SIMON FRASER UNIVERSITY

Digital Logic and Microcontroller Project Report

Scrolling LED and Servo Motors Project

Submitted to: Dr. Amr Marzouk

Submitted By:

Gagandeep Singh (301240541)

Arun Kumar (301255784)

Feyijimi Folarin (301159032)

Submission Date 11 Dec 2015

Abstract

For the digital logic and microcontroller part our team designed a digital logic circuit on the circuit board and breadboard for controlled dot matrix LED for displaying scrolling text which was given as an input to the microcontroller. Furthermore, we used the same microcontroller to read data from 2 thumb joysticks which control the rotation direction and speed of servo motors.

Table of Contents

Introduction

This project was a combination of digital logic circuit and microcontroller. Both parts of the projects were composed separately to build up the final project which includes digital logic circuit, LED Matrix, shift registers, transistors, timer, servo motors and microcontroller. The purpose of control unit for this project was served by microcontroller containing memory, programmable input/output ports, and microprocessor core. For its use, we need low power consumption. We used TI Tm4C123GXL microcontroller to control the whole circuit including servo motors part. The main aim for this project was to display a scrolling text “MSE ROCKS” on LED display from right to left and to control servo motors using thumb joystick. In this report, we are going to convey that microcontroller we used and the description about it, and then discuss each part such as IC circuit design, algorithm, flow charts and code in detail. The biggest and most useful takeaway from this project and report Is that you can information’s about how you can communicate between hardware and software simultaneously.

Microcontroller Description

For this project the microcontroller used was made by Texas Instruments called Tiva TM4C123G Launchpad with processor of ARM Cortex-M4 (TM4C123GH6PM) to get output data values for the given input.

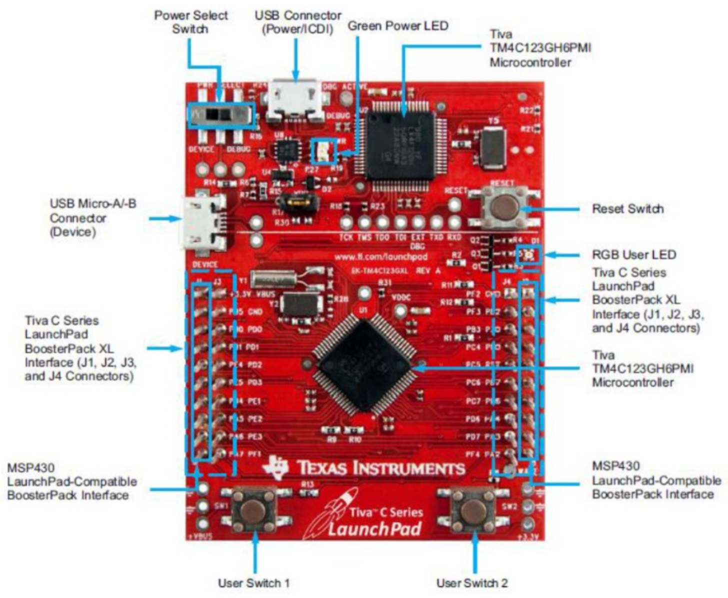

Power Select Switch
USB Connector (Power/ICDI)
Green Power LED
Tiva TM4C123GH6PMI Microcontroller
USB Micro-A/-B Connector (Device)
Reset Switch
RGB User LED
Tiva C Series LaunchPad BoosterPack XL Interface (J1, J2, J3, and J4 Connectors)
Tiva C Series LaunchPad BoosterPack XL Interface (J1, J2, J3, and J4 Connectors)
Tiva TM4C123GH6PMI Microcontroller
MSP430 LaunchPad-Compatible BoosterPack Interface
MSP430 LaunchPad-Compatible BoosterPack Interface
User Switch 1
User Switch 2

Figure 1 ARM Cortex Tm4C123GH6PM

On the board it has 2 processors; one is for debugging and another for the connecting device. USB port is there to connect t it to PC and run it with software. It also contains 40 input/output pins; two built in switches and built in RGB LEDs [1].

The schematic and peripherals we have to consider for this project are given below in the diagram. So for this we used GPIO peripherals for sending signals from microcontroller to the digital circuit for LED scrolling part. The voltage microcontroller provides is 3.3 V which is not very high and it won't be sufficient enough to light up all 512 LEDs so here we converted it from 3.3V to 5V. For controlling servo motors with potentiometers, we initiated ADC and PWM generators.

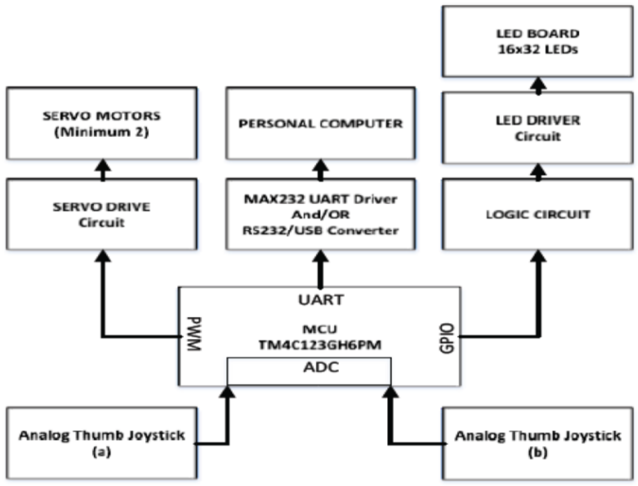


Figure 2 Layout

Overall System Overview

Display

For the display we built the circuit on circuit board. Resistors, header for LEDs, shift registers were all soldered as shown below. But due to some technical error (we think soldering might have been misconnected) the circuit didn't work and LEDs were not displayed. Our circuit looks like the shown picture below.

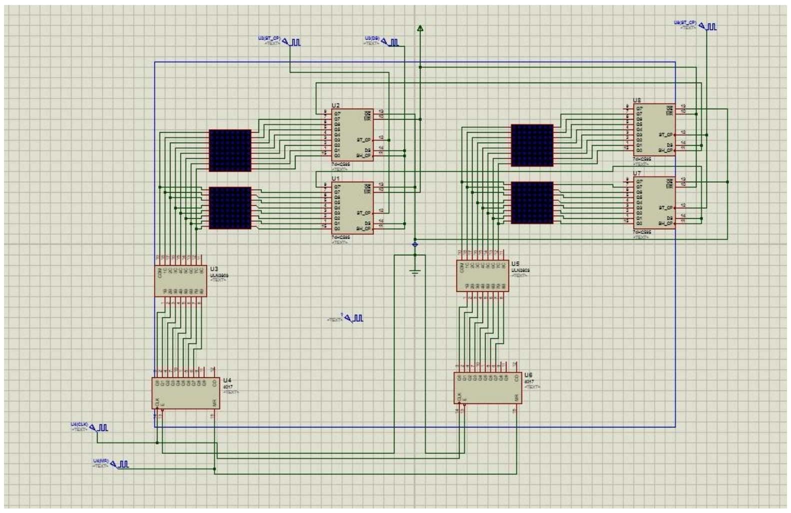


Figure 3 Proteus schematic

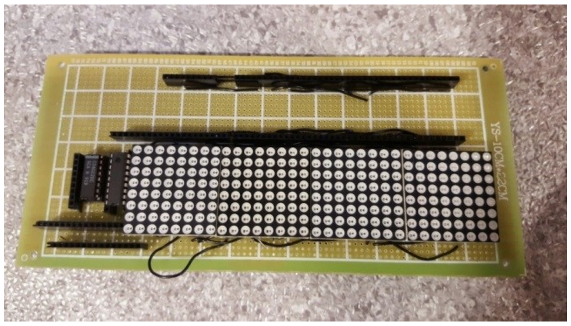


Figure 4 Top View of circuit board

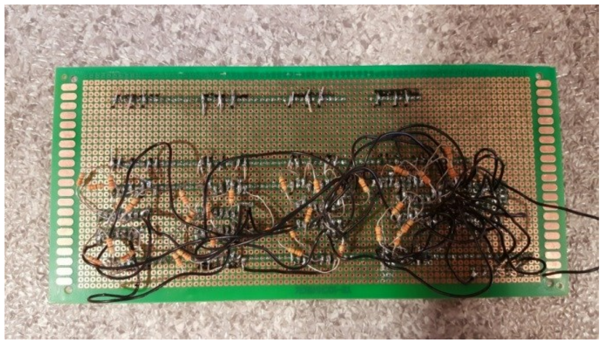


Figure 5 Behind the circuit

We had a backup circuit which was made on plastic breadboard but it was a small circuit like a piece out of a bigger circuit to show that our code works perfectly and all the connections are made right. WE used 2 LED matrix for the demo part and scrolling text was appeared as recorded by our instructor.

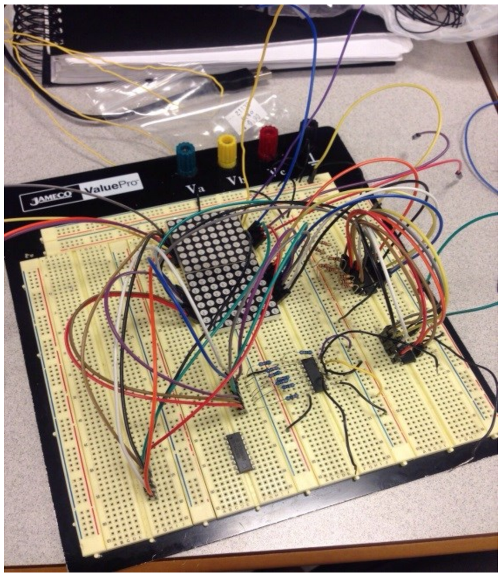


Figure 6 Breadboard Circuit

The schematic of each LED matrix is shown in the figure below. The figure shown below depicts common anode (left) and common cathode (right).

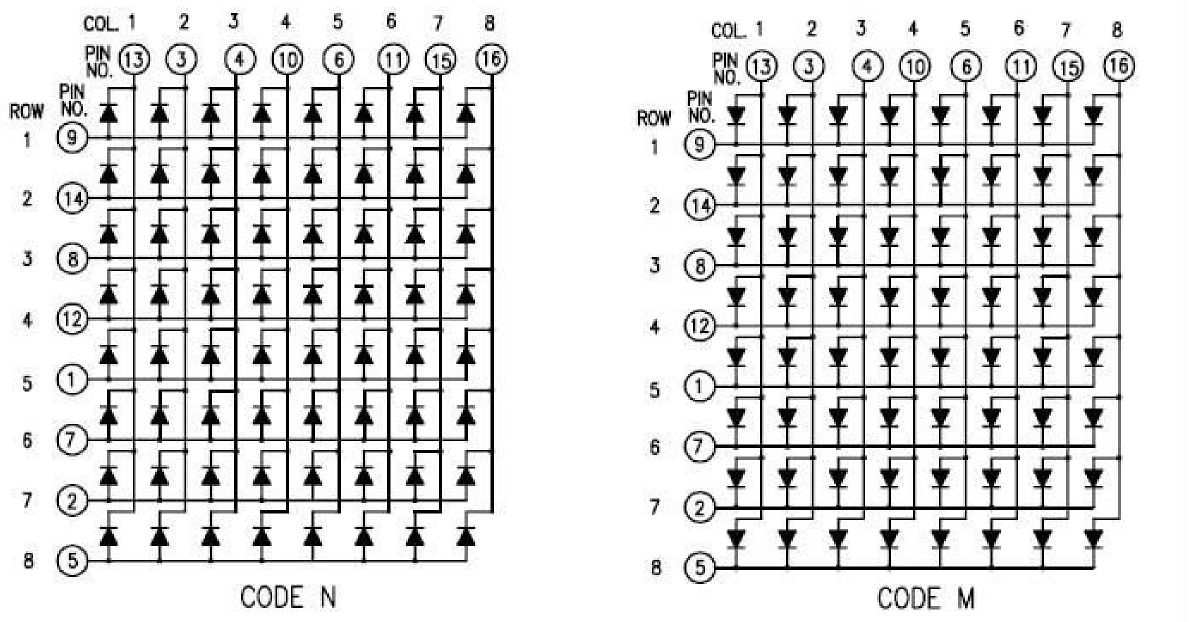


Figure 7 LED Matrix Schematic

For the left circuit, anodes of LEDs are connected in rows and cathodes are also connected in columns. Before going any further we came up with a schematic to control single LED and then a whole column.

The schematic for the overall design is shown in the proteus. The rough model that we made is shown below and on based of this we built our whole circuit. Powering and controlling every single diode in LED is too complicated and required 512 outputs, so to come up with a smart decision we made a schematic in proteus and made it work. We used 74HC595 for controlling anodes and we used CD4022BE for controlling the cathode. Also to invert the output from the counter to the dot matrix we used ULN 2803 for switching the value array. The data sheets for 74HC595, CD4022BE and ULN 2803 are provided in the appendix. To connect all the rows together we used ULN2803 which further connects to CD4022BE and which further goes to the clock. The columns are connected to the shift registers and some of the ports are combined together to reduce the input wires. The part of the circuit design is shown below.

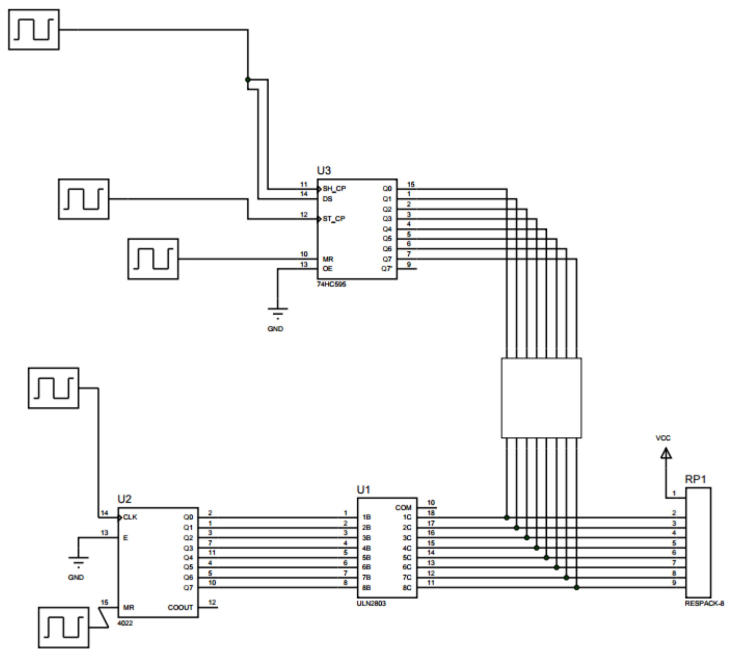
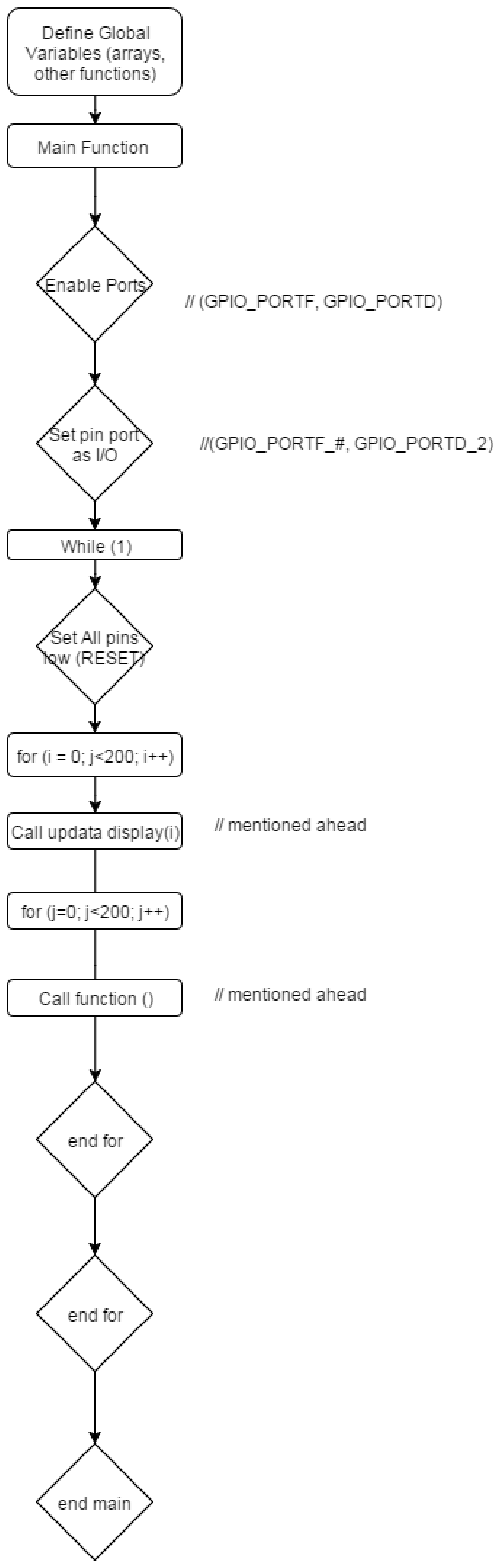


Figure 8 Layout portion of circuit

Flow Charts:

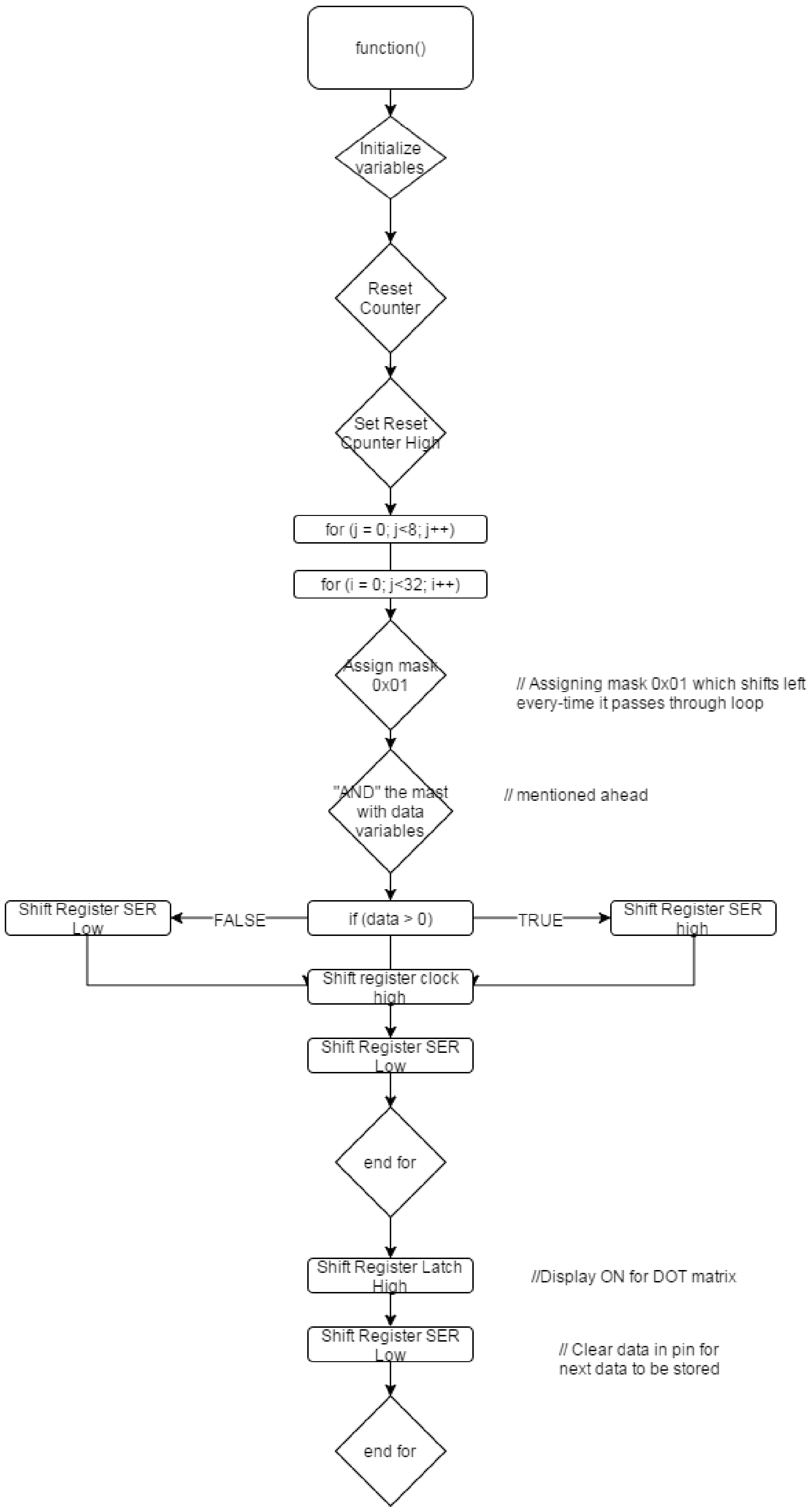

Define Global Variables (arrays, other functions)
Main Function
Enable Ports
// (GPIO_PORTF, GPIO_PORTD)
Set pin port as I/O
//(GPIO_PORTF_#, GPIO_PORTD_2)
While (1)
Set All pins low (RESET)
for (i = 0; j<200; i++)
Call updata display(i)
// mentioned ahead
for (j=0; j<200; j++)
Call function ()
// mentioned ahead
end for
end for
end main

The code initially starts here. The main function invokes serves as an avenue through which other functions are called. Our main function allowed us to initialize some of the variables and also enables the ports and peripherals used in the code. In order to use the PWM peripheral we were required to initialize it in order to allow the microcontroller initialize its PWM ports. We then enable the PWM ports we were to use in order to authorize them to carry out PWM activity whenever they were used.

The main function was used to initialize, enable and write to the following ports in order to control the LED matrix:

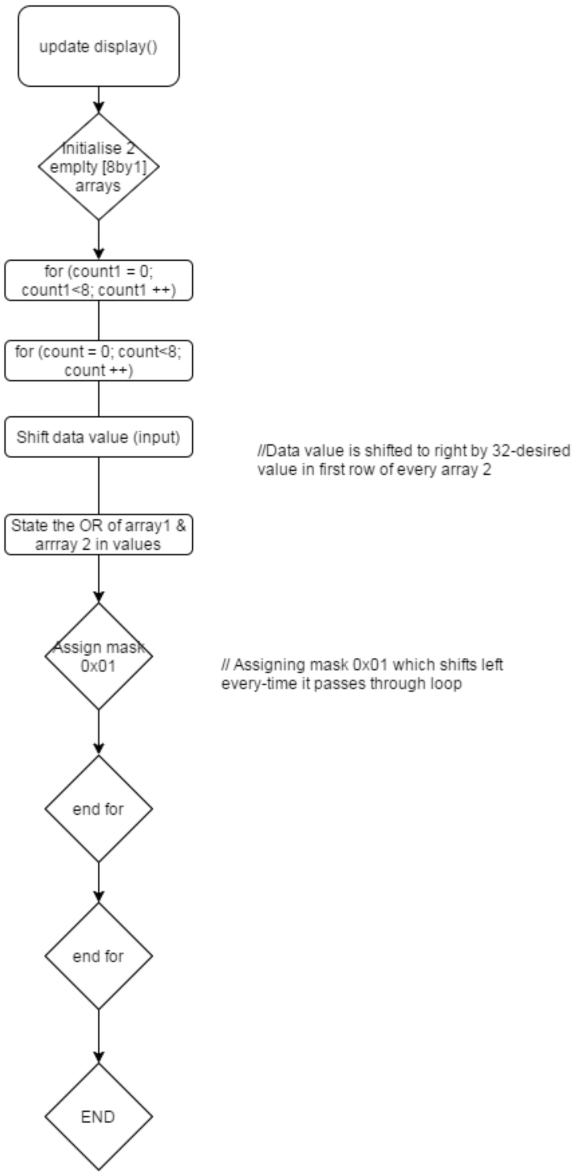
Port A, Port D and Port F With pins 1, 2 and 3 on those ports enabled. Hence, permission was given to those ports to send outputs from the microcontroller to our LED MATRIX system.

We took advantage of the while loop functionality in order to clear the data in our enables pins 3 whenever a cycles was complete. Main also contains a nested for look that allowed us to write to both our rows and columns. These were used to control the flow of data from our microcontroller by first completing the scan of one column before moving on to the next.

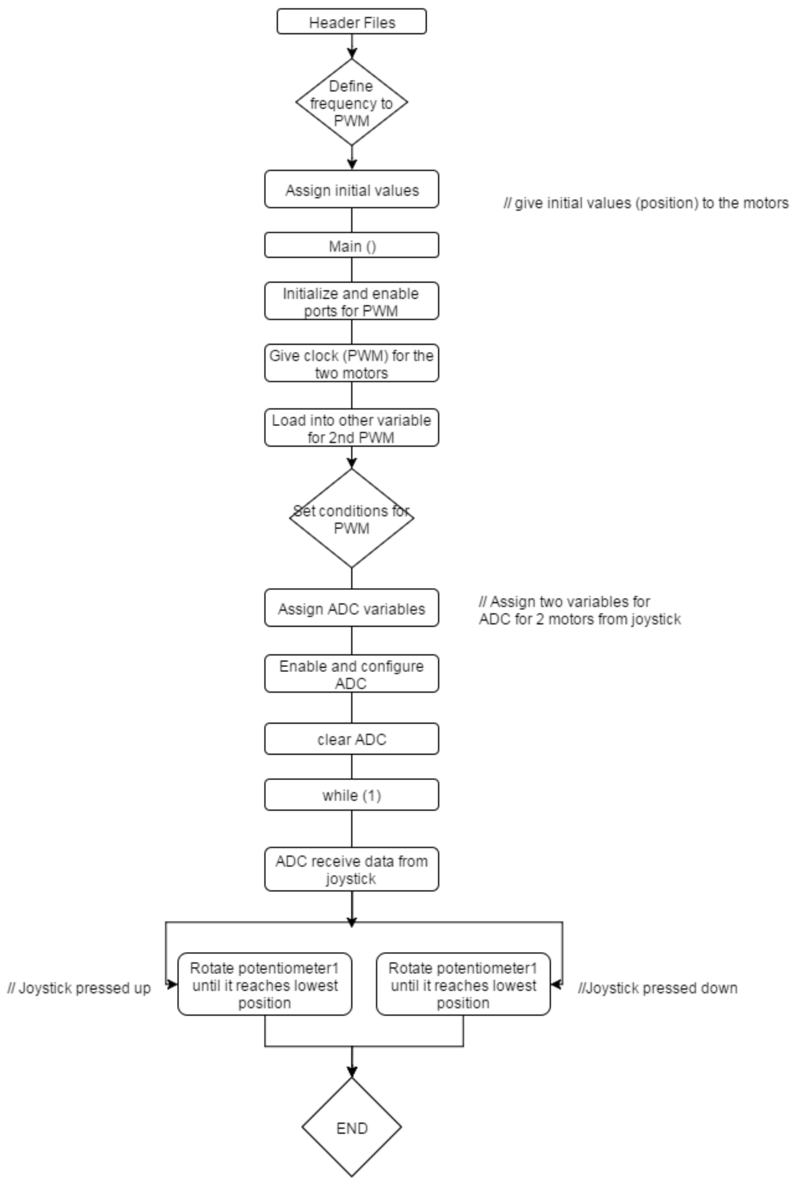

function()
Initialize variables
Reset Counter
Set Reset Cpunter High
for (j = 0; j<8; j++)
for (i = 0; j<32; i++)
Assign mask 0x01
// Assigning mask 0x01 which shifts left every-time it passes through loop
"AND" the mast with data variables
// mentioned ahead
if (data > 0)
FALSE
TRUE
Shift Register SER Low
Shift Register SER high
Shift register clock high
Shift Register SER Low
end for
Shift Register Latch High
//Display ON for DOT matrix
Shift Register SER Low
// Clear data in pin for next data to be stored
end for

The hardware working of shift registers and counters was explained in the sections above. However, the software part of it was coded to perform the similar operations that would duplicate the working of a hardware shift registers and counters.

The function ‘function’ will tell us about the working of it in a better way. We know, to send data into the shift register various steps need to be done. Firstly, we need to first reset all the ports of shift registers so that we do not input any wrong data. Now, since the shift data can be sent into the shift registers one bit at a time, we and our data values with a mask which gives us either a zero or a one. This mask is shifted to the left by one every time it passes the loop. The result of the ‘and’ goes through an ‘if’ loop, if the data is 1 it gives the data (SER) pin of shift register a high and vice versa. To push the data in the shift register we clock it high. The step is done 32 times and the mask is shifted and “anded” with the values the same time. The above told stores one row data into the shift register. To move to the next row we give a high to the clock of the counter (CD4022BE). This brings us to the next row and we start storing the data for second row using the same steps as above. This is done for 8 times, since there are in total 8 rows in the array of our data and values variables. After all the data is stored in the shift registers we latch the shift registers by giving it a high. This means that all the data is pushed out at once and displayed in our function.


update display()
Initialise 2 emplty [8by1] arrays
for (count1 = 0; count1<8; count1 ++)
for (count = 0; count<8; count ++)
Shift data value (input)
//Data value is shifted to right by 32-desired value in first row of every array 2
State the OR of array1 & arrray 2 in values
Assign mask 0x01
// Assigning mask 0x01 which shifts left every-time it passes through loop
end for
end for
END

The update display function is basically a function that controls all the moving part of the display. We shift our input array values left and right by a number and store the values in empty arrays of length 8 by 1. Its these values that are distorting our persistence of vision as the leds turn on and of really fast in the rows and columns.


Header Files
Define frequency to PWM
Assign initial values
// give initial values (position) to the motors
Main ()
Initialize and enable ports for PWM
Give clock (PWM) for the two motors
Load into other variable for 2nd PWM
Set conditions for PWM
Assign ADC variables
// Assign two variables for ADC for 2 motors from joystick
Enable and configure ADC
clear ADC
while (1)
ADC receive data from joystick
// Joystick pressed up
Rotate potentiometer1 until it reaches lowest position
Rotate potentiometer1 until it reaches lowest position
//Joystick pressed down
END

The above flowchart is used to control the PWM and ADC part of our program. Getting to run a PWM on the TIVA we need to initialize a lot of ports. A lot of help to create the code for this was received from the labs PWM and ADC & UART. We assign two ports of the microcontroller as GPIO and connect our two motors to them. The joystick receives its voltage from the Vbus of the microcontroller is grounded there. No external power is given to this circuit. The code converts the input Analog signal of the joystick to Digital, also the limits for the turning and PWM for the motors are defined and configured in the code. The results are seen when the motors rotate one with the vertical movement of the joystick and other with the horizontal.

Step by Step approach with the sample circuit schematic:

The photo below is a proof of concept of our final circuit. We did this in order to test the working of the system on a small scale circuit before building a larger LED matrix.

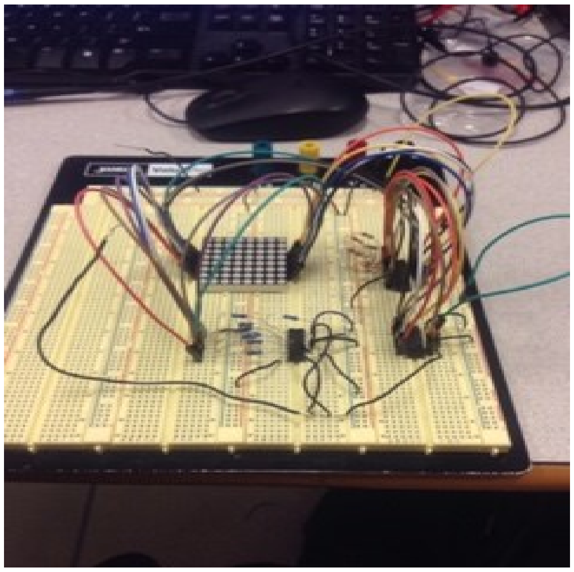


Figure 9

The Photo above shows a simulation which we carried out using an 8X8 Led dot matrix. We did this in order to gain an understanding of how the dot matrices work and how to control the switching of rows and columns using a function generator to provide inputs to our circuit. We connected all our rows to a shift register and all our columns to a decade counter which was then connected to a shift register. We then pulled the data pin of the shift register on the rows up and also that of the led on the columns up. We then connected the clock pin to a function generator and periodically set the latch pin high. We managed to generate some very fast scrolling of the lights with this.

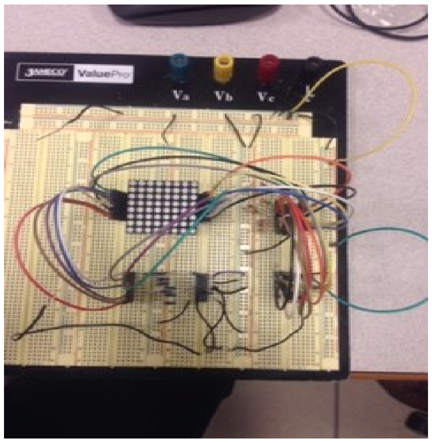


Figure 10

The next step was to use a micro controller to generate inputs which were to turn columns of the leds on periodically. We did this by connecting our data, clock and latch pins to the microcontroller and running the code called "BLINKY". This allowed us to continuously set the latch pin of our shift register high thus allowing the leds to periodically scroll across the columns.

The next step then involved combining 2 dot matrices. We connected two of our dot matrices side by side and added then connected them together. We managed to run the blinky code on this again and observed that the LED's scrolled though both matrices. We managed to power our shift registers with by using a voltage amplifier.

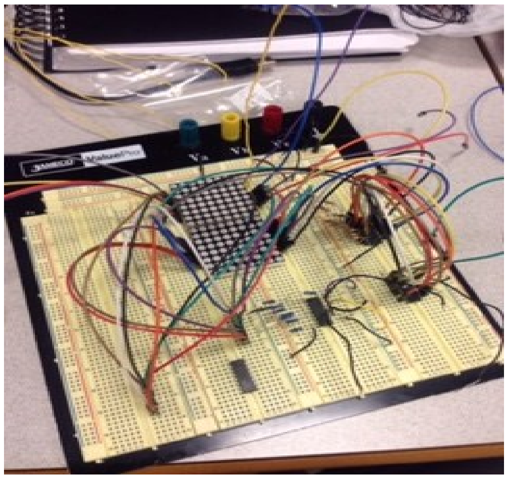


Figure 11

We connected the data, set and latch pins from our shift register to the input ports of the amplifier and then took outputs of those ports to our Tiva microcontroller. This allowed us to give the shift register its required 5v operating voltage, Hence by passing the weak 3.3 voltage the Tiva provided.

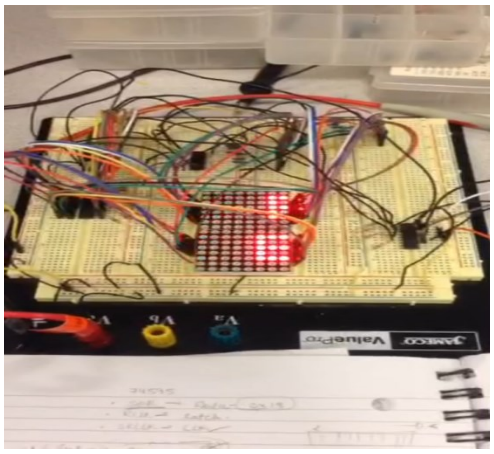


Figure 12

Finally, after writing code on the code composer to the inputs of our led’s we managed to generate a message across the two dot matrices we had. Unfortunately, as we had a time constraint on this project we were unable to incorporate our final step of a soldered 16X32 LED matrix by the time of demonstration.

Controlling servo motors using potentiometer

For the servo motor part we used the PWM generator on microcontroller by changing its duty cycle. In TM4C123G, 16 bit Up/Down counter with 2 PWM comparator are used to generate output signals. Pulse width gets changed with change in the settings on UP/Down counter. The data received from the movement of joysticks is used for actuating servo motors in synchronization. The servo motors can rotate 180 degrees with the voltage range around 5 V. It consists of 3 wires: Red, black and yellow. Black is ground, red is Vcc and yellow is signal.

ADC (Analog digital converter)

For controlling the operating period in PWM by rotating potentiometer we don’t have to reset the counter value receiving from potentiometer. For that we need enable ADC peripheral. In here ADC is converting analog signal from the input (joy stick) and change it to digital signal and saves it. The data received from the thumb joysticks would correspond to paddle positions.

Conclusion

Our team designed the scrolling LED matrix and controlled the servo motors using joysticks. But due to some hurdles we couldn’t do it completely. Like mentioned before, there was a flaw in the soldering circuit which didn’t let us successfully complete this project. Due to the error, still we were able to show a scrolling text on 2 LEDs and for potentiometer part we were successfully able to control servo motors using joystick. As the servo motor was rotating we can see its position was getting changed.

References

[1] Datasheet for TI TM4MC123GXL http://www.ti.com/lit/ds/spms376e/spms376e.pdf

[2] Datasheet for 74HC595 http://www.ti.com/lit/ds/symlink/sn74hc595.pdf

[3] Datasheet for CD4022BE http://pdf1.alldatasheet.com/datasheet-pdf/view/26871/TI/CD4022BE.html

[4] datasheet for ULN 2803 http://www.ti.com/lit/ds/symlink/uln2803a.pdf