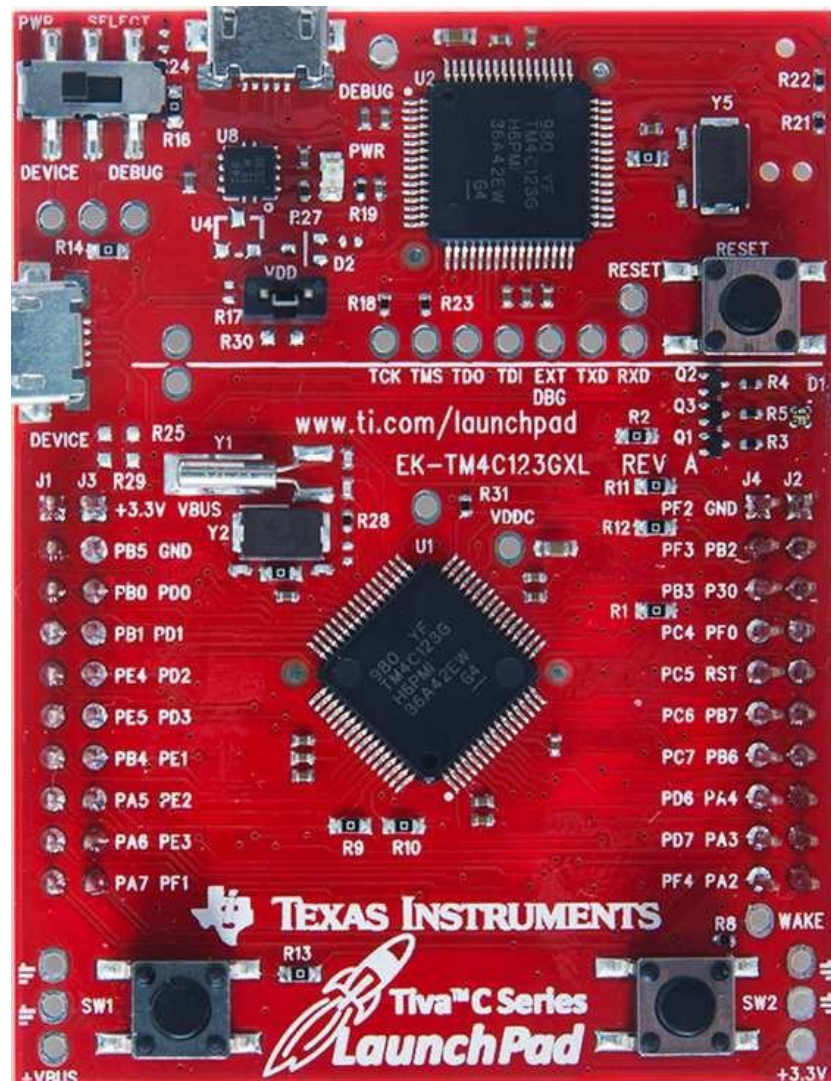


Embedded System Design using TM4C LaunchPad™ Development Kit



Experiment 5
Design and Development

Topics	Page
Objective	3
Introduction	3
Component Requirements	4
Software	5
Procedure.....	9
Observation	11
Summary	12
Exercise.....	12

- **Objective**

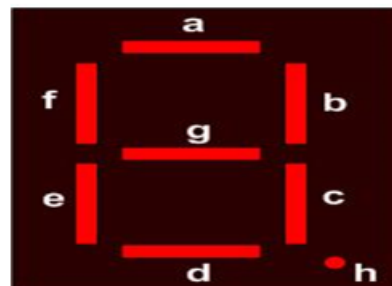
The main objective of this experiment is to learn how to write software that uses a 7-segment LED. You will learn and understand the steps required to initialize parallel ports.

- **Introduction**

The 7-seg LED can have common anode or common cathode. With common anode, the anode of the LED is driven by the positive supply voltage and the micro controller drives the individual cathodes LOW for current to flow through LEDs to light up. In this configuration, the sink current capability of the microcontroller is critical. With common cathode, the cathode of the LED is grounded, and the microcontroller is critical. With common cathode, the cathode of the LED is grounded, and microcontroller drives the individual anodes HIGH to light up the LED. In this configuration, the microcontroller pins must provide sufficient source current for each LED segment. In either configuration, if the microcontroller does not have sufficient drive or sink current capacity, we must add a buffer between the 7-seg LED and the microcontroller. The buffer for the 7-seg LED can be an IC chip or transistors.

A byte of data should be sufficient to drive all of the segments. In the example below, segment a is assigned to be bit D0, segment b is assigned to be bit D1, and so on as shown below:

Seven-segment LED interfacing and programming
(Seven Segment)



Seven-segment LED interfacing and programming
(Assignments of port pins to each segments of a 7-seg LED)

D7	D6	D5	D4	D3	D2	D1	D0
.	g	F	e	D	c	B	a

The D7 bit is assigned to decimal point. One can create the following patterns for numbers 0 to 9 for the common cathode configuration (example):

Seven-segment LED interfacing and programming
(Segment patterns for the 10 decimal digits for a common cathode 7-seg LED)

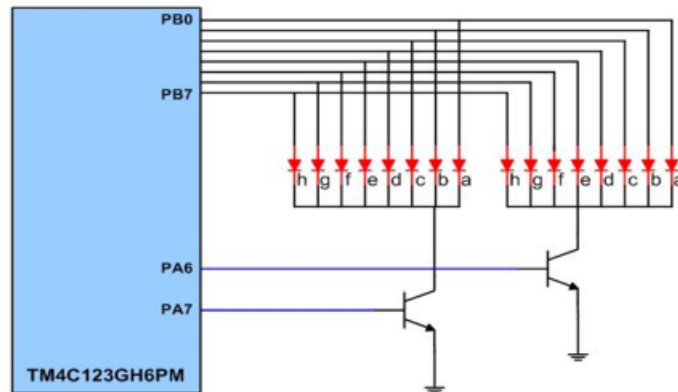
Num	D7	D6	D5	D4	D3	D2	D1	D0	Hex value
	.	g	f	e	d	c	b	a	
0	0	0	1	1	1	1	1	1	0x3F
1	0	0	0	0	0	1	1	0	0x06
2	0	1	0	1	1	0	1	1	0x5B
3	0	1	0	0	1	1	1	1	0x4F
4	0	1	1	0	0	1	1	0	0x66
5	0	1	1	0	1	1	0	1	0x6D
6	0	1	1	1	1	1	0	1	0x7D
7	0	0	0	0	0	1	1	1	0x07
8	0	1	1	1	1	1	1	1	0x7F
9	0	1	1	0	1	1	1	1	0x6F

Notice since the same segment for both digit 1 and digit 2 are connected to the same I/O port pin, the common cathode of each digit must be driven separately so that only one digit is on at a time.

The two digits are turned on alternatively. For example, if we want to display number 14 on the 7-seg LED, the following steps should be used:

1. Configure Port B as output port to drive the segments,
2. Configure Port A as output port to select the digit,
3. Write the pattern of numeral 1 from the table (above) to port B,
4. Write one bit of port A to activate the tens digit,
5. Delay for some time,
6. Write the pattern of numeral 4 from the table (above) to port B,
7. Write one bit of port A to activate the ones digit,
8. Delay for some time,
9. Repeat from step 3-8.

Seven-segment LED interfacing and programming (Microcontroller Connection to 7-segment LED)



Seven-segment LED interfacing and programming (Microcontroller Connection to 7-segment LED with Buffer Driver)

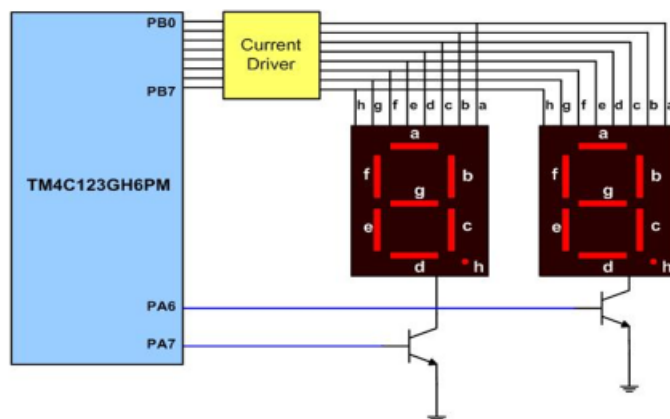


Figure 4-1 Functional Block Diagram

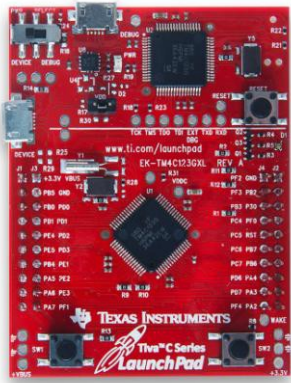
• Component Requirements

Software Requirements

1. [Keil](#)
2. [TivaWare_C_Series](#)

Hardware Requirements

Table 1-2: Components Required for the Experiment

S.No	Components	Specification	Image
1.	Tiva LaunchPad	EK-TM4C123GXL LaunchPad	
2.	USB Cable		

• Flowchart

[Please Read the Code and Provide the Flowchart](#)

• C Program Code

```

/* Seven-segment display counter
 *
 * This program counts number 0-3 on the seven segment display.
 * The seven segment display is driven by a shift register which is
 * connected to SSI2 in SPI mode.
 *
 * Built and tested with Keil MDK-ARM v5.28 and TM4C_DFP v1.1.0
 */

#include "TM4C123.h"

void delayMs(int n);
void sevenseg_init(void);
void SSI2_write(unsigned char data);

int main(void) {
    const static unsigned char digitPattern[] = {0xB0, 0xA4, 0xF9, 0xC0};
    int i;

    sevenseg_init(); // initialize SSI2 that connects to the shift registers

    while(1) {
        SSI2_write(digitPattern[i]); // write digit pattern to the seven segments
        SSI2_write((1 << i));        // select digit
        if (++i > 3)
            i = 0;
        delayMs(4); // 1000 / 60 / 4 = 4.17
    }
}

// enable SSI2 and associated GPIO pins
void sevenseg_init(void) {
    SYSCTL->RCGCGPIO |= 0x02; // enable clock to GPIOB
    SYSCTL->RCGCGPIO |= 0x04; // enable clock to GPIOC
    SYSCTL->RCGCSSI |= 0x04; // enable clock to SSI2

    // PORTB 7, 4 for SSI2 TX and SCLK
    GPIOB->AMSEL &= ~0x90; // turn off analog of PORTB 7, 4
    GPIOB->AFSEL |= 0x90; // PORTB 7, 4 for alternate function
    GPIOB->PCTL &= ~0xF00F0000; // clear functions for PORTB 7, 4
    GPIOB->PCTL |= 0x20020000; // PORTB 7, 4 for SSI2 function
    GPIOB->DEN |= 0x90; // PORTB 7, 4 as digital pins

    // PORTC 7 for SSI2 slave select
    GPIOC->AMSEL &= ~0x80; // disable analog of PORTC 7
    GPIOC->DATA |= 0x80; // set PORTC 7 idle high
    GPIOC->DIR |= 0x80; // set PORTC 7 as output for SS
    GPIOC->DEN |= 0x80; // set PORTC 7 as digital pin

```

```

SSI2->CR1 = 0;          // turn off SSI2 during configuration
SSI2->CC = 0;           // use system clock
SSI2->CPSR = 16;        // clock prescaler divide by 16 gets 1 MHz clock
SSI2->CR0 = 0x0007;     // clock rate div by 1, phase/polarity 0 0, mode freescale, data size 8
SSI2->CR1 = 2;          // enable SSI2 as master
}

// This function enables slave select, writes one byte to SSI2,
// wait for transmit complete and deassert slave select.
void SSI2_write(unsigned char data) {
    GPIOC->DATA &= ~0x80;    // assert slave select
    SSI2->DR = data;          // write data
    while (SSI2->SR & 0x10) {} // wait for transmit done
    GPIOC->DATA |= 0x80;      // deassert slave select
}

/* delay n milliseconds (50 MHz CPU clock) */
void delayMs(int n) {
    int i, j;
    for(i = 0 ; i < n; i++)
        for(j = 0; j < 6265; j++)
            {} /* do nothing for 1 ms */
}

```

- **Procedure**

1. Connect the EK-TM4C123GXL to the PC using the USB cable supplied.
2. Build, program and debug the code into the LaunchPad using Keil to view the status of the 7 segments LED.

- **Summary**

In this experiment, we have learnt the steps required to initialize serial ports.

- Complete the following table:

[illegible]