Interfacing a HD44780 Character-LCD to a MicroConverter

1.0 Introduction

There are a vast array of LCD displays available. Fortunately a vast array of these comply with the HD44780U standard. This standard referrers to the LCD controller chip, which accepts data from the MicroConverter and communicates with the LCD screen. HD44780 standard LCD screens are available in numerous formats, the most popular of which are the 16x2 and 20x2 formats. In this technical the various commands to control the basic functions of the LCD are illustrated.

2.0 Interfacing a HD44780 LCD

The databus that connects the HD44780 to the MicroConveter can be 8 or 4 bits wide, in this technical note we only examine the case of an 8 bit data bus. As well as the databus 3 control lines are required, this means that for our examples we will need to use a total of 11 pins to interface the LCD to the MicroConverter.

The 8 Data lines which form the data bus are referred to as DB0, DB1, ….. DB7

The 3 Control lines are referred to as EN, RS and RW. Their functions are described below.

EN is the *Enable Line*, This line is used to indicate the start of a transmission of a data byte to the LCD controller. To indicate the start of transmission this line is brought high, when transmission is complete the EN line is brought low indicating that transmission is complete.

RS is the *Register Select Line*, This line indicates to the LCD controller whether the data byte is to be treated as a command or as text data to be displayed on the screen. If the RS line is high the data byte is treated as text to be displayed, if the RS line is low the data byte is treated as a command.

RW is the *Read/Write Line*, When this line is low the information on the data bus is written to the LCD controller. If this line is high the LCD controller can be read, this is used to check the status of the LCD.

In this example we will connect the 8 data lines to port 3 of the MicroConverter and the 3 control lines to Port 2.6, 2.7 and 3.6. This is illustrated in figure 1.

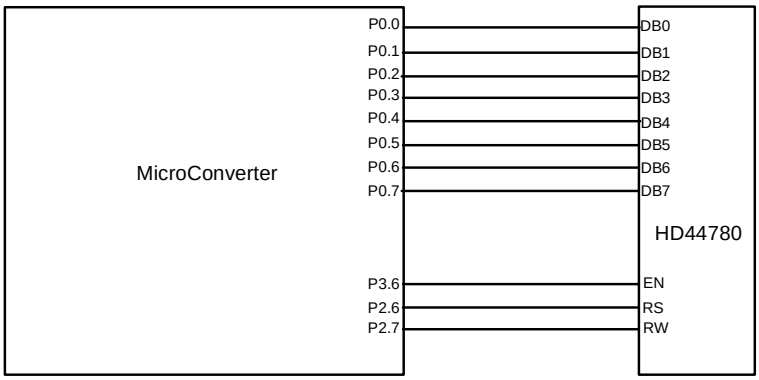

Figure 1. Connecting a HD44780 LCD to a MicroConverter.

In our assembly code this interface is defined using the following:

```
LCD_RW      EQU   P2.7
LCD_RS      EQU   P2.6
LCD_EN      EQU   P3.6
LCD_DATA    EQU   P0
CHAR        DATA  040H          ; Char buffer for LCD
```

3.0 Configuring the LCD Screen

In order to display text to the LCD screen it must be first configured. In the configuration the LCD controller is told what sort of LCD screen we are using, as well as the data bus format and the font that we want to use.

The various commands available are listed in detail in the HD44780 datasheet.

In this example we will configure the LCD to use a 8 bit data bus and to display in 5x10 dot character font. This configuration is achieved by sending 38H to the LCD controller.

To do this we us we use three functions, `SEND_CMD`, `CHKBUSY`, and `PULSE_E`

```
SEND_CMD:   MOV   CHAR,R0             ; SAVES R0 TO LOCATION CHAR
            CALL  CHKBUSY             ; CHECKS BUSY FLAG
            MOV   R0,CHAR             ; PULLS CONTENTS OF R0 FROM CHAR
            MOV   LCD_DATA,R0         ; SENDS TO PORT 3
            CLR   LCD_WR              ; CLEARS R/W LINE
            CLR   LCD_RS              ; CLEARS RS LINE
            CALL  PULSE_E             ; CLOCKS IN DATA INTO LCD
            RET                       ; RETURNS

CHKBUSY:    MOV   R0,#0FFH
            MOV   LCD_DATA,R0  ; SETS UP PORT2 AS I/P'S
            CLR   LCD_RS       ; CLEARS RS LINE
            SETB  LCD_WR       ; SETS R/W LINE
            NOP
            NOP
            NOP
BUSYREAD:   SETB  LCD_EN       ; CLOCKS E HIGH
            NOP
            NOP
            NOP
            MOV   ACC,LCD_DATA ; READS PORT 2
            NOP
            NOP
            NOP
            CLR   LCD_EN       ; CLOCKS E LOW
            NOP
            NOP
            JB    ACC.7,BUSYREAD      ; IF BUSY, LOOP
            RET

PULSE_E:    NOP
            NOP
```

```
            SETB   LCD_EN       ; CLOCKS E HIGH
            NOP
            NOP
            NOP
            NOP
            NOP
            CLR    LCD_EN       ; CLOCKS E LOW
            NOP
            NOP
            RET
```

SEND_CMD is a function which sends the command byte that is contained in R0 to the MicroConverter.
CHKBUSY is a function which checks the busy status of the LCD controller and waits for the instruction to be fully executed.
PULSE_E is used to clock in the data to the LCD controller.

Thus our code to initialise the LCD display is as follows,

```
; Display initialisation
MOV    R0,#038H            ; Function set
CALL   SEND_CMD            ;
MOV    R0,#06H             ; Entry set
CALL   SEND_CMD            ;
MOV    R0,#02H             ; Home
CALL   SEND_CMD            ;
MOV    R0,#0CH             ; Display on, cursor, etc
CALL   SEND_CMD
```

The additional commands are used to position the cursor to the start of the first row of the LCD. Now that the LCD is initialised we can write text to the screen.

4.0 Writing Text to the LCD Display

In the sample program we are going to write the following text to the LCD screen.
"ANALOG DEVICES
 LCD EXAMPLE CODE"

To do this we need to define this string in code memory,

```
TEXT0:      DB    ' ANALOG DEVICES '      ; welcome message
            DB    099h
            DB    'LCD EXAMPLE CODE '
            DB    099h
```

The 099h is used by the code to decide to print the rest of the string on the next line.

We also need three more functions in order to print the text. These are shown below

```
BOTHLINES:    PUSH   PSW                ;=>Save Carry
              ACALL  LINE1
              INC    DPTR
              ACALL  LINE2
FIN:          POP    PSW
```

```
            RET

LINE1: MOV    R0,#80h             ; CURSOR ADDRESS=
            CALL   SEND_CMD             ; beginning of first line
            CALL   WRITE                ; Write the full line
            RET

LINE2: MOV    R0,#0C0h                  ;set CURSOR ADDRESS
            CALL   SEND_CMD
            CALL   WRITE                ;Write the full line
            RET

WRITE:
write_loop:   MOV    A,R1
            MOVC   A,@A+DPTR
            CJNE   A,#99h,write_cont    ; stop writing condition
            RET

write_cont:   MOV    R0,A
            CALL   send_char            ; send one ASCII character
            INC    DPTR                 ; next character
            JMP    write_loop

SEND_CHAR:    MOV    CHAR,R0              ; SAVES R0 TO LOCATION CHAR
            CALL   CHKBUSY              ; CHECKS BUSY FLAG
            MOV    R0,CHAR              ; PULLS CONTENTS OF R0 FROM CHAR
            MOV    LCD_DATA,R0          ; SENDS TO PORT 2
            CLR    LCD_WR               ; CLEARS R/W LINE
            SETB   LCD_RS               ; SETS RS LINE HIGH FOR DATA
            CALL   PULSE_E              ; CLOCKS DATA INTO LCD
            RET
```

`BOTHLINES` is a function which prints the text across the two lines.
`LINE1 & LINE2` are functions which move the cursor position.
`WRITE and SEND_CHAR` are functions which writes the text on the lcd screen one character at a time.

Thus our code to write the text on the LCD screen is as follows,

```
MOV    DPTR,#TEXT0   ; Welcome message stored in code memory
LCALL  BOTHLINES
```

5.0 Sample programs

Accompanying this technote you will also find assembly code containing the LCD functions that we have used previously in this program. There is also a sample file containing similar functions in c.

Two sample programs are available which utilise the LCD interface described in this technical note.

The first program is a LCD temperature monitor. This program utilises the on chip temperature sensor of the ADuC834/ADuC824/ADuC816 and displays the die temperature on the LCD screen. A program flow diagram for this is shown in figure 2.
The second program that is available implements a frequency measurement using the Timer 2 input pin on the ADuC834/ ADuC824/ADuC816, though the code can easily be ported to the other MicroConverter products. This program is described in greater detail in technote uC013.

Temperature Monitor Routine

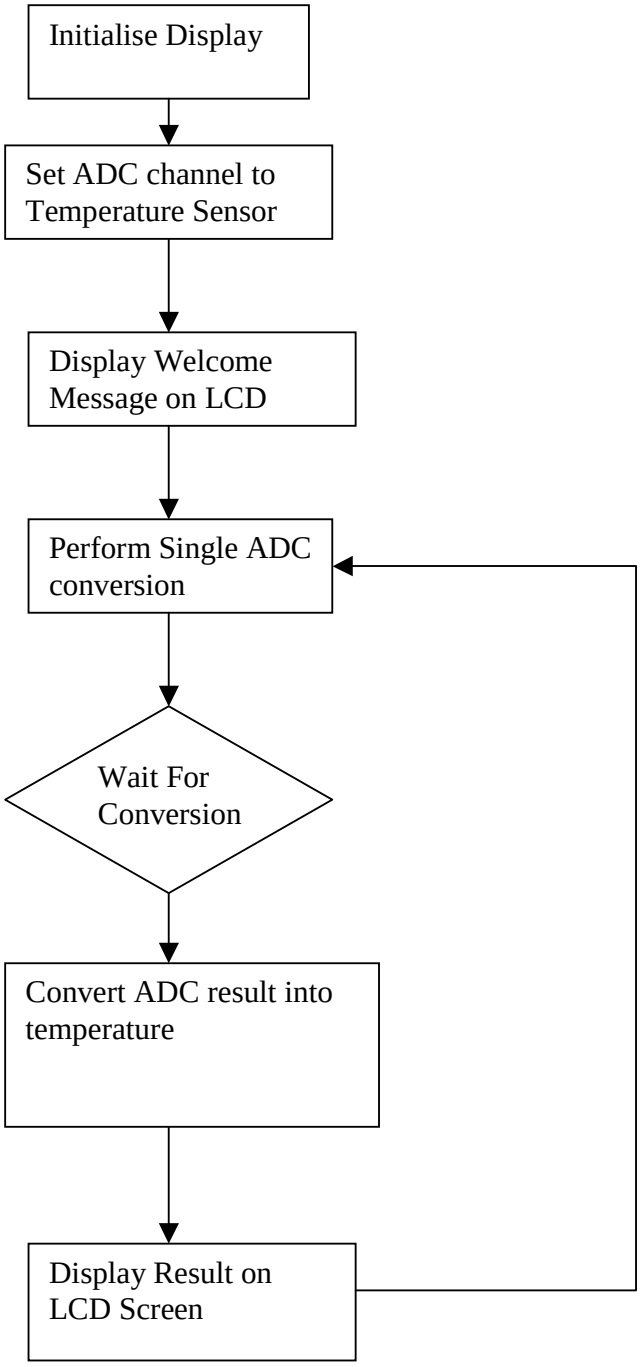

Figure 2. Flow Diagram for Temperature Monitor Routine.