Homework 9: Software Design Considerations

Team Code Name: RevGeo Multipurpose Puzzle Box **Group No. 11**

Team Member Completing This Homework: Joshua Marchi

E-mail Address of Team Member: jmarchi@purdue.edu

Evaluation:

SCORE	DESCRIPTION
10	***Excellent*** *– among the best papers submitted for this assignment. Very few corrections needed for version submitted in Final Report.*
9	***Very good*** *– all requirements aptly met. Minor additions/corrections needed for version submitted in Final Report.*
8	***Good*** *– all requirements considered and addressed. Several noteworthy additions/corrections needed for version submitted in Final Report.*
7	***Average*** *– all requirements basically met, but some revisions in content should be made for the version submitted in the Final Report.*
6	***Marginal*** *– all requirements met at a nominal level. Significant revisions in content should be made for the version submitted in the Final Report.*
*	***Below the passing threshold*** *– major revisions required to meet report requirements at a nominal level.* ***Revise and resubmit.***

** Resubmissions are due within **one week** of the date of return, and will be awarded a score of "6" provided all report requirements have been met at a nominal level.*

Comments:

1.0 Introduction

The RevGeo, or "Reverse Geocache", is a multipurpose puzzle box to be used for recreational purposes. Whereas normal geocaching involves searching for hidden objects at specific GPS locations, the RevGeo will consist of a sealed, locked box given to a user, and will only open after one or more predetermined GPS locations have been visited. The RevGeo box will direct the user to the next location by means of GPS coordinates, a compass heading, or text description displayed on an LCD screen on the outside of the box. The GPS coordinate route data will be stored on a MicroSD card located inside the box, and will be able to be taken out and reprogrammed by a PC, ensuring a high replay value for the device. Lastly, an RFID "master key" interface will be developed, providing a "backdoor" method to unlock the box without finishing the current puzzle route.

At the heart of the design is the PIC24FJ256GA106 microcontroller which is used to interface with several different peripherals and sensors. Two UART channels will be used to receive data from the GPS and RFID receivers, and a third UART channel will interface with a UART to USB bridge, allowing messages to be sent to a PC for debugging purposes. Two I2C channels will be used to interface with the digital compass peripheral and battery fuel gauge, and one SPI channel will interface with the MicroSD card. The LCD screen on the outside of the box will be controlled by 14 GPIO pins (8 data and 6 control signals), and the servo and buzzer will each be controlled by a PWM signal. The application code organization scheme for this project consists of an interrupt/polling loop hybrid, where the main code loop performs data manipulation and analysis in response to flags set by interrupt service routines.

2.0 Software Design Considerations

When developing software for the RevGeo box, several design aspects need to be considered. These include managing the limited amounts of Flash and SRAM available for program code and data, as well as managing the number of I/O pins provided by the chip. In addition, the software needs to ensure proper initialization of all peripheral modules used in the design. Finally, the application software must be organized in a way that allows the device to operate and interact in a real-time manner. These considerations will be discussed in the following subsections.

2.1 Memory Mapping (See Appendix A for diagrams)

The PIC24FJ256GA106 utilizes a 24-bit program memory address space, divided into word addressable blocks (16 bits) [1]. The User Flash Program Memory spans from word address 0x000200 to 0x02ABFF. Program instructions on the PIC24FJ256GA106 have a width of 24 bits (3 bytes) each, and therefore every fourth byte in the User Flash Program Memory section is unused to keep the instructions word-aligned. This results in 256KB worth of Flash available for program instructions, the equivalent of 87K 3-byte instructions. Constant data can also be stored in User Flash Program Memory, accessible by mapping the upper half of data memory space to a corresponding block in program memory.

The PIC24F series has a separate 16-bit wide data memory space, divided into byte addressable blocks (8 bits) [1]. Byte addresses 0x0000 to 0x07FF are reserved for Special Function Registers, which are used by the PIC24F core and peripherals for controlling the operation of the device. Byte addresses 0x0800 to 0x47FF correspond to Data RAM, resulting in 16KB of data memory available for the user application (local data, stack, heap, etc.). Byte addresses 0x8000 to 0xFFFF, a section known as the Program Space Visibility Area, can be used to map a 32KB page of program memory to the data memory space, allowing for easy access to constant data that may be stored in program memory.

As software is developed for the PIC24FJ256GA106 microcontroller, care needs to be taken to ensure that the application size does not exceed the maximum amount of Flash and RAM available on chip. After some initial code development for the RevGeo box, this does not seem like it will be an issue. The largest module of code at the moment is the MDD File System Library (MicroSD interface) which takes up approximately 25 percent of program memory and

33 percent of data memory. All other code modules each take up only 1-3 percent of available program and data memory. Therefore, plenty of free memory should still be available when all modules are eventually merged together into one application.

2.2 External Interfaces

The PIC24FJ256GA106 provides great flexibility when it comes to I/O pin mapping. Most on-chip peripherals allow their inputs and outputs to be mapped to Reconfigurable Peripheral (RP) pins, where any available RP pin can be assigned to any given peripheral. Taking this into consideration, the RP pins for these peripherals were assigned based on ease of routing for the PCB. The only on-chip peripherals that could not map to RP pins were the I2C modules, which have static pin assignments. Therefore, the pins chosen for the compass and fuel gauge correspond to the static pairs of I2C pins closest to their respective locations on the PCB. The GPIO pins for the LCD data and control were chosen so that all data pins were on the same I/O port, and all control pins were also together on another port. All other GPIO pin assignments (pushbutton, MicroSD interface) were chosen based on ease of routing for the PCB.

Table 1 – External Interface Mapping

Module	Pin Numbers	Pin Name	Pin Functionality
UART1 (PC debug)	45-46	RP12-RP11	Tx, Rx
UART2 (RFID receiver)	52	RP25	Rx
UART3 (GPS receiver)	49-50	RP24-RP23	Tx, Rx
I2C1 (Compass)	2-3	SCL3, SDA3	SCL, SDA
I2C2 (Fuel Gauge)	31-32	SDA2, SCL2	SDA, SCL
SPI1 (MicroSD card)	11-12, 14-16	RB5, RP28, RP13, RP1, RB0	CS, MOSI, SCK, MISO, CD
PWM (Buzzer)	4	RP21	PWM
PWM (Servo)	8	RP27	PWM
GPIO (LCD data)	21-24, 27-30	RB8-RB11, RB12-RB15	DB7-DB4, DB3-DB0
GPIO (LCD control)	60-64, 1	RE0-RE5	E, R/W, D/I, CS1, CS2, RST
GPIO (Pushbutton)	48	RC14	PB_IN

2.3 Integrated Peripheral Initializations

As stated previously, 3 UART channels, 2 I2C channels, and 1 SPI channel will be used in the RevGeo design, as well as GPIO pins for the LCD data and control lines, pushbutton input, and some MicroSD control signals. Initializations for UART1 (PC debug) involve setting the baud rate to 9600 using the U1BRG register, enabling UART1 for 8 bit data, no parity, and 1 stop bit using the U1MODE register, and enabling UART1 transmit in the U1STA register. Similarly, UART2 (RFID receiver) initializes the baud rate to 9600 using U2BRG, and enables UART2 for 8 bit data, no parity, and 1 stop bit using U2MODE. Since an interrupt is to be generated when a character has been received from the RFID receiver, the UART2 Rx interrupt priority is set in the corresponding IPC register, and UART2 Rx interrupts are enabled in the corresponding IEC register. Although code for initializing UART3 (GPS) has not yet been written, it will most likely initialize a baud rate of 38400 (U3BRG), 8 bit data, no parity, 1 stop bit (U3MODE), and set Rx interrupt priority similarly to UART2. The UART1 Rx module and UART3 Tx module, although mapped to pins on the microcontroller, are currently unused in the RevGeo application, but are available to provide extra functionality if needed.

Both I2C modules (compass, fuel gauge) will operate in identical fashion. Both will be initialized to operate in standard mode (100KHz) using the I2CxBRG registers, and enabled for master mode operation with 7-bit device addresses using the I2CxCON registers. The initializations for the SPI module (MicroSD) are handled by the Microchip MDD File System Library, which sets the appropriate values in the SPI1CON register for Master mode and clock polarity, and enables the SPI module with the SPIEN bit in the SPI1STAT register. The last peripherals to be utilized are the PWM channels (servo, buzzer), which will be implemented with two Output Compare modules. While code has not yet been written for these initializations, it is known that the OCxCON1 registers are used to select PWM mode, and the OCxR registers are used to control the duty cycle of the PWM signal. Further research will be required to fully understand initializations for these modules. Lastly, all used I/O pins (GPIO and peripheral) need to be set to either “input” or “output” by using the TRIS register associated with that pin. Setting the corresponding bit to a “1” configures the pin as an input, while clearing the corresponding bit to “0” configures the pin as an output.

2.4 Application Code Organization (See Appendix B for flowchart)

The last main software consideration is the overall organization of the application code. For the RevGeo design, the chosen organization scheme is an interrupt/polling hybrid. In this scheme, interrupts are generated to immediately respond to small events. These interrupts then set flags which cue the main loop to execute much larger sections of code. This organization is beneficial, since it allows for a quick response to some events, such as a character received in an Rx buffer, but also allows a much longer time for other events to compete, such as parsing a received message and taking appropriate action. For example, the RFID receiver sends a message to the microcontroller when an RFID tag is brought within range. An interrupt is generated to copy each received character into a message buffer, and a flag is set once the last character has been received. The main loop then parses this message, extracts the ID, compares it to a list of valid IDs, and unlocks the box and updates the LCD if a match is found.

There are 4 main "flags" to be set by interrupts for the RevGeo application. Two of these correspond to messages received from off-chip peripherals (GPS and RFID), which cause the main loop to parse the message and take appropriate action. The other two flags will be set by timer interrupts, which cue the microcontroller to poll the compass and fuel gauge for data and perform calculations once the data has been obtained. A flowchart of the main loop operation, as well as the initialization sequence for the RevGeo application is included in Appendix B.

3.0 Software Design Narrative (See Appendix C for diagrams)

Since the RevGeo box interfaces with a large number of off-chip devices, the application code is extremely modular. A separate code module will exist for all six serial communication channels (USB debug, RFID, GPS, compass, fuel gauge, MicroSD). In addition, a module will exist for the LCD display to handle LCD initialization and communication. Finally, small code modules will exist for the two PWM channels (servo, buzzer) to control servo position and audio output. The main loop will consist of function calls to the various modules, and the interrupt service routines will be located in their respective modules as well. Details of each of these modules including development status will be discussed in the following subsections. As code development is still in the modular development and testing phase, no true application "main" loop has been developed yet. However, smaller "main" functions that test each module individually are under current development.

3.1 PC Debug Interface (Link: PC Debug Interface)

When developing code for an embedded system, it can be extremely difficult to trace code execution as no screen is readily available for debug "print" statements. To solve this issue, the RevGeo box utilizes the UART1 module on the PIC24FJ256GA106 to output debugging statements to an FT232RL UART to USB bridge. This device converts the UART message to an equivalent USB message, which can then be viewed on a PC through a virtual COM port. The PC debug interface consists of two functions. The first initializes the UART1 interface as described earlier. The second function transmits a formatted string to the PC through the UART1 interface. This function is implemented by using a "printf" call from the <stdio.h> library for the MPLAB C30 compiler, which automatically outputs the formatted message to the UART1 module if UART1 has been properly initialized. Code for the PC debug interface has been written and functionality has been verified for each function.

3.2 RFID Interface (Link: RFID Interface)

The RFID interface consists of three functions. The first function deals with the initializations of the UART2 module as described earlier. The second function is an interrupt service routine that is called each time a character is received in the UART2 Rx buffer. This function copies the character from the Rx buffer to an RFID message buffer, and sets a flag once

the last character in the message has been received. The third and final function is used to check if the ID in the RFID message buffer matches the ID associated with the "master key" for the RevGeo box, returning a "1" on match, and "0" when a match is not found. Code for all three of these functions has been written, and proper execution of each has been verified.

3.3 GPS Interface

The GPS interface will consist of three main functions. The first will initialize the UART3 module as described earlier. The second function is an interrupt service routine very similar to the one in the RFID interface. This routine copies a character received in the UART3 Rx buffer to a GPS message buffer, and sets a flag when an entire message has been received. The final function is used to analyze the message in the buffer to determine the GPS coordinates received and if the coordinates match the desired location within a specific tolerance. If the current location does not match the desired location, a "distance to target" will be calculated and returned to the calling function. None of the code for the GPS interface has been written yet, but it is one of the highest priorities for the near future.

3.4 Compass Interface (Link: Compass Interface Testing, I2C APIs)

The compass interface consists of three main functions. The first contains initializations for the I2C1 module (described earlier) and timer used to initiate data acquisition from the compass device. The second function queries the compass for 3-axis accelerometer and 3-axis magnetometer data and receives the data via the I2C1 module. Lastly, the third function uses the accelerometer and magnetometer data to calculate a compass heading with tilt compensation. The contents of the first two functions have been written and tested, although the code still exists in the compass test "main" file rather than being neatly encapsulated into two functions. The third function has not been written yet, but is a high priority for the near future. To aid in communication via the I2C protocol, APIs to send and receive byte sequences were developed based on code from Robert B. Reese, Bryan A. Jones, and J. W. Bruce of Mississippi State University. [2]

3.5 Fuel Gauge Interface

The fuel gauge interface is extremely similar to the compass interface, consisting of at least two main functions. The first function initializes the I2C2 module and a timer used to initiate data acquisition from the fuel gauge. The second function queries the fuel gauge for battery status, and receives the data via the I2C2 module. Both functions have not been written yet, and are currently at a much lower priority than the GPS and compass, since the fuel gauge is an additional feature that is not necessarily required for device functionality. The fuel gauge interface will use the same APIs to send and received bytes via I2C as used by the compass interface.

3.6 MicroSD Interface (Link: MicroSD File R/W)

The MicroSD interface consists of three main functions. The first function initializes the SPI1 module as well as the FAT16 file system used by the MicroSD card. The second function opens a file on the MicroSD card, and reads the route data from the file. Both of these functions use the Microchip MDD File System Library calls to easily interface with the MicroSD card[3]. The final function verifies the route data read from the file and writes the data to flash memory. The contents of the first two functions have been written and tested, although the code still exists in the MicroSD "main" file rather than being neatly encapsulated into two functions. The third function has not been developed yet, but is a high priority since writing to flash is critical to the functionality of the RevGeo.

3.7 LCD Interface (Link: PIC24 LCD Interface, Generic KS0108 Driver)

The LCD interface consists of an initialization function for the LCD screen, as well as several functions to enable text and shapes to be written to the screen. The initialization function properly configures the GPIO pins used for the screen, and sends an instruction command to the LCD to turn the display on. Other functions in the interface deal with writing character strings, circles, rectangles, lines, and bitmap images to the screen. The code for this interface is based off of the KS0108 driver library developed by Radosław Kwiecień, and was modified to work for the PIC24 microcontroller series[4]. All functions have been successfully modified, and tested for full functionality

3.8 Miscellaneous Interfaces (PWM)

The last two interfaces deal with the PWM channels used to control the servo and buzzer. Each of these interfaces must have an initialization function to configure their respective output compare modules. In addition, the servo interface must have a function to adjust the PWM duty cycle. This will allow the servo arm to change the position, locking and unlocking the RevGeo box. Similarly, the buzzer interface must also have a function to adjust the PWM duty cycle, allowing the buzzer to turn on and off. The code for these interfaces has not been written yet, but should be able to be developed in a relatively short amount of time, due to their simple nature.

4.0 Summary

The RevGeo Multipurpose Puzzle Box intends to provide the user with a fun and engaging recreational activity, ensuring that all internal components interact without conflict. The PIC24FJ256 will manage serial communication with a USB to UART bridge, GPS receiver, RFID receiver, compass module, battery fuel gauge, and MicroSD card. In addition the microcontroller will update an LCD display through GPIO pins, and control a servo locking mechanism and buzzer with PWM signals. The application code will be organized as an interrupt/polling loop hybrid, ensuring that serial messages will be received in a timely manner, and in-depth calculations will have an adequate amount of time to complete. Finally, the application code will be developed in a modular fashion, allowing for easy testing of individual interfaces and simple merging of modules to produce the final build.

List of References

[1] Microchip Technology Inc, "PIC24FJ256GA106," [Online] Available: http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en531071

[2] PIC24 Support Libraries, "pic24_i2c.c," [Online] Available: http://www.ece.msstate.edu/courses/ece3724/main_pic24/docs/pic24__i2c_8c.html

[3] Microchip Technology Inc, " Memory Disk Drive File System for PIC18 PIC24 dsPIC PIC32," [Online] Available: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2680&dDocName=en537999

[4] Radosław Kwiecień, "Universal C Library for KS0108 based LCD displays," [Online] Available: http://en.radzio.dxp.pl/ks0108/

Appendix A: Program Space and Data Space Memory Maps for the PIC24FJ256GA106

Figure 1 – Program Space Memory Map

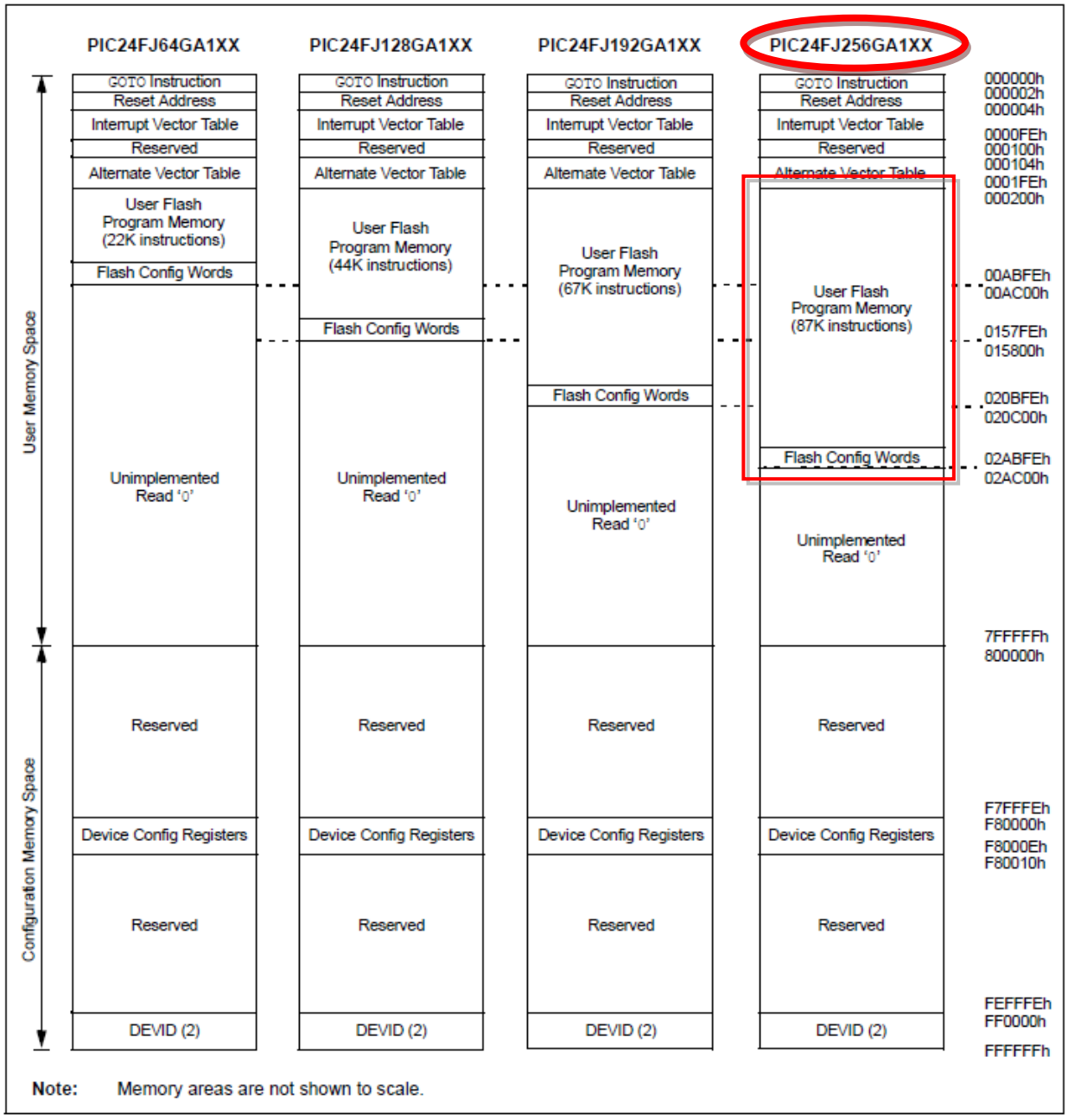
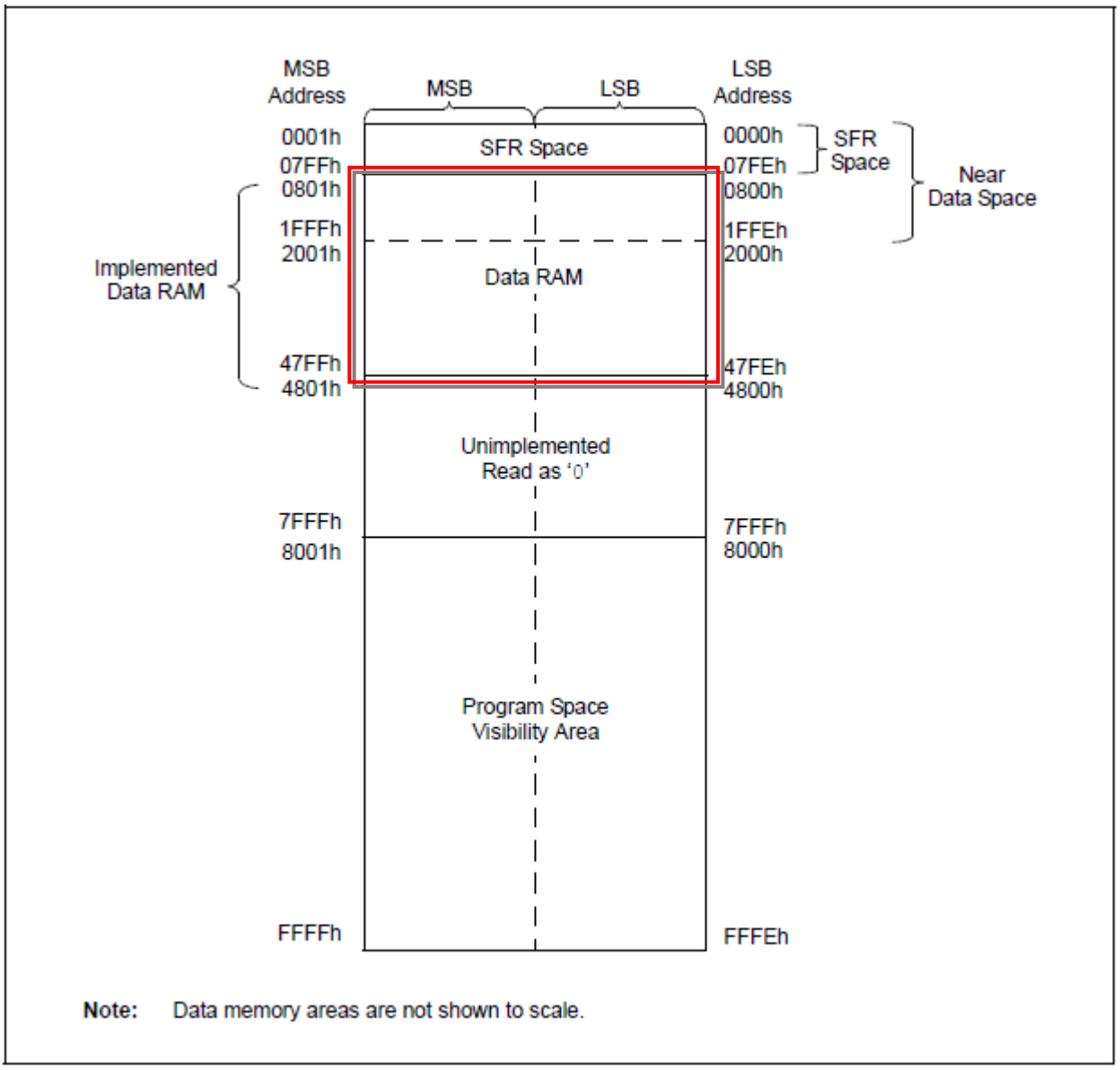


Figure 2 – Data Space Memory Map



Appendix B: Flowchart/Pseudo-code for Main Program

Figure 3 – Initialization Flowchart

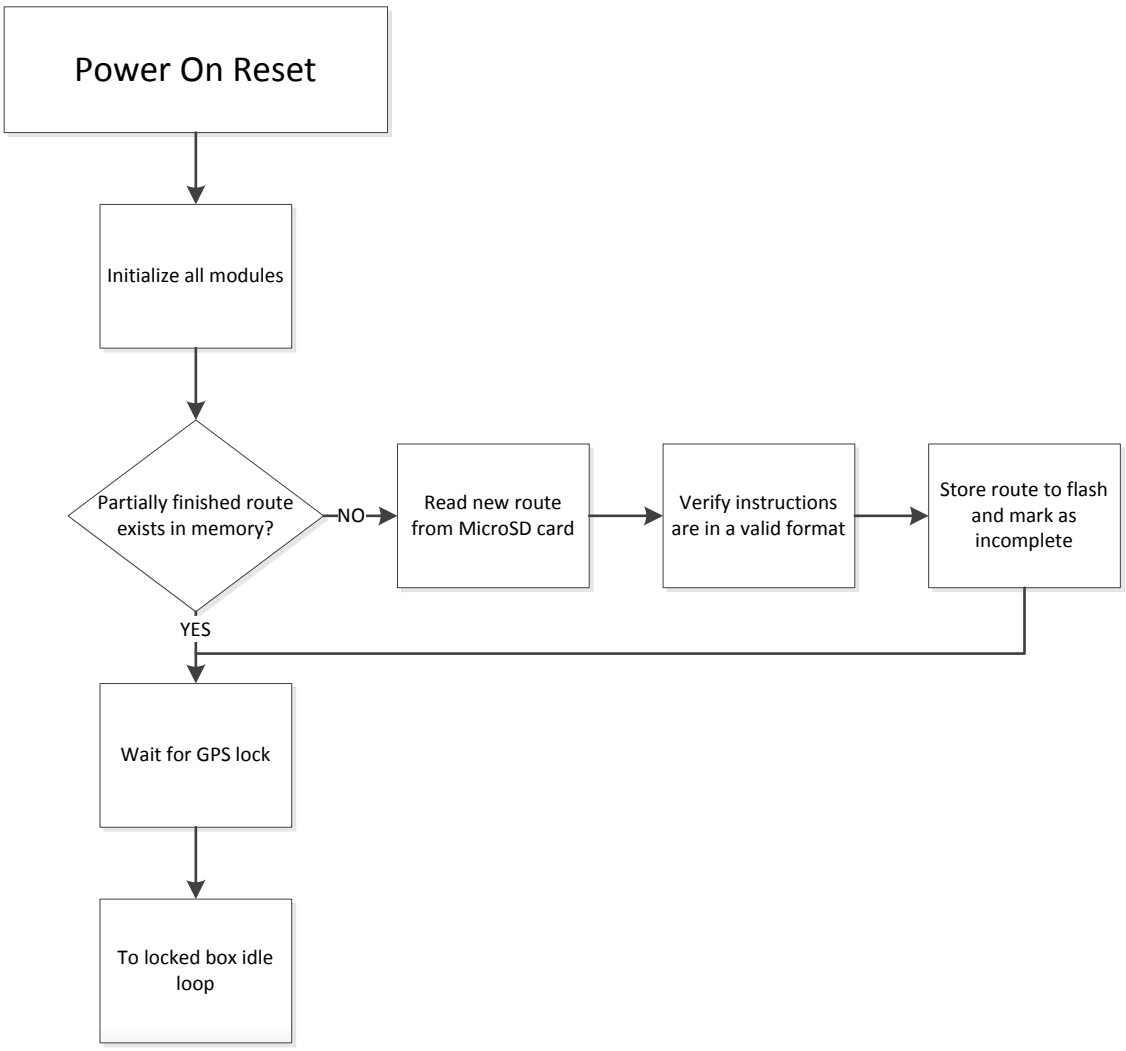
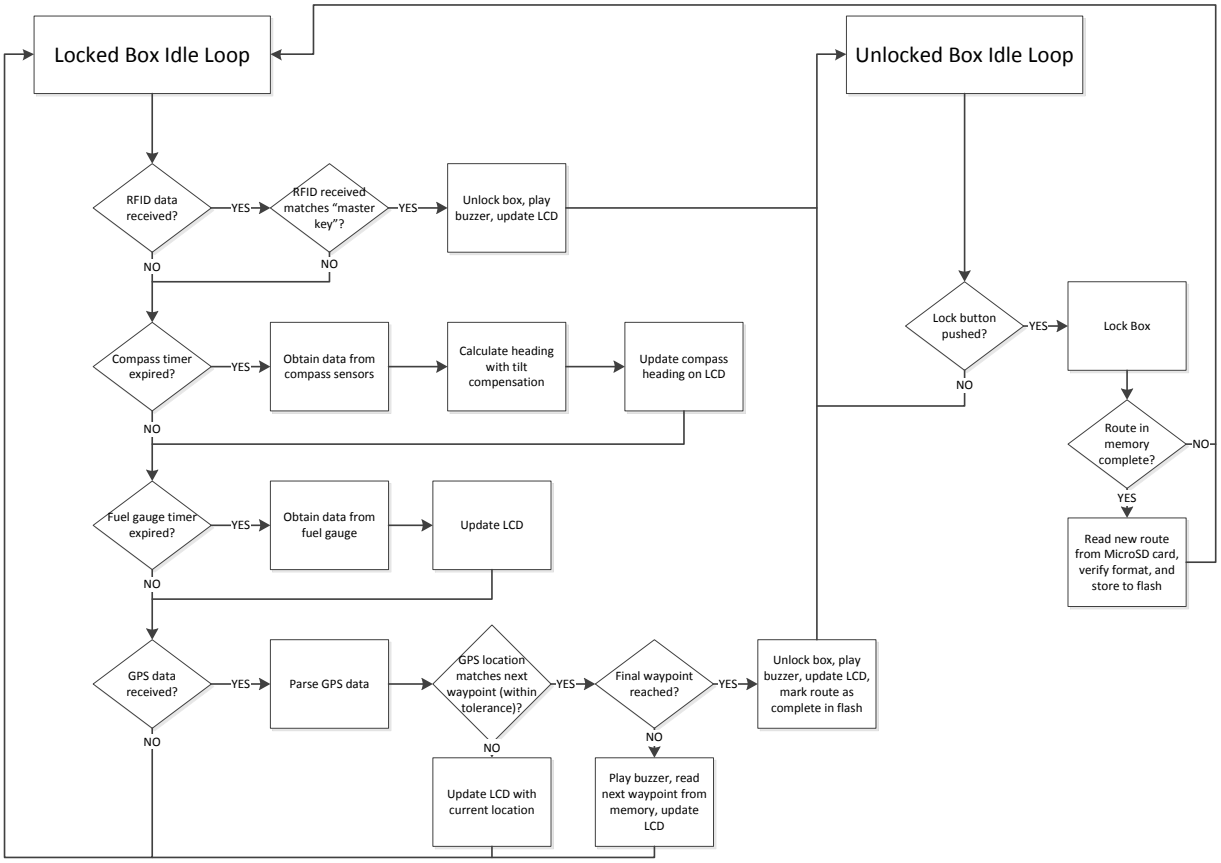
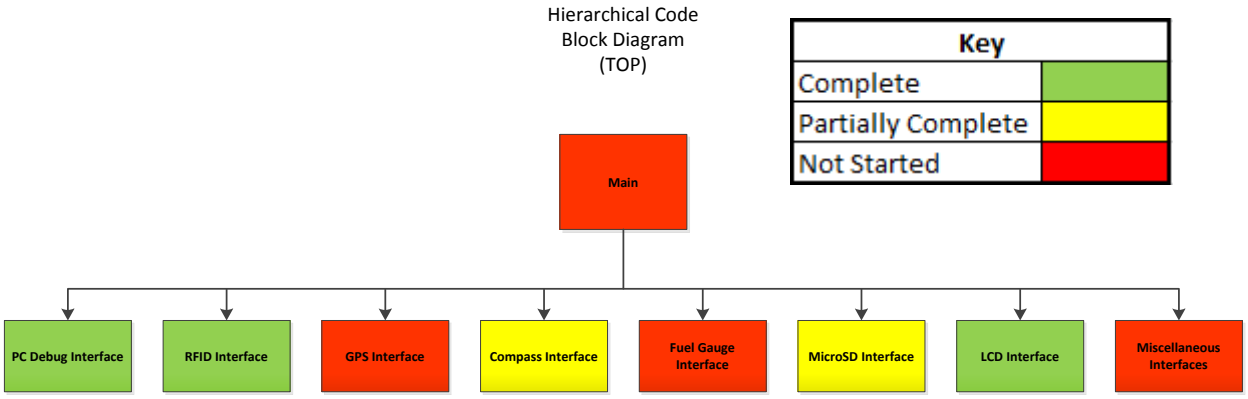
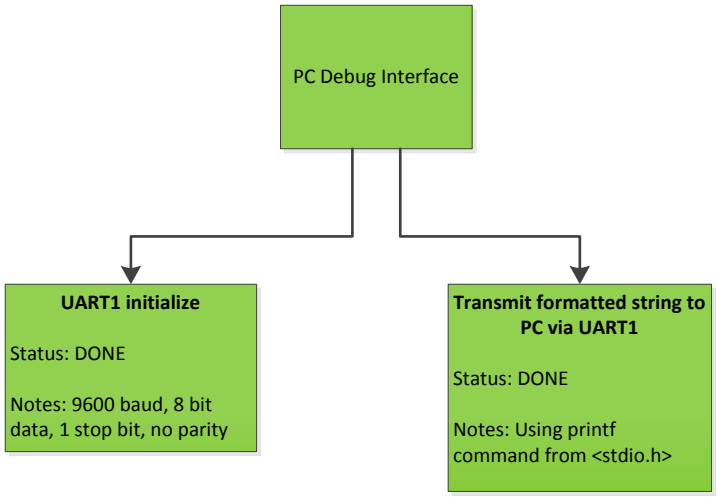


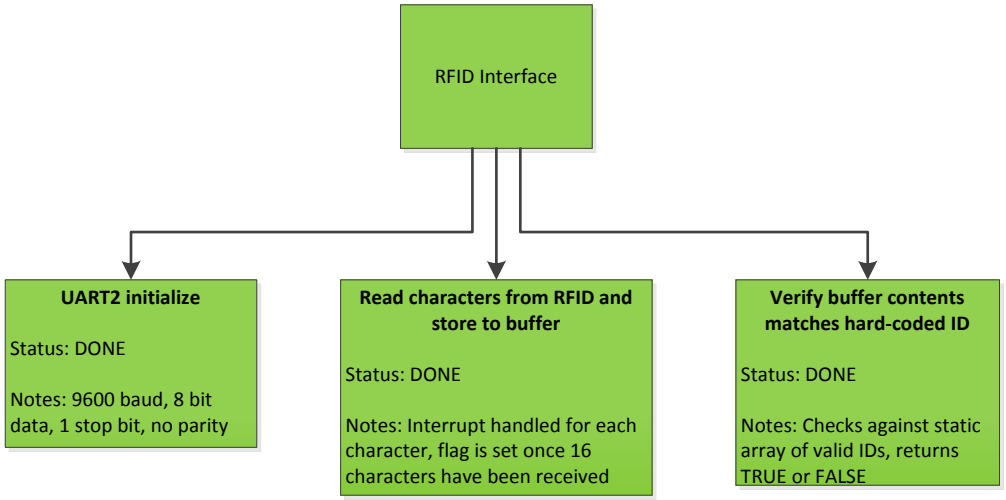
Figure 4 – Main Loop Flowchart

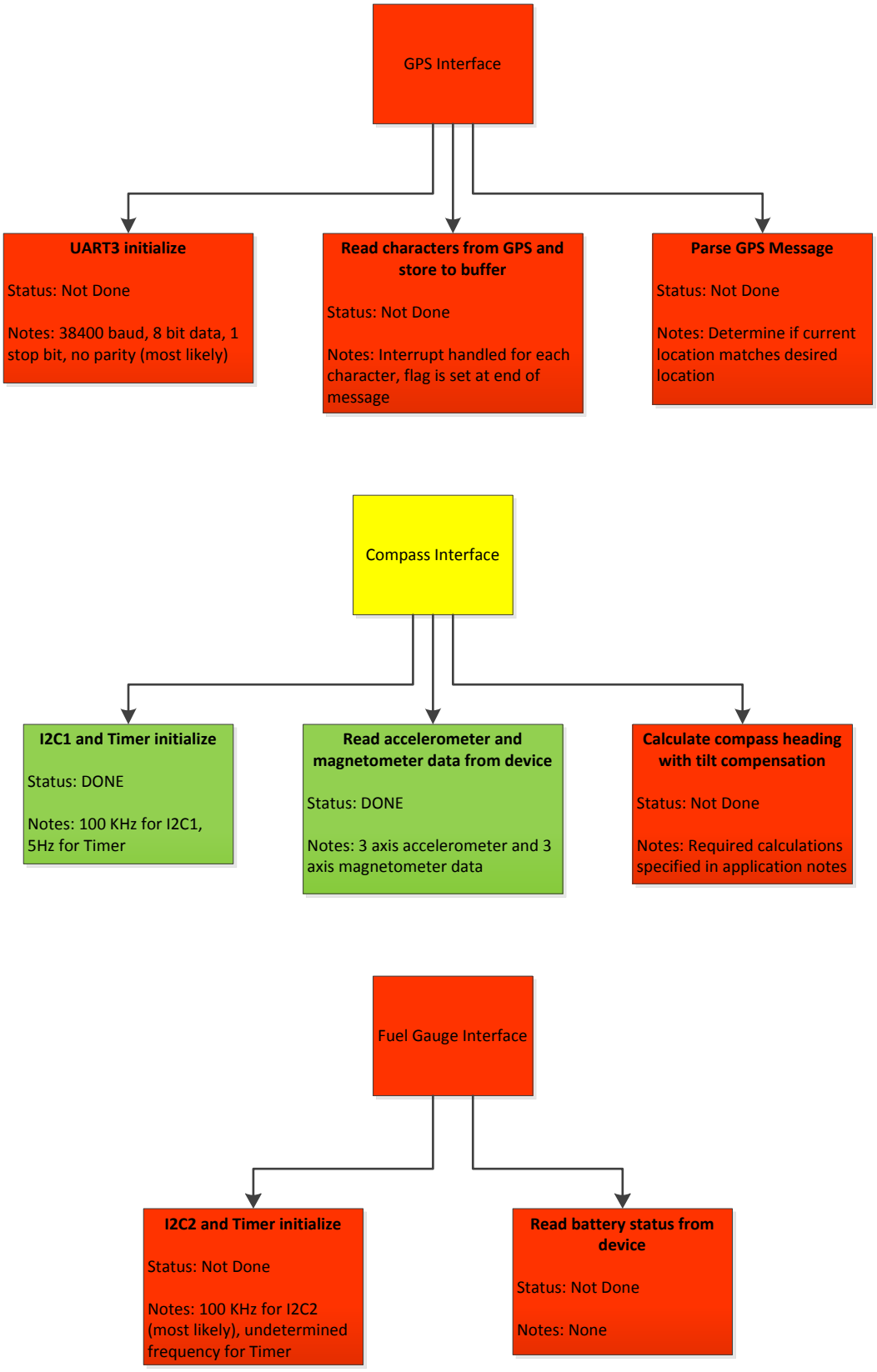


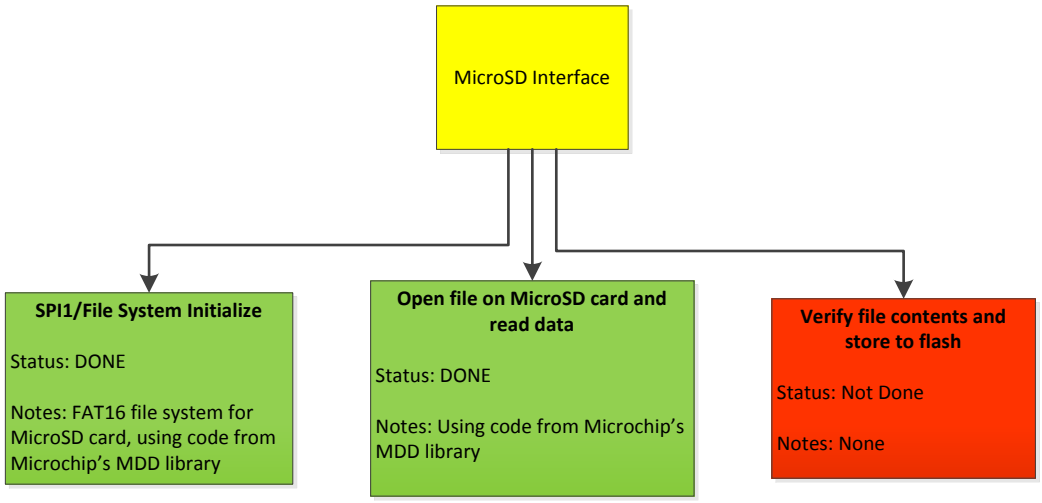
Appendix C: Hierarchical Block Diagram of Code Organization

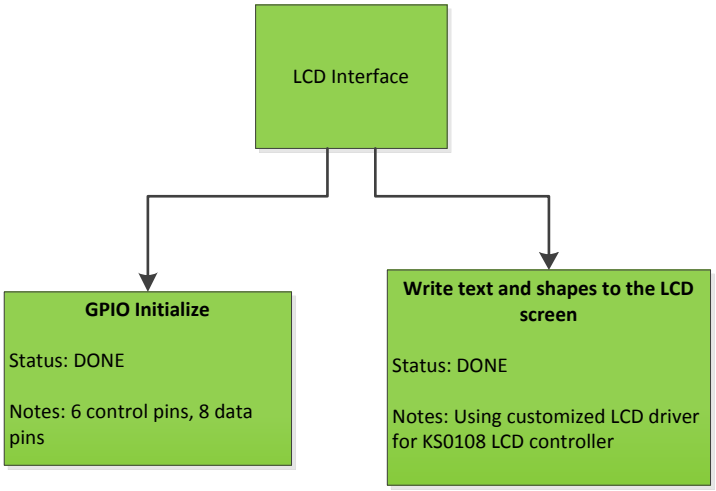


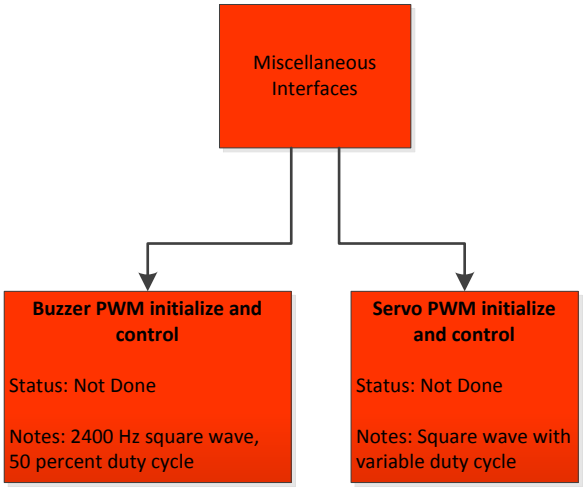





GPS Interface
UART3 initialize
Status: Not Done
Notes: 38400 baud, 8 bit data, 1 stop bit, no parity (most likely)
Read characters from GPS and store to buffer
Status: Not Done
Notes: Interrupt handled for each character, flag is set at end of message
Parse GPS Message
Status: Not Done
Notes: Determine if current location matches desired location
Compass Interface
I2C1 and Timer initialize
Status: DONE
Notes: 100 KHz for I2C1, 5Hz for Timer
Read accelerometer and magnetometer data from device
Status: DONE
Notes: 3 axis accelerometer and 3 axis magnetometer data
Calculate compass heading with tilt compensation
Status: Not Done
Notes: Required calculations specified in application notes
Fuel Gauge Interface
I2C2 and Timer initialize
Status: Not Done
Notes: 100 KHz for I2C2 (most likely), undetermined frequency for Timer
Read battery status from device
Status: Not Done
Notes: None


MicroSD Interface
SPI1/File System Initialize
Status: DONE
Notes: FAT16 file system for MicroSD card, using code from Microchip's MDD library
Open file on MicroSD card and read data
Status: DONE
Notes: Using code from Microchip's MDD library
Verify file contents and store to flash
Status: Not Done
Notes: None


LCD Interface
GPIO Initialize
Status: DONE
Notes: 6 control pins, 8 data pins
Write text and shapes to the LCD screen
Status: DONE
Notes: Using customized LCD driver for KS0108 LCD controller


Miscellaneous Interfaces
Buzzer PWM initialize and control
Status: Not Done
Notes: 2400 Hz square wave, 50 percent duty cycle
Servo PWM initialize and control
Status: Not Done
Notes: Square wave with variable duty cycle