Experiment 8

Parallel Interfacing: Interfacing Seven Segment Display

Objective

This lab provides an opportunity to learn about the use of a microcontroller (MCU) and its interfacing to external devices. We will use ARM Cortex-M4F based Stellaris LM4F120 LaunchPad to drive a 7-segment display.

7-Segment Display Construction

A 7-Segment display is a useful electronic component use to produce numeric, alphabetic and some non-alphabetic symbols using a specific arrangement of LEDs as shown in Figure 8.1.

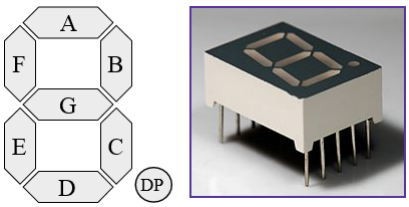

Figure 8.1: 7 Segment LED Display

A seven segment display consists of seven LEDs arranged in the form of a squarish 8 slightly inclined to the right and a single LED as the dot character. Different characters can be displayed by selectively glowing the required LED segments. Seven segment displays are of two types, common cathode and common anode. In common cathode type, the cathode of all LEDs are tied together to a single terminal which is usually labeled as com and the anode of all LEDs are left alone as individual pins labeled as a, b, c, d, e, f, g & dot. In common anode type, the anode of all LEDs are tied together as a single terminal and cathodes are left alone as individual pins. Both the configurations are shown in Figure 8.2

In this experiment, a GPIO port on the LM4F120H5QR MCU transmits 8 bits of data. These bits are applied to the seven-segment display to cause it to illuminate the appropriate segments to display the proper number or character. The seven segment used in this experiment is of

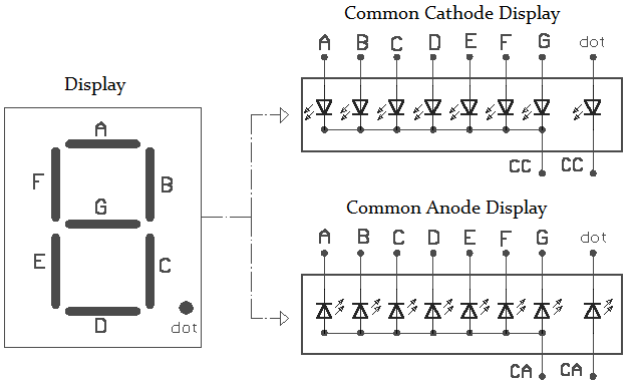

Figure 8.2: Common Anode and Common Cathode Configurations

common-cathode type. Segment intensity is dependent on the current flow and should not exceed the limit of the segment.

Digit Drive Pattern

Digit drive pattern of a seven segment LED display is simply the different logic combinations of its terminals. For a certain character, a combination of LED ON and LED OFF is generated to display the character for a short period of time. The pattern is loaded al isplay other characters. For example, to display the number 2, LEDs a, b, d, e, and g are illuminated. Table 8.3 provides the display pattern for numbers(0-9) and characters A through F.

How to Interface a Seven-Segment Display with the Z8 Encore!® MCU

zilog

Table 1. Display Pattern for Characters 0-F

Characters	DP	G	F	E	D	C	B	A	Hexadecimal
0	0	0	1	1	1	1	1	1	3F
1	0	0	0	0	0	1	1	0	06
2	0	1	0	1	1	0	1	1	5B
3	0	1	0	0	1	1	1	1	4F
4	0	1	1	0	0	1	1	0	66
5	0	1	1	0	1	1	0	1	6D
6	0	1	1	1	1	1	0	1	7D
7	0	0	0	0	0	1	1	1	07
8	0	1	1	1	1	1	1	1	7F
9	0	1	1	0	1	1	1	1	6F
A	0	1	1	1	0	1	1	1	77
B	0	1	1	1	1	1	0	0	7C
C	0	0	1	1	1	0	0	1	39
D	0	1	0	1	1	1	1	0	5E
E	0	1	1	1	1	0	0	1	79
F	0	1	1	1	0	0	0	1	71

Figure 8.3: 7 Segment Decoder Table

To control four 7-segment displays, multiplexing can reduce the number of GPIO pins required.

In this setup, the four multiplexed seven-segment displays are turned on one at a time to output the appropriate display. Because of the visual phenomenon known as *persistence of vision*, rapid switching of the seven-segment display can appear as if all four displays are turned on.

duces the illusion that all four 7-segment displays are turned on.

Multiplexing the Seven Segments

To control four 7-segment displays, multiplexing can reduce the number of GPIO pins required. In this setup, the four multiplexed seven-segment displays are turned on one at a time to output the appropriate display. Because of the visual phenomenon known as *persistence of vision*, rapid switching of the seven-segment display can appear as if all four displays are turned on.

Launchpad Interface

Before you build your circuit, you should carefully note the non-sequential pin-out diagram of the LaunchPad shown in Figure 8.4. Although the LM4F120H5QR MCU has 43 GPIO, only 35 of them are available through the LaunchPad. They are: 6 pins of Port A (PA2-PA7), 8 pins of Port B (PB0-PB7), 4 pins of Port C (PC4-PC7), 6 pins of Port D (PD0-PD3, PD6-PD7), 6 pins of Port E (PE0-PE5) and 5 pins of Port F (PF0-PF4). In addition, there are two ground, one 3.3V, one 5V (VBUS), and one reset pins available on the LaunchPad.

Pins PC0-PC3 are left off as they are used for JTAG debugging. Pins PA0-PA1 are also left off as they are used to create a virtual COM port to connect the LaunchPad to PC. These pins should not be used for regular I/O purpose.

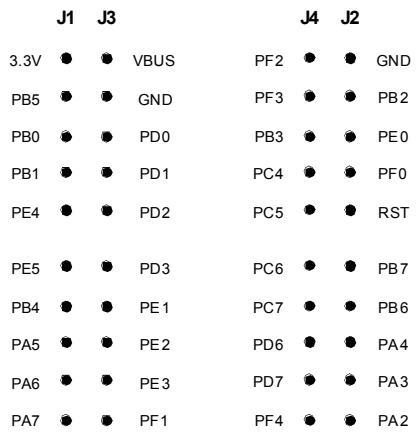

Figure 8.4: Header Pins on the Launchpad (EK-LM4f120H5QR)

Display System Using Seven Segments

In this experiment, we will program the LaunchPad and four-digit 7-segment display module to display "UOFS". In addition, when SW1 is pressed, the display is to flash the letters "H", "E", "L", "O" then settle back to "UOFS".

Note carefully the diagram of the display module. Connect the circuit as shown in Figure 8.5. The LaunchPad board is sufficient to supply +5V to Vdd of the 7-segment display module. The display power pins are connected directly to VBUS and GND of J3 On the expansion board. The experiment uses C language to program the MCU. A program for this task is included in

the lab manual. The display latches are connected to PA2-PA5 and the data pins are connected to PB0-PB7. Using the Latch Enable (LE) pin, your program can update the display digits one at a time.

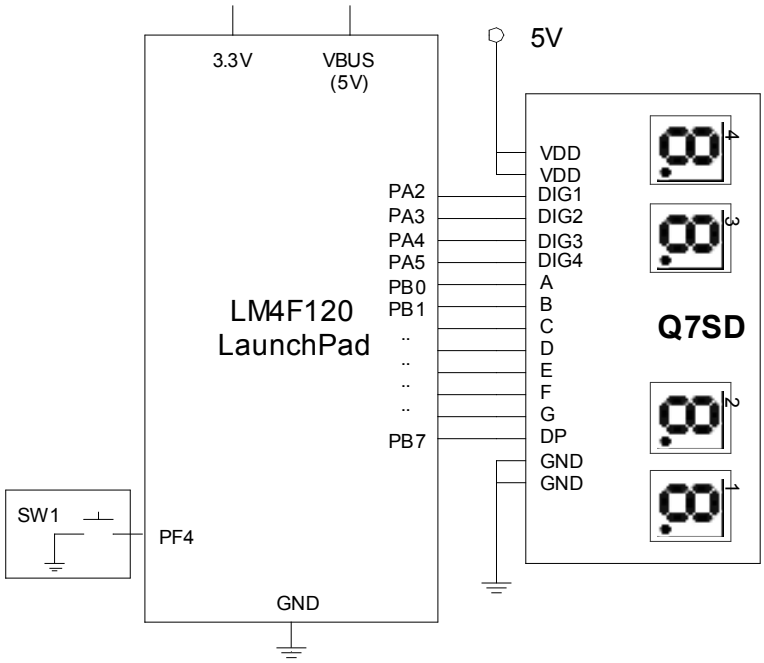

Figure 8.5: Connection to Seven Segment Display

Source Code

Consult the datasheet for the proper understanding of this code.

```
1  #define SYSCTL_RCGCGPIO_R          (*((volatile unsigned long *)0x400FE608))
2
3  #define GPIO_PORTB_DATA_R          (*((volatile unsigned long *)0x400053FC))
4  #define GPIO_PORTB_DIR_R           (*((volatile unsigned long *)0x40005400))
5  #define GPIO_PORTB_AFSEL_R         (*((volatile unsigned long *)0x40005420))
6  #define GPIO_PORTB_DEN_R           (*((volatile unsigned long *)0x4000551C))
7  #define GPIO_PORTB_PCTL_R          (*((volatile unsigned long *)0x4000552C))
8
9  #define GPIO_PORTA_DATA_R          (*((volatile unsigned long *)0x400043FC))
10 #define GPIO_PORTA_DIR_R           (*((volatile unsigned long *)0x40004400))
11 #define GPIO_PORTA_AFSEL_R         (*((volatile unsigned long *)0x40004420))
12 #define GPIO_PORTA_DEN_R           (*((volatile unsigned long *)0x4000451C))
13 #define GPIO_PORTA_PCTL_R          (*((volatile unsigned long *)0x4000452C))
14
15 #define GPIO_PORTF_DATA_R          (*((volatile unsigned long *)0x400253FC))
16 #define GPIO_PORTF_DIR_R           (*((volatile unsigned long *)0x40025400))
17 #define GPIO_PORTF_AFSEL_R         (*((volatile unsigned long *)0x40025420))
18 #define GPIO_PORTF_DEN_R           (*((volatile unsigned long *)0x4002551C))
19 #define GPIO_PORTF_PCTL_R          (*((volatile unsigned long *)0x4002552C))
```

```
20 #define GPIO_PORTF_PUR_R          (*((volatile unsigned long *)0x40025510))
21
22 #define SEG_1     0xFB
23 #define SEG_2     0xF7
24 #define SEG_3     0xEF
25 #define SEG_4     0xDF
26
27 void init_gpio (void);
28 void wait_for_key (void);
29 void display_uofs (void);
30 void display_hello (void);
31 void delay (unsigned long value);
32
33 // PortB pins:   76543210
34 //               1edcbafg
35
36 // Look up table for UOFS
37 const char lut_uofs [4] = {0xC1,  // U  11000001
38                            0xC0,  // O  11000000
39                            0x8E,  // F  10001110
40                            0x92   // S  10010010
41                           };
42
43
44 // Look up table for HELLO
45 const char lut_hello [5] = {0x89,  //H  10001001
46                             0x86,  //E  10000110
47                             0xC7,  //L  11000111
48                             0xC7,  //L  11000111
49                             0xC0   //O  11000000
50                            };
51
52 // Initialization function for ports
53 void init_gpio (void) {
54   volatile unsigned long delay_clk; // delay for clock, must have 3 sys
       clock delay
55         SYSCTL_RCGCGPIO_R |= 0x23;
56         delay_clk = SYSCTL_RCGCGPIO_R;
57
58         GPIO_PORTB_PCTL_R &= 0x00000000;
59         GPIO_PORTB_AFSEL_R &= ~0xFF;
60         GPIO_PORTB_DIR_R |= 0xFF;
61         GPIO_PORTB_DEN_R |= 0xFF;
62
63         GPIO_PORTA_PCTL_R &= 0x00000000;
64         GPIO_PORTA_AFSEL_R &= ~0x3C;
65         GPIO_PORTA_DIR_R |= 0x3C;
66         GPIO_PORTA_DEN_R |= 0x3C;
```

```
67
68          GPIO_PORTF_PCTL_R &= 0xFFF0FF0F;
69          GPIO_PORTF_AFSEL_R &= ~0x12;
70          GPIO_PORTF_DEN_R |= 0x12;
71          GPIO_PORTF_DIR_R |= 0x02;
72          GPIO_PORTF_PUR_R |= 0x10;
73
74 }
75
76 // other functions
77 void wait_for_key (void) {
78
79   while (GPIO_PORTF_DATA_R & 0x10); // wait for SW1 press, poll low
80   GPIO_PORTF_DATA_R |= 0x02; // RED on, make PF1 high
81   delay (1000000);
82   GPIO_PORTF_DATA_R &= ~0x02; // RED off, make PF1 low
83   delay (1000000);
84 }
85
86 // displays UOFS on display board
87
88 void display_uofs (void) {
89   while(GPIO_PORTF_DATA_R & 0x10)
90   {
91
92          GPIO_PORTA_DATA_R = 0xFF;
93          GPIO_PORTB_DATA_R = lut_uofs [0];
94          GPIO_PORTA_DATA_R = SEG_1;
95          delay (10000);
96
97
98          GPIO_PORTA_DATA_R = 0xFF;
99          GPIO_PORTB_DATA_R = lut_uofs [1];
100         GPIO_PORTA_DATA_R = SEG_2;
101         delay (10000);
102
103
104         GPIO_PORTA_DATA_R = 0xFF;
105         GPIO_PORTB_DATA_R = lut_uofs [2];
106         GPIO_PORTA_DATA_R = SEG_3;
107         delay (10000);
108
109         GPIO_PORTA_DATA_R = 0xFF;
110         GPIO_PORTB_DATA_R = lut_uofs [3];
111         GPIO_PORTA_DATA_R = SEG_4;
112         delay (10000);
113   }
114 }
```

```
115
116 // displays HELO on display board
117
118 void display_helo (void) {
119   int i=0;
120   while(i<100)
121   {
122         GPIO_PORTA_DATA_R = 0xFF;
123         GPIO_PORTB_DATA_R = lut_hello [0];
124         GPIO_PORTA_DATA_R = SEG_1;
125         delay (10000);
126
127         GPIO_PORTA_DATA_R = 0xFF;
128         GPIO_PORTB_DATA_R = lut_hello [1];
129         GPIO_PORTA_DATA_R = SEG_2;
130         delay (10000);
131
132         GPIO_PORTA_DATA_R = 0xFF;
133         GPIO_PORTB_DATA_R = lut_hello [2];
134         GPIO_PORTA_DATA_R = SEG_3;
135         delay (10000);
136
137         GPIO_PORTA_DATA_R = 0xFF;
138         GPIO_PORTB_DATA_R = lut_hello [4];
139         GPIO_PORTA_DATA_R = SEG_4;
140         delay (10000);
141         i++;
142   }
143 }
144
145 void delay (unsigned long value) {
146   unsigned long i;
147   for (i = 0; i < value; i++);
148 }
149
150 int main (void) {
151
152   init_gpio ();
153
154   while (1) {
155     display_uofs ();
156     wait_for_key (); // polling SW1 (PF4)
157     display_helo ();
158   }
159 }
```