1.0 INTRODUCTION

LCD modules have become a popular way to display system messages and status in embedded applications. This application note shows how to interface an SST FlashFlex® microcontroller1 to a typical character LCD module. The SST FlashFlex is an industry-standard, 8051-compatible MCU and thus gives full control of its I/O ports allowing for a variety of hardware/firmware implementations of the LCD interface. For simplicity, our sample implementation uses ports P1 and P3 of the MCU as “bit-banged” I/O. However it is also possible to implement the interface with a memory-mapped I/O scheme, using ports P0 and P2 as a data and address bus.

2.0 HARDWARE SCHEMATICS

The hardware schematic in Figure 2-1 is applicable to most character LCD modules because they conform to the same industry-standard pin assignments and instruction set. The signals to the LCD module consists of an 8-bit data bus, power signals, three control signals, and optionally two pins for a backlight.

![FIGURE 2-1: SST MCU Interfacing to Character LCD Module](image)

In this implementation, the data bus of the LCD module is connected to port P1 and the three control signals (RS, R/W, and EN) are connected to port pins P3.3, P3.4, and P3.5 respectively. Trimpot R2 can be used to adjust the contrast of the LCD display, but it can be omitted and simply shorted to ground for maximum contrast.

For LCD modules that have a backlight LED, a current limiting resistor is mandatory. The resistor R1 value can be calculated as follows:

\[ R1 = \frac{(V_{DD} - V_{LED})}{I_{LED}} \]

The \( V_{LED} \) (or \( V_F \)) and \( I_{LED} \) (or \( I_F \)) can be determined from the LCD module data sheet². Typically, the value of \( V_{LED} \) is 4.2V and \( I_{LED} \) varies from 120mA to 160mA. For example, when \( I_{LED} = 120 \text{ mA} \), the resistor R1 would be 6.8 Ω.

3.0 FIRMWARE

The sample program provided in Appendix A was written in Keil 8051 assembler format and will display a series of messages with a 5 second delay spacing. The main program utilizes subroutines to initialize the LCD module, load the character table and display text. These subroutines, in turn, make use of lower level functions to write a command, write data, read data, read ram and busy wait. The algorithms for each of these subroutines are shown on Figures 3-1 to 3-4.

Although the LCD module has built-in power-on reset circuitry, software initialization is strongly recommended to assure reliable operation of the module. Without software initialization, if (1) the rise time of the power supply is out of range from 0.1 ms to 10 ms when V\textsubscript{DD} rises from 0.2V to 4.5V at power-up, or (2) power-off time is less than 1 ms, the module may not behave properly. When initializing the LCD module, the firmware cannot check the busy flag until the FunctionSet command is executed three times with appropriate delays.

Another consideration in the firmware is that the relation between DDRAM address allocations and character display position can vary depending on the size of the display. For example, on a 16 character x 2 line display, the LCD module will not display anything in the address range of 10H to 3FH as shown in the table below. Review the data sheet of the specific LCD module for information of display mapping.

<table>
<thead>
<tr>
<th>(Left)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line1</td>
<td>00H</td>
<td>01H</td>
<td>02H</td>
<td>03H</td>
<td>04H</td>
<td>05H</td>
<td>06H</td>
<td>07H</td>
<td>08H</td>
<td>09H</td>
<td>0AH</td>
<td>0BH</td>
<td>0CH</td>
<td>0DH</td>
<td>0EH</td>
<td>0FH</td>
</tr>
<tr>
<td>Line2</td>
<td>40H</td>
<td>41H</td>
<td>42H</td>
<td>43H</td>
<td>44H</td>
<td>45H</td>
<td>46H</td>
<td>47H</td>
<td>48H</td>
<td>49H</td>
<td>4AH</td>
<td>4BH</td>
<td>4CH</td>
<td>4DH</td>
<td>4EH</td>
<td>4FH</td>
</tr>
</tbody>
</table>

Lastly, it should be noted that some LCD modules allow for the display of user defined characters that are stored as bit-map data into CGRAM. In the sample program, the BuildChar subroutine demonstrates how this data is loaded into CGRAM. For details on constructing the user defined characters, please refer to the data sheet for the LCD module\textsuperscript{2}.
FIGURE 3-1: Initialization and Main Program Flowcharts

Start Initialize LCD Module

- Wait for longer than 15 ms
- #1 Write FunctionSet command
  - Wait for longer than 4.1 ms
  - #2 Write FunctionSet command
    - Wait for longer than 100 µs
    - #3 Write FunctionSet command
    - #4 Write FunctionSet command
      - Write Display Off command
  - Write Display Control command

**Initialization is done**

Main Program

Lcall BuildChar subroutine to create custom-made chars

- Reset CGRAM address
  - Read CGRAM data, Compare with original data
    - Any mismatch?
      - No
      - Write DDRAM address cmd to access DDRAM
        - Lcall string subroutine to display message1
        - Write Shift command
        - Wait 5 seconds to display
        - Write ReturnHome cmd
          - Cursor goes to address 0
        - Write ReturnHome cmd
          - Cursor goes to address 2
        - Wait 5 seconds to display
        - Lcall string subroutine to display message 3
        - Wait 5 seconds to display
        - Restart Again

**Busy Flag can be checked from now on**

LCD module Internal reset sequences

P1 = 00h indicate error

Any mismatch?

YES

- P1 = 00h indicate error

NO

- Write Display Off command
- Write Clear Display command
- Write Entry Mode command
- Write Display Control command

Initialization is done

**2035 INIT/MAIN 1.1**
FIGURE 3-2: BuildChar and String Subroutine Flowcharts
FIGURE 3-3: WriteCommand and WriteData Subroutine Flowcharts
FIGURE 3-4: Read, Busy, and ReadRam Subroutine Flowcharts

Read Subroutine

LCDEN=0
LCDRS=0
LCDRW=0
P1=0FFh
LCDEN=1

LCRDW=1
NOP
Read P1 into ACC
LCDEN=0
LCDRW=0

Return

Busy Subroutine

LCALL Read subroutine

ACC.7=1?
Yes

LCALL Read subroutine

ACC.7=1?
No

Return

ReadRam Subroutine

LCALL Busy subroutine

LCDEN=0
LCDRS=0
LCDRW=0
P1=0FFh
LCDEN=1
LCDRW=1
LCDEN=1
NOP
Read P1 into ACC
LCDEN=0
LCDRS=0
LCDRW=0

Return
APPENDIX A. SAMPLE PROGRAM

; This project is LCD interfaces to standard 8051 MCU such as SST89E516RDx, SST89E58RDx.
; assume that working frequency is 12MHz, 1 machine cycle=1us.

;---------------------------------------------------------------
LCDRS bit P3.3 ; 0 for instruction, 1 for data
LCDRW bit P3.4 ; 0 for write, 1 for read
LCDEN bit P3.5 ; falling edge to latch commands/data
; P1.0 to P1.7 connect to LCD data bus DB0 to DB7

ClearDisplay EQU 01h
ReturnHome EQU 02h
EntryMode EQU 06h ; I/D=1(increment), SH=0(no shift)
DisplayControl EQU 0Fh ; D=1(Display ON), C=1(Cursor ON), B=1(Blinking)
Shift EQU 14h ; S/C=0(Cursor Move), R/L=1(Right)
FunctionSet EQU 38h ; 8 bit, 2 line, 5x8 dot.
CGRAMaddress EQU 40h ; DB6 must be 1, valid address is AC5...AC0
DDRAMaddress EQU 80h ; DB7 must be 1, valid address is AC6...AC0

org 0000h
ljmp LCDinit

org 0050h

;---------------------------------------------------------------
LCDinit: mov r5, #15
lcall delay ; wait >15ms
anl P3, #11100111b ; RS=0, RW=0
mov P1, #FunctionSet ; #1 Function Set
lcall wrtcmd ; BF(Busy Flag) can not be checked here
mov r5, #5
lcall delay ; wait >4.1ms
mov P1, #FunctionSet ; #2 Function Set
lcall wrtcmd ; BF(Busy Flag) can not be checked here
mov r6, #50
djnz r6, $ ; wait >100us
mov P1, #FunctionSet ; #3 Function Set
lcall wrtcmd ; BF can not be checked BEFORE this instruction
;---------------------------------------------------------------

; BF can be checked from now on.
; If no BF check, waiting time must be longer than instruction’s execution time.
;---------------------------------------------------------------

mov P1, #FunctionSet ; #4 Function Set,
lcall writecommand
mov P1, #0000$1000b ; display off
lcall writecommand
mov P1, #ClearDisplay
lcall writecommand ; clear display
mov P1, #EntryMode
lcall writecommand ; entry mode: inc, no shift
mov P1, #DisplayControl
lcall writecommand ; turn on display, cursor and blinking
mov dptr, #customchar
lcall buildchar

mov P1, #CGRAMaddress ; Set CGRAM address=0
lcall writecommand

mov dptr, #customchar ; point to start of customchar
clr f0 ; flag of mismatch

compare:
cir a
movc a, @a+dptr
inc dptr
cjne a, #80h, more
sjmp complete

more:
mov r6, a ; save original data
lcall busy
lcall readram
xrl a, r6
jz compare
setb f0
sjmp compare

complete:
jnb f0, good
here:
mov P1, #0
sjmp here

good:
mov P1, #DDRAMaddress ; switch to DDRAM address
lcall writecommand

mov dptr, #message1
lcall string

mov P1, #Shift ; Shift control
lcall writecommand ; cursor is not visible at this time
lcall wait5s
mov P1, #ReturnHome
lcall writecommand

mov dptr, #message2
lcall string
lcall wait5s

mov dptr, #message3
lcall string
lcall wait5s
ljmp LCDinit

;=======================================================================
buildchar: mov P1, #CGRAMaddress ; create 8 custom-made characters
lcall writecommand ; defined in label customchar:

lcall wait5s
mov P1, #ReturnHome
lcall writecommand

mov dptr, #message2
lcall string
lcall wait5s

mov dptr, #message3
lcall string
lcall wait5s

ljmp LCDinit

;=======================================================================
dotmatrix: cir a
movc a, @a+dptr
inc dptr
cjne a, #60h, continue
ret

continue: mov P1, a
lcall writedata
sjmp dotmatrix
display one message string

string:
clr a
movc a, @a+dptr
inc dptr
setb acc.7 ; ensure DB7=1 when set DDRAM address
mov P1, a
lcall writecommand ; set starting address
text:
clr a
movc a, @a+dptr
inc dptr
cjne a, #80h, display ; 80h as terminator of text
ret
display:
mov P1, a
lcall writedata
cursor:
lcall busy
cjne a, #10h, text ; reach end of 1st line, continue at 2nd line
mov P1, #0C0h ; reach end of 1st line, continue at 2nd line
lcall wrtcmd
sjmp text
message1: DB 00h, "SST FlashFlex 8052 Compatible", 80h
message2: DB 00h, "This LCD module is 16 x 2 lines", 80h
message3: DB 00h, "user custom-made symbols:\n00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h, 08h",
customchar: DB 0Ch, 12h, 04h, 08h, 1Eh, 00h, 00h, 00h, 00h; first char: superscript 2
DB 16h, 09h, 08h, 08h, 08h, 09h, 06h, 00h; second char: °C
DB 17h, 04h, 04h, 07h, 04h, 04h, 04h, 00h; third char: °F
DB 00h, 00h, 00h, 01h, 00h, 00h, 00h, 00h; fourth char: ellipsis ...
DB 04h, 0Eh, 04h, 04h, 04h, 04h, 04h, 00h; fifth char: up arrow ↑
DB 04h, 04h, 04h, 04h, 15h, 0Eh, 04h, 00h; sixth char: down arrow ↓
DB 04h, 04h, 1Fh, 04h, 04h, 00h, 1Fh, 00h; seventh char: plus minus ±
DB 00h, 01h, 05h, 09h, 1Fh, 08h, 08h, 00h; eighth char: return ↵
DB 80h
SUBROUTINES
; Write one command into LCD
writecommand:
mov r5, P1 ; save original command
lcall busy
mov P1, r5 ; restore command byte
wrtcmd:
clr LCDRS ; LCDRS=0 for command
clr LCDRW ; LCDRW=0 for write
clr LCDEN
setb LCDEN
clr LCDEN ; falling edge latch data into LCD
ret
Application Note

FlashFlex Microcontroller
Control of Character LCD Module

;=======================================================================
; Write one data byte into LCD
;=======================================================================
writeData:
  mov r6, P1 ; save original data
  lcall busy
  mov P1, r6 ; restore data
  wrtData:
  setb LCDRS ; LCDRS=1 for data
  clr LCDRW ; LCDRW=0 for write
  clr LCDEN
  setb LCDEN
  clr LCDEN ; falling edge latch data into LCD
  ret

;=======================================================================
; Read one data byte from CGRAM
;=======================================================================
readCGRAM:
  lcall busy
  clr LCDEN
  clr LCDRS
  clr LCDRW
  mov P1, #0FFh ; ready for input
  setb LCDRS
  setb LCDRW
  setb LCDEN
  nop
  mov a, P1 ; "a" returns read data
  clr LCDEN ; end of this operation
  clr LCDRW
  ret

;=======================================================================
; Read one data byte from DDRAM
;=======================================================================
read:
  clr LCDEN
  clr LCDRS ; LCDRS=0 for command
  clr LCDRW
  mov P1, #0FFh ; set port P1 as input port
  setb LCDRW
  setb LCDEN
  nop
  mov a, P1 ; "a" contains read data
  clr LCDEN ; end of this operation
  clr LCDRW
  ret

;=======================================================================
; Check the status - wait until not busy
;=======================================================================
busy:
  lcall read
  jb acc.7, busy ; check one more time,
  lcall read
  jb acc.7, busy ; confirm BUSY=0 before exit
  ret

;=======================================================================
; Wait 5 seconds
;=======================================================================
wait5s:
  mov r4, #50 ; 5s
  loadr5:
  mov r5, #200 ; 0.1s
  loadr6:
  mov r6, #250 ; 0.5ms
djnz r6, $
djnz r5, loadr6
djnz r4, loadr5
ret
; Delay "r5" milliseconds
;=======================================================================
delay:    lcall dly1ms
         djnz r5, delay
         ret
;=======================================================================
; Delay 250x4us=1ms for 12MHz crystal
;=======================================================================
dly1ms:   mov    r6, #250
loop:     nop    ; 1µs
         nop    ; 1µs
         djnz   r6, loop ; 2µs
         ret
;=======================================================================
END