Spring 2022

Haroutun Haroutunian
Andres Lopez

California State University Northridge
Department of Electrical and Computer Engineering

Experiment 8
LCD Display

Professor Flynn
15 April, 2022

Introduction

The objective of this lab is to control the LCD display using several subroutines. The LCD(Liquid Crystal Display) is a very important peripheral device used in alot of consumer electronics.Along microprocessors, the LCD provides a practical method for the user/programmer to to verify operations and display strings or characters.

Procedure

A similar process that was previously developed for lighting the LEDs is used for initialization of the pins for writing data to the LCD display. PINSEL1 and PINSEL2 are introduced as new control registers that are used for configuring the pins that connect from the EduBoard to the LCD display as shown in the schematic below.

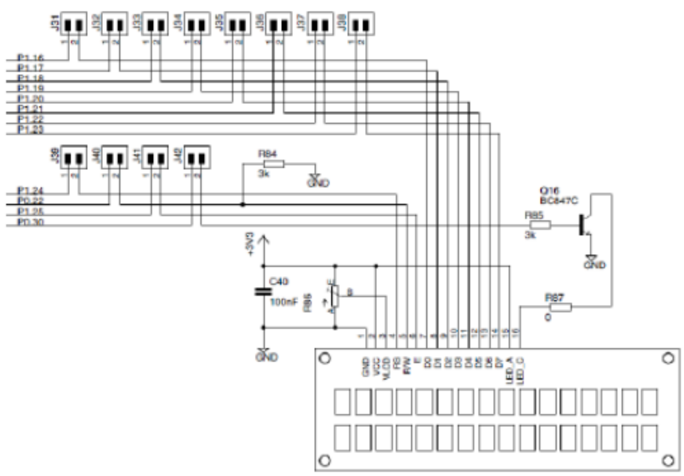
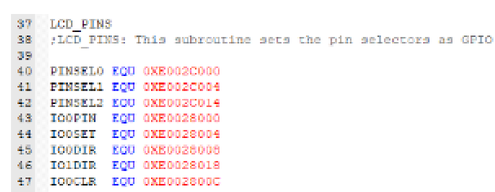


Figure 9.1: LPC 2148 Education Board Schematic: LCD

Task 1 requires us to write subroutines for the LCD display and group them in one file named lcd_subs.s. A series of subroutines must be developed in order to initialize, turn on, and write to the LCD display. These subroutines can override the data currently held in registers, so it's imperative to save the data of specific registers and program counters used in the subroutines. At the end of a subroutine, the saved registers and PC must be loaded back into memory so the processor can direct the algorithm to the next instruction. For the LEDs on the Eduboard, only port 0 was configured as GPIO; however, for this lab, the LCD requires port 1, as well as port 0 where each port controls 16 pins, where each pin is represented by 2 bits which totals 32 bits.



```
LCD_PINS
;LCD_PINS: This subroutine sets the pin selectors as GPIO

PINSEL0 EQU 0XE002C000
PINSEL1 EQU 0XE002C004
PINSEL2 EQU 0XE002C014
IO0PIN  EQU 0XE0028000
IO0SET  EQU 0XE0028004
IO0DIR  EQU 0XE0028008
IO1DIR  EQU 0XE0028018
IO0CLR  EQU 0XE002800C
```

Figure 9.2: LCD_pins EQU's

First, students use LCD mnemonics in order to simplify the program flow. LCD_DATA controls pins 16-23, which are the data pins D0:D7 that contains the data to be sent to the LCD as mentioned in the manual. LCD_E, refers to the enable pin, which initiates the flow of data into the LCDs processor.When the microcontroller changes states from HIGH/LOW and vice versa, it will take data as input. LCD_RW is the read/write mnemonic that controls whether or not data is to be read from the LCD, or data is to be written to the LCD. LCD_RS is the register select line, which controls the type of data sent to the LCD. If the LCD_RS line is HIGH, text data is to be sent to the LCD, and if it is LOW, the data sent to the LCD is a command. Using these mentioned commands of the LCD, students can now initialize and perform subroutines.

```
LCD_CMD
;;-This subroutine sends a command to the LCD.

LCD_DATA    EQU 0X00FF0000 ;P1.16-23
LCD_RS      EQU 0X01000000 ;P1.24
LCD_E       EQU 0X02000000 ;P1.25
LCD_RW      EQU 0X00400000 ;P0.22
LCD_LIGHT   EQU 0X40000000 ;P0.30
IO1PIN      EQU 0XE0028010 ;ALLOWS US TO OUTPUT 0/1 ON PORT 1 PINS
```

Figure 9.3: LCD_CMD EQU's

<u>Below are all the subroutines used to complete assigned tasks.</u>

LCD_PINS

- *Sets the pins as GPIO*

LCD_CMD

- Sends data/commands to the LCD
- D[7:0] are command lines wired to pins 16-31
- Set RS = RW = E = 0
    - Delays for 6ns using a separate delay subroutine
- Set E = 1, data can be sent
    - Delays for 6ns using a separate delay subroutine
- Set E= 0
    - completes the transition from HIGH to LOW of the enable pin

LCD_CHAR

- Sends one character to be displayed on the LCD.
    - RS = 1,signals the LCD that a character is being sent.

LCD_INIT

- initializes LCD by calling LCD_CMD subroutine in order to wake-up the LCD and prepare it for displaying characters.
    - Set E = RS = RW = 0, and delay for 15 ms using delay subroutine

1. Send command 0x30 via pins D[7:0] using the LCD_CMD subroutine and delay for 6ns using delay subroutine

2. Send command 0x30 again, then delay for 100 us
3. -Send command 0x30 again, then delay for 4.1 ms.
4. Send command 0x38 to the LCD
    a. combination of 0x20 (function), 0x10 (indicates an 8-bit data bus) and 0x80 (selects two-line display)
5. Send command 0x0C to the LCD
    a. combination of 0x08 (display on/off control) and 0x04 (to turn the LCD on)
6. Sends c 0x01 to the LCD,
    a. clears the LCD screen and sets the cursor back to its home/default position
7. Sends 0x06 to the LCD, this is a combination of 0x04 for the entry mode set, plus 0x02 to configure the LCD so that whenever a character is sent, the cursor position automatically moves over to the right.

LCD_STRING

- used to read and display a character of a user defined string, one character at a time, until the null character or zero of the string is read.
- Sends LCD command to place cursor on first character
- Calls LCD_CHAR subroutine to send a single character.
    - Loops until character is read

LCD_CLR

- Clears the LCD display
- sets the cursor back to its home position.

These subroutines are all written in one file for organization purposes. It's much easier to have them organized in one subroutine file, which is then imported by a main file called mycode. Task 3 rotates the display on the first line from right to left by passing the command 0x18 into the subroutine LCD_CMD. This rotates the string on the first line from left to right just like what you might see at your local drugstore or taco truck.

LCD Pins, as seen in figure 9.4, is a subroutine which consists of various Read-Modify-Writes to configure pins as GPIO. Pins 16-30 are utilized for the LCD

```
LCD_pinss       STMFD   SP!, {r0-r2, LR}
                MRS     r0, CPSR
                STMFD   SP!, {r0}

                ;Set P0.22 & P0.30 as GPIO
                LDR     r1, =PINSEL1
                LDR     r0, [r1]                ;read contents of PINSEL1 save into r0
                LDR     r2, =0x30003000
                BIC     r0, r0, r2  ;modify conents by clearing [13:12] and [29:28]
                STR     r0, [r1]                ;write value back into PINSEL1
                ;Set P1.16-P1.23 as GPIO
                LDR     r1, =PINSEL2
                LDR     r0, [r1]                ;read contents of PINSEL2 save into r0
                BIC     r0, r0, #0x8                ;modify contents by clearing bit 3
                STR     r0, [r1]                ;write value back into PINSEL2
                ;Set P0.22 & P0.30 as outputs
                LDR     r1, =IO0DIR
                LDR     r0, [r1]                ;read contents of IO0DIR
                LDR     r2, =0x40400000
                ORR     r0, r0, r2      ;modify contents to set Bit 22 and 30
                STR     r0, [r1]                ;write back into IO0DIR
                ;Set P1.16 - P1.23 as outputs
                LDR     r1, =IO1DIR
                LDR     r0, [r1]                ;read contents of IO1DIR
                LDR     r2, =0x3FF0000
                ORR     r0, r0, r2      ;modify contents to set Bits [25:16]
                STR     r0, [r1]                ;write back into IO1DIR
                ;Turn backlight on
                LDR     r3, =IO0SET
                LDR     r0, =LCD_LIGHT
                STR     r0, [r3]

                LDMFD   SP!, {r0}
                MSR     CPSR_f, r0
                LDMFD   SP!, {r0-r2, PC}
```

Figure 9.4: LCD_pins subroutine setting LCD pins as GPIO

while the lower 15 pins are utilized for LED’s. For the sake of this lab, Pins 16-30 are utilized for GPIO, as seen below in figure. All subroutines begin and end with the similar format of saving registers, including the link register, and flags, then storing at the end of the subroutine. This process essentially allows the program to branch to a subroutine and branch back to the “call” of the subroutine as if it never happened

LCD_cmds is a subroutine which enables the LCD control lines, allowing programmers to utilize the LCD. In figure 9.5, programmers initially enable the control line, then specify in the “Register Select” line whether the input is going to be a command or text. After specifying the RS line, the “Read/Write” line is adjusted, which specifies to the program whether the code will read or write to the LCD.

```
LCD_cmds        STMFD   SP!, {r0-r5, LR}
                MRS     r0, CPSR
                STMFD   SP!, {r0}

                LDR     r1, =IO1SET
                LDR     r2, =IO1CLR
                LDR     r3, =IO0SET
                LDR     r4, =IO0CLR

                LSL     r6, #16         ;r6 contains command, shift by 16 to align

                LDR     r0, =LCD_DATA
                STR     r0, [r2]                ;Clear data line
                STR     r6, [r1]                ;Send command

                LDR     r0, =LCD_RS
                STR     r0, [r2]                ;RS = 0
                LDR     r0, =LCD_RW
                STR     r0, [r4]                ;RW = 0
                LDR     r0, =LCD_E
                STR     r0, [r2]                ;E = 0
                BL      delay10uss

                LDR     r0, =LCD_E
                STR     r0, [r1]                ;E = 1
                BL      delay10uss

                LDR     r0, =LCD_E
                STR     r0, [r2]                ;E = 0
                MOV     r5, #4
delay40         BL      delay10uss
                SUBS    r5, r5, #1
                BNE     delay40

                LDMFD   SP!, {r0}
                MSR     CPSR_f, r0
                LDMFD   SP!, {r0-r5, PC}
```

Figure 9.5: LCD_cmds subroutine

LCD_init begins similarly to all other subroutines by saving registers and flags, then storing them back to the stack pointer as if nothing happened. This subroutine initializes and configures the LCD to allow programmers to use it. The RS, R/W and Enable lines must be set to 0 before initializing the LCD. From lines 173 to 178 in figure 9.6, a 0x30 is sent to the LCD which “wakes” up the module. Lines 180 through 194 sends a command for function select, which configures the LCD to be utilized for an 8-bit data bus and a two-line display. Line 194-195 sends the command 0x0C, which turns the display on. Lines 196-197 clears the display from any previous characters or strings and resets the cursor to “home.” Finally, the command 0x06 is sent on lines 198-199 to set “Entry mode” and automatically position the cursor to the right as strings are sent to the display.

```
152 LCD_inits       STMFD   SP!, {r0-r6, LR}
153                 MRS     r0, CPSR
154                 STMFD   SP!, {r0}
155
156                 LDR     r1, =IO1SET
157                 LDR     r2, =IO1CLR
158                 LDR     r3, =IO0SET
159                 LDR     r4, =IO0CLR
160
161
162                 LDR     r0, =LCD_RS
163                 STR     r0, [r2]                ;RS = 0
164                 LDR     r0, =LCD_RW
165                 STR     r0, [r4]                ;RW = 0
166                 LDR     r0, =LCD_E
167                 STR     r0, [r2]                ;E = 0
168                 LDR     r5, =0x5DC
169 delay15m        BL      delay10uss
170                 SUBS    r5, r5, #1
171                 BNE     delay15m
172
173                 MOV     r6, #0x30
174                 BL      LCD_cmds                ;send 1st time
175                 LDR     r5, =0x19A
176 delay4m         BL      delay10uss
177                 SUBS    r5, r5, #1
178                 BNE     delay4m
179
180                 BL      LCD_cmds                ;send 2nd time
181                 MOV     r5, #10
182 delay100u       BL      delay10uss
183                 SUBS    r5, r5, #1
184                 BNE     delay100u
185
186                 BL      LCD_cmds                ;send 3rd time
187                 LDR     r5, =0x19A
188 delay4ml        BL      delay10uss
189                 SUBS    r5, r5, #1
190                 BNE     delay4ml
191
192                 LDR     r6, =0x38
193                 BL      LCD_cmds
194                 LDR     r6, =0x0C
195                 BL      LCD_cmds
196                 LDR     r6, =0x01
197                 BL      LCD_cmds
198                 LDR     r6, =0x06
199                 BL      LCD_cmds
200
201                 LDR     r0, =LCD_LIGHT
202                 STR     r0, [r4]
203
204                 LDMFD   SP!, {r0}
205                 MSR     CPSR_f, r0
206                 LDMFD   SP!, {r0-r6, PC}
```

Figure 9.6: LCD_init initialization and configuration of the display

The LCD_char subroutine positions the cursor on the LCD display to its correct position prior to displaying the desired string. As seen in figure 9.7, the Memory Map consists of two lines for the display with the cursor's corresponding location, from address 0x00 to 0x4F. To set the cursor position, a location address must be added to command 0x80, for example, 0x80 + 0x00 would result in 0x80, which corresponds to the first character of the first line. As per usual, the subroutine begins with saving the registers and flags, seen in figure 9.8. Line 123 would shift Register 6 from the main myCode file, which corresponds to the cursor location, by 16 to align with the LCD display. The LCD is then cleared, initialized and enabled, sending strings to the LCD, shifting the cursor to the right or to the next line accordingly.

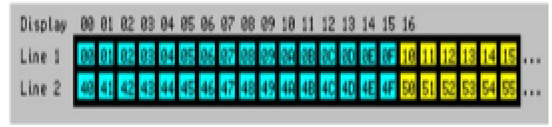


Figure 9.7: 2x16 LCD memory map

```
114 LCD_chars     STMFD   SP!, {r0-r5, LR}
115               MRS     r0, CPSR
116               STMFD   SP!, {r0}
117
118               LDR     r1, =IO1SET
119               LDR     r2, =IO1CLR
120               LDR     r3, =IO0SET
121               LDR     r4, =IO0CLR
122
123               LSL     r6, #16       ;r6 contains command, shift by 16 to align
124
125               LDR     r0, =LCD_DATA
126               STR     r0, [r2]          ;Clear data line
127               STR     r6, [r1]          ;Send command
128
129               LDR     r0, =LCD_RS
130               STR     r0, [r1]          ;RS = 1
131               LDR     r0, =LCD_RW
132               STR     r0, [r4]          ;RW = 0
133               LDR     r0, =LCD_E
134               STR     r0, [r3]          ;E = 0
135               BL      delay10uss
136
137               LDR     r0, =LCD_E
138               STR     r0, [r1]          ;E = 1
139               BL      delay10uss
140
141               LDR     r0, =LCD_E
142               STR     r0, [r2]          ;E = 0
143               MOV     r5, #4
144 delay40m      BL      delay10uss
145               SUBS    r5, r5, #1
146               BNE     delay40m
147
148               LDMFD   SP!, {r0}
149               MSR     CPSR_f, r0
150               LDMFD   SP!, {r0-r5, PC}
```

Figure 9.8: LCD_chars subroutine

```
208 LCD_strings   STMFD   SP!, {r0-r5, LR}
209               MRS     r0, CPSR
210               STMFD   SP!, {r0}
211               BL      LCD_cmds    ;send command to set cursor position
212               MOV     r5, #200
213 delay200m     BL      delay10uss
214               SUBS    r5, r5, #1
215               BNE     delay200m
216
217 stringloop    LDRB    r6, [r8], #1    ;get a character from string, post index by 1 passed through r8
218               CMP     r6, #0          ;check for end of string
219               BEQ     exit            ;leave if end found
220               BL      LCD_chars
221               B       stringloop
222 exit
223               LDMFD   SP!, {r0}
224               MSR     CPSR_f, r0
225               LDMFD   SP!, {r0-r5, PC}
226
227 LCD_clear     STMFD   SP!, {r0-r6, LR}
228               MRS     r0, CPSR
229               STMFD   SP!, {r0}
230
231               LDR     r6, =0x01
232               BL      LCD_cmds
233
234               LDMFD   SP!, {r0}
235               MSR     CPSR_f, r0
236               LDMFD   SP!, {r0-r6, PC}
237
238               END
```

Figure 9.9: Prep subroutines for the display

LCD_strings subroutine in figure 9.9 is utilized as a “middle-man,” which sends the desired strings to the LCD_cmds and LCD_chars subroutines, furthermore displaying them on the LCD display. LCD_clear subroutine clears the display prior to writing to the LCD. Lines 208 through 236 are essentially treated as an initializer and prep for the display, which are called in the main file located in figure 9.10.

In figure 9.10, programmers run the main code which implements all previously explained subroutines and displays the strings on lines 31 and 32 on lines 1 and 2 of the LCD display, respectively. Lines 12 and 13 branch link to the LCD_pins and LCD_init subroutines, which initialize and configure the LCD by turning it on and enabling it to allow programmers to write to the display. Lines 22 through 28 simply send the strings and cursor location to the LCD_strings subroutine which finally displays “I LOVE/ ASSEMBLY” on the screen. On Lines 32 and 33, a 0 is added to the end of the string because it is used as the “pointer” for the end of the string, which the subroutines check for a “0” and declares that that's the end of the string.

```
1           GLOBAL user_code
2           IMPORT  LCD_pinss
3           IMPORT  LCD_inits
4           IMPORT  LCD_clear
5           IMPORT  LCD_cmds
6           IMPORT  LCD_strings
7               AREA mycode, CODE, readonly
8
9   IO0PIN EQU 0XE0028000
10
11  user_code
12          BL LCD_pinss
13          BL LCD_inits
14          LDR R0,=IO0PIN
15          MOV R1, #0X40000000
16          LDR R2,[R0]
17          ORR R2,R2,R1
18          STR R2,[R0]
19          MOV R6, #0X80000000
20          BL LCD_cmds
21          BL LCD_clear
22          MOV R6, #0XF0;CURSOR SET AT LOCATION 00 LINE 1
23          LDR R8, =STRING1
24          BL LCD_strings
25          ;BL LCD_clear
26          MOV R6, #0XC0;CURSOR SET AT LOCATION 40 LINE 2
27          LDR R8, =STRING2
28          BL LCD_strings
29
30  STOP B STOP
31
32  STRING1 DCB "I LOVE",0
33  STRING2 DCB "ASSEMBLY",0
34          ALIGN
35          END
```

Figure 9.10: Main code implementation of all subroutines

Discussion

This lab has been one of the more challenging ones because it requires students to configure the LCD’s pins properly.In order to do so students need to study the LCD datasheet and all the commands in order for it to work. First the LCD wakes up first by turning on its backlight. Now you would think that you would get the LCD to display text by simply writing strings to it but in reality we need to write one character at a time and move the cursor. The cursor is constantly moving as characters are being written, so that after one character is set, the next one is ready to go

Conclusion

Experiment 10 challenges programmers by creating several subroutines which work together in turning an LCD screen on and writing 2 lines of string to the display. Various steps were needed to be taken prior to writing to the LCD, such as enabling the display, setting it to “write,” turning the backlight on, etc. A lot of real world devices have LCD screens, and assembly is one of many programming languages where programmers can configure and play around with displays.