Winter 2013 – Basic Stamp Multi Sensor Data Logging

Introduction

This project has been the most challenging task I have taken on, in regards to Basic Stamp system design. It required a full understanding of the platform and the interactions with peripheral devices. At times I was faced with the nagging question, "is this really possible"? My initial view of the project was what the entire system would ultimately do. However, during the two months of work, it became clear that this was a discipline of component development and their interactions with one another.

The project that I submitted would be a system that would retrieve data from six sensors. Then the system would store the data to an external high capacity memory device. The idea of using the Basic Stamp module for this task is a challenge due to resources constraints. The system is limited to 16Kbits of program memory and 256bits of variable RAM. Design considerations were crucial to getting all the components to work together.

The Multi Sensor Logger was used to monitor and store the environmental conditions of a bicycle trailer. The system usage requirements are, but not limited to:

- Reliability in a high vibration environment.
- Resistance or protection from temperature and moisture.
- Stabilized mounting of hardware to mobile platform.
- Ease of setup, operation, monitoring, and removal of system.

The system is operator initiated through the use of a control switch. The system makes use of all 16 I/O pins from the Board Of Education. It collects data from the following sensors:

- Tilt Sensor for tracking vibration, angle, and motion.
- Ultrasonic Sonar Sensor for detecting objects in range.
- Magnetic Compass Sensor to track bearing and to detect magnetic interferences.
- Speedometer Reed Sensor to track motion and distance.
- Temperature Sensor to record air temperature of environment.
- Voltage Level Sensor to record system voltage use during entire data logging process.

The sensor data is collected at 3 set time intervals, these are as follows:

- 250 msec intervals for Tilt and Sonar readings.
- 1.5 sec intervals for Compass and Speed readings.
- 10 sec intervals for Temperature and Voltage readings.

As the data is polled from the sensors, the results are stored to the external 512Kbit EEPROM. The EEPROM addresses are counted during each pass so data can be correctly stored. In addition, the task of tracking memory usage is derived from the pass count. The memory usage is displayed on a LCD to allow the operator to view progress.

Project Object Code

Below is the code used to detect sensor data based on time intervals and write that data values to EEPROM. Care was taken to segment each task into a logical group. The objective was to make reading and modifying the code easier.

```
' -----[ Title ]-------------------------------------------------------------
'
'   File....... MulitSensor_Logging_Final.bs2
'   Purpose.... Winter 2013 Paper - Records Sensor Readings at set times to memory and display
space usage
'   Author..... Patrick Gilfeather, OrangeLine Solutions
'   E-mail..... patrick@orangelinesolutions.com
'   Updated.... 8 APR 2013
'
' {$STAMP BS2}
' {$PBASIC 2.5}

' -----[ Declarations ]--------------------------------------------------
sign                    VAR       Bit                 ' Memsic Sign bit
I2C_LSB                 VAR       Bit                 ' Compass Variables for I2C LSB
i2cAck                  VAR       Bit                 ' Ack bit from device
MSec                    VAR       Nib                 ' Median Second Counter
LSec                    VAR       Nib                 ' Long Second Counter
devNum                  VAR       Nib                 ' device number (0 - 7)
addrLen                 VAR       Nib                 ' 0, 1 or 2
Ssec                    VAR       Byte                ' Short Second Counter
I2C_VAL                 VAR       Byte                ' Compass Variables for I2C VAL
I2C_REG                 VAR       Byte                ' Compass Variables for I2C REG
Recycle3                VAR       Byte                ' Multi re-use Byte Variable
Recycle4                VAR       Byte                ' Multi re-use Byte Variable
Recycle5                VAR       Byte                ' Multi re-use Byte Variable
outVal                  VAR       Byte
inVal                   VAR       Byte
slvAddr                 VAR       Byte                ' slave address
RPM                     VAR       Byte
z                       VAR       Word                ' Compass z-axis measurement
Recycle1                VAR       Word                ' Multi re-use Word Variable
Recycle2                VAR       Word                ' Multi re-use Word Variable
RSWTime                 VAR       Word
devAddr                 VAR       Word                   ' address in device

ADC_CS                  PIN       0                   ' Temperature ADC Chip Select pin P0
ADC_Clk                 PIN       1                   ' Temperature ADC Clock pin P1
ADC_Dout                PIN       2                   ' Temperature ADC Data output pin P2
VoltPin                 PIN       3                   ' Voltmeter RCTime pin P3
pMaxEnable              PIN       4                   ' Ultra Sonic Range enable pin P4
pMaxPWM                 PIN       5                   ' Ultra Sonic Range PWM pin P5
XAxis                   PIN       6                   ' Memsic X Axis of tilt to pin P6
YAxis                   PIN       7                   ' Memsic Y Axis of tilt to pin P7
SDA                     PIN       8                   ' Compass SDA of gyro to pin P8
SCL                     PIN       9                   ' Compass SCL of gyro to pin P9
ReedSwitch              PIN       10                  ' Speedometer reed sensor on pin P10
mSCL                    PIN       11                  ' I2C serial clock line IC Pin 11
mSDA                    PIN       12                  ' I2C serial data line IC Pin 12
LED                     PIN       13                  ' Green LED on Pin 13
Switch                  PIN       14                  ' Control Switch on Pin 14
LCDPin                  PIN       15                  ' Spark Fun 16x2 LCD on Pin 15

WRITE_Data              CON       $3C                 ' Compass Requests Write operation
READ_Data               CON       $3D                 ' Compass Requests Read operation
MODE                    CON       $02                 ' Compass Mode setting register
X_MSB                   CON       $03                 ' Compass X MSB data output register
Cn1                     CON       48576               ' Voltmeter first constant
Cn2                     CON       8                   ' Voltmeter second constant
Line1                   CON       0
Line2                   CON       64
LCDCls                  CON       12
Baud                    CON       84                  '+ Inverted 8,N,1 inverted
lcd_cmd                 CON       $FE                 'command prefix
lcd_cmd2                CON       $7C                 'special command prefix
clrLCD                  CON       $01                 'Clear entire LCD screen
displayOff              CON       $08                 'Display off
displayOn               CON       $0C                 'Display ON
BackLite10              CON       $83                 '10% backlight
noCurs                  CON       $0C                 'Make cursor invisible
curpos                  CON       $80                 'set cursor  + position  (row 1=0 TO 15, row 2
= 64 TO 79)
scrollRight             CON       $1C
scrollLeft              CON       $18
SaveLCDScr              CON       $0A                 'Save Splash screen
Ack                     CON       0                      ' acknowledge bit
Nak                     CON       1                      ' no ack bit
EE24LC32                CON       %1010 << 4
LCDBlock                CON       255
```

```
' -----[ Initialize ]-----------------------------------------------------
SEROUT LCDPin,Baud,[lcd_cmd, displayOff]
PAUSE 10
SEROUT LCDPin,Baud,[lcd_cmd, displayOn]
PAUSE 10
SEROUT LCDPin,Baud,[lcd_cmd, clrLCD]
PAUSE 10
SEROUT LCDPin,Baud,[lcd_cmd2, BackLite10]
PAUSE 10
SEROUT LCDPin,Baud,[lcd_cmd, curpos + Line1, " Multi-Sensor ", lcd_cmd, curpos + Line2, " Memory
Logger  "]
SEROUT LCDPin,Baud,[lcd_cmd2, SaveLCDScr]
PAUSE 3000                                          ' Allow connection to stabilize
SEROUT LCDPin,Baud,[lcd_cmd, clrLCD]: PAUSE 20                          ' Clear LCD
SEROUT LCDPin,Baud,[lcd_cmd, curpos + Line1, " Press to Start ", lcd_cmd, curpos + Line2,
" ................"]

Ssec = 0
RSWTime = 0
RPM = 0
devAddr = 0

RCTIME VoltPin,0,Recycle1
LOW VoltPin
Recycle1 = 0

Reset:
  #IF ($stamp >= BS2P) #THEN
    #ERROR "Use I2COUT and I2CIN!"
  #ENDIF

  devNum = 0                                        ' device address %000
  slvAddr = EE24LC32 | (devNum << 1)
  addrLen = 2

PAUSE 100                                     ' Power up delay
I2C_REG = MODE                                ' Set operating mode to continuous
I2C_VAL = $0
GOSUB I2C_Write_Reg


' -----[ Start Routine ]---------------------------------------------------
Pushbutton:

Ssec = 0
RSWTime = 0

DO

IF Switch = 1 THEN
  HIGH LED
  Ssec = Ssec + 1
  PAUSE 250
  IF (Ssec > 5) THEN
    FOR RSWTime = 0 TO 25
      RSWTime = RSWTime + 1
      LOW LED
      PAUSE 120
      HIGH LED
      PAUSE 120
    NEXT
    LOW LED
    GOSUB Main
    ENDIF
ELSE
  LOW LED
ENDIF

LOOP
```

```
' -----[ Main Routine ]----------------------------------------------------
Main:

Ssec = 0
RSWTime = 0

DO
  Ssec = Ssec + 5
  RSWTime = RSWTime + 1
  IF (RSWTime > 64000) THEN                         ' Counter reset to avoid overflow
    RSWTime = 0
  ENDIF
  IF ReedSwitch = 0 THEN
    GOSUB ReedSensor                                ' Speedometer readings from Reed Sensor
  ENDIF
  IF (Ssec > 240) THEN
    GOSUB UltraSonic                                ' Distance from MaxBotix Range Finder
    GOSUB Tilt                                      ' Tilt axis from Memsic
    Msec = Msec + 1
    Ssec = 0
  ENDIF
  IF (Msec > 9) THEN
    GOSUB Speedometer                                ' Speedometer readings from Reed Sensor
    GOSUB Compass                                   ' Magnetic readings from Compass Sensor
    Lsec = Lsec + 1
    Msec = 0
  ENDIF
  IF (Lsec > 5) THEN
    GOSUB Voltage                                   ' Voltage from RCTime Cicuit
    GOSUB Temperature                               ' Temperature Read ADC 0831
    Lsec = 0
  ENDIF
LOOP

' -----[ Sensor Routines ]--------------------------------------------------
UltraSonic:
      'Read and Create Data
  HIGH pMaxEnable
  PULSIN pMaxPWM, 1, Recycle1
  LOW pMaxEnable                                                    ' Rcm = ToF / 58  Convert
Time of Flight to cm
  Recycle3 = Recycle1 / 74                                         ' Convert Time of Flight
to Inches
      'Log Data
  outVal = Recycle3
  GOSUB Logging
RETURN

Tilt:
      'Read and Create Data
  PULSIN XAxis, 1, Recycle1                                         ' x-axis measurement
  PULSIN YAxis, 1, Recycle2                                         ' y-axis measurement
      ' Scale and offset x and y-axis values to 0 to 255.
  Recycle3 = (Recycle1 MIN 1890 MAX 3214) - 1890 ** 13500           'Measured scale 3124-1890
  Recycle4 = (Recycle2 MIN 1899 MAX 3130) - 1899 ** 13533           'Measured scale 3130-1899
      'Log Data
  outVal = Recycle3
  GOSUB Logging
  outVal = Recycle4
  GOSUB Logging
RETURN

Compass:
      'Read and Create Data
  GOSUB GetRawReading                                               ' Get raw Compass reading
from I2C
  Recycle3 = Recycle1 + 160 ** 46420
  Recycle4 = Recycle2 + 308 ** 40268
  Recycle5 = z + 650
      'Log Data
  outVal = Recycle3
  GOSUB Logging
  outVal = Recycle4
  GOSUB Logging
  outVal = Recycle5
  GOSUB Logging
RETURN
```

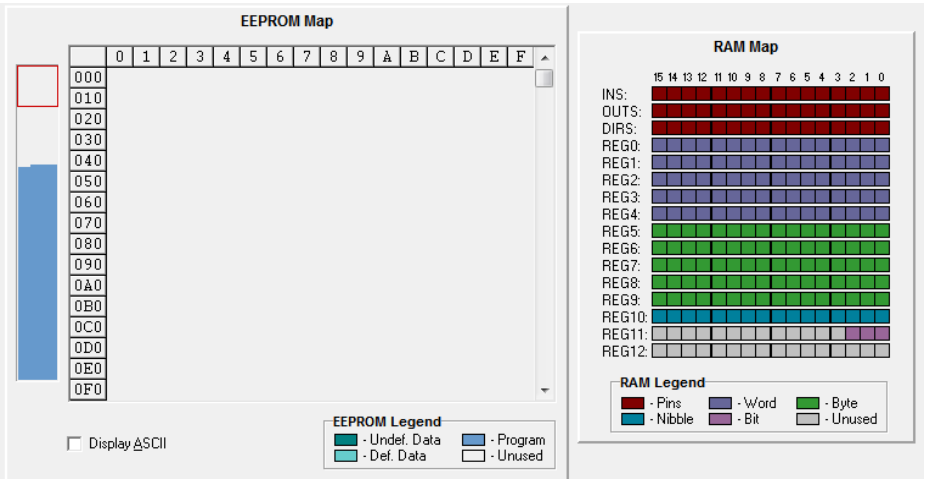
```
Speedometer:
      'Log Data
  outVal = RPM
  GOSUB Logging
RETURN

Temperature:
      'Read and Create Data
  LOW ADC_CS                                          ' Enable chip
  SHIFTIN ADC_Dout, ADC_Clk, MSBPOST,[Recycle3\9]     ' Clock in data from ADC
  HIGH ADC_CS                                         ' Disable ADC
  Recycle3 = Recycle3 * 9 / 5 + 32                    ' Convert C to F degrees
  Recycle3 = Recycle3 - 27                            ' Offset to correct value
      'Log Data
  outVal = Recycle3
  GOSUB Logging
RETURN

Voltage:
      'Read and Create Data
  RCTIME VoltPin,0,Recycle1
  LOW VoltPin
  Recycle3= Cn1 / Recycle1 + Cn2
      'Log Data
  outVal = Recycle3
  GOSUB Logging
RETURN

ReedSensor:
      ' Calcutate Time only
  DO
    Ssec = Ssec + 5
    RSWTime = RSWTime + 1

    IF ReedSwitch = 1 THEN
      RSWTime = 60000 / RSWTime
      RPM = RSWTime / 6
      RSWTime = 0
      RETURN
    ENDIF
  LOOP

' -----[ EEPROM Routines ]-------------------------------------------------
Logging:
  IF devAddr > 65535 THEN
    END
  ENDIF

  Recycle3 = outVal
  GOSUB mWrite_Byte
  GOSUB mRead_Byte
  inVal = Recycle3

  SEROUT LCDPin,Baud,[lcd_cmd, clrLCD]
  PAUSE 10
  Recycle5 = devAddr / 655
  SEROUT LCDPin,Baud,[lcd_cmd, curpos + Line1, "Memory = ", DEC Recycle5, "%"]
  SEROUT LCDPin,Baud,[lcd_cmd, curpos + Line2, "Address = ", DEC5 devAddr]

  PAUSE 10

  devAddr = devAddr + 1
RETURN
```

```
' -----[ Compass I2C High Level Functions ]-------------------------------------------------------
GetRawReading:
  PAUSE 50                                ' Wait for new data

  ' Send request to X MSB register
  GOSUB I2C_Start
  Recycle3 = WRITE_Data
  GOSUB I2C_Write
  Recycle3 = X_MSB
  GOSUB I2C_Write
  GOSUB I2C_Stop

  'Get data from register (6 bytes total, 2 bytes per axis)
  GOSUB I2C_Start
  Recycle3 = READ_Data
  GOSUB I2C_Write

  ' Get X
  GOSUB I2C_Read
  Recycle4 = Recycle3
  GOSUB I2C_ACK
  GOSUB I2C_Read
  GOSUB I2C_ACK
  Recycle1 = (Recycle4 << 8) | Recycle3

  ' Get Z
  GOSUB I2C_Read
  Recycle4 = Recycle3
  GOSUB I2C_ACK
  GOSUB I2C_Read
  GOSUB I2C_ACK
  Z = (Recycle4 << 8) | Recycle3

  ' Get Y
  GOSUB I2C_Read
  Recycle4 = Recycle3
  GOSUB I2C_ACK
  GOSUB I2C_Read
  GOSUB I2C_NACK
  Recycle2 = (Recycle4 << 8) | Recycle3

  GOSUB I2C_Stop
RETURN

' -----[ Compass I2C Low Level Functions ]----------------------------------------------------
' Set I2C_REG & I2C_VAL before calling this
I2C_Write_Reg:
  GOSUB I2C_Start
  Recycle3 = WRITE_DATA
  GOSUB I2C_Write
  Recycle3 = I2C_REG
  GOSUB I2C_Write
  Recycle3 = I2C_VAL
  GOSUB I2C_Write
  GOSUB I2C_Stop
RETURN

' Set I2C_REG before calling this, I2C_DATA will have result
I2C_Read_Reg:
  GOSUB I2C_Start
  Recycle3 = WRITE_DATA
  GOSUB I2C_Write
  Recycle3 = I2C_REG
  GOSUB I2C_Write
  GOSUB I2C_Stop
  GOSUB I2C_Start
  Recycle3 = READ_DATA
  GOSUB I2C_Write
  GOSUB I2C_Read
  GOSUB I2C_NACK
  GOSUB I2C_Stop
RETURN

I2C_Start:
  LOW SDA
  LOW SCL
RETURN

I2C_Stop:
  LOW   SDA
```

```
  INPUT SCL
  INPUT SDA
RETURN

I2C_ACK:
  LOW   SDA
  INPUT SCL
  LOW   SCL
  INPUT SDA
RETURN

I2C_NACK:
  INPUT SDA
  INPUT SCL
  LOW   SCL
RETURN

I2C_Read:
  SHIFTIN SDA, SCL, MSBPRE, [Recycle3]
RETURN

I2C_Write:
  I2C_LSB = Recycle3.BIT0
  Recycle3 = Recycle3 / 2
  SHIFTOUT SDA, SCL, MSBFIRST, [Recycle3\7]
  IF I2C_LSB THEN INPUT SDA ELSE LOW SDA
  INPUT SCL
  LOW SCL
  INPUT SDA
  INPUT SCL
  LOW SCL
RETURN

' -----[ EEPROM I2C High Level Functions ]-------------------------------------------------------
' Random location write
' -- pass device slave address in "slvAddr"
' -- pass address bytes (0, 1 or 2) in "addrLen"
' -- register address passed in "devAddr"
' -- data byte to be written is passed in "i2cData"

mWrite_Byte:
  GOSUB mI2C_Start                              ' send Start
  Recycle4 = slvAddr & %11111110                ' send slave ID
  GOSUB mI2C_TX_Byte
  IF (i2cAck = Nak) THEN mWrite_Byte            ' wait until not busy
  IF (addrLen > 0) THEN
    IF (addrLen = 2) THEN
      Recycle4 = devAddr.BYTE1                  ' send word address (1)
      GOSUB mI2C_TX_Byte
    ENDIF
    Recycle4 = devAddr.BYTE0                    ' send word address (0)
    GOSUB mI2C_TX_Byte
  ENDIF
  Recycle4 = Recycle3                            ' send data
  GOSUB mI2C_TX_Byte
  GOSUB mI2C_Stop
  RETURN


' Random location read
' -- pass device slave address in "slvAddr"
' -- pass address bytes (0, 1 or 2) in "addrLen"
' -- register address passed in "devAddr"
' -- data byte read is returned in "i2cData"

mRead_Byte:
  GOSUB mI2C_Start                              ' send Start
  IF (addrLen > 0) THEN
    Recycle4 = slvAddr & %11111110              ' send slave ID (write)
    GOSUB mI2C_TX_Byte
    IF (i2cAck = Nak) THEN mRead_Byte           ' wait until not busy
    IF (addrLen = 2) THEN
      Recycle4 = devAddr.BYTE1                  ' send word address (1)
      GOSUB mI2C_TX_Byte
    ENDIF
    Recycle4 = devAddr.BYTE0                    ' send word address (0)
    GOSUB mI2C_TX_Byte
    GOSUB mI2C_Start
  ENDIF
  Recycle4 = slvAddr | %00000001                ' send slave ID (read)
  GOSUB mI2C_TX_Byte
```

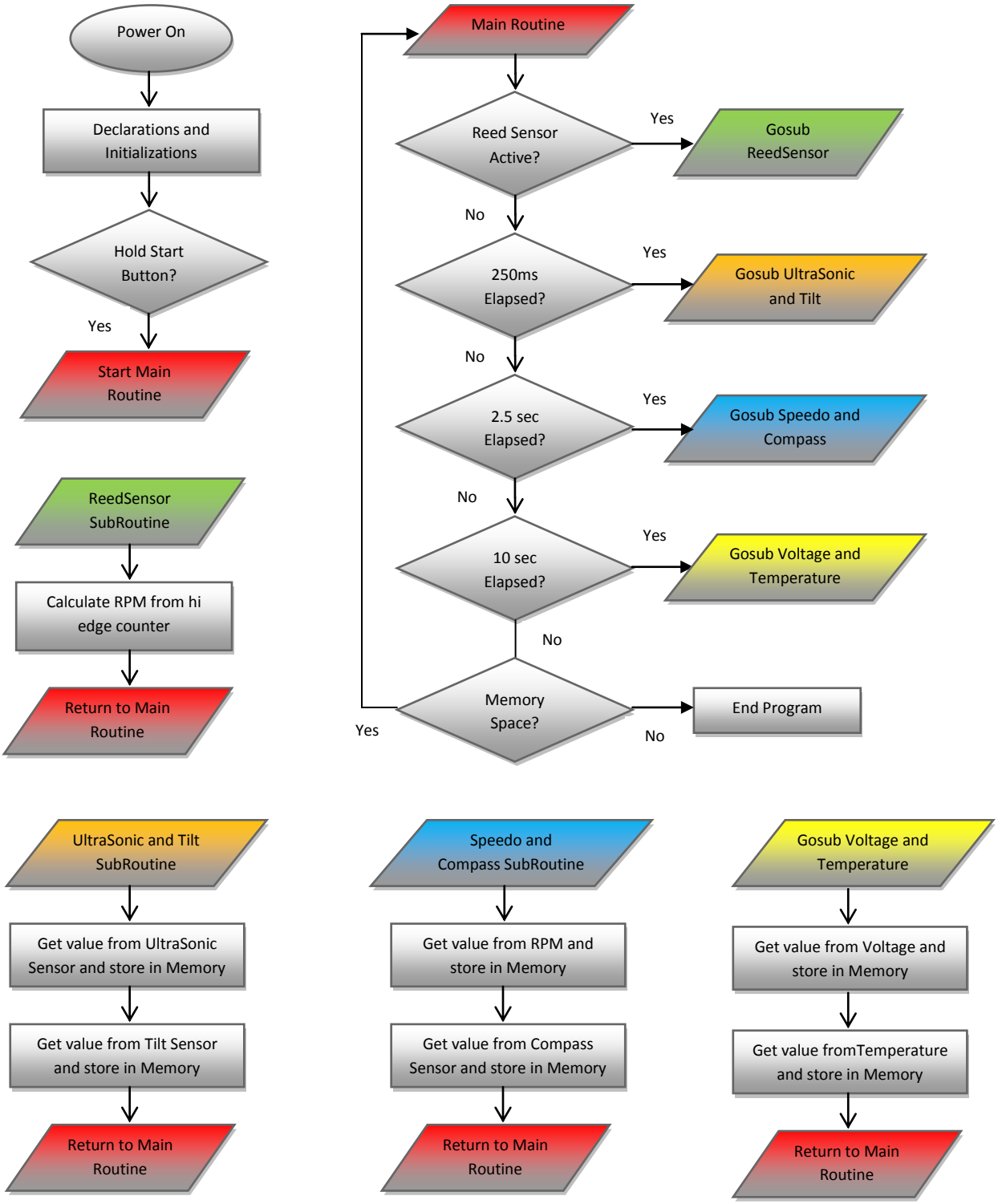
```
  GOSUB mI2C_RX_Byte_Nak
  GOSUB mI2C_Stop
  Recycle3 = Recycle4
  RETURN

' -----[ EEPROM I2C Low Level Functions ]-------------------------------------------------------
' *** Start Sequence ***

mI2C_Start:                                         ' I2C start bit sequence
  INPUT mSDA
  INPUT mSCL
  LOW mSDA

mClock_Hold:
  DO : LOOP UNTIL (mSCL = 1)                        ' wait for clock release
  RETURN


' *** Transmit Byte ***

mI2C_TX_Byte:
  SHIFTOUT mSDA, mSCL, MSBFIRST, [Recycle4\8]       ' send byte to device
  SHIFTIN mSDA, mSCL, MSBPRE, [i2cAck\1]            ' get acknowledge bit
  RETURN


' *** Receive Byte ***

mI2C_RX_Byte_Nak:
  i2cAck = Nak                                     ' no Ack = high
  GOTO mI2C_RX

mI2C_RX_Byte:
  i2cAck = Ack                                     ' Ack = low

mI2C_RX:
  SHIFTIN mSDA, mSCL, MSBPRE, [Recycle4\8]          ' get byte from device
  SHIFTOUT mSDA, mSCL, LSBFIRST, [i2cAck\1]         ' send ack or nak
  RETURN


' *** Stop Sequence ***

mI2C_Stop:                                          ' I2C stop bit sequence
  LOW mSDA
  INPUT mSCL
  INPUT mSDA
  RETURN
```

The memory map of the program shows how the limited resources available were used to their full extent. This recycling of variables was the most effective method to keep RAM usage to a minimum.



Flow Chart

Below is the flow chart of the operational logic and control of the multi sensor data logger.



Theory of Operation

The system will load all the variables, pin assignments, and constants into RAM after powering on. The LCD will also initialize and display a splash message "Multi Sensor" on the first line and "Memory Logger" on the second line. After three seconds, the message will clear and the LCD will display a prompt "Push to Start". Some initial variables will be set to zero and the memory will be initialized.

From this point, the system enters into a loop which will depend on the push button state. If the push button is pressed, a LED will turn on to indicate user input. This was needed to know if the system was ready and responding to input. If the push button is held for more than 1.5 seconds, the loop is broken. This is indicated by a series of flashes on the LED.

The system now enters the Main routine loop of the program. There are five conditions in the main routine. The first is a time counter housekeeping function, which prevents overflow of the counter variable. The second is the reed sensor state, which is triggered when the reed sensor is active. The last three are time functions that are triggered based on the time counters, which reset after firing.

In the time counter housekeeping function, the counter value is reset to zero when it reaches a value in excess of 64000. This ensures that the WORD variable does not overflow. When the counter value is reset, it simply continues on with the Main routine loop.

The reed sensor function breaks the Main routine loop and follows into a Reed Sensor routine loop. Here the counter continues to add values with each pass. It remains in this loop until the reed sensor is no longer active. When the reed sensor becomes deactivated, the reed switch counter total is divided by 60000ms, the result is the RPM value. The reed sensor counter is set to zero. Here, the reed sensor counter starts it count with each deactivation of the reed sensor, giving a complete time between each wheel rotation. At this point the Reed Sensor routine is broken from its loop state and the function returns to the Main routine.

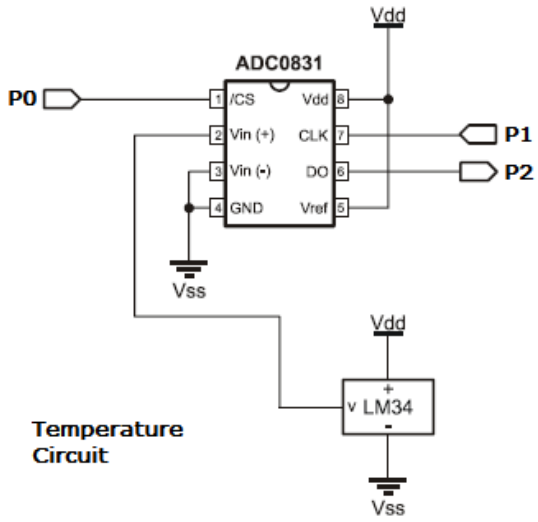
The time based functions only differ in that they are triggered at different times and they activate different sensors. Those differences aside, they last three functions operate exactly the same way. If a time counter is reached, the Main routine is broken and it branches out to the corresponding function.

As an example, the UltraSonic function will break the loop of the Main routine and will start if the counter is at 250ms. Here the UltraSonic function will poll the sensor for a value and store it in a variable. That variable will have some calibration math applied to it and the resulting value will be stored in a Byte variable. That Byte variable will then be passed to the Logging function.

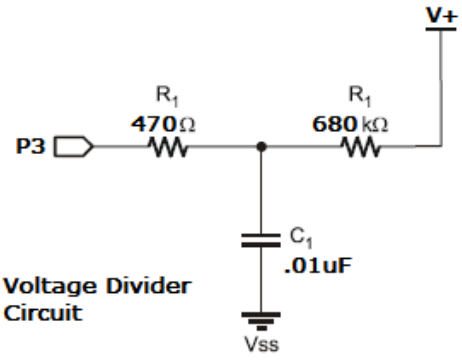
In the Logging function, the Byte variable will be stored in memory address space that is determined from an address counter. With each pass of the Logging function, the address counter is incremented. This allows the Byte values to be stored in each successive memory address space on the EEPROM. After the value is written to EEPROM, the address space in memory is displayed on the LCD. The function then returns back to the Main routine loop only if memory space remains on the EEPROM. If no space remains, then the program terminates.

This example is the same from the Tilt, Speedometer, Compass, Temperature, and Voltage readings. Again, the Main routine will continue to loop until it is ultimately broken from exhausting all of the EEPROM space. Based on the capacity of the 512Kbit EEPROM and the frequency of the readings, the system should log date for 60 minutes.

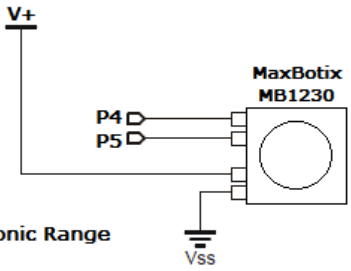
Hardware Schematics of Multi Sensor Data Logger



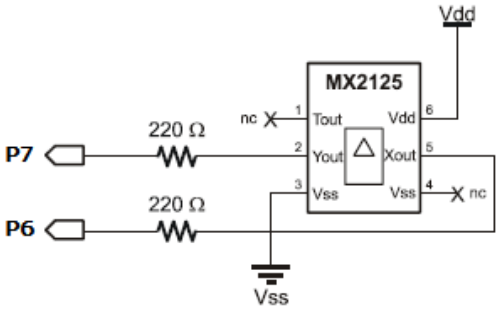
Temperature Circuit



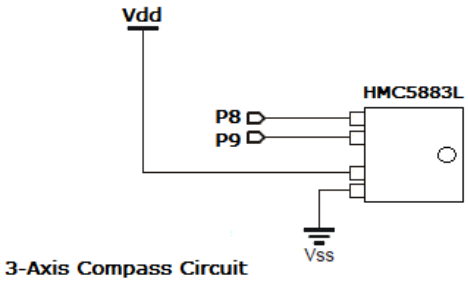
Voltage Divider Circuit



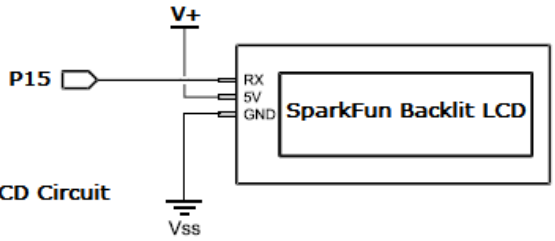
UltraSonic Range Circuit



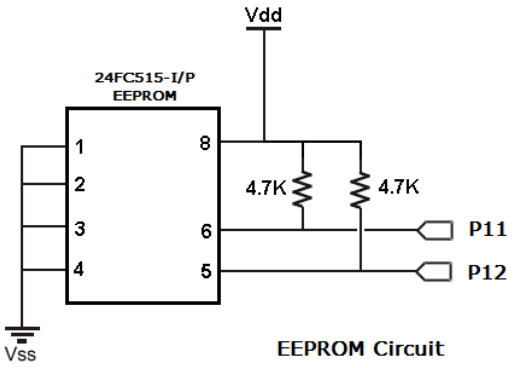
Tilt Circuit



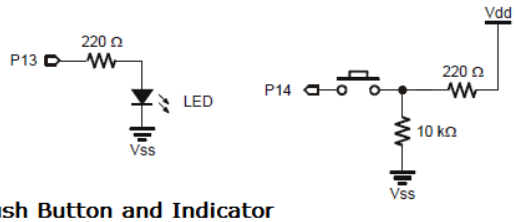
3-Axis Compass Circuit



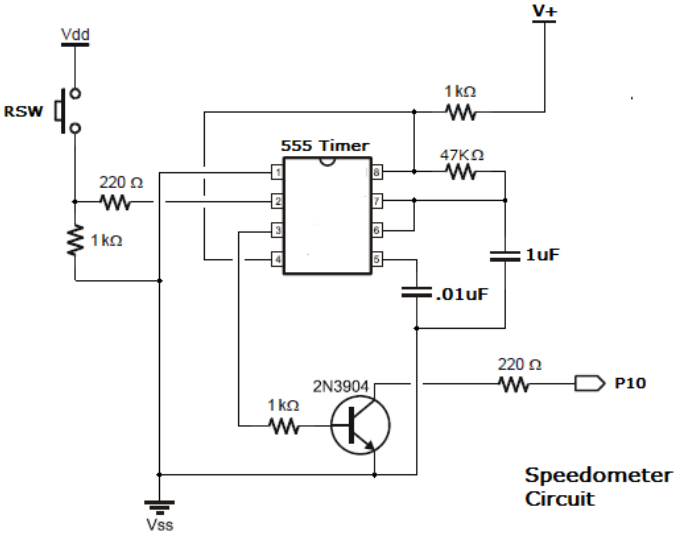
LCD Circuit



EEPROM Circuit



Push Button and Indicator Circuit



Speedometer Circuit

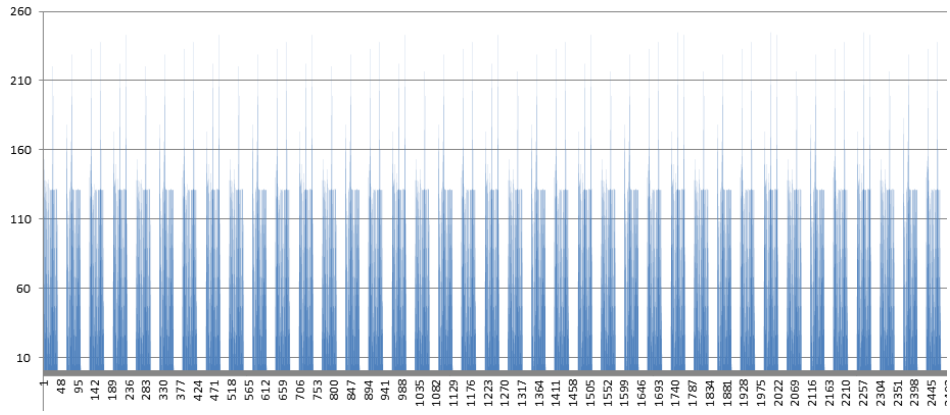
Equipment Setup of Multi Sensor Data Logger

Placement of the equipment had its own set of challenges. The environment was wet, cold, and prone to shaking. Outfitting a bicycle wasn't practical for this application since time was short and materials were not available. Use of a bike trailer proved to be the best method. Rigging the speedometer, ultrasonic sensor, and camera were the only challenge. The remainder of the sensors were housed in the cabin of the trailer.

Removal of the trailer seat revealed Velcro that was already in place on the bread board platform. This made it a snap to adhere the backboard to the trailer seat housing. Side pockets inside the cabin were used to hold the external battery packs for the GPS and camera devices.

Operational Test and Results

Weather, without fail, added a burden to the initial test run of the system. The sensor data from the ultrasonic sensor was erroneous due to interferences from rain drops entering the senor field of view. In addition, the rear facing camera had fogged up during the run, a result of condensation from the high humidity and low temperature conditions.



Data gathering appears to be successful, but the representation has proved to be too difficult to process and interpret within the time constrain of the project. Memory space had no identifier data and was comprised entirely of the sensor data readings. This approach allowed the system to have all of the available memory dedicated to storing sensor readings. The consequence is that each reading cannot be distinguished from each other. Albeit the results could be processed to isolate readings from each sensor, there are anomalies that post processing will not correct. The data graph above shows zero readings throughout the memory, these cannot be explained.

Summary

The last month of instruction for the winter quarter moved entirely away from the text and into the realm of possibilities. This opened up an entirely different point of view of the basic stamp. The problem of multi sensor data logging on a platform as confined presented challenges that could only be experienced this way.

The objective of the project was not successful because the data is jumbled and not discernible. Had portions of the memory storage been used as sensor identifiers, it's possible that some sensor data could have been represented in the results. Clearly this is a result of time running out on the project. It was best to present the project in this fashion than to delay, or worse abandon it all together. There are some good points worth noting and milestones achieved during the endeavor.

The design of the project far surpassed expectations. It proved that the project could be done. The principles were elegant and well presented. Each component became an object, and by doing so the entire project design was manageable.

The circuit design followed a similar path, but that actual circuitry fell far short. The bread board should only be used in a stable lab environment, such as a test bench. Placing it on a mobile platform pushed the limits of reliability which may have led to the erroneous readings stored in memory. In addition, the hardware was haphazardly connected in a scattered fashion which made identification of components extremely difficult. The circuitry would have been in par with the project had it been modularized with interconnects in a stackable fashion.

The project's use of a bicycle trailer was refreshing following the circuitry short comings. Placement and protection of the sensors and devices was adequate. The supplemental hardware did not damage or alter the trailer and this was ideal for this project goal.

The greatest challenge for any undertaking can be the vast quantity of hurdles. This appeared to be the theme this project followed. Achievement was not measured in reaching the goal, but by overcoming each obstacle as it presented itself. The project was scattered with doubt, frustration, exhaustion, disappointment, and grief. I have a sincere relief from its completion.

It is to my dear brother Kenneth that I dedicate this project to. His untimely passing a few weeks before the project's completion is a bold reminder that nothing in life will go as expected. Although he would have probably shaken his head at the sight of the equipment, he would have shared in the delight of discovery.