Lab 1 - Calculator

EE138

Colin Chen & Daniel Graham

2/17/2015

Introduction

The objective of Lab 1 is to demonstrate the ability to properly interface the SAMD20 microcontroller board with a 7-segment display and a keypad. In this lab we fulfilled the following tasks.

- Using Address/Pointer, light up the LED
- Operate a 4 digit 7-segment display with a Keypad
- Make it into a calculator

By doing the following tasks, it will demonstrate that we have the knowledge to code onto the SAMD20 board, enabling of GPIO pins, interfacing of GPIOs with a 7 segment display and 16 button keypad, as well as basic state machine programming.

Procedure/Methodology

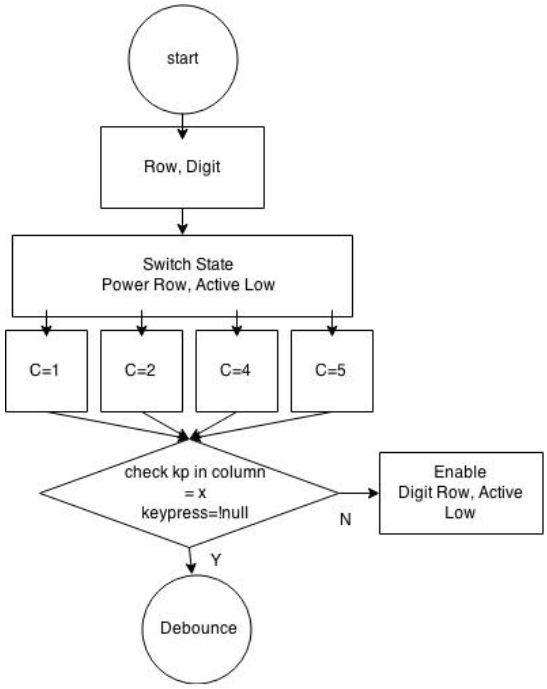
Task 1

The purpose of task 1 is to get acquainted with programming the SAMD20 board. In this task we simply set up the GPIOs so that our LED would be assigned as an output and the onboard button is assigned as an input. Using the pointer groups that the onboard peripherals are assigned to, we created a program that would turn on the LED whenever the button is pressed.

Task 2

The main goal for task 2 is to allow the 7-segment display to light up the LEDs that correspond to the numbers pressed on the keypad. To do this, we viewed task 2 as three separate parts; properly displaying the numbers, receiving keypad inputs, and software debouncing the keypad to prevent the microcontroller from registering faulty inputs.

The 7 segment display has 4 connections that correspond the the powering of each of the 4 digits, and 7 connections that correspond to the individual LEDs of the 7 segment display. By powering one of the digits at a time, and enabling (active low) the proper connections for the 7 LEDs, we can show a number on a single digit on the 7 segment display. To properly show all 4 digits at the same time, we can cycle through the 4 digits at high rates, rates higher than the human eye can recognize, and we can emulate a continuous stream of numbers being displayed rather than a single digit at a time.

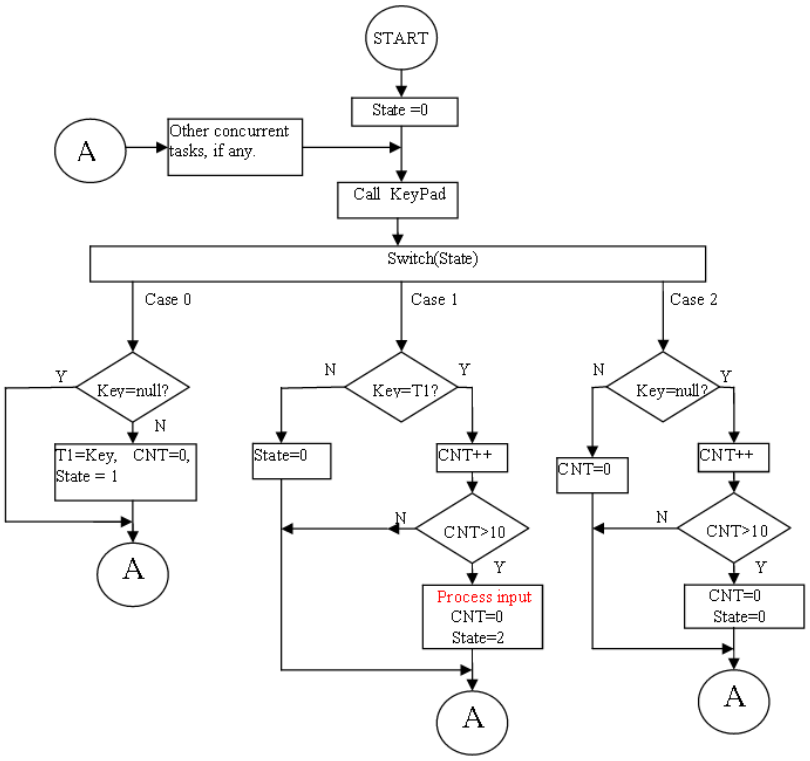


To properly display the numbers, we created a function that would accept two integers as well as pass the port pointers for LEDs and keypad ports. The two integers correspond to the row it will print out and the integer value that it would point out. Using switch

statements, we plotted out all the different possible integer printouts. In our main function, we looped this function four times, one for each of the different positions on the 4x7 segment LED display,

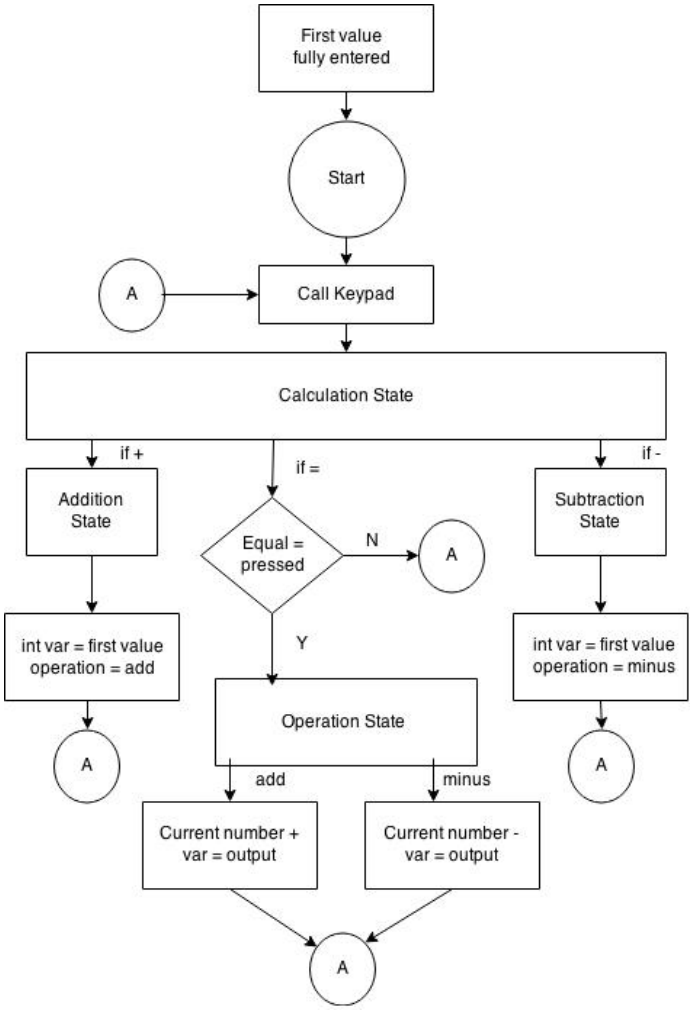
To properly receive keypad inputs from the keypad, we attached a number to each of the LED rows and on the powering of each LED, the 4 possible inputs are also read. By having the read function within the writing function, we would ensure that the LEDs are bright as possible. In addition to this, we also set the delay in such a way so that the duty cycle for the each LED being lit was as long as possible in comparison to the period for switching. The debounce state machine is identical to the state machine shown below.

P. Hsu © 2015 SJSU EE138



Task 3

In task 3, we took the basic functions of task 2 and attached a calculator function to our program. Due to the fact that our debouncing state machine is ingrained in our display function, we had to adapt our calculator into the display function as well. Using two extra states, calculation state and operation state, we were able to add on the extra buttons, the buttons that correspond to calculator functions such as +,-, delete, and equals to, into our debouncing state machine. The first input into our calculator program will be stored into a variable integer upon the press of one of the operating functions.



In Lab 1, our group has properly completed all tasks. Our calculator has the basic functions, addition, subtraction, delete, and clear. Going through this lab, we ran into difficulties that delayed the completion of the lab such as technical difficulties and coding issues. The technical issues that we faced involved our board having a faulty fuse that occasionally caused our circuit to malfunction before burning out. Upon replacing the fuse with a more power tolerant one, all the hardware issues vanished. From a coding standpoint, Implementing our state machine for our debouncing code was more difficult than we had planned. Our display code consists of one function that takes in the inputs of row and digit number. By doing this, testing our display code was fairly simple as one could simply type in the row and digit one wanted to see. However, when trying to implement the state machine for the software debouncing, we ran into issues as the check for debounce would be slightly inconsistent as the check portion for key press could not properly check until the next time the digit came around. We eventually solved this issue by carefully going through the code and adding a switch state into our state machine to deal with issues.