-285-0075

Dallas

Microchip Technology Inc.
14651 Dallas Parkway, Suite 816
Dallas, TX 75240-8809
Tel: 972-991-7177 Fax: 972-991-8588

Dayton

Microchip Technology Inc.
Two Prestige Place, Suite 150
Miami, OH 45342
Tel: 937-291-1654 Fax: 937-291-9175

Los Angeles

Microchip Technology Inc.
18201 Von Karman, Suite 1090
Irvine, CA 92612
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

San Jose

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

Toronto

Microchip Technology Inc.
5925 Airport Road, Suite 200
Mississauga, Ontario L4V 1W1, Canada
Tel: 905-405-6279 Fax: 905-405-6253

ASIA/PACIFIC

Hong Kong

Microchip Asia Pacific
RM 3801B, Tower Two
Metroplaza
223 Hing Fong Road
Kwai Fong, N. T., Hong Kong
Tel: 852-2-401-1200 Fax: 852-2-401-3431

India

Microchip Technology India
No. 6, Legacy, Convent Road
Bangalore 560 025, India
Tel: 91-80-229-0061 Fax: 91-80-229-0062

Korea

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

Shanghai

Microchip Technology
RM 406 Shanghai Golden Bridge Bldg.
2077 Yan'an Road West, Hongqiao District
Shanghai, PRC 200335
Tel: 86-21-6275-5700
Fax: 86-21-6275-5060

Singapore

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

Taiwan, R.O.C

Microchip Technology Taiwan
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: 44-1628-851077 Fax: 44-1628-850259

France

Arizona Microchip Technology SARL
Zone Industrielle de la Bonde
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 München, Germany
Tel: 49-89-627-1440 Fax: 49-89-627-144-44

Italy

Arizona Microchip Technology SRL
Centro Direzionale Colleone
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-39-6899939 Fax: 39-39-6899883

JAPAN

Microchip Technology International Inc.
Benex S-1 6F
3-18-20, Shin Yokohama
Kohoku-Ku, Yokohama
Kanagawa 222 Japan
Tel: 81-4-5471-6166 Fax: 81-4-5471-6122

5/8/97



MICROCHIP

All rights reserved. © 1997, Microchip Technology Incorporated, USA. 6/97

Information contained in this publication regarding device applications and the like is intended for suggestion only and is not intended to be construed as a warranty, expressed or implied. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of the information contained herein. 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. in the U.S.A. and other countries. All other trademarks mentioned herein are the property of their respective companies.