Memo to: Randy Larimer
From: Erik Andersen
Date: March 17, 2015
Regarding: EELE 465, Lab2 Project – Liquid-Crystal Display and Keypad

Summary:

The purpose of this lab was to display hexadecimal values 0 through F on an LCD display as well as the sixteen binary codes for each of those hexadecimal values on four LEDs using the MC9S08QG8 microprocessor and a breadboard with a custom built circuit. The sixteen different hexadecimal values and binary codes will be selected based on which button is pressed on the keypad. A fifth LED will simply act as a heartbeat LED blinking on once every second as long as there is power to the microprocessor.

Preliminary Solutions:

First, according to the user manual of the LCD on power up of the microprocessor the LCD has to always be properly initialized. According to the main.asm flowchart below the solution involved translating the LCD initialization portion of the flowchart into code located in the startup code section of the main.asm file of the project.

Second, developing a flowchart gave us a place to start from when it came to building the keypad read write code. The solution involved creating a new file called keypad read write within the sources folder of the project that contained all the code for reading from the keypad. The startup code jumps to the keypad read write subroutine which then after a long process of determining which keypad button was pressed that eight bit code is stored into memory. Based on what that eight bit code is the main code will branch to the correct subroutine. Each subroutine includes the correct code to write the correct value to the LED's and display the correct character on the display.

Setup:

The setup involved interfacing the MC9S08QG8 microprocessor with the desktop computer by USB cable to be used for downloading the firmware to the processor for debugging and wiring the keypad, LED array, LCD display and DEMO9S08QG8 onto the breadboard. The breadboard circuit contains three d-flip-flop packages, one bus transceiver package and one de-multiplexer package as well as an LCD display, potentiometer and some capacitors in order to make the LCD display visible. The LCD display is attached to the breadboard by soldering a 16-pin header to the pins so the LCD can be attached to the breadboard. The d-flip-flops and bus transceiver are used to read the keypad press store it and then write the correct value to the LED's and LCD display based on the key pressed. All instructions from the MC9S08QG8 microprocessor go through the de-multiplexer before they go to the three d-flip-flops and the bus transceiver. Further, the instructions for the LCD display come directly from the MC9S08QG8 microprocessor the four bit bus line and an inverted signal from the de-multiplexer. The reset button on the MC9S08QG8 microprocessor was also enabled.

Solution – Keypad Read Write:

First, set the PTBDD register to an output and store binary number 11110111 into the correct memory address in order to read the first row of the keypad. Then, logical shift left that binary number in order to test the next row of the keypad if the first row of the keypad had no press. Next, check to make sure all rows have been tested if not reinitialize memory address sixty until all rows have been checked. Once all rows have been checked branch to the roughly one second delay loop which was determined by using a stop watch and guessing. After the delay loop branch to the re read subroutine to recheck the button press to verify that it is still pressed after one second. Lastly, store the pattern to the correct memory address and based on the pattern stored to memory branch to one of sixteen subroutines 0 through F_write to load to PTBD in order to drive the LED's and write the correct character to the LCD (refer to the main.asm flowchart at the end of this report).

Solution – LCD Initialization

First, after power is on load values into the correct addresses for the delay loop subroutine to delay for 15 milliseconds. Then, clear bits 0 and 1 of PTAD and move binary number 00111100 into PTBD and then bring the clock back up by setting bit 3 of PTBD (function set command). Now, store the correct values for the delay loop subroutine to delay for 4.1 milliseconds. Next, write the function set command code as written on the second line of this paragraph. Then, store the correct values for the delay loop subroutine to delay for 100 microseconds. Next, write the function set command code same as before. Now, check the busy flag by jumping to the busy flag check subroutine. Refer to the busy flag check flowchart at the end of the report to see how the busy flag is checked. Then, clear bits 0 and 1 of PTAD and move binary number 00101100 into PTBD and then bring the clock back up by setting bit 3 of PTBD (function set). Again, check the busy flag by jumping to the busy flag check subroutine. Next, write the function set code same as before. Then, clear bits 0 and 1 of PTAD and move binary number 10101100 into PTBD and then bring the clock back up by setting bit 3 of PTBD (set N and F). Now, check the busy flag by jumping to the busy flag check subroutine. Then, clear bits 0 and 1 of PTAD and move binary number 00001100 into PTBD and then bring the clock back up by setting bit 3 of PTBD; again, clear bits 0 and 1 of PTAD and move binary number 10001100 into PTBD and then bring the clock back up by setting bit 3 of PTBD (Display off command). Now, check the busy flag by jumping to the busy flag check subroutine. Then, clear bits 0 and 1 of PTAD and move binary number 00001100 into PTBD and then bring the clock back up by setting bit 3 of PTBD; again, clear bits 0 and 1 of PTAD and move binary number 00011100 into PTBD and then bring the clock back up by setting bit 3 of PTBD (Clear display command). Now, check the busy flag by jumping to the busy flag check subroutine. Then, clear bits 0 and 1 of PTAD and move binary number 00001100 into PTBD and then bring the clock back up by setting bit 3 of PTBD; again, clear bits 0 and 1 of PTAD and move binary number 01101100 into PTBD and then bring the clock back up by setting bit 3 of PTBD (Entry Mode Set). Now, check the busy flag by jumping to the busy flag check subroutine. Then, clear bits 0 and 1 of PTAD and move binary number 00001100 into PTBD and then bring the clock back up by setting bit 3 of PTBD; again, clear bits 0 and 1 of PTAD and move binary number 11111100 into PTBD and then bring the clock back up by setting bit 3 of PTBD (Display On). Lastly, jump to subroutine done which simply jumps to keypad read write subroutine with some initial setup. The LCD is finally properly setup.

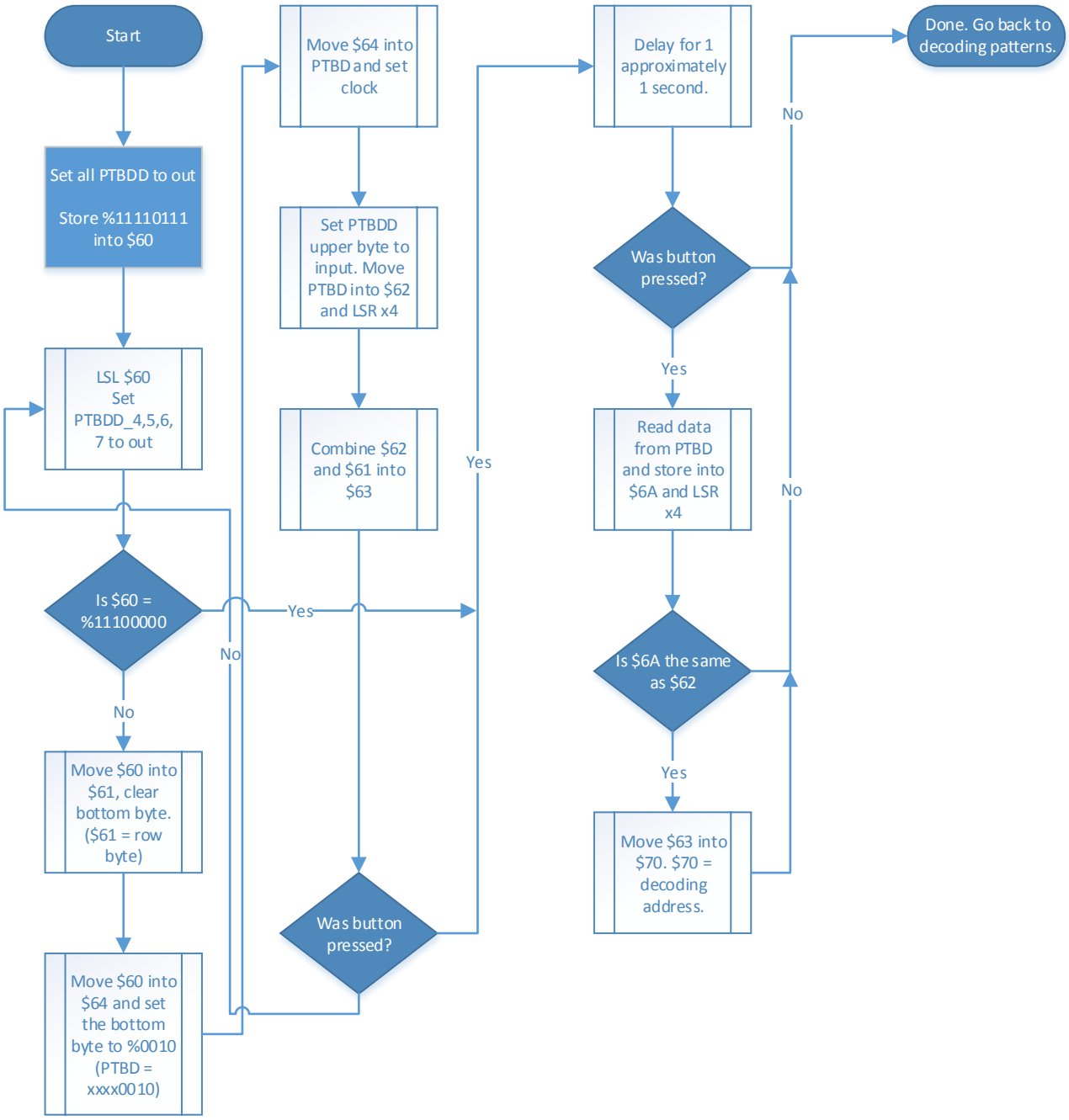
Solution – Write to LCD and LED's

First, move the correct binary value into PTBD in order to display 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F on the LED's depending on which button on the keypad is pressed. Next, set bit 3 of PTBD for a rising edge of the clock. Then, reset the button press to zero by moving 0 into the correct address. Also, set PTADD to an output and set bit 1 of PTAD and clear bit 0 of PTAD. Then, move the correct value into PTBD (upper four bits) to select the column then bring the clock back up by setting bit 3 of PTBD. Next, move the correct value into PTBD (lower four bits) to select the row then bring the clock back up by setting bit 3 of PTBD. By selecting the correct row and column characters 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F are written to the LCD based on which button on the keypad is pressed. Lastly, jump to the busy flag check subroutine and return to main.

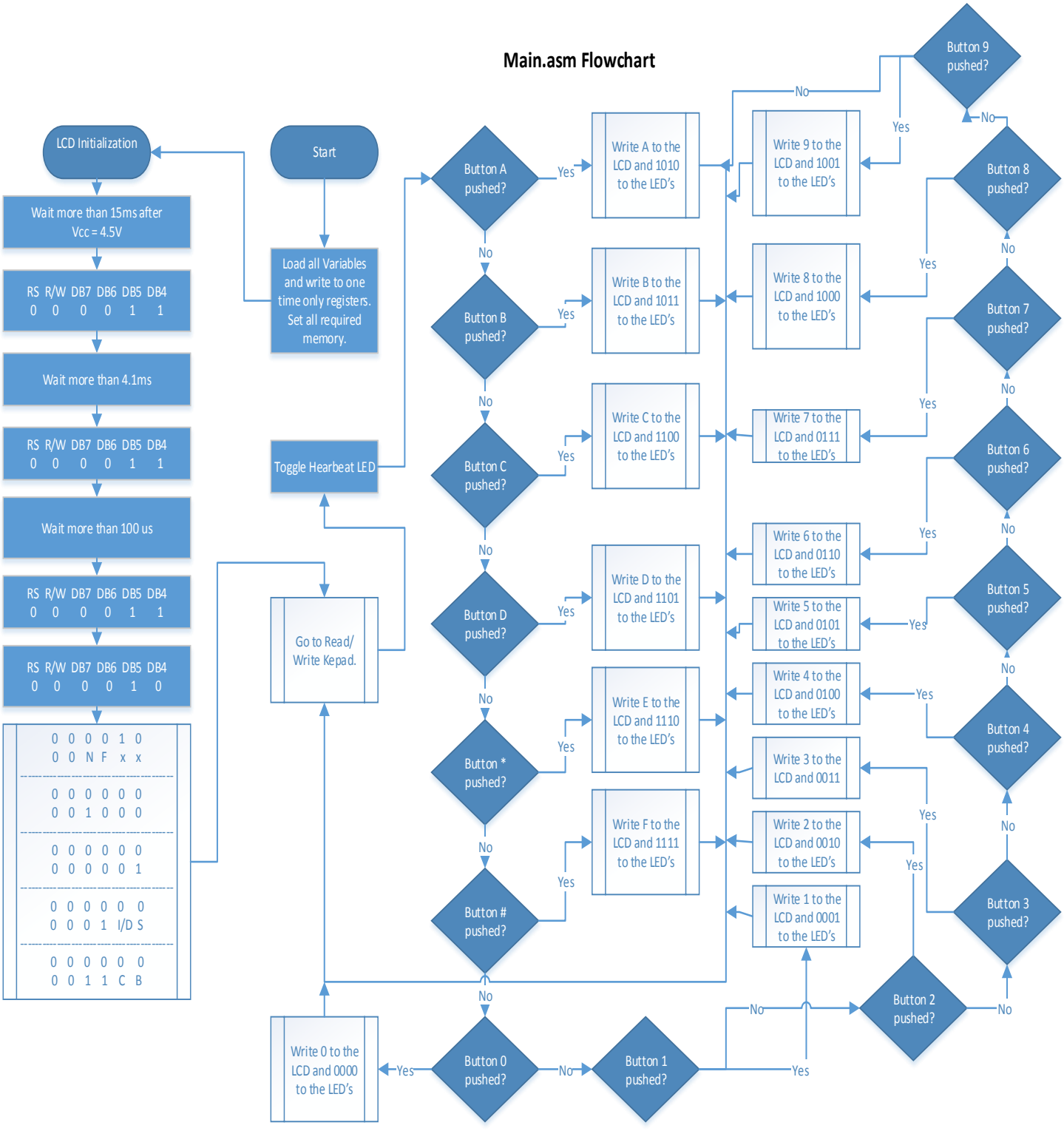
Final Comments:

While performing this lab, many problems and errors in the code were encountered. Some took multiple hours to troubleshoot and debug. The most bothersome and frustrating was figuring out and understanding how to design the solution to jump to the second line of the LCD and then when the LCD was full of characters how to completely clear the display and move the cursor back to the top left corner of the LCD. Both of the above problems were difficult to solve. We ended up writing the code for the above problems in the busy flag check subroutine. For example, is the cursor at address $10 if so jump to second line? This would not always work. For some reason when the curser was at the last location on line 1 on the display the address would not always be $10 so we had to add a few more checks for address $11 and $12. Although, the display always cleared when it was full of characters.

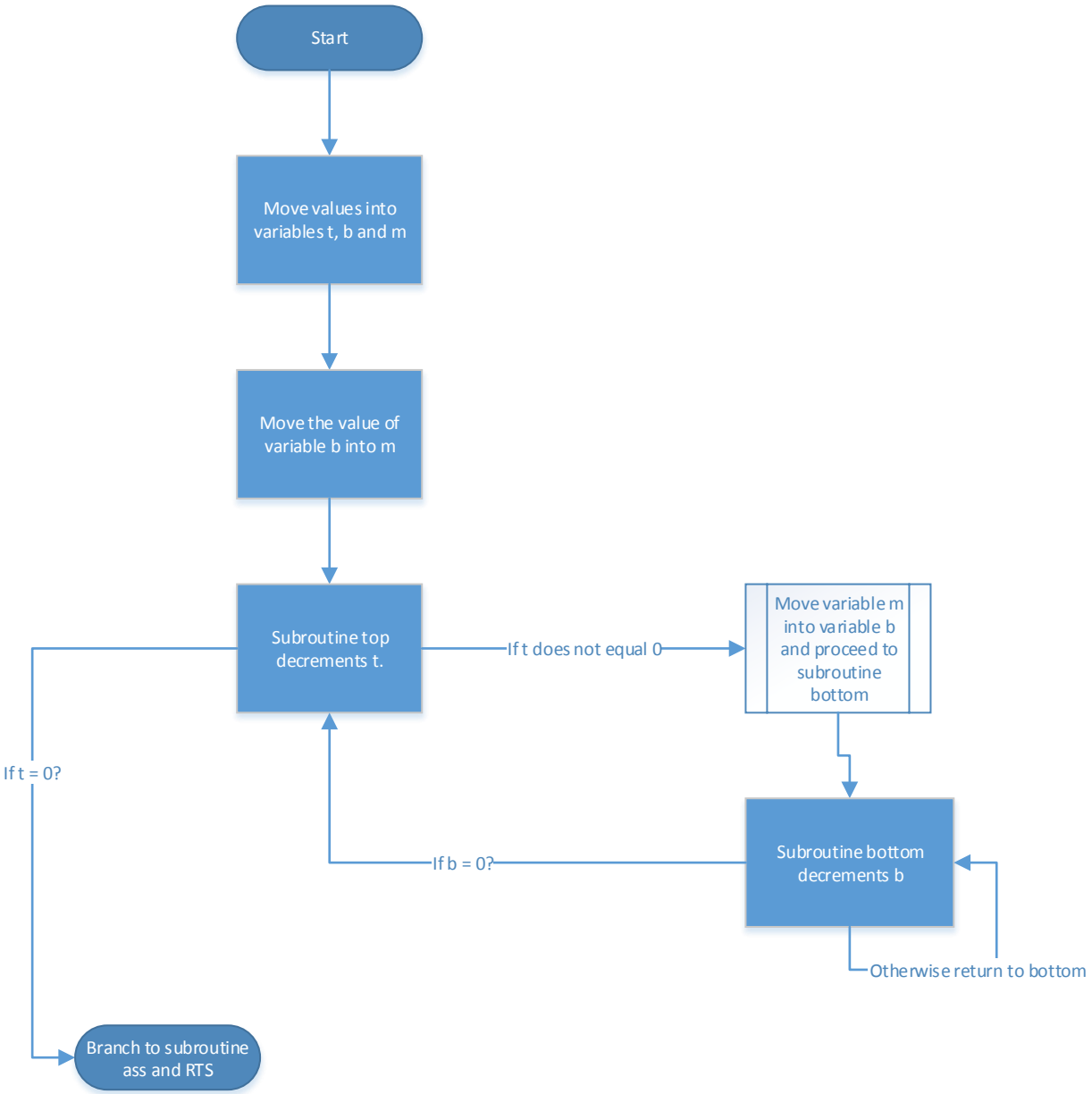
Reading from Keypad Flowchart

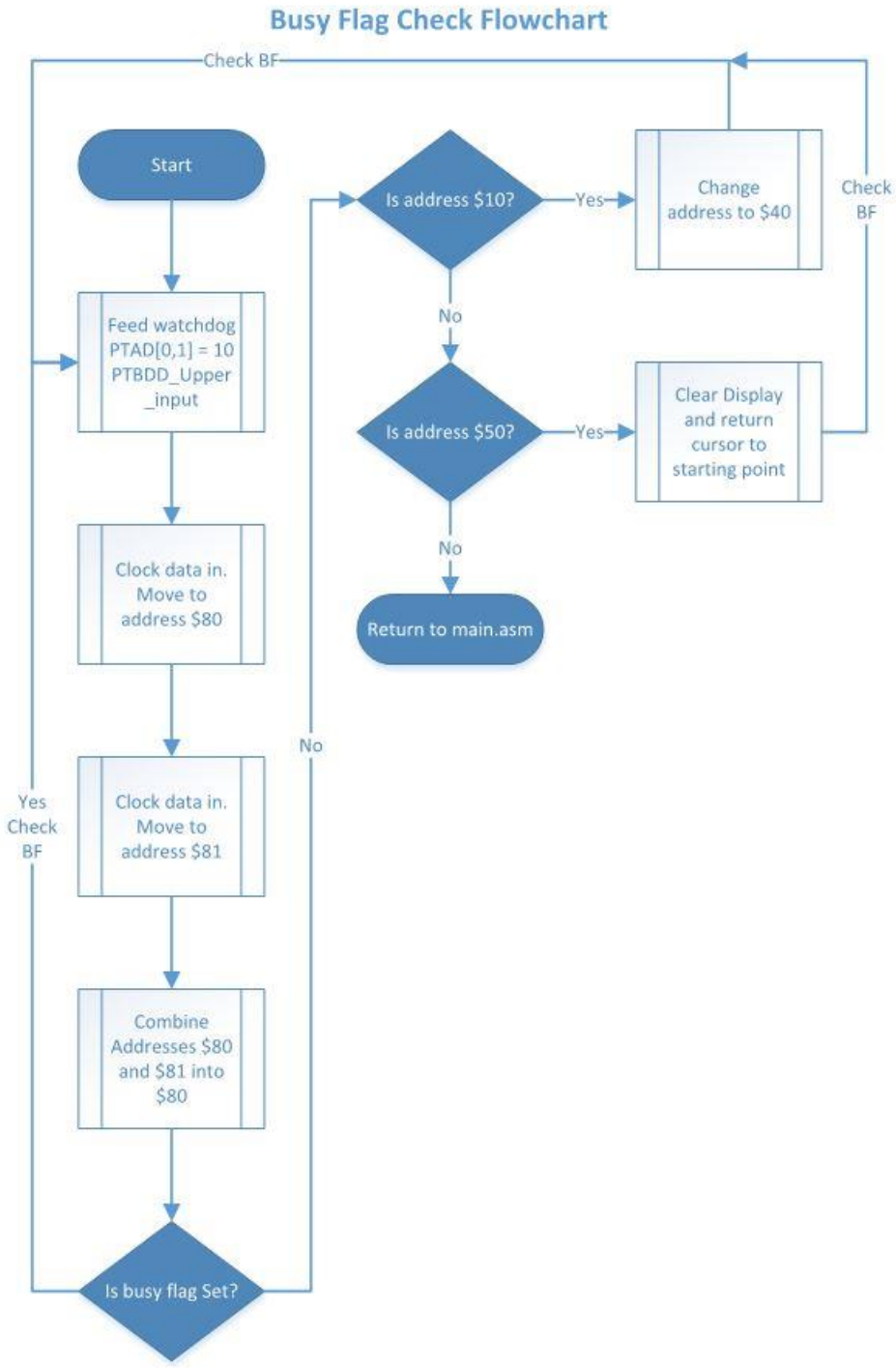


Main.asm Flowchart



Delay.asm


Start
Move values into variables t, b and m
Move the value of variable b into m
Subroutine top decrements t.
If t does not equal 0
Move variable m into variable b and proceed to subroutine bottom
If t = 0?
Subroutine bottom decrements b
If b = 0?
Otherwise return to bottom
Branch to subroutine ass and RTS


Busy Flag Check Flowchart
Check BF
Start
Feed watchdog
PTAD[0,1] = 10
PTBDD_Upper
_input
Clock data in.
Move to
address $80
Clock data in.
Move to
address $81
Combine
Addresses $80
and $81 into
$80
Is busy flag Set?
Yes
Check
BF
No
Is address $10?
Yes
Change
address to $40
Check
BF
No
Is address $50?
Yes
Clear Display
and return
cursor to
starting point
No
Return to main.asm