Experiment 6

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. The lab uses the Tiva-C TM4C123 LaunchPad that has an ARM Cortex-M4F based TM4C123GH6PM MCU in it. This lab uses C language programming 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 6.1.

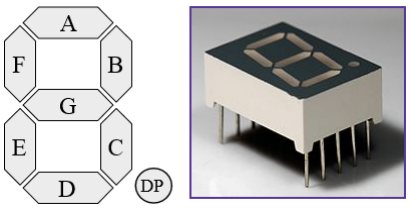

Figure 6.1: 7 Segment LED Display

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 6.2

In this experiment, a GPIO port on the TM4C123GH6PM 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

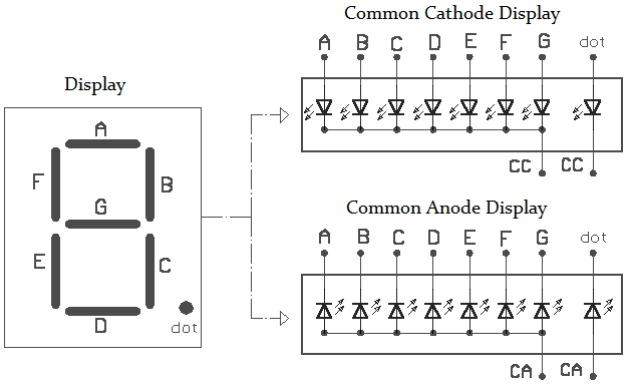

Figure 6.2: Common Anode and Common Cathode Configurations

common-anode 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 alternately to display other characters. For example, to display the number 2, LEDs a, b, d, e, and g are illuminated. Table 6.3 provides the display pattern for numbers(0-9) and characters A through F.

Characters	DP	G	F	E	D	C	B	A	Hexadecimal
0	1	1	0	0	0	0	0	0	0xC0
1	1	1	1	1	1	0	0	1	0xF9
2	1	0	1	0	0	1	0	0	0xA4
3	1	0	1	1	0	0	0	0	0xB0
4	1	0	0	1	1	0	0	1	0x99
5	1	0	0	1	0	0	1	0	0x92
6	1	0	0	0	0	0	1	0	0x82
7	1	1	1	1	1	0	0	0	0xF8
8	1	0	0	0	0	0	0	0	0x80
9	1	0	0	1	0	0	0	0	0x90
A	1	0	0	0	1	0	0	0	0x88
B	1	0	0	0	0	0	1	1	0x83
C	1	1	0	0	0	1	1	0	0xC6
D	1	0	1	0	0	0	0	1	0xA1
E	1	0	0	0	0	1	1	0	0x86
F	1	0	0	0	1	1	1	0	0xs8E

Figure 6.3: 7 Segment Decoder Table

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 6.4. Although the TM4C123GH6PM 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.

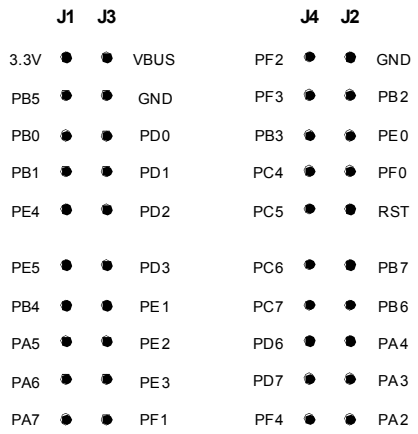

Figure 6.4: Header Pins on the Launchpad (EK-TM4C123GH6PM)

Display System Using Seven Segments

In this experiment, we will program the LaunchPad and four-digit 7-segment display module to display "UOFS" or "H", "E", "L", "O" depending upon the function selected.

Note carefully the diagram of the display module. Connect the circuit as shown in Figure 6.5. The LaunchPad board is sufficient to supply +5V to Vdd of the 7-segment display module - connect the display power pins directly to VBUS and GND of J3. The experiment uses C language to program the MCU. Write a program to accomplish the task. You will need to open a new project on Keil μVision. Note that 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.

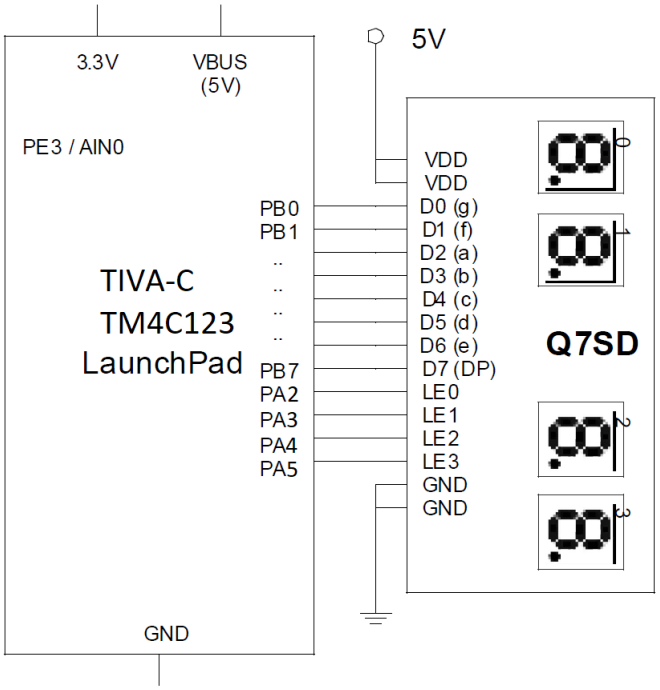

Figure 6.5: Connection to Seven Segment Display

Source Code

Source code for displaying UOFS and HELO is given below. Two files are included here. gpio.h includes the register definitions and display_UOFS.c displays UOFS. Create a new project in Keil and add both files in it. Build your code and observe the output.

Example 6.1 (**gpio.h**).

```
/* ********************************************************
* Macros for Register Addresses and Values
 ******************************************************** */

/* Register for clock */
#define SYSCTL_RCGCGPIO_R    (*((volatile unsigned long *)0x))

/* GPIO Registers for port B */
#define GPIO_PORTB_DATA_R    (*((volatile unsigned long *)0x))
#define GPIO_PORTB_DIR_R     (*((volatile unsigned long *)0x))
#define GPIO_PORTB_DEN_R     (*((volatile unsigned long *)0x))

/* GPIO Registers for port A */
#define GPIO_PORTA_DATA_R    (*((volatile unsigned long *)0x))
#define GPIO_PORTA_DIR_R     (*((volatile unsigned long *)0x))
#define GPIO_PORTA_DEN_R     (*((volatile unsigned long *)0x))

/* Values for enabling seven segments */
#define SEG_1    0x
#define SEG_2    0x
```

```
#define SEG_3   0x
#define SEG_4   0x
#define SEG_OFF 0xFF

/* Function Declarations */
void init_gpio(void);
void display_1(void);
void display_2(void);
void delay(unsigned long value);
```

Example 6.2 (display_UOFS.c).

```
/*****************************************************************
*This program is written for common anode type sevensegment display*
*     Seven segment digits pins:    PA2-PA5*
*     Seven segment data pins:      PB0-PB7*
*     Port B pins:                  76543210*
*                                   pgfedcba*
*****************************************************************/

#include ''gpio.h''

void SystemInit(){}

/*Lookup tables for common anode display */
/*lut for display1*/
const char lut_display1[4]={0x,//U --> 11000001
              0x,//O --> 11000000
              0x,//F --> 10001110
              0x//S --> 10010010
              };

/* lut for display2 */
const char lut_display2[4]={0x,//0x89,H --> 10001001
              0x,//E --> 10000110
              0x,//L --> 11000111
              0x//O --> 11000000
              };

/* lut for segment selection */
const char seg_select[]={0x,//SEG_1
             0x,//SEG_2
             0x,//SEG_3
             0x//SEG_4
            };

/*initialization function for ports */
void init_gpio(void){
```

```
    volatile unsigned long delay_clk;

    /*enable clock for PortA and PortB */
    SYSCTL_RCGCGPIO_R |= 0x;
    // dummy read for delay for clock,must have 3sys clock delay
    delay_clk=SYSCTL_RCGCGPIO_R;

    /* Enable the GPIO pin for PortB pins 0-7 for digital function
    and set the direction as output. */
    GPIO_PORTB_DEN_R |=0x;
    GPIO_PORTB_DIR_R |= 0x;

    /* Enable the GPIO pin for PortA pins 2-5 for digital function.
    and set the direction as output. */
    GPIO_PORTA_DIR_R |=0x;
    GPIO_PORTA_DEN_R |=0x;
}

/* display_1 on seven segments using Macros */
void display_1(void){
    GPIO_PORTA_DATA_R =SEG_OFF;
    GPIO_PORTB_DATA_R =lut_display1[0];
    GPIO_PORTA_DATA_R =SEG_1;
    delay(10000);
    GPIO_PORTA_DATA_R=SEG_OFF;
    GPIO_PORTB_DATA_R=lut_display1[1];
    GPIO_PORTA_DATA_R=SEG_2;
    delay(10000);
    GPIO_PORTA_DATA_R=SEG_OFF;
    GPIO_PORTB_DATA_R=lut_display1[2];
    GPIO_PORTA_DATA_R=SEG_3;
    delay(10000);
    GPIO_PORTA_DATA_R=SEG_OFF;
    GPIO_PORTB_DATA_R=lut_display1[3];
    GPIO_PORTA_DATA_R=SEG_4;
    delay(10000);
}

/* display_2 on seven segments using for loop */
void display_2(void){
    int i;
    for(i=0;i<4;i++)
    {
        GPIO_PORTA_DATA_R=SEG_OFF;
        GPIO_PORTB_DATA_R=lut_display2[i];
        GPIO_PORTA_DATA_R=seg_select[i];
        delay(10000);
    }
}

/* Delay function */
```

```
void delay(unsigned long value){
  unsigned long i ;
  for(i=0;i<value;i++);
}

/* Main function */
int main(void){
  int i;
  init_gpio();
  while(1)
    display_1();
    // display_2();

}
```

Exercises

Program the LaunchPad and four-digit 7-segment display module to alternatively display "HELO" and after some delay,"2021".