# Development of a Digital Pressure Sensor Emulator for Altimeter Testing

Altimeter Buddies Again Team
T-813
Dan Wolf - NAR 24516
Mary Kuzel - NAR 46379
NARCON 2017
2/24/2017

## Project Abstract

The purpose of this project was to develop a hardware device that could be used to emulate a pressure sensor. One limitation in performance analysis of altimeters has been understanding the details of how the altimeter software processes the barometric pressure data. In my team's NARAM 58 R&D report, using a pressure chamber, I2C protocol logic analyzer, and a reference altimeter, we were able to understand the capability of the current altimeters approved for NAR contest use to accurately compute altitude.

However both flight and altitude chamber testing done by my team and others has identified some gaps in understanding how commercial altimeters operate in two key areas. 1) Launch detection and robustness of the launch detect algorithm to prevent false triggering. 2) Filtering of the altitude data including the ability to filter out spikes in the data that often occur at motor ejection. These spikes in the data can cause over/under reporting of the peak altitude. The other limitation in testing done to date is that most readily available pressure chambers cannot simulate the actual atmospheric pressure changes that occur during a rocket flight.

To overcome these limitations, this project proposes taking an altimeter and removing its pressure sensor and replace it with a piece of hardware called a pressure sensor emulator. This device "looks" to the altimeter software just like the sensor it replaced. However, the emulator can be programmed to send raw pressure data to the altimeter processor that looks like any flight profile desired, including data with ejection spikes, noisy data during launch, etc. In this way the ability of the altimeter to handle these issues can be better understood. It can also simulate slow launches, fast launches, and launches to any altitude.

To date, the process for creating the flight profiles has been developed and the hardware completed for the pressure sensor emulator. The next step is to complete the software of the emulator itself so that actual testing can be completed.

## Contents

## 1.0 Introduction

There are several commercial altimeters on the market that are used in the sport rocketry hobby. Initially these altimeters were most popular with high power rocket enthusiasts for both reporting altitude, recording flight profile, and also offering features like dual recovery system deployment. As these devices matured and grew in popularity, the desire and interest to use them for contest rocketry came about, both in the United States and Internationally. Like in the early days of optical tracking, there is still room for growth in using altimeters for contest use. Whereas a sport rocketry flier is mostly concerned about the altimeters basic functionality (did it report an altitude that looks reasonable, did it deploy the chutes at the right times/altitudes, etc.), errors of +/-10 or even 20% are not fatal and probably not noticeable. When altimeters started to be used for contest flights, they suddenly became more heavily scrutinized. How accurate were they, what were the limitations, what issues are there, and general mistrust that is found with any new technology. Further, they were black boxes whose process for actually determining the altitude was a mystery.

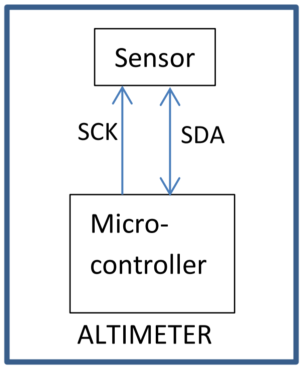
Several early R&D reports were done to study altimeters, their accuracy, and suitability for use in rocketry competition. Notable among them were references 1 through 5 (see Appendix C). These reports found altimeters as a suitable alternative to optical tracking. However, these reports considered the accuracy of the entire instrument (altimeter). In an attempt to identify the error budget and error sources of altimeters, our NARAM 54 R&D report (reference 21) examined the technology used. The key finding was that the latest digital pressure sensors do an excellent job of measuring pressure quite accurately over the range that contest rockets fly. Recommendations from that report were that only digital pressure sensors be used that met a certain set of criteria. The NAR adopted those recommendations in the current approved altimeter list for contest use. This eliminated of the key accuracy and performance limitations among altimeters used for NAR competition.

Having studied and mitigated some of the altimeter risk with the NARAM 54 report, our work in our NARAM 58 report (reference 23) focused on the accuracy of altimeters in taking raw pressure data from the sensor and computing the altitude. This work resulted in one manufacturing correcting an error in their altitude calculations that caused altitudes to report too high the higher the flight (divergence from the actual altitude). Another altimeter was found to have an over aggressive noise filter that cause the peak altitude reported to contain significant error. However, due to limitations of controlling the vacuum chamber to simulate an actual flight profile, the errors for a typical rocket flight would most often not be significant. This report retired another risk of most of today's commercial altimeters. Most do a great job of accurately converting and reporting altitude data.

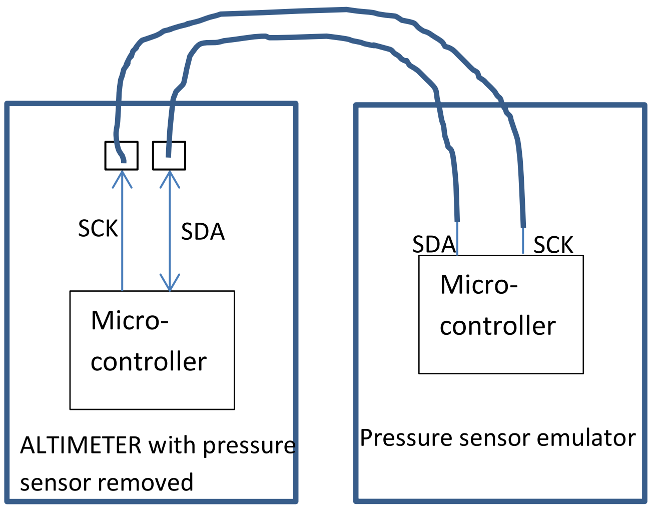
In spite of the success that altimeters exhibited in the all the projects cited, there are still some areas where the performance of commercial altimeters is not fully understood. One is in the area of launch detection. Common issues here are of two types, 1) launch detected prematurely, due to a wind gust or pressure change that occurred when the altimeter was placed in the rocket and 2) launch not detected at all and the altimeter is returned reporting 0 altitude, prior flight's altitude, or no altitude. The second issue is the performance of the altimeter software to filter out different types of noise, and specifically, spikes in the data caused by ejection.

In our NARAM-58 report we suggested that one way to measure performance for both launch detection and filtering would be to build a hardware device that “emulates” a pressure sensor. The sensor would be removed from the altimeter and the pressure sensor emulator would be connected in its place. The emulator would have the capability of sending raw data to the altimeter processor in the same format as the sensor. The altimeter would function as usual. However, the emulator, when started would send raw data for a desired flight profile. Profiles could be created for 100m, 200,m, 300m, 400, and so forth peak altitudes. In fact flight profiles could be generated for most any flight conceivable including taking a flight profile from Rocksim, Space Cad, Open Rocket of any contest rocket design from 1/8A through G power and back converting it to raw data. The sensor emulator would then take this raw data file and send it to the altimeter. If the altimeter does its job properly, the resulting flight profile should exactly match the flight profile from the flight simulator. The next step would be to add pressure spikes to these flight profiles and see how they affect the ability of the altimeter to filter those out properly. The other types of tests to run would be launch detect profiles to see how robust the altimeter would be to noise likely to be encounter on the pad and to see at what altitude the altimeter actually detects launch.

The figures below illustrate the concept:



**Typical altimeter with I2C Sensor Interface**



**Sensor removed and wires added to pads for board connections to sensor emulator board.**

## 2.0 Design Considerations and Approach Taken

All altimeters approved for NAR and FAI contest use a digital pressure sensor that has an I2C interface. See table 1 for altimeter sensor summary.

| Manufacturer | Model No | Sensor | Absolute Accuracy Typical | Relative Accuracy Typical | Resolution | Interface |
|---|---|---|---|---|---|---|
| Adrel | ALT-BMP | Bosch Sensortec BMP180 | 1 mBar | 0.12 mBar | < 1 meter | I2C |
| Altus Metrum | MicroPeak | Measurement Specialties MS5607 | 1.5 mBar | 0.5 mBar | < 1 meter | I2C/SPI |
| Jolly Logic | Altimeter One | Bosch Sensortec BMP85[1] | 1 mBar | 0.2 mBar | < 1 meter | I2C |
| Jolly Logic | Altimeter Two | Bosch Sensortec BMP85 ornewer[2] | 1 mBar | 0.2 mBar | < 1 meter | I2C |
| Jolly Logic | Altimeter Three | Bosch Sensortec BMP85 or newer[2] | 1 mBar | 0.2 mBar | < 1 meter | I2C |
| Perfectflite | ARPA | Believe to be Measurement Specialties MS5611[3] | 1.5 mBar | 0.5 mBar | < 1 meter | I2C/SPI |
| Perfectflite | Pnut | Measurement Specialties MS5611 | 1.5 mBar | 0.5 mBar | < 1 meter | I2C/SPI |
| Perfectflite | Stratologger | Measurement Specialties MS5611 | 1.5 mBar | 0.5 mBar | < 1 meter | I2C/SPI |
| Perfectflite | Firefly | Believe to be Measurement Specialties MS5611[3] | 1.5 mBar | 0.5 mBar | < 1 meter | I2C/SPI |

Table 1 - NAR and FAI altimeters sensor specification summary

*Notes:*

1. *The original Jolly Logic Altimeter One used the Bosch Sensortec BMP85. However, the unit has been redesigned and forward production may use a newer sensor.*
2. *Has not been verified. Assume all Jolly Logic Altimeters use a sensor from Bosch Sensortec but they may be of newer vintage (BMP180, BMP280).*
3. *Pefectflite Pnut and Stratologger use the Measurement Specialties MS5611. Although not verified, it is assumed that the APRA and Firefly also use the MS5611 since the specifications for all of the altimeters are the same.*

All of the sensors support the I2C interface as a slave device. Therefore, a pressure sensor emulator needs to support I2C communication as a slave device. In addition, all the commands or the subset of the commands used by the altimeter employing the sensor must be supported. See appendix B for details on the I2C interface.

Several design approaches were considered for how to build the pressure sensor emulator. One approach was to build a true hardware based emulator. This could be easily done using a field programmable gate array (FPGA) or Complex Programmable Logic Device (CPLD). The advantage of this approach is that the emulator needs to emulate the hardware wave forms of an I2C bus. Due to the teams experience with FPGAs, and implementation of I2C interfaces in FPGAs this was a strong consideration. By using VHDL (VHSIC Hardware Design Language) the basic design could be implemented fairly quickly. However, the limitation of this approach is that it would require developing a board that supports the FPGA or CPLD. Further, a mechanism for changing the flight profiles would need to be implemented. Finally, this would be a highly specialized technique that would be difficult to be maintained.

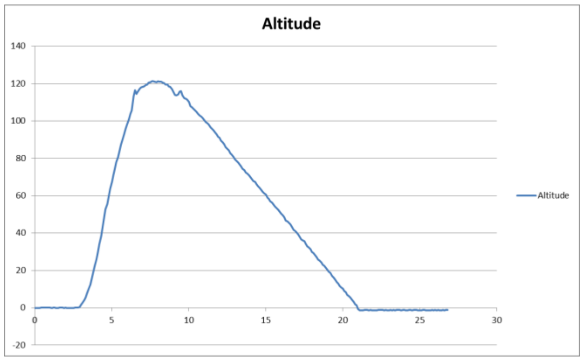
The next approach considered was to use the Atmel ATUSBKEY development board and write a program that would "bit bang" two of the I/O pins to send the raw data to the altimeter. However, it was decided that this was not an optimal approach because using software loops may not allow the data lines to be toggled fast enough.

Finally, it was decided to use the ATUSBKEY development board but to use the built in I2C port as a slave port. Altimeters use their onboard processors I2C port as a master to interface to the sensors. Because the emulator must look like a sensor, it must operate in slave mode. The advantage of using the ATUSBKEY is that data files can easily be written into the storage flash memory on the device. When the ATUSBKEY is plugged into a computer, it looks like a flash drive. Files from the computer can be copied to the ATUSBKEY. Then it can be removed from the computer, and connected to the altimeter under test. When operating, the emulator's onboard processor can read the data file transferred from the PC and send the raw data flight profile to the altimeter as the altimeter goes through its power and, launch detect, and flight data collection sequence.

## 2.1 Detailed Design Approach

Going from a desired flight profile for the pressure sensor emulator to emulate is a multistep approach. The steps are listed below:

1. Obtain altitude versus time data.



2. Convert altitude back to pressure. Note that for this step, the starting pressure can be set to sea level or the launch site level or any other value.  Here it is the elevation of Bong Recreation Area, where the launch this data was collected took place.

| TIME | ALTITUDE | PRESSURE |
|---|---|---|
| | ⋮ | |
| 2.6 | 0.034001 | 99175 |
| 2.7 | 0.204007 | 99173 |
| 2.8 | 0.034001 | 99175 |
| 2.9 | 0.459022 | 99170 |
| | ⋮ | |
| 7.2 | 119.3795 | 97779 |
| 7.3 | 119.6374 | 97776 |
| 7.4 | 120.4973 | 97766 |
| 7.5 | 120.7553 | 97763 |
| 7.6 | 121.1853 | 97758 |
| | ⋮ | |
| 20.7 | 3.264595 | 99137 |
| 20.8 | 1.98924 | 99152 |
| 20.9 | 1.309116 | 99160 |
| 21 | -0.646 | 99183 |
| 21.1 | -1.32595 | 99191 |

3. Convert the pressure values back to the raw data that would come from the sensor. The calculations will be dependent upon two things. 1) The manufacturer and model of the sensor, and 2) The calibration coefficients. One way to do this is to collect the calibration coefficients from an actual sensor and use those. As part of the function of the emulator, it must send those calibration coefficients to the altimeter processor when it collects them. Below is a set of calibration coefficients that came from the BMP180 sensor used in our NARAM-54 report. It doesn't matter which set of coefficients is used. However the same set of coefficients used to do the calculation of the raw sensor data from the pressure value must be the same as the ones used in the emulator that it sends to the altimeter processor.

   Example:
   Altitude = 0 meters (launch site)
   244 meters above sea level
   Pressure reading
      99176 Pascals from altimeter used (it recorded both altitude and pressure)
      or use standard atmospheric model formula – 98430 Pascals

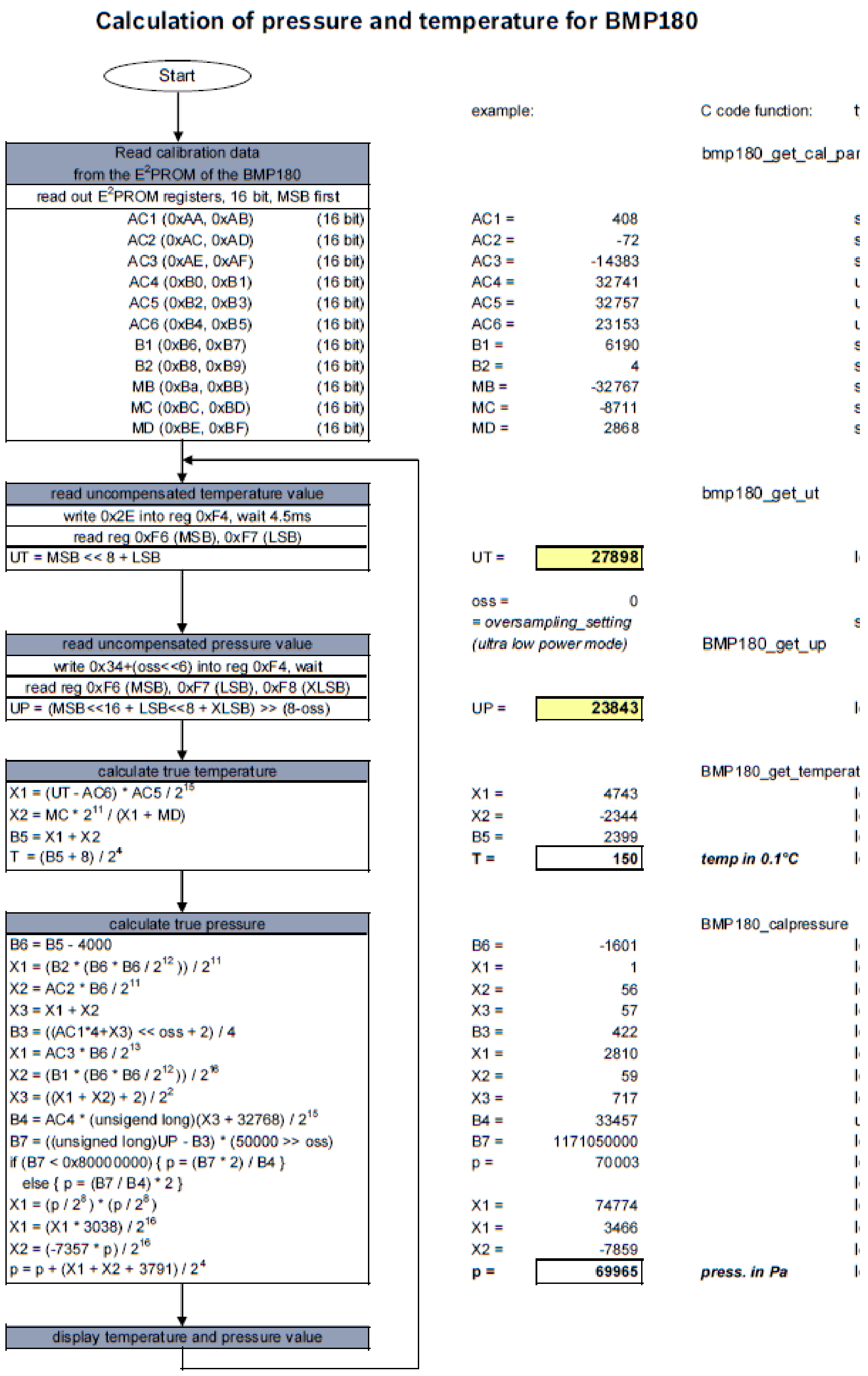
   | BMP180 Coefficients | |
   |---|---:|
   | ac1 | 8261 |
   | ac2 | -1127 |
   | ac3 | -14723 |
   | ac4 | 32456 |
   | ac5 | 24356 |
   | ac6 | 19323 |
   | b1 | 5498 |
   | b2 | 59 |
   | mb | -32768 |
   | mc | -11075 |
   | md | 2432 |

   Reverse the conversion of algorithm from raw data to pressure and temperature data and save to a .csv file.

4. Copy .csv file to emulator, connect emulator to altimeter and run.
5. The altimeter flight profile should match the starting point flight profile.

## 2.2 Conversion of raw data to pressure

For completeness this section walks through converting the raw data to pressure. That is the algorithm the altimeter uses. For the emulator, the reverse process is needed to create the data set that it sends to the altimeter. That will be covered in the next section. The algorithm for converting the raw data from the sensor to pressure is published in the sensor data sheet.



**Calculation of pressure and temperature for BMP180**

| | example: | | C code function: | t |
|---|---|---|---|---|
| Start | | | | |
| Read calibration data from the $E^2$PROM of the BMP180 | | | bmp180_get_cal_par | |
| read out $E^2$PROM registers, 16 bit, MSB first | | | | |
| AC1 (0xAA, 0xAB) (16 bit) | AC1 = | 408 | | s |
| AC2 (0xAC, 0xAD) (16 bit) | AC2 = | -72 | | s |
| AC3 (0xAE, 0xAF) (16 bit) | AC3 = | -14383 | | s |
| AC4 (0xB0, 0xB1) (16 bit) | AC4 = | 32741 | | u |
| AC5 (0xB2, 0xB3) (16 bit) | AC5 = | 32757 | | u |
| AC6 (0xB4, 0xB5) (16 bit) | AC6 = | 23153 | | u |
| B1 (0xB6, 0xB7) (16 bit) | B1 = | 6190 | | s |
| B2 (0xB8, 0xB9) (16 bit) | B2 = | 4 | | s |
| MB (0xBa, 0xBB) (16 bit) | MB = | -32767 | | s |
| MC (0xBC, 0xBD) (16 bit) | MC = | -8711 | | s |
| MD (0xBE, 0xBF) (16 bit) | MD = | 2868 | | s |
| read uncompensated temperature value | | | bmp180_get_ut | |
| write 0x2E into reg 0xF4, wait 4.5ms | | | | |
| read reg 0xF6 (MSB), 0xF7 (LSB) | | | | |
| UT = MSB << 8 + LSB | UT = | 27898 | | l |
| | oss = | 0 | | |
| | = *oversampling_setting* | | | s |
| | *(ultra low power mode)* | | BMP180_get_up | |
| read uncompensated pressure value | | | | |
| write 0x34+(oss<<6) into reg 0xF4, wait | | | | |
| read reg 0xF6 (MSB), 0xF7 (LSB), 0xF8 (XLSB) | | | | |
| UP = (MSB<<16 + LSB<<8 + XLSB) >> (8-oss) | UP = | 23843 | | l |
| calculate true temperature | | | BMP180_get_temperat | |
| $X1 = (UT - AC6) * AC5 / 2^{15}$ | X1 = | 4743 | | l |
| $X2 = MC * 2^{11} / (X1 + MD)$ | X2 = | -2344 | | l |
| $B5 = X1 + X2$ | B5 = | 2399 | | l |
| $T = (B5 + 8) / 2^4$ | **T =** | **150** | ***temp in 0.1°C*** | l |
| calculate true pressure | | | BMP180_calpressure | |
| $B6 = B5 - 4000$ | B6 = | -1601 | | l |
| $X1 = (B2 * (B6 * B6 / 2^{12})) / 2^{11}$ | X1 = | 1 | | l |
| $X2 = AC2 * B6 / 2^{11}$ | X2 = | 56 | | l |
| $X3 = X1 + X2$ | X3 = | 57 | | l |
| B3 = ((AC1*4+X3) << oss + 2) / 4 | B3 = | 422 | | l |
| $X1 = AC3 * B6 / 2^{13}$ | X1 = | 2810 | | l |
| $X2 = (B1 * (B6 * B6 / 2^{12})) / 2^{16}$ | X2 = | 59 | | l |
| $X3 = ((X1 + X2) + 2) / 2^2$ | X3 = | 717 | | l |
| $B4 = AC4 * (\text{unsigend long})(X3 + 32768) / 2^{15}$ | B4 = | 33457 | | u |
| B7 = ((unsigned long)UP - B3) * (50000 >> oss) | B7 = | 1171050000 | | l |
| if (B7 < 0x80000000) { p = (B7 * 2) / B4 } | p = | 70003 | | l |
| else { p = (B7 / B4) * 2 } | | | | l |
| $X1 = (p / 2^8) * (p / 2^8)$ | X1 = | 74774 | | l |
| $X1 = (X1 * 3038) / 2^{16}$ | X1 = | 3466 | | l |
| $X2 = (-7357 * p) / 2^{16}$ | X2 = | -7859 | | l |
| $p = p + (X1 + X2 + 3791) / 2^4$ | **p =** | **69965** | ***press. in Pa*** | l |
| display temperature and pressure value | | | | |

Figure 1 - Conversion of raw data to pressure algorithm from BMP180 datasheet

Applying this to raw data…

Uncompensated temperature value, UT = 27623

Uncompensated pressure value, UP = 40382

First calculate true temperature (this is the temperature of the sensor die)

X1 = (UT – AC6) * AC5 / $2^{15}$ = ((27623 – 19323)* 24356)$2^{15}$ = 6169

X2 = MC*$2^{11}$/(X1+MD) = (-11075*$2^{11}$)/(6169+2432) = -2637

B5 = X1 + X2 = 6169 + (-2637) = 3532

T = (B5 + 8)/$2^4$ = (3532+8)/16 = 221 which is degrees C x 10 or 22.1 C

Next calculate the true pressure

B6 = B5 – 4000 = 3532-4000 = -468

X1 = (B2*(B6*B6/$2^{12}$))/$2^{11}$ = (((59*(-468)*(-468))/4096)/2048) = 1

X2 = AC2 * B6 / $2^{11}$ = (-1127)*(-468)/2048 = 257

X3 = X1+ X2 = 1 + 257 = 258

B3 = ((AC1*4)+X3 << oss +2)/4 = ((8261*4)+258+2)/4 = 8326

X1 = (AC3*B6)/$2^{13}$ = (-14723)*( -468)/8192 = 841

X2 = (B1*(B6*B6/$2^{12}$))/$2^{16}$ = (5498)*((-468)*(-468)/4096)/65536 = 4

X3 = ((X1+X2)+2/$2^2$ = (841+4+2)/4 = 211

B4 = (AC4 * (X3 + 32768))$2^{15}$ = ((32456)*(211+32768))/32768 = 32664

B7 = (UP-B3)*(50000 >> oss) = (40382 – 8325) * (50000) = 1602800000

B7 < 0x80000000 so

P = (B7/B4)*2) = (1602800000/32664)*2 = 98138

X1 = (P/$2^8$) * (P/$2^8$) = 146689

X1 = (X1*3038)/$2^{16}$ = (146689*3038)/65536 = 6799

X2 = (-7357*P)/$2^{16}$ = -11016

P = P + (X1+X2+3791)/$2^4$ = 98138+ (6799 -11016+3791)/16 = 98712

## 2.3 Converting pressure back to uncompensated pressure and temperature

The above procedure is reversed for computing the uncompensated pressure and temperature from the pressure value. Any suitable temperature can be used, such as a value from 20 to 25 C. However, because the final value of pressure must be worked backwards to find the final X1 and X2 values, it is not as straightforward. A matrix of equations needs to be solved.

Equation 1: 98712 = Pold + (X1+X2+3791)/16

Equation 2: X2 = (-75357*Pold)/65536

Equation 3: X1 = (Pold*Pold)*3038/65536

Three equations with three unknowns, however equation 3 is nonlinear. Further investigation will be needed to determine if some adjustments or other techniques will be needed here.

## 3.0 Data Collection and Analysis

### 3.1 Creating the raw data arrays

The complexity of reversing the order of the equations to create the uncompensated pressure and temperature values was a surprise. Perhaps the process can be simplified if the coefficients were changed to simple values such that the calculations are easier. This needs to be investigated further.

### 3.2 Emulator software development

Currently the emulator board software is still under development. The code for operation of the I2C slave interface is still not operational. The main issue is compiler differences between the Atmel reference design for the I2C slave and the Winavr compiler. The incompatibility is not a major issue, but will take time to resolve.

## 4.0 Conclusions and Next Steps

The key next steps in this project are:

1. Resolve issues with the I2C slave interface code and get the link operational between the altimeter and the emulator.
2. Develop a workable algorithm for creating the uncompensated pressure and temperature values from the pressure values.
3. Once 1 and 2 are completed, create a series of altitude profiles of different altitudes, with noise and without to evaluate the altimeter performance of the ALT-BMP
4. Repeat 1 through 3 with the MS5611 and MS5607 sensors so that the Perfectflite and Micropeak altimeters can also be tested.
5. Develop a test procedure for altimeter performance evaluation using the sensor emulator.
6. Use the data collected to find how well different altimeters handle spikes in the data caused by ejection and to determine robustness of launch detection.

## Appendix A – References

1. Kidwell, Christopher and Ash-Poole, Jennifer 2004, A Comparison of Altimeters and Optical Tracking. R&D report for Mostly Harmless for NARAM 46 http://www.narhams.org/library/rnd/Altimeters.pdf
2. Kidwell, Christopher and Ash-Poole, Jennifer 2005, Measuring Absolute Accuracy of Altimeters R&D report for Slightly Harmful for NARAM 47
3. Curcio, Larry 2008, Analysis of Flight Computer Data From Off-Vertical Trajectories
4. Curcio, Larry 2008, Using Numerical Methods to Explore Barometric Flight
5. Schulz, David 2004, Application of the Kalman Filter to Rocket Apogee Detection
6. Graham Jackson and Chris Crocker, 2000, The Use of Altimeters in Height Measurement web document: http://www.hills-database.co.uk/altim.html
7. U.S. Standard Atmosphere 1976, U.S. Government Printing Office, Washington, D.C., 1976
8. United States Committee on Extension to the Standard Atmosphere, U.S. Standard Atmosphere 1976, web document: http://modelweb.gsfc.nasa.gov/atmos/us_standard.html
9. Kansas Flyer Web Site, Altitude Theory, web page: http://www.kansasflyer.org/Documents/AltitudeTheory.pdf
10. Texas Instruments, 1998, Low-Power Signal Conditioning for a Pressure Sensor, Application Note SLA0034, web page http://www.ti.com/lit/an/slaa034/slaa034.pdf
11. WinAVR Website: http://winavr.sourceforge.net
12. LUFA Website: http://www.fourwalledcubicle.com/LUFA.php
13. Atmel Website: http://www.atmel.com
14. Bosch Sensortec Website: http://www.bosch-sensortec.com
15. Freescale Website: http://www.freescale.com
16. Measurement Specialties, Website: http://www.meas-spec.com
17. ST Micro Website: http://www.st.com
18. CSG Shop Website: http://www.csgshop.com
19. DataSheets and application notes from Atmel, Bosch Sensortec, Measurement Specialties, ST Micro for the development kits and devices used (key sheets included)
20. Altimeter suppliers websites and manuals from Adept, Adrel, Featherweight, G-wiz, Jolly Logic, Missileworks, Perfectflite, and Transolve
21. Wolf, Dan & Wolf, Mary, 2012, On the Use of Barometric Sensor Based Altimeters for NAR Altitude Contest Events
22. Wolf, Dan & Wolf, Mary, 2015, Further Studies on Barometric Sensor Based Altimeters for NAR Competition
23. Wolf, Dan & Wolf, Mary, 2016, Analysis of Altimeter Performance Using Vacuum Chamber Testing and Sensor Monitoring

## Appendix B – Equipment Used & Expenditures

1. Personal Computer (Already had)
2. Microsoft Excel (Already had)
3. Microsoft Word (Already had)
4. Soldering Iron and Solder (Already had)
5. Two Adrel ALT-BMP Altimeters with download unit (Already had)
6. Two AT90USBKEY development boards (Already had)

Total Expenses for project - none

## Appendix C – I2C Specification

I²C (Inter-Integrated Circuit), pronounced I-squared-C, is a multi-master, multi-slave, single-ended, serial computer bus invented by Philips Semiconductor (now NXP Semiconductors). It is typically used for attaching lower-speed peripheral ICs to processors and microcontrollers in short-distance, intra-board communication.

I²C uses only two signals (wires), Serial Data Line (SDA) and Serial Clock Line (SCL), pulled up with resistors. Typical voltages used are +5 V or +3.3 V. I²C bus speeds are the 100 kbit/s standard mode and the 10 kbit/s low-speed mode, but arbitrarily low clock frequencies are also allowed. Recent revisions of I²C can host more nodes and run at faster speeds (400 kbit/s Fast mode, 1 Mbit/s Fast mode plus or Fm+, and 3.4 Mbit/s High Speed mode).

I²C defines basic types of messages, each of which begins with a START and ends with a STOP:

- Single message where a master writes data to a slave;
- Single message where a master reads data from a slave;

Virtually all modern pressure sensors use the I2C protocol for interfacing with single messages as described above.

For more details on the I2C bus, refer to the I2C specification here:
http://www.nxp.com/documents/user_manual/UM10204.pdf