Lab #5: 7-Segment LED

School of Engineering

Electrical and Computer Engineering Department

ECEG 721-61 Embedded Systems

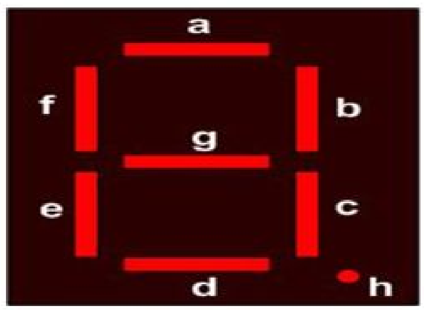
Jamie Quinn & Lucia Rosado-Fournier

November 15, 2020

Objective
The objective of this experiment was to understand the steps necessary to initialize parallel ports and use them with a 7-segment LED.

Introduction
There are several different configurations used to power on the individual segments on the LED correctly. However, generally, in order to turn on the different segments in the 7-segment LED, a byte of data is sent to the corresponding bit by the microcontroller. Depending on which bit the data is sent to, a number will be displayed on the LED. The different bits are detailed in the image 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

Figure *- 7-Segement LED interfacing.*

Component Requirements
Software Requirements
1. Keil uVision4
2. TivaWare_C_Series

Hardware Requirements
1. EK-TM4C123GXL LaunchPad
2. USB cable
3. 7-Segment LED

Software

Flowchart

The program used had four functions, all illustrated below. Function A initialized the clocks needed for the ports used, function B wrote the letter, and function C delayed the main function for n ms, where n is provided as an input. The flowchart for all four of these functions can be seen below.

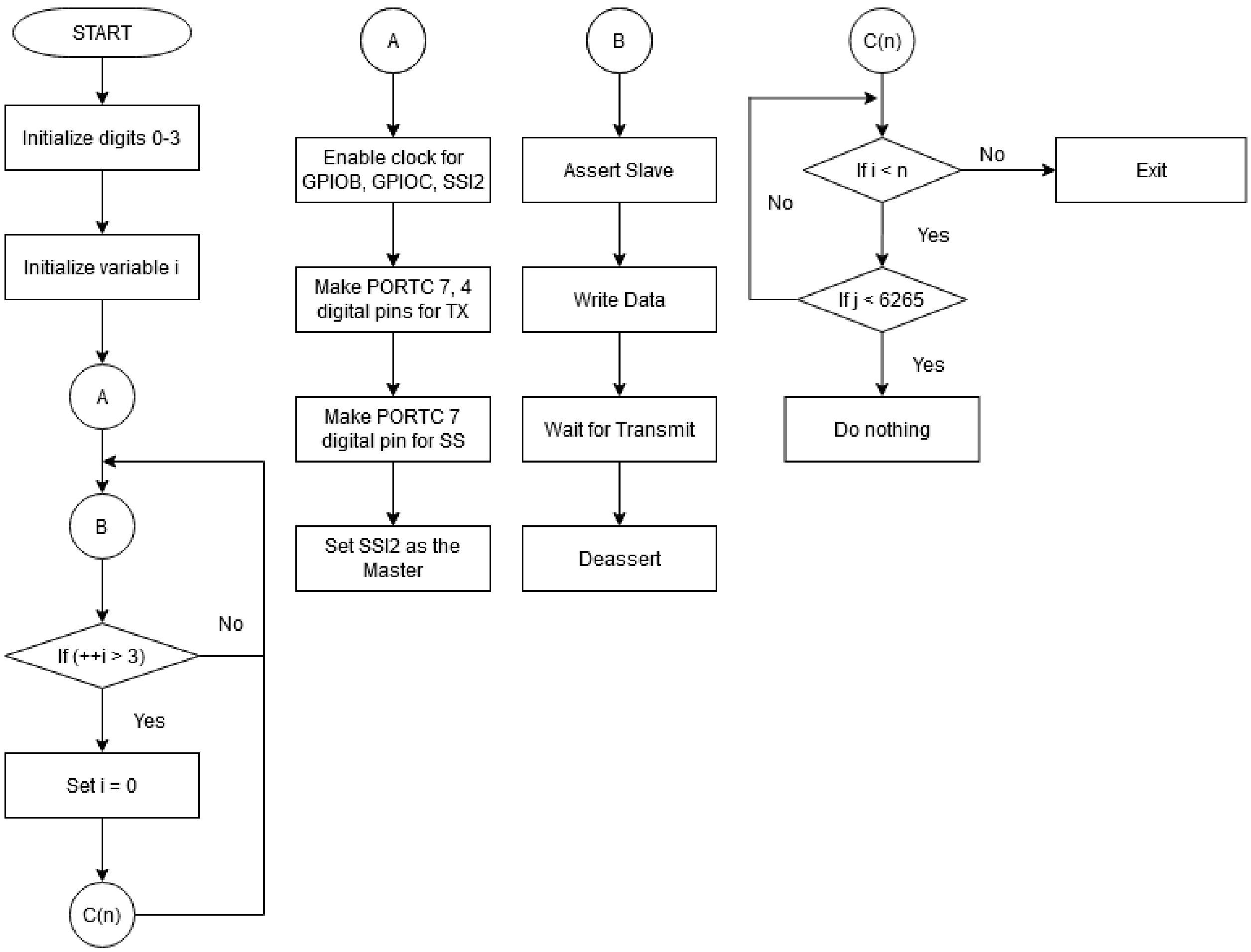


Figure *- Flowchart of the code to run a seven segment LED.*

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
Though we did not have the hardware necessary to complete the experiment, the procedure below is what we would follow to complete it.

1. Plug in the microcontroller into the PC using the provided USB cable.
2. Build the code and observe.

Observation
As stated earlier, we didn’t have the hardware required to complete the lab.

Summary
Despite the lack of hardware, being able to read the code and create a flowchart for it helped us get a better understanding of how to initialize parallel ports. Overall, this lab took about an hour to complete and write up.

Exercise
Complete the following table:

Number	.	g	f	e	d	c	b	a	Hex
0	1	1	0	0	0	0	0	0	C0
1	1	1	1	1	1	0	0	1	F9
2	1	0	1	0	0	1	0	0	A4
3	1	0	1	1	0	0	0	0	B0
4	1	0	0	1	1	0	0	1	99
5	1	0	0	1	0	0	1	0	92
6	1	0	0	0	0	0	1	0	82
7	1	1	1	1	1	0	0	0	F8
8	1	0	0	0	0	0	0	0	80
9	1	0	0	1	0	0	0	0	90