

## Experiment 8

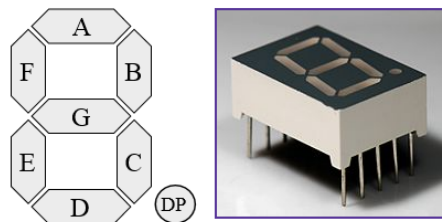
# Parallel Interfacing: Interfacing Seven Segment Display

### Objective

This lab provides an opportunity to learn about the use of a microcontroller (MCU) and interfacing it to external devices. The lab uses the Stellaris LM4F120 LaunchPad board, that has an ARM Cortex-M4F (LM4F120H5QR) MCU on it, and C language programming to drive a 7-segment display.

### Seven Segment Display Construction

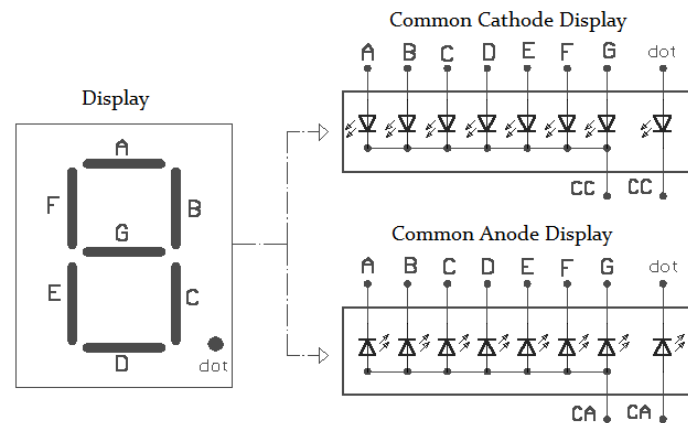
Seven segment display is a useful electronic component used to produce numeric, alphabetic and some non-alphabetic symbols using a specific arrangement of LEDs as shown in Figure 8.1



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



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 alternately to display 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.

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

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 8.4. Although the LM4F120H5QR MCU has 43 GPIOs, 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 pin 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.

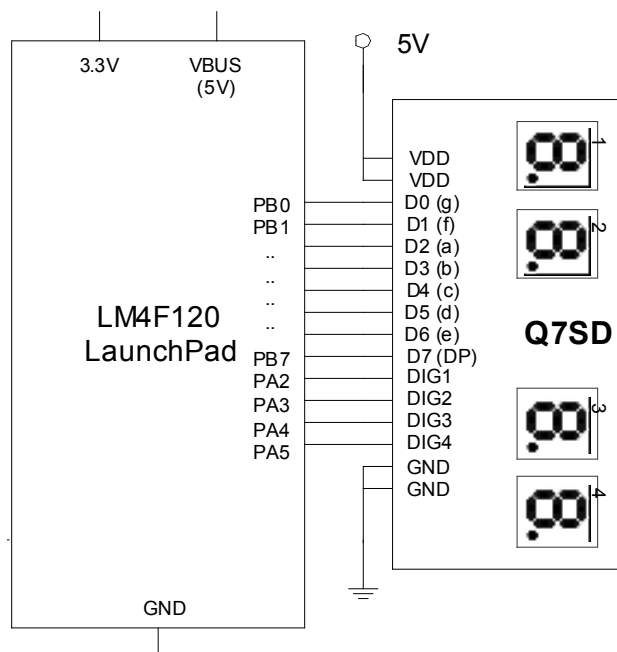
J1	J3	J4	J2
3.3V	VBUS	PF2	GND
PB5	GND	PF3	PB2
PB0	PD0	PB3	PE0
PB1	PD1	PC4	PF0
PE4	PD2	PC5	RST
PE5	PD3	PC6	PB7
PB4	PE1	PC7	PB6
PA5	PE2	PD6	PA4
PA6	PE3	PD7	PA3
PA7	PF1	PF4	PA2

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, after some delay, the display is to flash the letters “H”, “E”, “L”, “O” and 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 - 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



Connection to Seven Segment Display

a new project on Sourcery Codebench. Note that display latches are connected to PA4-PA7 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.

## Source Code

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

### Example 8.1: `gpio.h`

```
#define SYSCTL_RCGCGPIO_R      (*((volatile unsigned long *)0x400FE608))

#define GPIO_PORTB_DATA_R      (*((volatile unsigned long *)0x400053FC))
#define GPIO_PORTB_DIR_R      (*((volatile unsigned long *)0x40005400))
#define GPIO_PORTB_DEN_R      (*((volatile unsigned long *)0x4000551C))

#define GPIO_PORTA_DATA_R      (*((volatile unsigned long *)0x400043FC))
#define GPIO_PORTA_DIR_R      (*((volatile unsigned long *)0x40004400))
#define GPIO_PORTA_DEN_R      (*((volatile unsigned long *)0x4000451C))

#define SEG_1      0xFB
#define SEG_2      0xF7
```

```
#define SEG_3      0xEF
#define SEG_4      0xDF
#define SEG_OFF    0xFF

void init_gpio (void);
void display_uofs (void);
void display_hello (void);
void delay (unsigned long value);
```

### Example 8.2: display\_UOFS.c

```

/*
 * This program is written for common anode type seven segment display
 * Seven segment digits pins: PA2 — PA5
 * Seven segment data pins: PB0 — PB7
 * PortB pins: 76543210
 *
 * pgfedcba
 *
 *
 */

#include "gpio.h"

// Look up tables for common anode display
// lut for UOFS
const char lut_uofs [4] = {0xC1, // U 11000001
                           0xC0, // O 11000000
                           0x8E, // F 10001110
                           0x92, // S 10010010
                           };

// lut for HELO
const char lut_helo [4] = {0x89, //H 10001001
                           0x86, //E 10000110
                           0xC7, //L 11000111
                           0xC0, //O 11000000
                           };

const char seg_select [] = {0xFB, // SEG_1
                            0xF7, // SEG_2
                            0xEF, // SEG_3
                            0xDF, // SEG_4
                            };

// initialization function for ports
void init_gpio (void) {
    volatile unsigned long delay_clk; // delay for clock, must have 3 sys
    clock delay

```

```

        SYSCTL_RCGCGPIO_R |= 0x03;
        delay_clk = SYSCTL_RCGCGPIO_R;

        GPIO_PORTB_DIR_R |= 0xFF;
        GPIO_PORTB_DEN_R |= 0xFF;

        GPIO_PORTA_DIR_R |= 0x3C;
        GPIO_PORTA_DEN_R |= 0x3C;
    }

    // displays UOFS on display board
    void display_uofs (void) {

        GPIO_PORTA_DATA_R = SEG_OFF;
        GPIO_PORTB_DATA_R = lut_uofs [0];
        GPIO_PORTA_DATA_R = SEG_1;
        delay (10000);

        GPIO_PORTA_DATA_R = SEG_OFF;
        GPIO_PORTB_DATA_R = lut_uofs [1];
        GPIO_PORTA_DATA_R = SEG_2;
        delay (10000);

        GPIO_PORTA_DATA_R = SEG_OFF;
        GPIO_PORTB_DATA_R = lut_uofs [2];
        GPIO_PORTA_DATA_R = SEG_3;
        delay (10000);

        GPIO_PORTA_DATA_R = SEG_OFF;
        GPIO_PORTB_DATA_R = lut_uofs [3];
        GPIO_PORTA_DATA_R = SEG_4;
        delay (10000);
    }

    // displays HELO on display board
    void display_helo (void) {
        int i;
        for(i = 0; i < 5; i++)
        {

            GPIO_PORTA_DATA_R = SEG_OFF;
            GPIO_PORTB_DATA_R = lut_helo [i];
            GPIO_PORTA_DATA_R = seg_select[i];
            delay (10000);
        }
    }

    void delay (unsigned long value) {
        unsigned long i;

```

```
    for (i = 0; i < value; i++);  
  
}  
  
int main (void) {  
    init_gpio ();  
  
    while (1)  
        display_uofs ();  
    // display_helo ();  
  
}
```