Memo to: Randy Larimer
From: Erik Andersen
Date: April 24, 2015
Regarding: EELE 465, Lab5 Project – Real-Time Clock with I2C Serial Two-Wire Interface

Summary:

The purpose of this lab is to read the time and date from a DS1337 real-time clock in order to display the latest values of the time and date on the LCD display using the MC9S08QG8 microprocessor and 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 sixteen position keypad will be used to enter a value 1 through 9 for the time which includes the number of hours, minutes and seconds and the date which includes the month, day of the month and year. These six different values will be stored in memory. Then using the I^2C serial bus transfer protocol the six values mentioned previously will by written to the DS1337 real-time clock. Next, the current value of the date and time will be continuously read from the DS1337 real-time clock using the I^2C serial bus transfer protocol. The time and date read from the DS1337 real-time clock will be continuously displayed on the LCD. Thus, transforming the LCD into a clock that also display the date.

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 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 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 – Write to and read from the DS1337 real time clock using I^2C

First, enable Port A2 as an output (set bits one and two) so it can be used to write data to the DS1337 real time clock and clear all information stored in Port A. Then, set the data direction for Port A3 as an output as well so it can be used as a clock line in order to communicate with the DS1337 real time clock using I^2C serial bus communication protocol. Then, the characters "Enter MM/DD/YY:" are written to the first line of the LCD display as described in the next section. Now, the heartbeat LED is toggled with an XOR mask at the rising edge of a clock. Next, after the month, day and year are entered and stored into memory as hex values as well as displayed on the LCD display the LCD display is cleared and the characters "Enter hh/mm/ss:" are written to the first line of the LCD display and the values entered for the hours, minutes and seconds of the clock are stored into memory as hex values as well as displayed onto the LCD display and then the display is cleared.

Second, every time a value is entered and stored into memory the X register is incremented by one and once it reaches seven the date information or time information has been entered and we branch to the i2c_write information subroutine. The i2c_write subroutine first moves the date and time information into the correct order in memory to be sent to the DS1337 real-time clock so that the date and time information is stored into the

correct registers on the DS1337 real-time clock. Then, through the implementation of the bit banging method which simply sends the correct conditions in the proper order by setting the clock line low or high and the data line low or high to either transfer or not transfer data in the i2c_write subroutine.

Next, the start condition is sent first followed by the slave address of in this case "1101000". The R/W bit is set to a 0 to indicate a write. Then, an acknowledge is received. Next, a word address of "00000000" is sent. Then, another acknowledge is received. Next, the date and time information are sent as long as an acknowledge is received between each sending of the six 8 bits of data. Once the date and time has been sent to the DS1337 real-time clock a stop condition is sent to signify no more data is to be sent to the DS1337. Lastly, after the stop condition we branch to the read subroutine which continuously reads from the DS1337 real-time clock in order to display the correct time on the LCD display.

Third, the read subroutine is implemented using the same method as described above, bit banging. The read subroutine is designed by first sending a start condition followed by the slave address of in this case "11010000". The R/W bit is set to a 0 to indicate a write. Then, an acknowledge is received. Next, a word address of "00000000" is sent. Then, another acknowledge is received. Next, the start condition is sent again followed by the same slave address as before. The R/W bit is set to a 1 to indicate a write. Then, an acknowledge is received. Next, the date and time information are read as long as an acknowledge is received between each reading of the six 8 bits of data. Once the date and time has been read from to the DS1337 real-time clock a not acknowledge condition and then a stop condition are sent to signify no more data is to be read from the DS1337. Lastly, we jump to the display subroutine and once we return from that we read from the DS1337 real time clock again. What happens in the above paragraph happens every second.

Fourth, the display subroutine properly formats the data being read from the DS1337 and then writes the characters "Date is:" using the method described in the section below. Followed by the date read from the DS1337 real time clock in the format MM/DD/YY all on the first line of the LCD display. Then, jump to the second line and the characters "Time is:" using the method described in the section below are written to the LCD display. Followed by the time read from the DS1337 real time clock in the format hh/mm/ss all on the second line of the LCD display. Lastly, we return to the read subroutine to read the next second in time from the DS1337 real time clock.

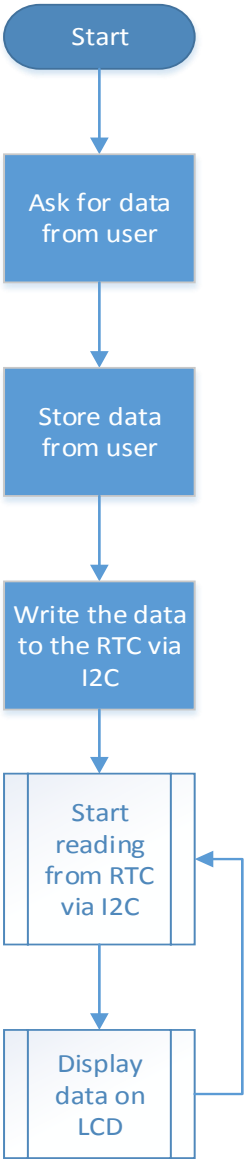
Solution – Write to LCD and LED's

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, hours minutes, month, day of the month and year entered by the user. The characters "E", "n", "t", "e", "r" " ", "M", "M", "/", "D", "D", "/", "Y", "Y" ":" are also written to the LCD display on power up with the same method as above. After, those values are entered the characters "E", "n", "t", "e", "r" " ", "h", "h", "/", "m", "m", "/", "s", "s" ": are also written to the LCD display with the same method as described above. The date and time are then read from the DS1337 real time clock and are also 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.

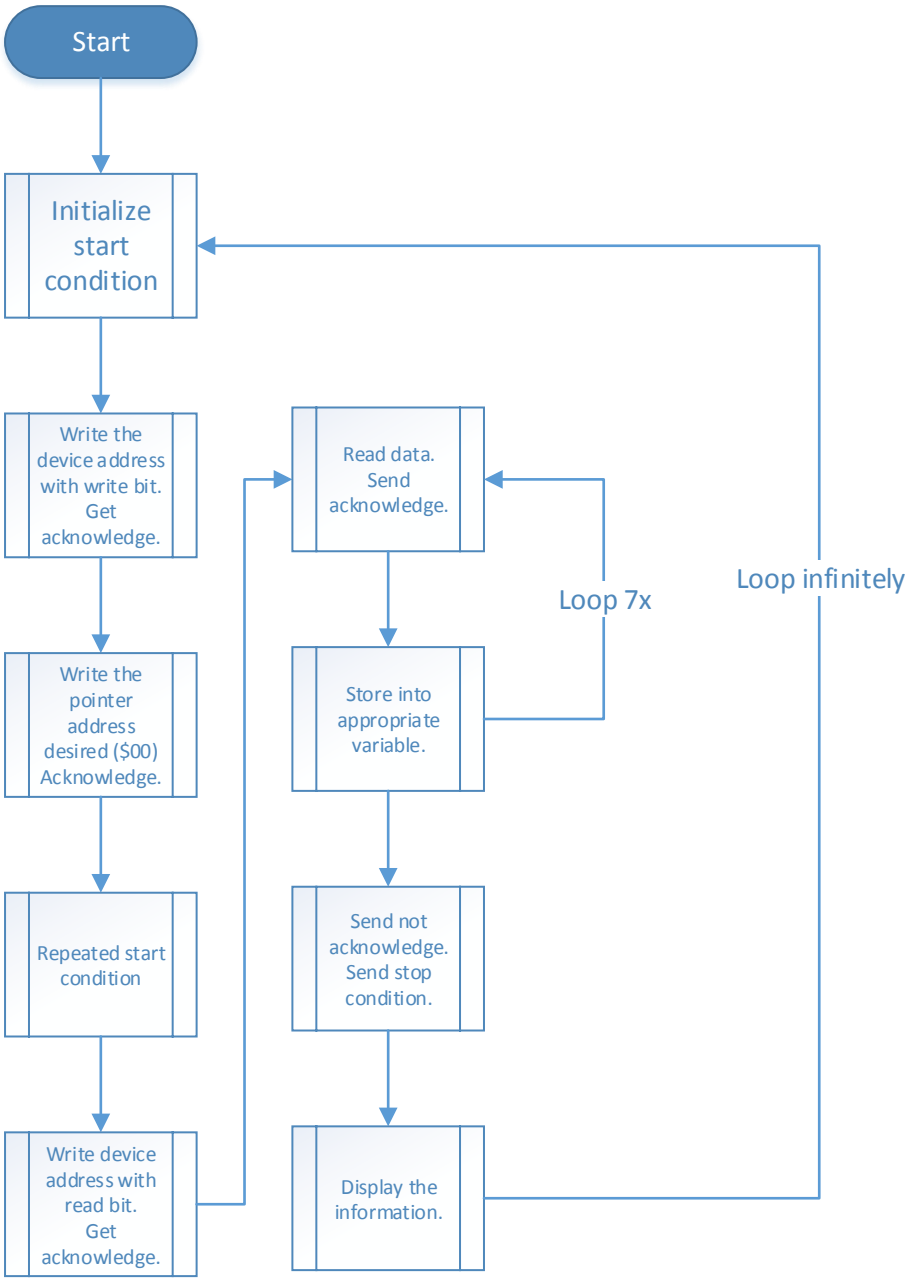
Final Comments:

While performing this lab, many problems and errors in the code were encountered. Some took multiple hours to troubleshoot and debug. The most bothersome and frustrating was figuring out how to read from the DS1337 correctly with our implementation of I^2C. I^2C was implemented in this project with the bit banging approach which gave the programmer direct control over whether the data line or clock line was high or low. For us this made it challenging to read from the DS1337 correctly using this approach.

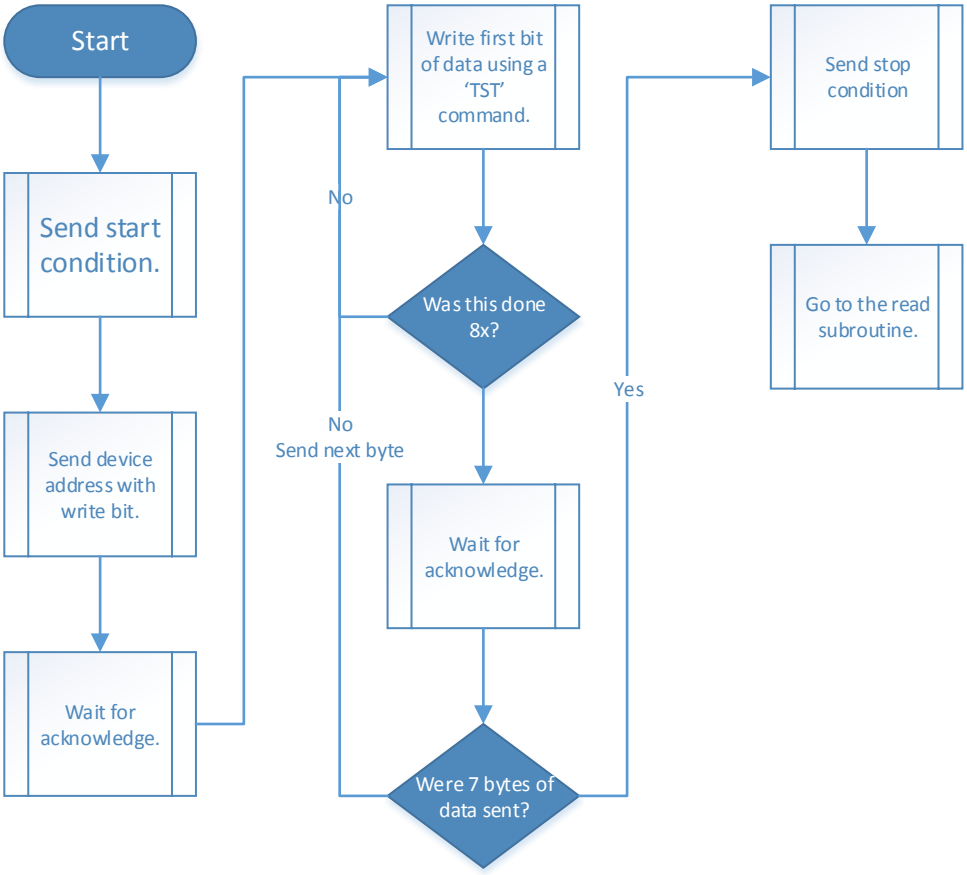
Overview of Lab 5, Real Time Clock (RTC) with I2C



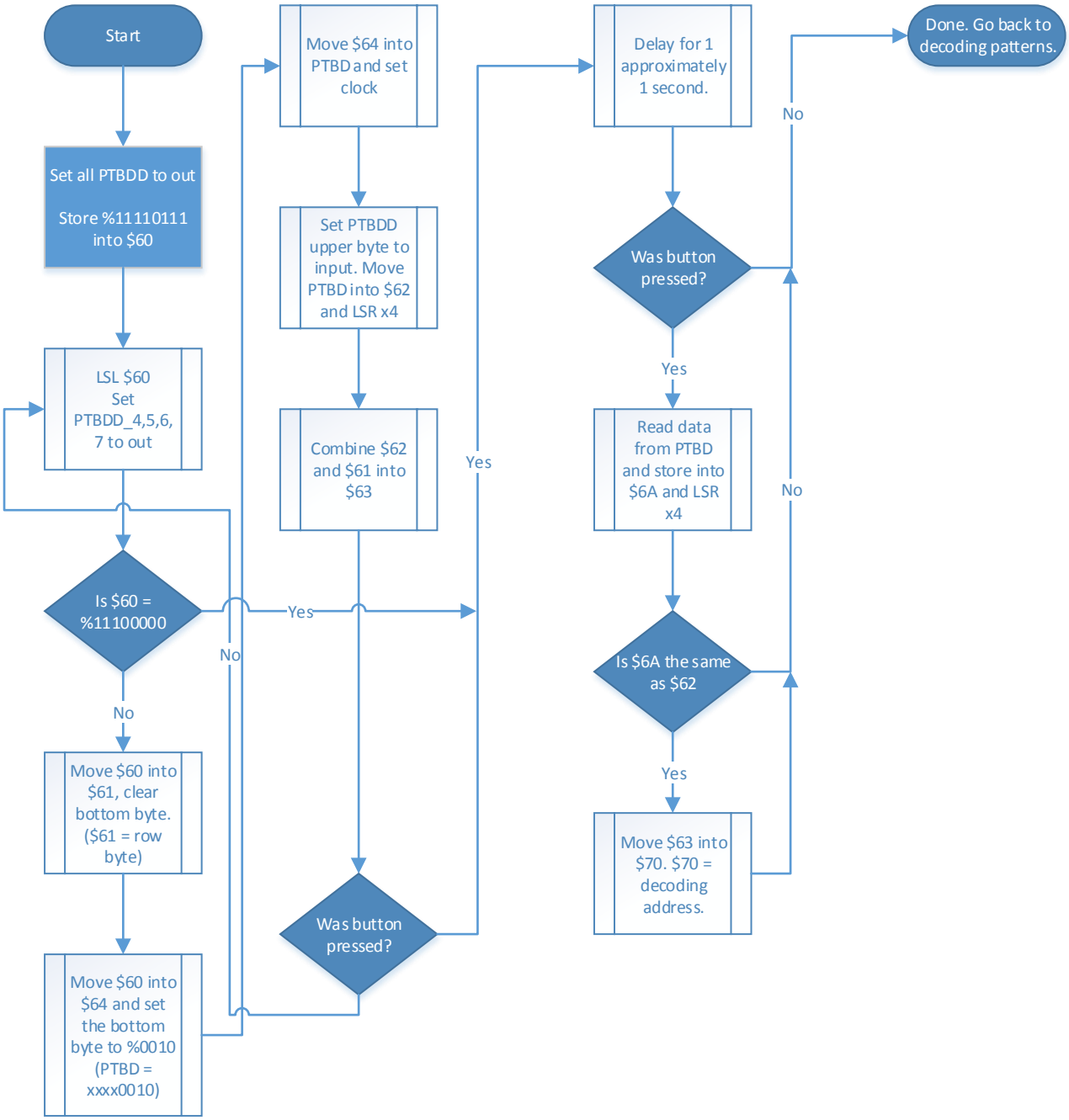
I2C reading protocol flowchart

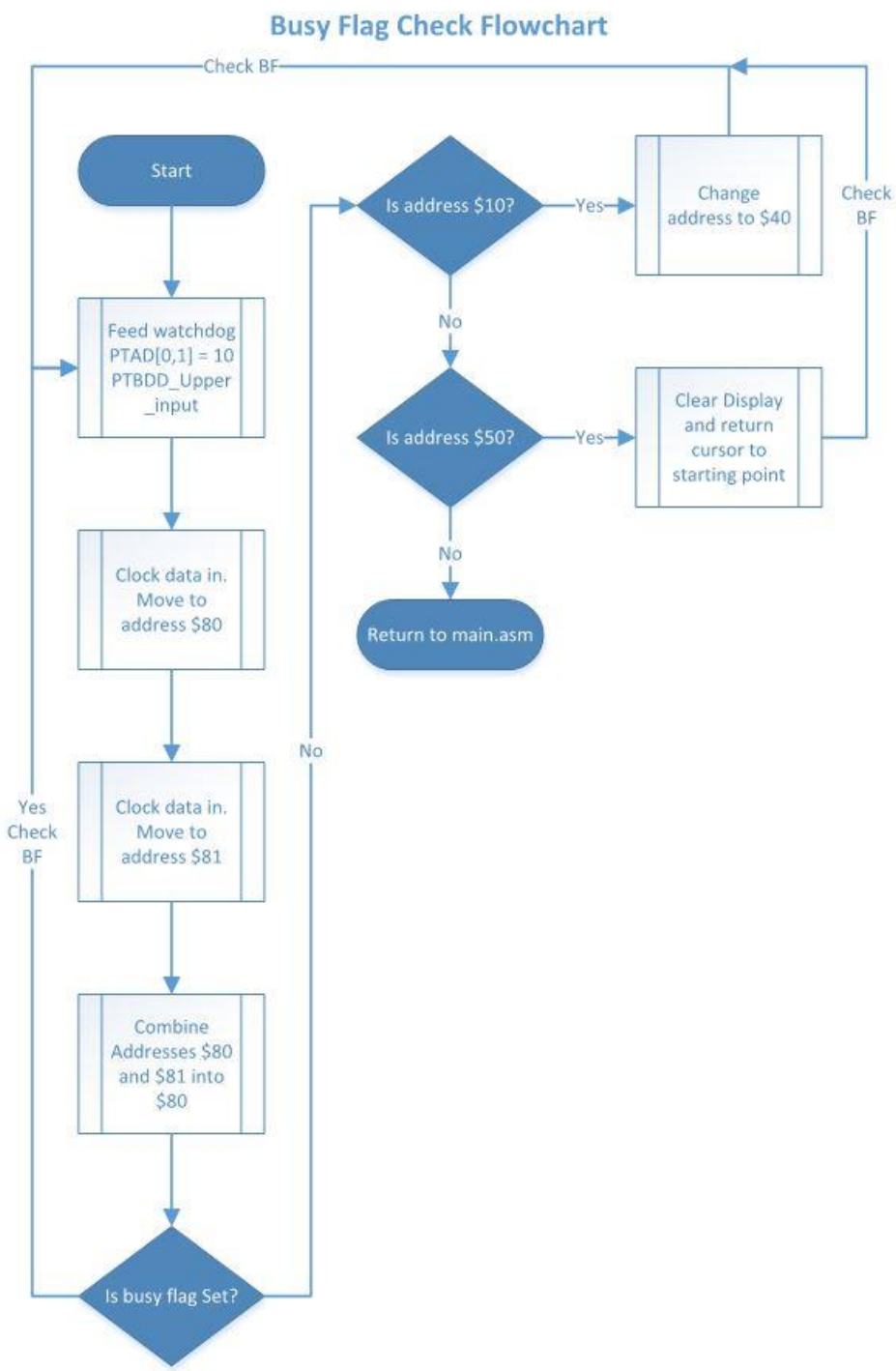


I2C writing protocol flowchart



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