Memo to: Randy Larimer
From: Erik Andersen
Date: May 5, 2015
Regarding: EELE 465, Lab6 Project – Digital Temperature Sensor with I2C Serial Two-Wire Interface

Summary:

The purpose of this lab is to read the temperature from an LM92 digital sensor via I^2C communication in order to control the temperature of an aluminum block using the MC9S08QG8 microprocessor and a TEC controller as well as a breadboard with a custom built circuit. An LED on the LED array will simply act as a heartbeat LED blinking on once every second as long as there is power to the microprocessor.

Preliminary Solutions:

First, the project requirements indicate two solutions "Mode A" and "Mode B" must be developed in order to meet the requirements of the project. "Mode A" involves reading the temperature from an LM92 digital temperature sensor via I^2C communication and then displaying the latest value on the LCD display. Also read the time from the DS1337 real-time clock and display the elapsed seconds on the LCD. Control the TE cooler with one of three possible single-digit commands entered from the keypad. Display the TE cooler status on the LCD. Display the LM92 temperature in Kelvin and time in seconds. Further "Mode B" involves entering a set point temperature T_{set} and controlling the TE cooler until the top aluminum block equals the set point temperature and then holding that temperature by heating and cooling the aluminum block as appropriate. Read the temperature from an LM92 digital temperature sensor via I^2C communication and then display the latest value on the LCD display. Also, read the time from the DS1337 real-time clock and display the elapsed seconds on the LCD. Display the LM92 temperature in Celsius and time in seconds for "Mode B".

Second, according to the user manual of the LCD on power up of the microprocessor the LCD has to always be properly initialized. The LCD was initialized the same way as it was in the last project.

Third, 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 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 DS1337 real-time clock, a crystal, three resistors and DEMO9S08QG8 onto the breadboard. The breadboard circuit contains three d-flip-flop packages, one bus transceiver package, one DS1337 real-time clock and one de-multiplexer package as well as an LCD display, potentiometer and some capacitors in order to make the LCD display visible as well as three resistors to properly enable I^2C serial bus communication with the DS1337 real-time clock. The crystal that keeps track of time was soldered to a 2-pin header in order to be attached to the breadboard and wired to the DS1337 real-time clock. 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 LM92 digital temperature is part of the TEC controller box and was wired to the breadboard appropriately with two resistors to properly enable I^2C serial bus communication. Two, RFD14N05L MOSFET's and two 1N5817 feedback diodes were used to interface the d-flip-flops correctly with the TEC controller heat relay and TEC controller cool relay. 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. Additionally, the DS1337 real-time clock and LM92 digital temperature sensor is controlled directly from Port A2 (data line) and Port A3 (Clock Line) of the MC9S08QG8 microprocessor. 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 subroutine LCD write which displays the number pressed onto the LCD display. Then, branch to subroutine write to memory in order to store the value pressed to memory in hexadecimal format.

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 – Mode A

First, "Mode A" is solved by getting the temperature value from the read-only temperature register of the LM92 using I^2C as described in the previous memo report every 2 seconds. Next, display the latest value on the bottom line of the LCD along with the elapsed time since the TEC controller state (heat or cool) was entered. The display format for Mode A is as follows: the top line displays "TEC state: <sp> <sp> <sp> <sp> <sp>

<sp> ” where <sp> indicates a space. The bottom line of the LCD reads: “T92: <sp> <sp> <sp> K@T=<sp> <sp> <sp> s”.

The project is designed to first prompt the user (using the LCD display) to choose either “Mode A” or “Mode B” in this case “Mode A” was chosen and solved as described below in the “Mode A” flowchart. Once Mode A is selected using the keypad (pressing button “A” on the keypad) the main code branches to the mode_a subroutine where first the real time clock is reset. Then, zeros are moved onto the bus and clock into the d-flip-flop. Next, we check which key was pressed. If a 1 is pressed on the keypad the heat bit is moved onto the bus and clocked into the d-flip-flop. If a 0 is pressed on the keypad the zeros are moved onto the bus and clocked into the d-flip-flop. If a 2 is pressed on the keypad the cool bit is moved onto the bus and clocked into the d-flip-flop. If a “*” is pressed we return to the main.asm file and ask the user to choose either “Mode A” or “Mode B”. After, a 0, 1 or 2 is pressed and what is described above occurs we check whether the last state was different. If the last state is different we reset the real time clock and display the TEC state, temperature and time as described above for the “Mode A” display format. If the last state is not different we just display the TEC state, temperature and time as described above for the “Mode A” display format. Basically, “Mode A” allows the user to manually control the temperature of the aluminum block.

Solution – Mode B

First, when “Mode B” is selected the top line of the LCD display reads “Target Temp?” and the bottom line of the LCD reads “Enter 10-40C” This prompts the use to enter the set temperature followed by a “#” to indicate the end of the entry. A “*” entry causes a return to the main menu at any time. After entering the set temperature the mode_b subroutine begins to drive the actual temperature as measured from the LM92 to the set temperature and the LCD then reads the following: top line “TEC state:HeatXX” or “TEC state:CoolXX” where XX is the set temperature (the target temperature for the aluminum block) bottom line: “T92: <sp> <sp> <sp> C@T=<sp> <sp> <sp> s” The time either heating or cooling is then displayed in seconds. Once the set temperature is reached the following is displayed on the LCD and we either continue to heat or cool to hold the desired actual temperature. Display “TEC state:HoldXX” where XX is the set temperature (the target temperature for the aluminum block) on the top line when the target temperature, XX, is attained. Bottom line: “T92:<sp> <sp> <sp> C@T=<sp> <sp> <sp> s” The time holding is displayed in seconds.

The project is designed to first prompt the user (using the LCD display) to choose either “Mode A” or “Mode B” in this case “Mode B” was chosen and solved as described below in the “Mode B” flowchart. Once Mode B is selected using the keypad (pressing button “B” on the keypad) the main code branches to the mode_b subroutine where first the temperature is received from the keypad. Then, that temperature is stored into memory in Binary Coded Decimal (BCD). Next, the real time clock is reset. Then, we read from the LM92 digital temperature sensor as well as read from the real time clock. Now, we compare what the LM92 reads to what is stored in memory. If what is stored in memory is greater than what the LM92 reads than the cool bit is moved onto the bus and clocked onto the d-flip-flop. If what is stored in memory is less than what the LM92 reads than the heat bit is moved onto the bus and clocked onto the d-flip-flop. If what is stored in memory is the same as what the LM92 reads than the zeros are moved onto the bus and clocked into the d-flip-flop. After, what is described above occurs we check whether the last state was different. If the last state is different we reset the real time clock and display the TEC state, set temperature, actual temperature and time as described above for the “Mode B” display format. If the last state is not different we just display the TEC state, set temperature and actual time as described above for the “Mode B” display format. Basically, Mode B allows the user to set a temperature that the aluminum block will go to and then maintain that temperature by either heating, cooling or doing nothing to the block.

Solution – Write to LCD and LED’s

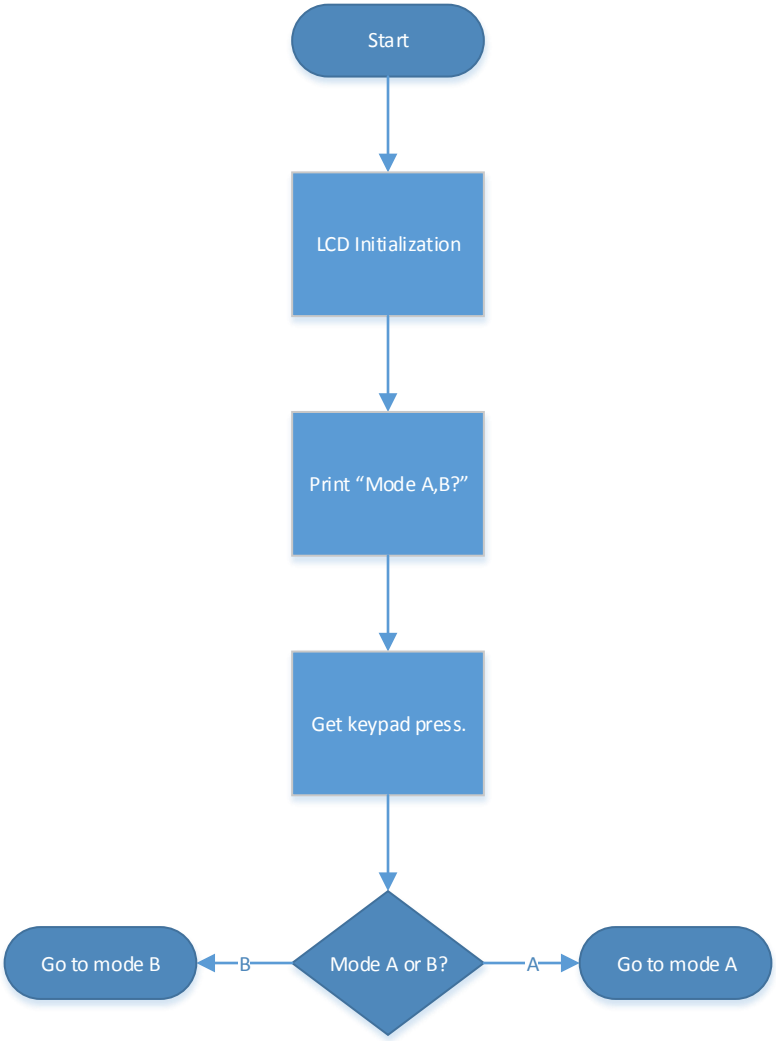
First, set PTADD to an output by setting bits 0 and 1 and then set bit 1 of PTAD and clear bit 0 of PTAD. Then, move the correct value into PTBD (upper four bits) to select the column then bring the clock back up by setting bit 3 of PTBD. Next, move the correct value into PTBD (lower four bits) to select 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 are written to the LCD based on the two values stored in memory which is the seconds in this

case. The characters described in the previous sections are all written to the LCD display with the method described above. The time is then read from the DS1337 real time clock and is written to the LCD display the same way that is described above. Lastly, jump to the busy flag check subroutine and clear address seventy and return to main.

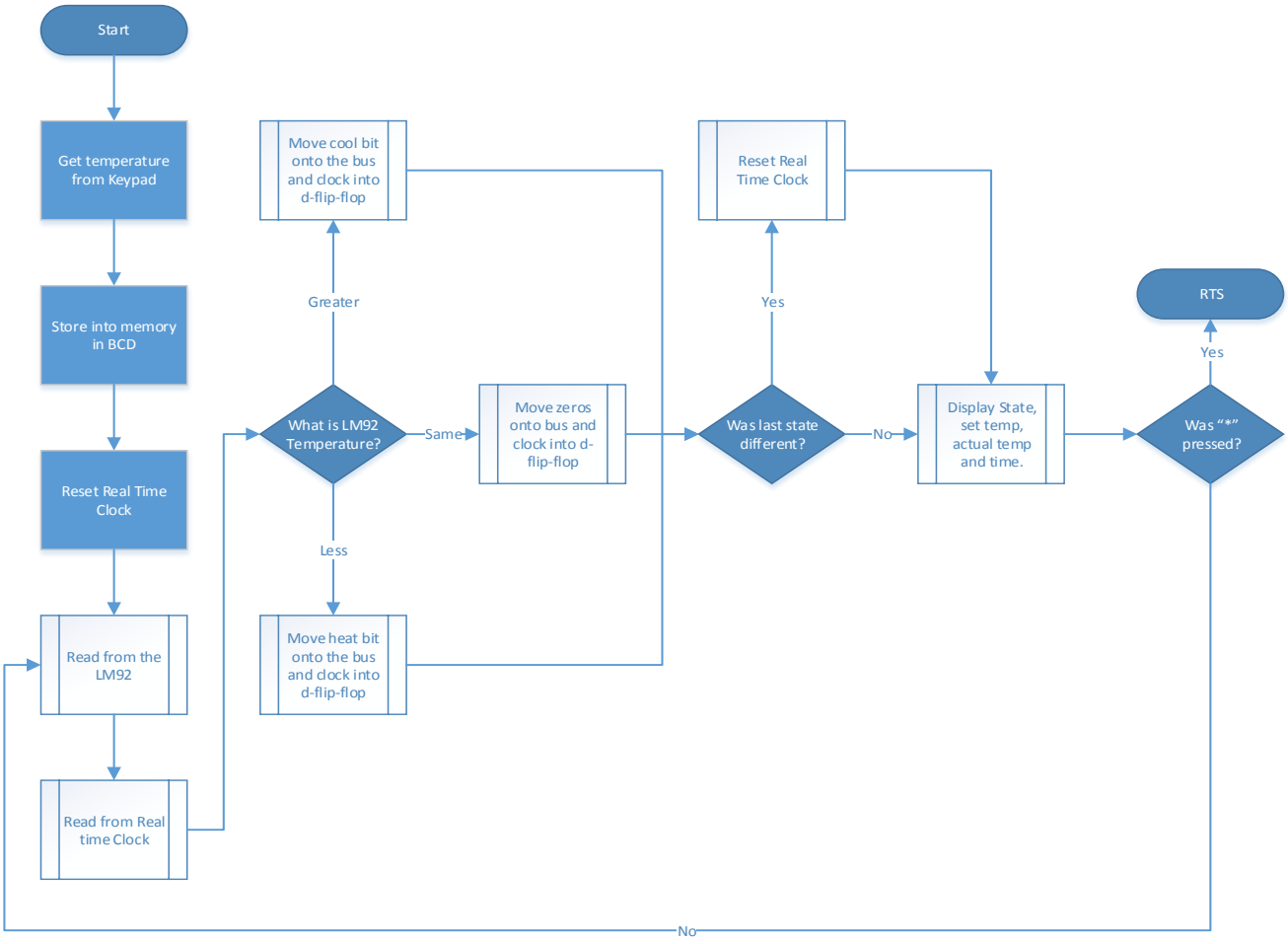
Final Comments:

While performing this lab, many problems and errors in the code were encountered. Some took multiple hours to troubleshoot and debug. One problem with "Mode A" that was never fixed was the fact that the LM92 always displayed a temperature of 273 degrees Kelvin no matter what. Now, by entering 0, 1 or 2 we were definitely able to get the aluminum block to heat up or cool down or just turn off the TEC controller, but the temperature was just never displayed correctly.

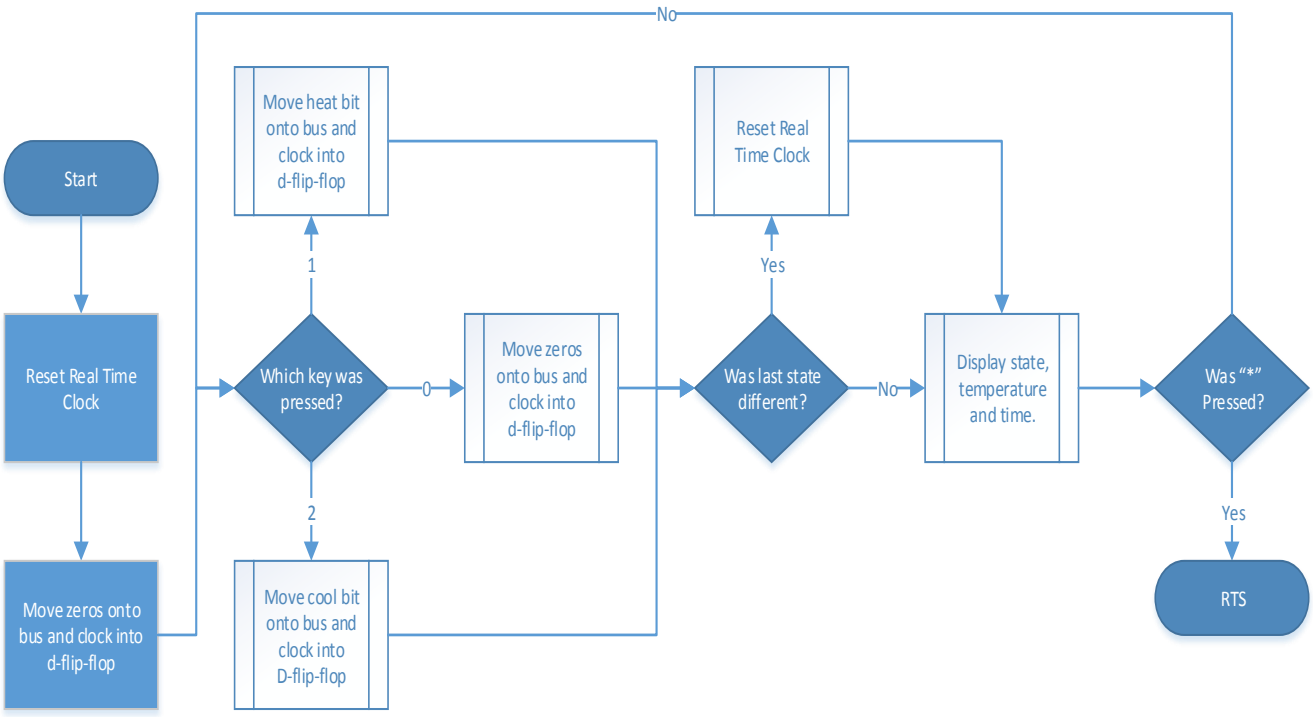
Flowchart for EELE 465, Lab 6, TEC controller



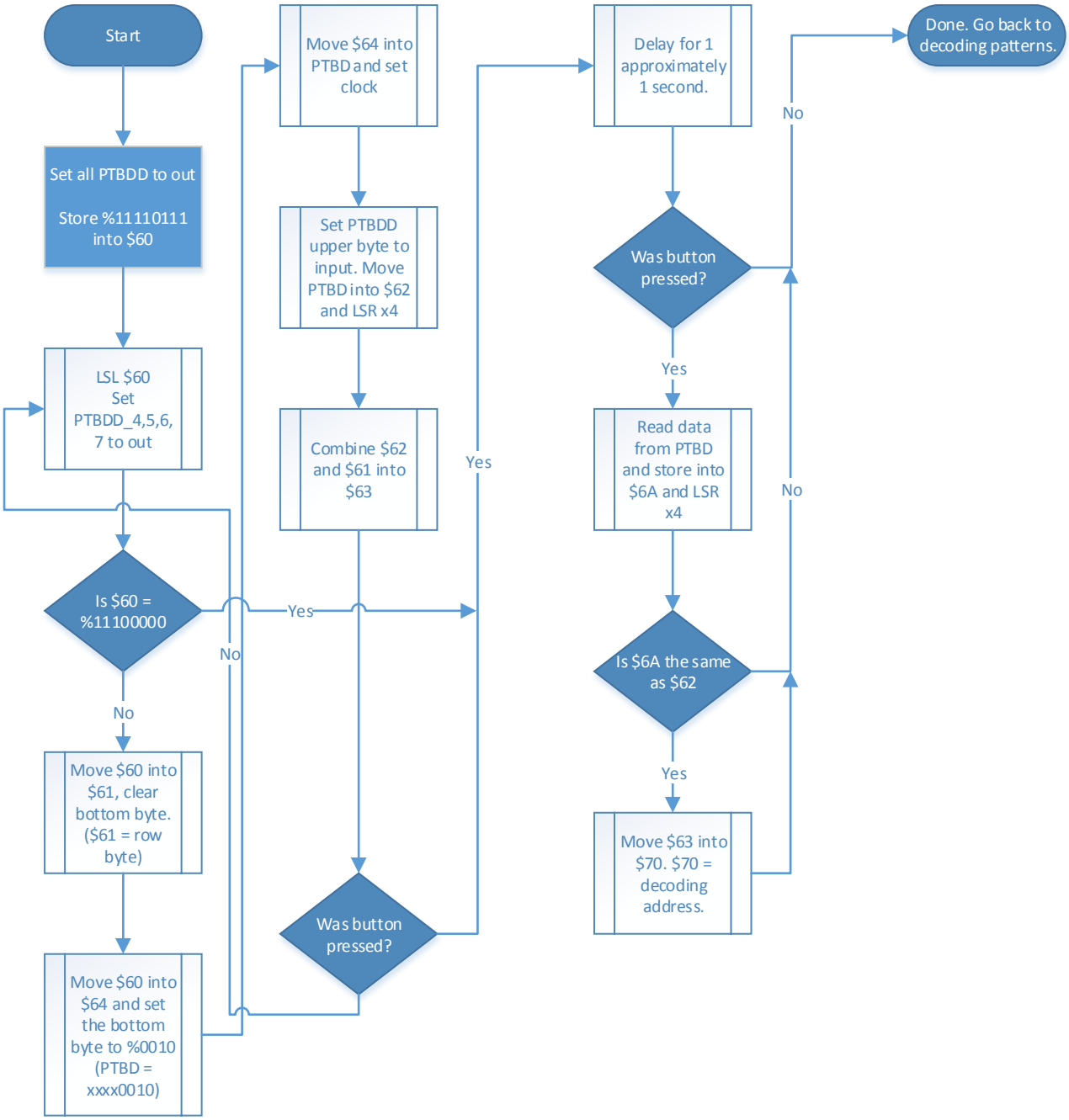
Flowchart for Mode B

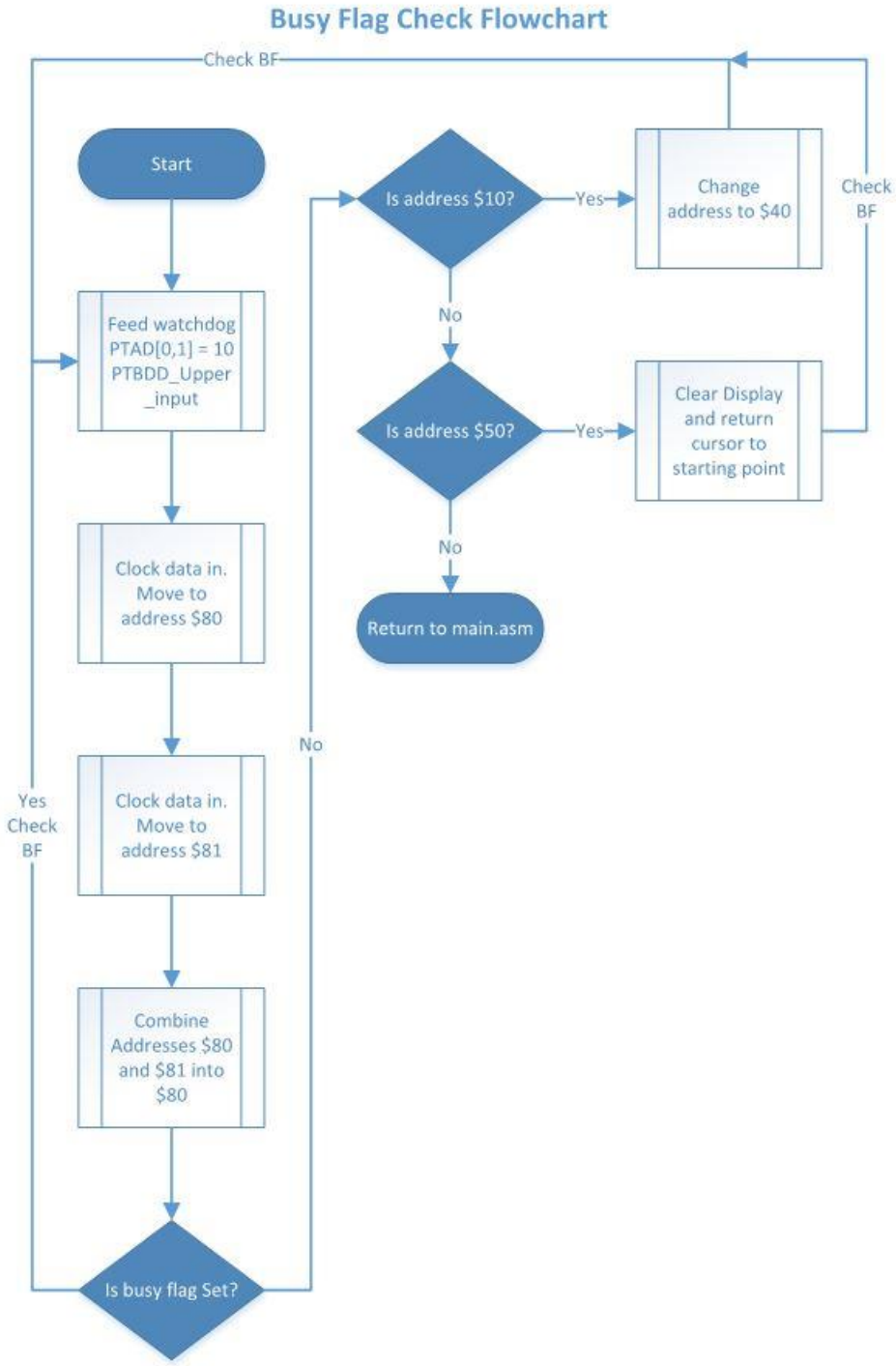


Flowchart for Mode A



Reading from Keypad Flowchart




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