5-1-2010

Development of a high altitude balloon payload data collection, telemetry, and recovery system

Nathan Michael King

DEVELOPMENT OF A HIGH ALTITUDE BALLOON PAYLOAD DATA COLLECTION, TELEMETRY, AND RECOVERY SYSTEM

By

Nathan Michael King

A Thesis
Submitted to the Faculty of
Mississippi State University
In Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Aerospace Engineering
in the Department of Aerospace Engineering

Mississippi State, Mississippi

May 2010

DEVELOPMENT AND TESTING OF A HIGH ALTITUDE BALLOON PAYLOAD DATA COLLECTION, TELEMETRY, AND RECOVERY SYSTEM

By

Nathan Michael King

Approved:

Keith Koenig Professor of Aerospace Engineering (Major Professor)	Randolph F. Follett Assistant Professor of Electrical and Computer Engineering (Committee Member)
Jerry W. Bruce, II Associate Professor of Electrical and Computer Engineering (Committee Member)	Bryan A. Jones Assistant Professor of Electrical and Computer Engineering (Committee Member)
J. Mark Janus Associate Professor and Graduate Coordinator for Aerospace Engineering	Lori Bruce Associate Dean for Research and Graduate Studies

Name: Nathan Michael King

Date of Degree: May 1, 2010

Institution: Mississippi State University

Major Field: Aerospace Engineering

Major Professor: Dr. Keith Koenig

Title of Study: DEVELOPMENT OF A HIGH ALTITUDE BALLOON PAYLOAD DATA COLLECTION, TELEMETRY, AND RECOVERY SYSTEM

Pages in Study: 113

Candidate for Degree of Master of Science

High altitude balloons are an effective, inexpensive and readily available conduit for conducting near space and low Reynolds number experimentation. Experiments are being developed that will use high altitude balloons as carriers for near space and low Reynolds test vehicles. The first step in developing this capability is to create a system that is able to log collected data and track and control a high altitude balloon payload. It is also beneficial that this system be flexible enough to accept different sensor types, communication methods and connection and release linkages.

By combining the flexibility of microcontroller biased circuitry and the availability of commercial off the shelf products an economical design solution to this problem has been be achieved. Analysis of this system has been performed and the design has been fabricated, tested and specially modified to withstand the extreme conditions of high altitude flight.

Key words: balloon payload, data collection, high altitude balloon, telemetry

ACKNOWLEDGEMENTS

The author expresses his sincere gratitude to the many people without whose assistance this thesis could not have been completed. First of all, sincere thanks is due to Dr. Keith Koenig, my committee chairman, for the time and effort he spent to guide and assist me throughout the project and the writing of this document. The author would also like to thank Graham Mitchell of Digital-DIY and Brian Bremer for their help in the development of the firmware programs.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

MISSION OVERVIEW AND SYSTEM BACKGROUND

Project Context

It is the purpose of this project to serve as a preparatory step in the development of a system to explore aircraft flight in the lower Martian atmosphere. An airplane can provide information about Mars at levels of extent and detail that are intermediate to those provided by orbiters and rovers. NASA has been interested in developing this capability and has created one prototype vehicle that has undergone some Earth testing[1]. The current project is intended to support additional research into this concept.

Mars airplane concepts need to first be tested on or near Earth, but in an environment similar to what will be experienced at Mars. Nominal values of Mars surface pressure and density are 14.6 lbf/ft^2 and $2.93 \cdot 10^{-5}$ $slug/ft^3$, respectively[2]. There are only a few wind tunnels on Earth that can simulate these conditions for testing. However, these values do correspond to conditions in the Earth's atmosphere near 100,000 ft (more precisely, standard atmosphere at 110,000 ft and 102,000 ft, respectively). Consequently, if an aircraft can be operated at altitudes near 100,000 ft in the Earth's atmosphere it will experience aerodynamic loads similar to those it would encounter near the surface of Mars.

It is extremely difficult for an airplane to takeoff from the surface of the Earth and climb to 100,000 ft altitude. Only a few altitude record attempts by modified military aircraft have actually achieved this height, although the extremely large (247 ft wingspan) Helios unmanned solar airplane has ascended to 96,000 ft[3]. A small aircraft needs assistance to reach this altitude; this assistance can be provided by a high altitude balloon. For example, a half-scale version of the NASA proposed Mars aircraft has been carried to 103,500 ft by a balloon, released, successfully flown and recovered[1].

The present project focuses on the development of a high altitude balloon system. This system, or subsequent generations of this system, will eventually be incorporated into a high altitude balloon that will carry a fixed wing glider to altitudes near 100,000 ft above the Earth. Of particular interest here are control, communication, sensor, data acquisition and actuator subsystems.

The system discussed in this document is designed to be placed inside a payload box instead of a glider and flown to approximately 100,000 feet, using latex, or other high altitude balloon. At approximately 100,000 feet, the balloon ruptures and the payload descends using a parachute. During the flight the payload system performs three main tasks recording data, telemetering data, and releasing the balloon and parachute. To perform these tasks, five in-flight subsystems, or modules, have been created. They are the GPS/Telemetry Module, Servo Driver Module (SDM), Low Speed Module (LSM), High Speed Module (HSM), and the Ground Station Module (GSM). Another module exists on the ground and is used to receive the telemetered data and track the balloon, while in flight.

Payload Enclosure Design

High altitude ballooning is conducted throughout the world by a wide variety of amateur as well as governmental and corporate organizations. As an example, Edge of Space Sciences, or EOSS, is a non-profit worldwide organization that promotes amateur and university high altitude ballooning[4]. EOSS provides guidelines on successful strategies and methods for payload box and subsystem design and construction. The design described here is guided by EOSS principles where appropriate.

Throughout the flight, the balloon payload will be subjected to temperatures ranging from -60ºC to 30ºC. The balloon payload will also be subjected to moderate turbulence and to atmospheric pressures as low as 0.1 psi. Constructing the payload box to withstand these conditions requires a unique construction technique. Therefore, it is recommended that the payload box be made of foam and covered in a plastic covering such as Monokote. However, the payload box should have minimal gaps between pieces to allow for maximum heat retention. The payload box should not be completely airtight so that the internal box pressure does not differ greatly from the atmospheric pressure. If the box is built airtight it might explode when subjected to very low atmospheric pressures.

Because of the low temperatures that the balloon payload structure will be exposed to, special considerations must be used when selecting an adhesive for constructing the payload box and for mounting components. Most adhesives, such as epoxy and super glue, become very brittle at low temperatures; in contrast low temperature hot glue is able to withstand the low temperatures and maintain its bonding

strength. Larger components are mounted within the balloon box using hook and loop straps. When connecting components to a foam board using a hook and loop straps, it is best to glue a small piece of wood (such as a tongue depressor) that will more evenly distribute the force the strap will exert on the balloon. If the pieces of wood are not glued in place, it is likely that the strap will pull through the foam and the mounted component will become loose within the payload box.

To minimize the possibility of component damage when the payload box contacts the ground, the payload box has been designed with a piece of foam that runs crosswise within the box. This allows the components to be suspended in the center of the box and for them to be more protected from the cold and from landing damage. Figure 1.1 depicts the payload box and the cross brace as viewed from above.

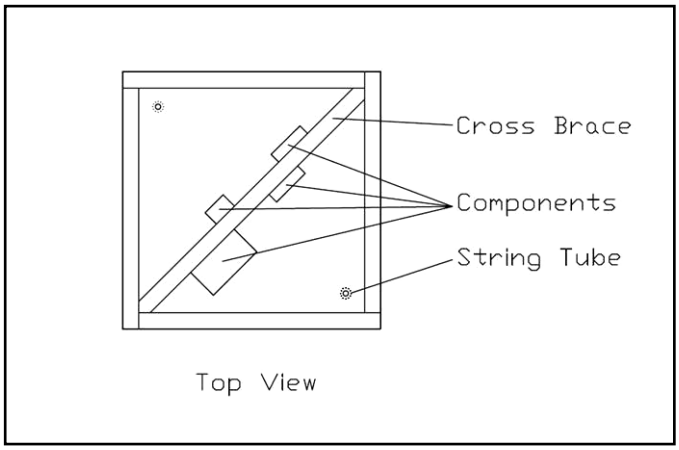

Figure 1.1 View from the Top of the Payload Box with the Payload Box Top Removed

To distribute the shock load of the parachute deployment a metal bottom should be added to the base of the balloon payload box. This metal piece will distribute the force exerted by the parachute strings on the payload box and prevent the sudden motion of the parachute deployment from pulling the mounting strings through the foam and "gutting" the payload box. Figure 1.2 shows the payload box as viewed from the side.

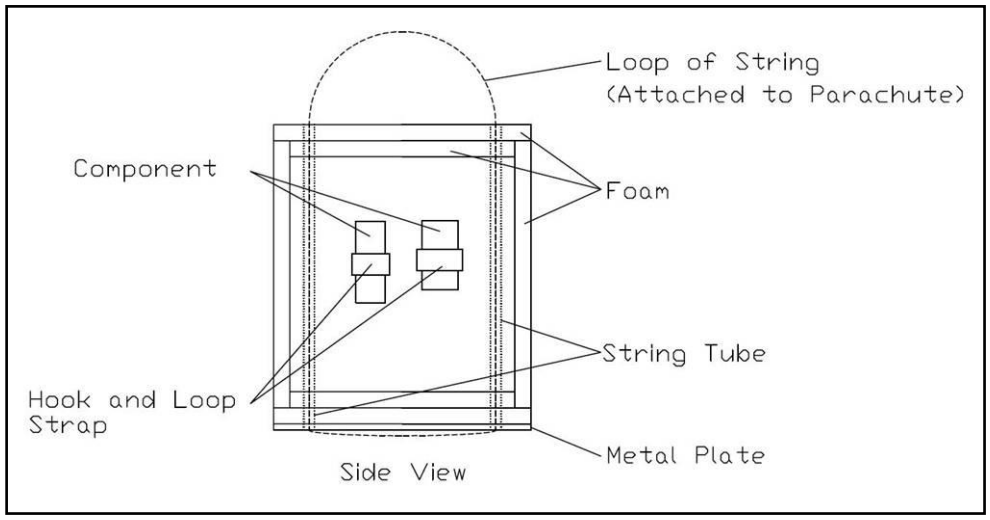

Figure 1.2 Side View Drawing of the Payload Box

The FAA sets certain guidelines that must be followed when performing a balloon launch. These guidelines are contained within Part 101 of the Federal Aviation Regulations[5]. These guidelines set weight, density, construction and other regulations for the operation of unmanned balloons. The specifics of the guidelines for launching an unmanned balloon will not be discussed in this document. It is the responsibility of the balloon operator to review and comply with these regulations.

Balloon and Parachute Release Mechanism

A typical high altitude balloon flight lasts approximately 3 hours. It takes about two hours for the balloon to reach its maximum altitude and it takes about one hour for the payload to descend with a parachute. To reduce the descent time, and the associated drift, the current system incorporates a parachute deployment mechanism that is triggered after the payload has reached its peak and then fallen through 70,000 feet. The balloon is released once it has ruptured and the descent has begun. This prevents the balloon remnants from entangling the parachute. Both the balloon release and the parachute deployment are programmed to be automatically triggered. However, it is possible to send a signal (through DTMF tones) from the ground station radio to trigger either the balloon release or the parachute deployment.

The release mechanism is used to release the balloon and to deploy the parachute. The release mechanism described, including the servo, represents two separate release mechanisms. These mechanisms are located on opposite sides of the balloon. The two release mechanism servos are connected to the same module output through a servo "Y-harness". Both mechanisms are operated using a standard hobby servo. (The details and use of the servos will be discussed in Chapter 3.) The actual release mechanism consists of a metal wire and a metal tube. The metal tube is mounted to a piece of wood that is mounted to the balloon payload box. Two gaps are cut through the metal tube and through the wood. A loop of string that connects the balloon to the payload box is placed in the gap. A second loop of string that keeps the parachute packed is placed through the second slit. The metal rod is then inserted through the tube, locking the loops of string in

place. When a release is triggered, the servo retracts the rod and the loop of string is released. Figures 1.3 and 1.4 show the details of the release mechanism. Figures 1.5, 1.6 and 1.7 show how the release mechanism operates using a side view perspective.

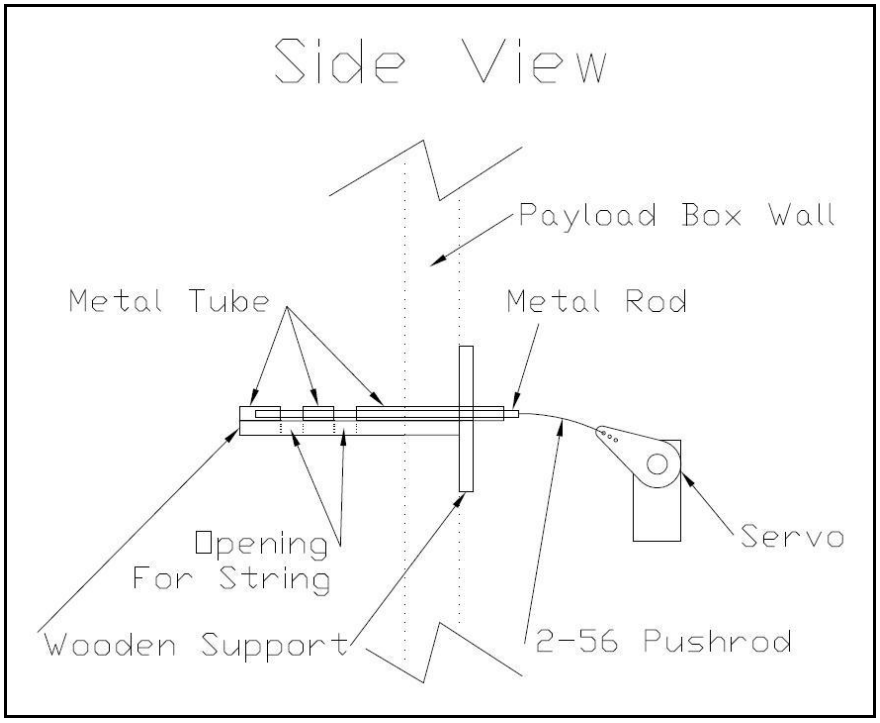

Figure 1.3 Side View Detail Drawing of Release Mechanism Design

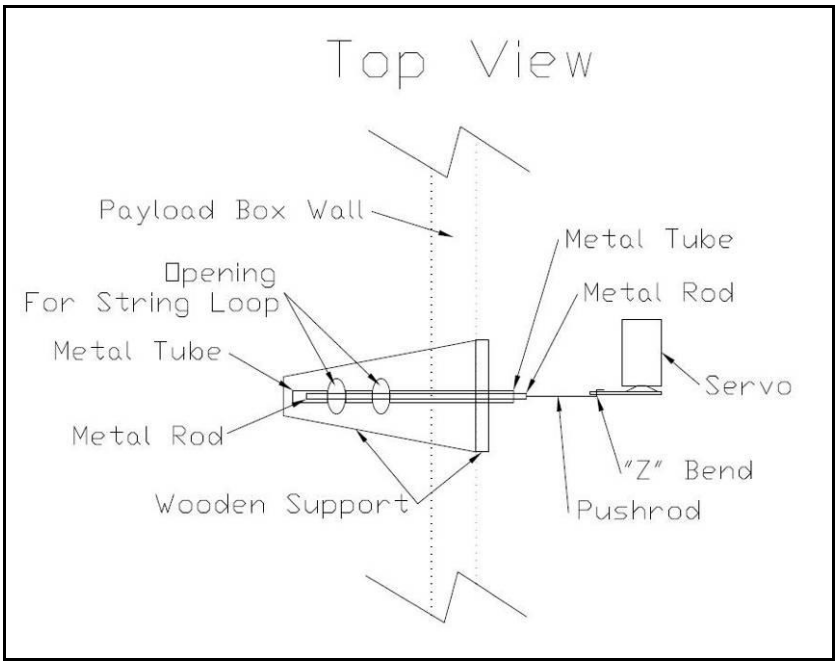

Figure 1.4 Top View Detail Drawing of Release Mechanism Design.

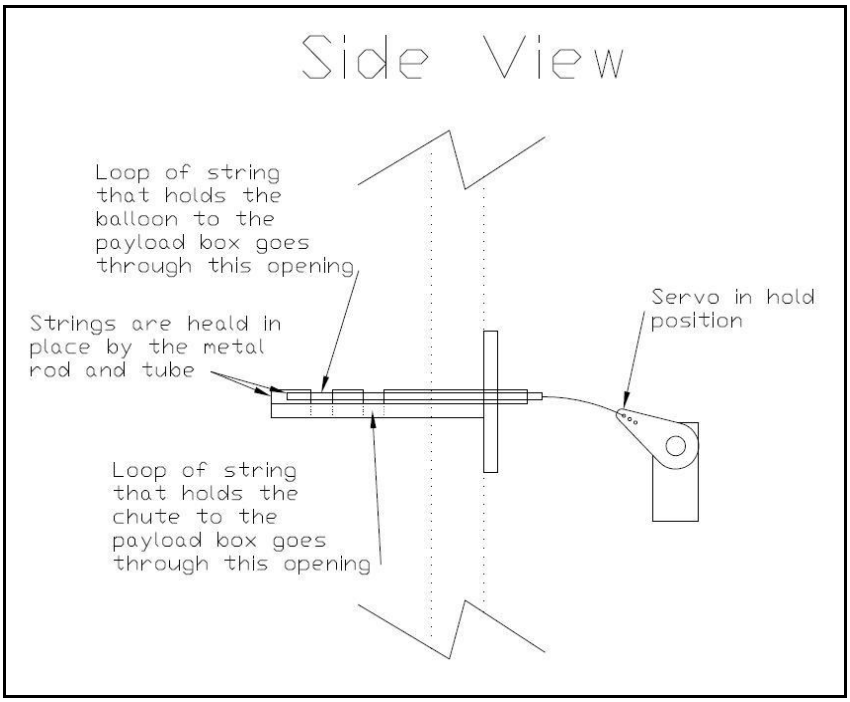

Figure 1.5 Release Mechanism Shown in Initial, Hold, Position

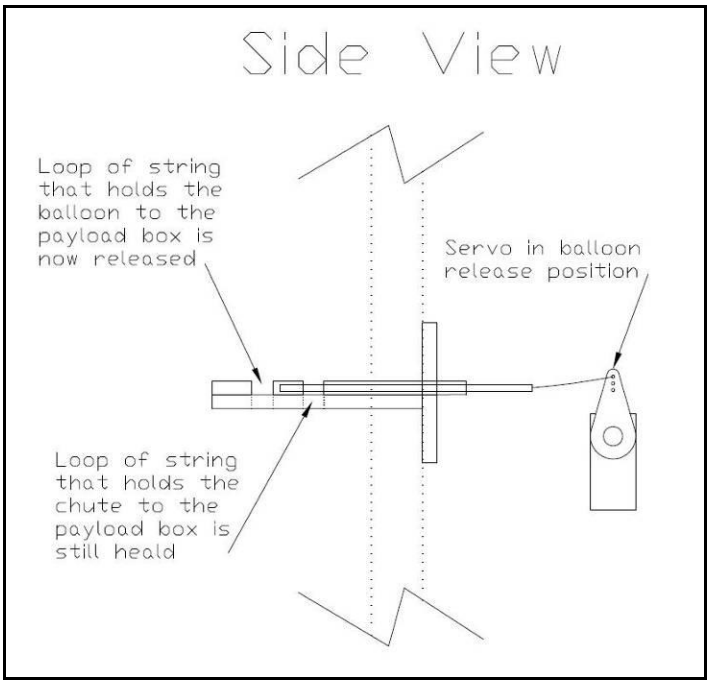

Figure 1.6 Release Mechanism Shown in Balloon Release Position

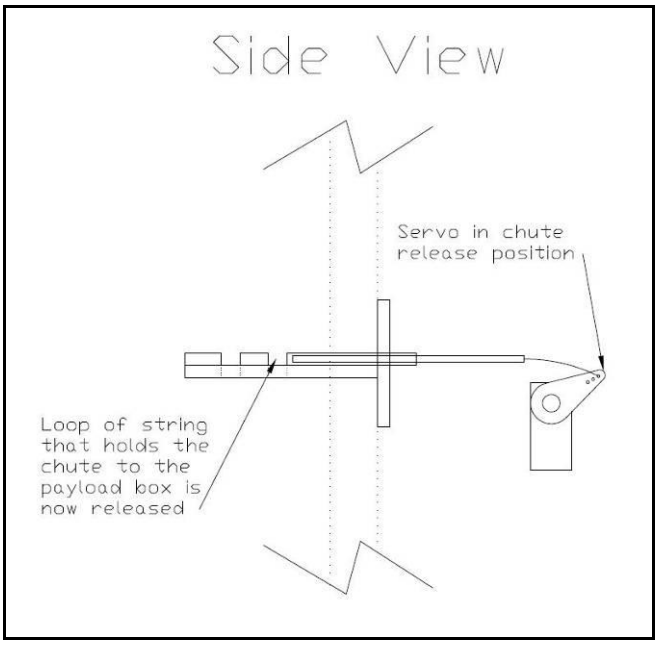

Figure 1.7 Release Mechanism Shown in Parachute Release Position

Figure 1.5 shows the release mechanism in its initial position. In this position the loops of string (not shown) are held in place by the metal rod. The balloon and parachute are held in place by the string connected to the loops. To release balloon, the servo is moved to its midpoint. This retracts the metal rod and allows the loop of string to pull through the opening, releasing the balloon. This release mechanism position is shown in Figure 1.6. Figure 1.7 shows the release mechanism in its final position. When a parachute release or a release all is commanded the servo moves to its final position. In this position the string holding the parachute to the side of the balloon is released, deploying the parachute. The parachute is still attached to the payload by the loop of string shown in Figure 1.2.

System Breakdown and Module Overview

This section will discuss the organization of the payload box and give a general description of the subsystems (modules). The payload data recording and telemetry system has been broken down into four flight modules and one ground module. The flight modules are contained within the payload box attached to the balloon and the ground module is located on the ground. Figure 1.8 shows the different modules and the general flow of information throughout the system.

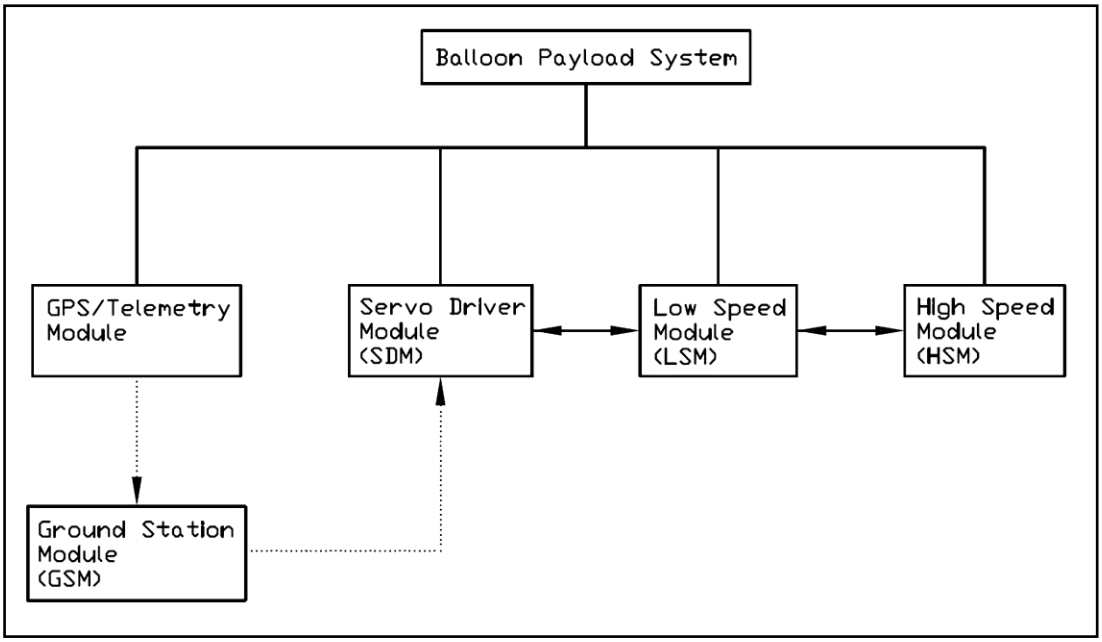

Figure 1.8 Payload System Breakdown and Information Paths

The Ground Station Module is connected to the GPS/Telemetry Module and the Servo Driver Module by dotted lines, signifying that there is no physical link between GSM and the other flight stations. Information is sent/received by the GSM by using a ham radio 2 meter transceiver. Information is exchanged among the flight modules through wire connections.

The Ground Station Module (GSM) is used to receive telemetered data from the balloon payload. Among the data sent are the balloon's ground track position and its altitude. This data is used by recovery teams on the ground to track and recover the balloon payload after the flight. The ground station may also trigger a release of the balloon or the parachute in the event that the payload system malfunctions.

The GPS/Telemetry Module relays collected data to the ground station. The main purpose of this information is to track and recover the payload. The GPS/Telemetry module also relays payload temperature, payload battery voltage and one other analog voltage to the ground.

The Servo Driver Module (SDM) controls the release servo. The SDM receives commands from the ground station that can be used to trigger a release or perform another, yet to be determined action. The SDM may also receive release commands from the Low Speed Module.

The Low Speed Module (LSM) is the most complex module. It reads data from various analog and digital sensors and records this data to an EEPROM array. The recording rate is 1 Hz. The data can be downloaded and analyzed once the flight has finished and the payload box is recovered. This module also provides timing information to the High Speed Module.

The High Speed Module (HSM) samples a 6 degree of freedom inertial measurement unit (IMU). The IMU collects acceleration and angular rate information about the balloon payload box. This information is recorded by the HSM to an EEPROM array so that the data can be downloaded and analyzed once the flight has finished and the payload box is recovered.

Comparison to Previous Systems

There have been two systems that were previously assembled and flown earlier in the ballooning program. The first system consisted of a simple GPS/TNC module without digital data telemetry. The only telemetry data (other than GPS) that was sent was an analog temperature tone. This tone was generated using a 555 timer connected to a thermistor. As the temperature changed, the thermistor resistance value changed and the tone responded. The tone was received and analyzed on the ground using expensive LabVIEW software.

The transmitter used in the first system, was a Kenwood TH-D7A. This transmitter has an internal TNC. The cut down mechanism for this payload was a hotwire cut down. During the flight, failures of the TH-D7A and the hot wire cut down occurred. Transmissions from the TH-D7A were sporadic after launch and completely stopped after 20 minutes. It was later discovered that other ballooning programs had also experienced reliability issues with the TH-D7A when used in similar systems.

This flight failure, combined with the reliability and cost associated with the TH-D7A component prompted a telemetry system redesign. The second payload system used a TinyTrak3 TNC and a TH-K2A transceiver for telemetering data. The flight was moderately successful. The payload was tracked to an altitude of over 90,000 feet. However, the GPS telemetry signal was lost during the descent at an altitude of 20,000 feet due to battery depletion. This prevented recovery of the payload.

These flights showed that a fully integrated, reliable and affordable system needed to be designed, in house. It was also determined that it was necessary to develop an

integrated, task specific system to achieve the goal of a stable, reliable balloon payload system. A system comprised of both COTS (commercial off the shelf) and engineered circuit components was developed to realize these goals, while reducing overall system cost. A more specific battery analysis was also performed to remedy the problems of the second flight.

The system discussed here is improves on previous systems by providing a fully capable telemetry, data logging, and communication solution. Costs have been reduced through component selection. Previous systems contained no onboard data collection. The new design records data from multiple data sources and sensors. The mechanical cut down mechanism eliminates the power and fragility problems of a hot wire cut down. Using the newly developed TinyTrak4 allows for building on the proven success of the TNC system while adding the ability to telemeter other sensor data along with the GPS data. Battery testing and current draw calculations were performed on this system to reduce the probability of repeating the second flight failure.

PIC Microcontrollers

The SDM, LSM and HSM must be built from electronic components. A circuit diagram and detailed system explanation is contained in the chapter of each module. However, each one of these modules has one, very important, component in common. The SDM, LSM and HSM all use a PIC Microcontroller to control the module. A microcontroller is basically a very small computer that is capable of being programmed and carrying out simple computing tasks. Microcontrollers have become very common and are found in devices such as calculators, automobiles and televisions.

A microcontroller must be loaded with an assembly language program. This program is run once the microcontroller is powered. In the event of a power interruption, the microcontroller will reset and run the program from the beginning. The assembly programming language can be very hard to understand and follow. This is why assembly compilers have been created. A compiler is simply a computer program that transforms a program written in one computer language into another. Many compilers have been created for use with the PIC microcontrollers. The Swordfish Compiler is a compiler that converts code written in Swordfish BASIC (very similar to most forms of BASIC) into an assembly program that can be run on series 18 PIC's. All of the code written for the SDM, LSM and HSM is written in Swordfish BASIC and compiled using the Swordfish Compiler. The programs written for the SDM and HSM can be compiled using the free evaluation version of the Swordfish Compiler (Swordfish SE). The LSM record program must be compiled using the full version of the Swordfish Compiler, because this program uses a large amount of PIC RAM.

Once the program has been written in Swordfish BASIC it must be loaded onto the microcontroller. Microchip, the makers of the PIC microcontrollers, provides a free program to do this. It is named MPLAB. This project was completed using MPLAB IDE V8.10. The MPLAB program must be setup to work with the Swordfish Compiler. Instructions for this are included on the Swordfish Compiler website. Once the Swordfish Compiler has been setup to run with MPLAB and a program to be downloaded to an 18 series PIC has been written, MPLAB is used to call on the Swordfish Compiler, compile the program to assembly, and then download the program into the PIC.To do this, the PIC must be connected to the computer running MPLAB through a PIC programmer. The PIC programmer used for this project is the PICkit 2. This programmer can also be used in conjunction with the PICkit 2 software to provide a UART interface with the PIC microcontroller. The following is a list of the steps to programming and checking the program run with a PIC.

- Write a program using the Swordfish editor.
- Connect the PICkit 2 to the microcontroller and to the computer.
- Open the MPLAB IDE program.
- Compile (Build) the program and program the microcontroller using MPLAB.
- Close MPLAB and Open the PICkit 2 software
- Open the PICkit 2 software UART interface to view UART communication sent by the microcontroller.

More information about using MPLAB, the Swordfish Compiler and the PICkit 2 software can be found through either the Microchip website, the Swordfish Compiler Forum or through digital-diy.com. Most of the Swordfish programming libraries used in the development of this system are available with the Swordfish compiler. The other libraries have been developed by digital-diy.com. All of the libraries used are available through the Department of Aerospace Engineering at MSU.

Battery Selection

Possibly the most important component in the balloon payload box is the flight battery. This battery provides power to all of the flight systems including all of the modules, the release servo, the GPS and the radio transceiver. It is important that the battery chosen be at a voltage that is compatible with all of these components and have enough capacity to run all of these systems for at least 4 hours. It is also important that the battery consists of a chemistry that is tolerant of cold temperatures and be light weight. The battery chosen for use with the balloon payload system is a Lithium Ion 7.2V 5200mAh pack. This pack consists of four 3.6V 2600mah cells. These cells are wired in a 2S2P configuration, meaning that two sets of two cells are wired in series. Then the two sets are wired in parallel. Figure 1.9 shows the battery pack cell configuration.

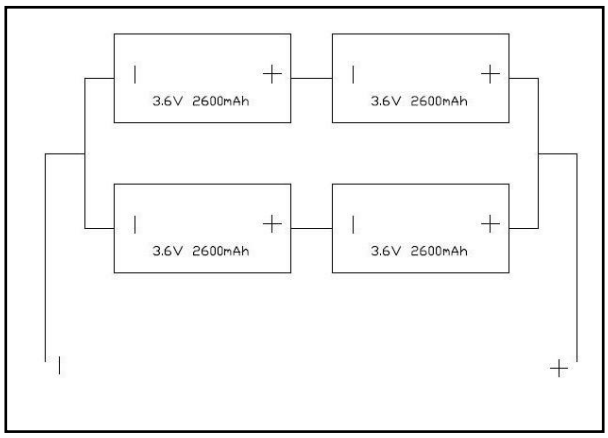

Figure 1.9 Battery Pack Cell Configuration

It is important to test the system with the flight battery at the temperatures that are expected during the flight. For this test, the completed system was taken to the Gulfstream High Altitude Test Chamber in Savannah, GA. The entire payload box was placed inside the high altitude chamber and turned on. For three hours the payload box and the electronics inside were subjected to pressures and temperatures that simulated a balloon flight to 100,000 feet and return. Once the simulation was complete, the once fully charged payload system battery was allowed to stabilize to room temperature, and then was recharged. During the recharge the payload battery received 1863mAh. This is a good approximation to the amount of battery capacity that was used during the simulation. According to the 3 hour test the battery should be able to power the payload electronics for a period of over 8 hours, showing that the 7.2V 5200mAh Lithium Ion battery pack is appropriate for use with the payload system. A secondary, component buildup/duty cycle, analysis is provided in Appendix A.

Payload Environment

Thermal

A major challenge faced by high altitude balloons is the low temperature that exists at the altitudes at which they are intended to operate. An analysis of the temperature in the payload box as the balloon and box ascend has been performed and is described here.

The equation for temperature in the payload box as a function of time will now be derived. The box is assumed to be a cube with constant thickness walls. Airflow into and out of the box will be neglected. The fundamental equation is the energy equation applied to a control volume which has a control surface composed of the interior surface of the box. The analysis will be done for the balloon ascent.

Since the box has constant contents and size the energy equation reduces to

$$\frac{d}{dt}(thermal\ energy) = \dot{Q}_{source} - \dot{Q}_{conduction} \quad (1)$$

where the source is heat dissipated by the electrical components within the box. Conduction is heat conducted through the walls of the box.

The thermal energy is

$$thermal\ energy = M \cdot C_v \cdot T \quad (2)$$

where M is the mass of air inside the box, C_v is the specific heat at constant volume of the air in the box and T is the temperature inside the box. M and C_v are assumed to be constant.

The rate at which heat is conducted through the walls is

$$\dot{Q}_{conduction} = k \cdot \frac{\Delta T}{\Delta x} \cdot A \tag{3}$$

where

$$\Delta T = T - T_{atm} \tag{4}$$

and T_{atm} is the local atmospheric temperature. The quantities k, Δx and A are the conductivity, thickness and effective surface area of the walls of the box, respectively. The surface area and thickness appear in Eq. (4) in the form A/□Δx, which is the effective conduction length L_c. This length has been determined for rectangular boxes[6] in terms of their dimensions and is

$$\frac{A}{\Delta x} = L_c = 6 \cdot \frac{S^2}{\Delta x} + 6.48 \cdot S + 1.2 \cdot \Delta x \tag{5}$$

when evaluated for a cube with side S.

Equations (2) through (5) are substituted into Eq. (1) to give

$$`M \cdot C_v \cdot \frac{dT}{dt} = \dot{Q}_{source} - k \cdot L_c \cdot (T - T_{atm}) \tag{6}$$

This is a first order, ordinary differential equation for the temperature in the box, T, as a function of time.

One step remains before Eq. (6) can be solved. As the balloon climbs it will experience a continuously changing atmospheric temperature. The atmospheric temperature, T_{atm}, must be expressed as a function of time. This expression is found by evaluating the following integral.

$$T_{atm} = \int \frac{d}{dt}(T_{atm}) \cdot dt + C \tag{7}$$

C in Eq. (7) is a constant of integration. The time derivative in Eq. (7) is evaluated using the chain rule.

$$\frac{d}{dt}(T_{atm}) = \frac{d}{dh}(T_{atm}) \cdot \frac{dh}{dt} \tag{8}$$

The derivatives on the right hand side of Eq. (8) are obtained with the standard atmosphere model and experimental data on balloon ascent rates.

The Earth's temperature varies with altitude in a series of nearly linear segments. The standard atmosphere model[7] is

$$T = 288.16\ K - 0.0065\ \frac{K}{m} \cdot h \quad \text{for h} \leq 11{,}000 \text{ m}$$

$$T = 216.66\ K \quad \text{for 11,000 m} \quad \text{for 11,000} < \text{h} \leq 25{,}000 \text{ m} \tag{9}$$

$$T = 141.66\ K + 0.003\ \frac{K}{m} \cdot h \quad \text{for 25,000} < \text{h} \leq 47{,}000 \text{ m}$$

The temperatures at the end points of the segments are used to find the constant of integration in Eq. (7). The segment temperature expressions also give

$$\frac{d}{dt}(T_{atm}) = -0.0065\ \frac{K}{m} \quad \text{for h} \leq 11{,}000 \text{ m}$$

$$\frac{d}{dt}(T_{atm}) = 0.0 \quad \text{for 11,000} < \text{h} \leq 25{,}000 \text{ m} \tag{10}$$

$$\frac{d}{dt}(T_{atm}) = 0.003\ \frac{K}{m} \quad \text{for 25,000} < \text{h} \leq 47{,}000 \text{ m}$$

The derivative dh/dt in Eq. (8) is the ascent rate of the balloon. Remarkably, a typical small high altitude balloon has a virtually constant ascent rate of about 300 m/min[8,9] until very near its peak altitude. Therefore, during the ascent

$$\frac{dh}{dt} = 300\ \frac{m}{min} \tag{11}$$

Equations (8) through (11) are substituted into Eq. (7) to give the temporal variation in atmospheric temperature that the balloon will experience as it ascends. This result is

$$T_{atm} = 288.16\ K - 0.0325\ \frac{K}{s} \cdot t \quad \text{for } t \leq 36.7 \text{ min}$$

$$T_{atm} = 216.66\ K \quad \text{for } 36.7 \text{ min} < t \leq 83.3 \text{ min} \qquad (12)$$

$$T_{atm} = 216.66\ K + 0.0150\ \frac{K}{s} \cdot (t - 83.3\ min) \quad \text{for } 83.3 \text{ min} < t \leq 157 \text{ min}$$

Equation (6) with Eqs. (5) and (12) has been solved for the case of a balloon payload box that is 0.305 m on a side and has a wall thickness of 0.0254 m. The walls are made of Owen Corning "pink" Foamular™ insulation board. This insulation has a conductivity k = 0.03 W/(m K)[10]. The box contains 0.033 kg of air (based on standard sea level conditions) at 15 C. The specific heat at constant volume is 717 J/(kg K). The power dissipated is assumed to be 1 W. This is based on assuming that a 7.2 volt battery is supplying an average of 0.4 amps to the systems in the box (see App. A) and that about 1/3 of this power is lost as heat. The solution of Eq. (6) for these conditions, done numerically in Mathcad with a time step of 1 second, appears in Fig. 1.10.

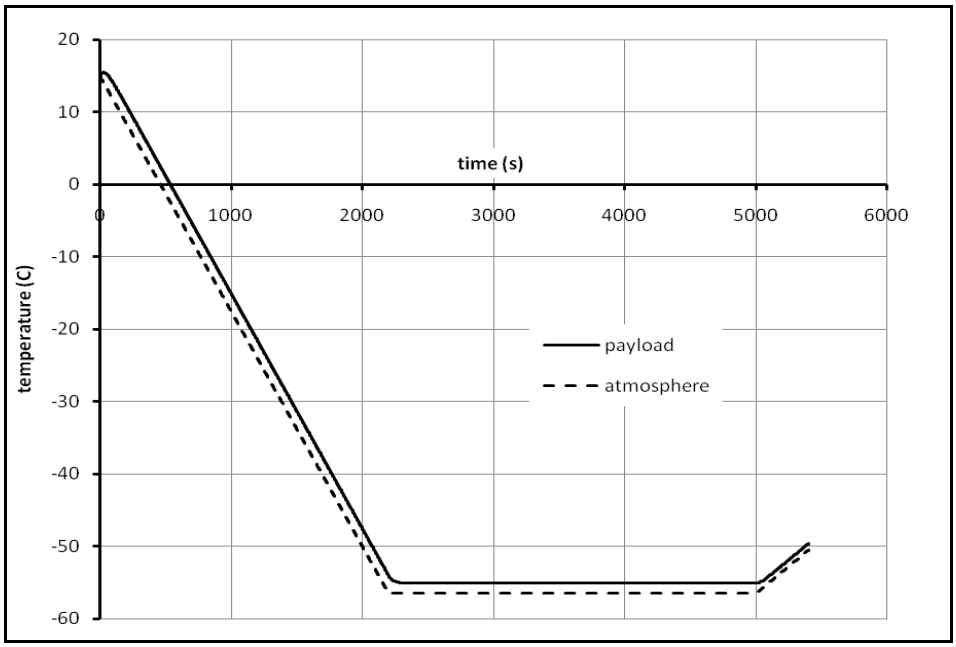

Figure 1.10 Temperature Time Histories for a Typical Balloon Ascent.

The heat from the electrical systems causes the box temperature to rise for a very brief period of time. This heat is not sufficient to balance the conduction losses once the balloon has gained a little altitude. Consequently, the box temperature follows the atmospheric temperature with only a very small offset. The average air temperature is below the operating values for typical electronic components (–40 C) during much of the ascent.

For comparison Figures 1.11 and 1.12 show results from a relatively recent EOSS flight, EOSS-138[11]. The balloon was launched from near Greeley, Colorado on May 9, 2009 and carried a payload for the National Oceanic and Atmospheric Administration. The altitude history appears in Fig. 1.11 and histories of internal and external temperatures are shown in Fig. 1.12. No information is available on the payload box for

this flight (nor for most of the other EOSS flights). Consequently, the internal temperature sensor location and heater details are not known. That there is a heater is fairly clear from the high internal temperatures.

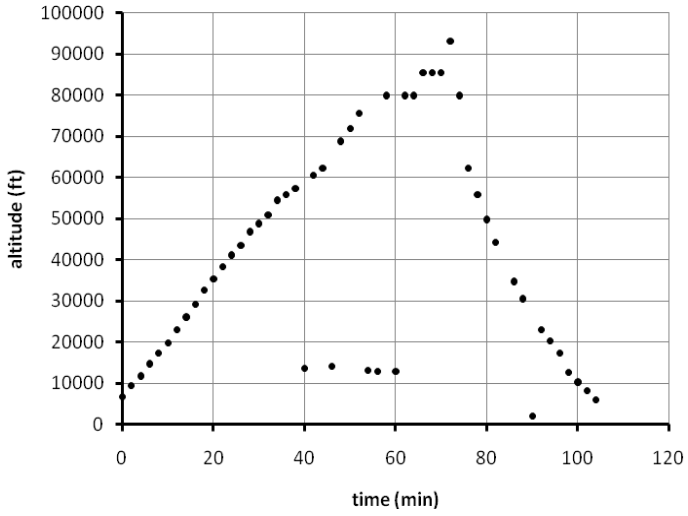

Figure 1.11 Altitude Profile of EOSS Balloon Flight May 9, 2009.

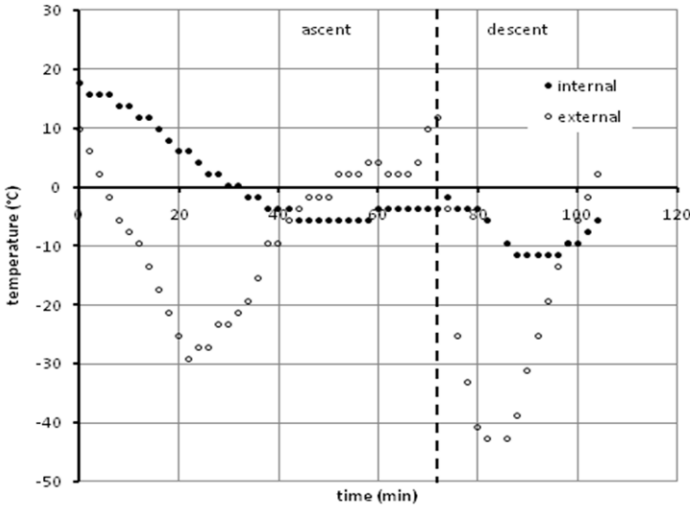

Figure 1.12 Temperature Time Histories of EOSS Balloon Flight May 9, 2009.

To gain a sense of the heater that would be required to create the temperature shown in Fig. 1.12, the calculations have been repeated for a box that has 1/3 the volume of the box used for Fig. 1.10. (The boxes used by EOSS tend to be considerably smaller than the one used in Fig. 1.10.) A 5 W heater was added to the 1 W heat dissipated to give a total source strength of 6 W. The result appears in Fig. 1.13.

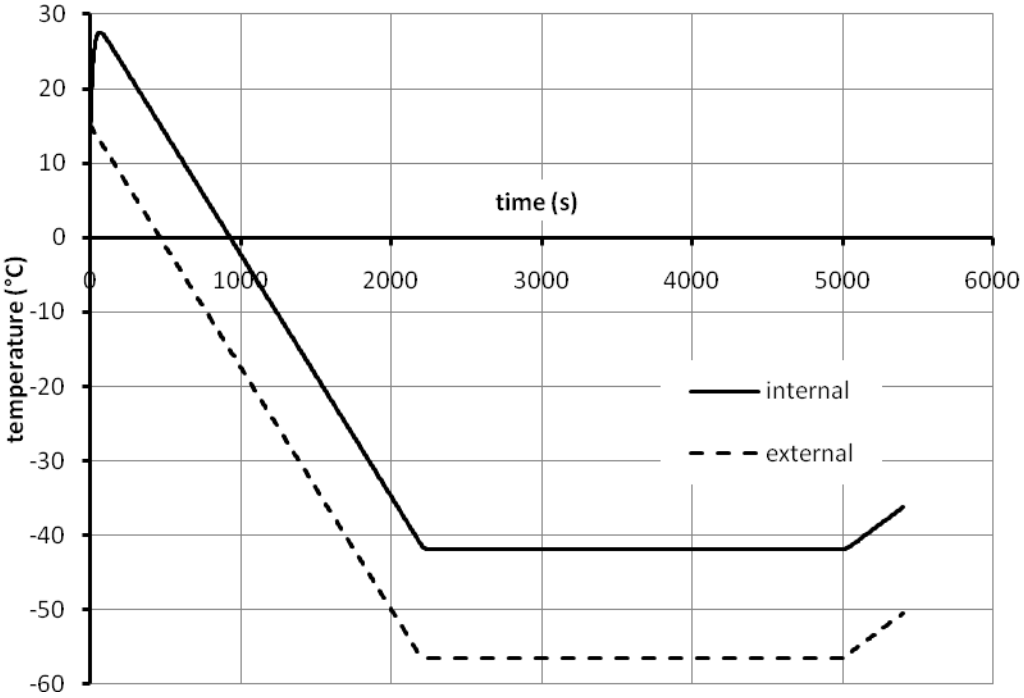

Figure 1.13 Calculated Temperature Time Histories for a Smaller Box with Heater.

There is a significant improvement in the internal temperature, although not to the level reported by EOSS-138. The minimum temperature in Fig. 1.13 is at the limit of most electronics, but may be tolerable. Judicious placement of a heater should eliminate temperature concerns.

Structural

The most severe structural situation the payload box will encounter in normal operation is the sudden load imposed upon it when the parachute deploys. An analysis of this situation has been performed to assess the adequacy of the design.

When the parachute deploys there will be a rapid increase in the aerodynamic drag acting on the system. The increase in drag over the value that exists without the parachute must be supported by the rope which encircles the payload box. The rope will transfer this load to the foam box. If the stress applied to the box by the rope is too large, the foam may collapse or be cut. The following analysis determines this stress and compares it to empirical data for the foam used here.

The analysis is performed by integrating Newton's second law for the box/parachute system during the descent. The temporal variation of altitude, velocity and drag are results of this integration. The drag increase during parachute deployment creates an impulse acting on the payload box and this impulse is found. Finally, the stress of the rope on the box is determined.

Newton's second law applied during descent is

$$M\frac{dV}{dt} = \frac{1}{2}\rho V^2 C_D A - Mg \qquad (13)$$

Here M is the mass of the box/parachute system, V is velocity (positive up), t is time, ρ is the air density, C_D is drag coefficient, A is frontal area and g is gravity. The air density changes as the system descends and the drag coefficient and frontal area change as the parachute is deployed. Variations in drag coefficient due to Reynolds number and

compressibility effects are neglected in the present analysis. Reynolds number and compressibility effects will make the chute opening impulse smaller. By neglecting these effects the solution here is conservative in nature.

A second equation describing the descent comes from kinematics.

$$\frac{dh}{dt} = V \tag{14}$$

Here h is altitude and is positive up from the ground. Equations (13) and (14) form a pair of coupled, nonlinear, first order, ordinary differential equations for h and V as functions of time. This system must be solved numerically.

Density variation with altitude is modeled here with a simple expression commonly used in space vehicle reentry analyses[12].

$$\rho = \rho_0 \cdot e^{-h/h_0} \tag{15}$$

Here ρ_0 is sea level density, 1.225 kg/m^3, and h_0 is scale height, 7,386 m.

Prior to parachute deployment the box has a drag coefficient C_{Db} and frontal area A_b. After the parachute has deployed and the flow field has adjusted there are new values, C_{Dp} and frontal area A_p. For the present analysis the drag coefficient and area vary linearly with time between these limits during the deployment process as shown in Fig. 1.14. The deployment begins at time t_{di} and ends at time t_{df}.

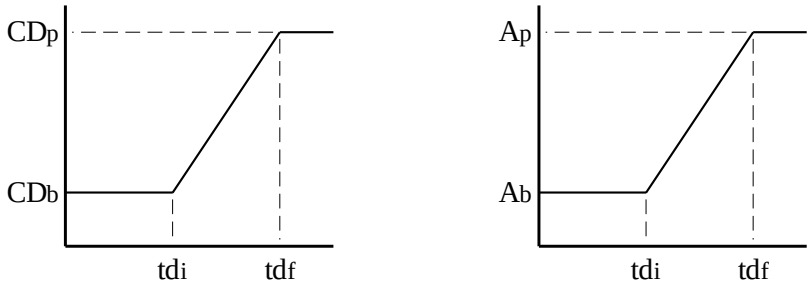

Figure 1.14 Drag Coefficient and Frontal Area Time Histories.

Equations (13) and (14) have been solved for the following conditions.

$M = 2.72\ kg \quad A_b = 0.929\ m^2 \quad A_p = 1.17\ m^2 \quad C_{Db} = 0.4 \quad C_{Dp} = 1.6 \quad t_{df} - t_{di} = 5\ s$

These conditions are typical of small payload/parachute combinations. The deployment time is based on experiments done at MSU which involved dropping payload/parachute combinations from a helicopter. The parachute deployment altitude was chosen to give a relatively severe deceleration load. For the solution the balloon starts its descent at 30,480 m (100,000 ft) and the parachute begins to deploy at 9,144 m (30,000 ft). The integration is numerical (in Mathcad) and the time step is 0.1 second. The results are shown in Fig. 1.15.

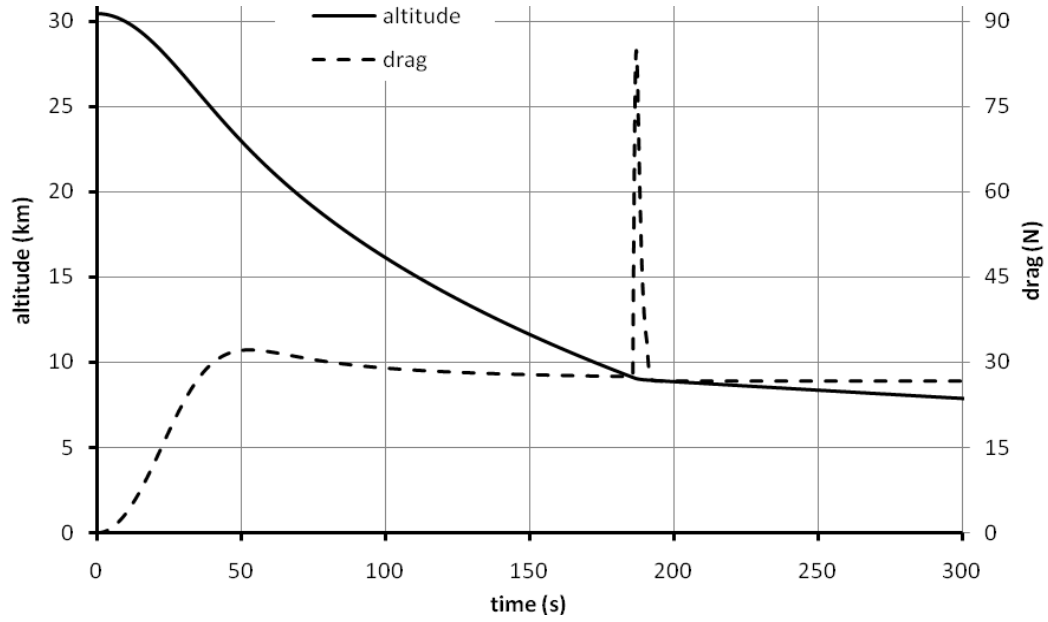

Figure 1.15 Balloon Descent with Parachute Deployment.

In these results the payload velocity immediately prior to the parachute release is 199 m/s and the drag force is 28 N. At the end of the 5 s deployment process the velocity has decreased to 11 m/s and the drag force is 32 N. The peak drag during the deployment is 85 N.

For a conservative estimate it is assumed here that the drag during the deployment process acts entirely on the parachute and is, therefore, transferred to the payload by the rope contacting the two components. (In reality, some of the drag acts directly on the box and will not be applied via the rope.) It is also assumed here that, because of the short time of action, the effective force applied to the box by the rope is the average of the drag during the deployment process, as given by Eq. (16).

$$F = \frac{1}{t_{df} - t_{di}} \int_{t_{di}}^{t_{df}} drag(t)\, dt \qquad (16)$$

Equation (16) gives 56 N for the trajectory in Fig. 1.12. The rope to be used will be at least 3 mm in diameter and will span the box, which here is a distance of 305 mm. These values give a stress distributed under the rope of 60,700 Pa. Experiments done at MSU have shown that the material from which the payload box will be made can handle concentrated linear compressive loads of more than 500,000 Pa before damage becomes noticeable. The conclusion is that the payload box should be more than capable of tolerating the deceleration stresses.

CHAPTER II

GPS / TELEMETRY MODULE

The GPS / Telemetry Module is a part of the balloon payload that decodes GPS signals received from the GPS satellites and relays the information to the ground station. Along with the GPS data, other information is also sent to the ground station. The information is received by the ground station and is used to track and monitor the balloon payload. The GPS / Telemetry Module also provides GPS time information to the Low Speed Data Collection Module.

The GPS / Telemetry Module is made of the following components:

- GPS Receiver – Garmin 18X LVC - Receives GPS signals and calculates position
- TNC – TinyTrak4 - Packets GPS and other data for transmission
- Radio Transceiver – Kenwood TH-K2 - Transmits Data to Ground Stations

These components are connected, as shown in the system diagram Figure 2.1, to form the GPS / Telemetry Module. Figure 2.1 shows only the basic system configuration of the GPS / Telemetry Module. The specific wiring details are located throughout the chapter.

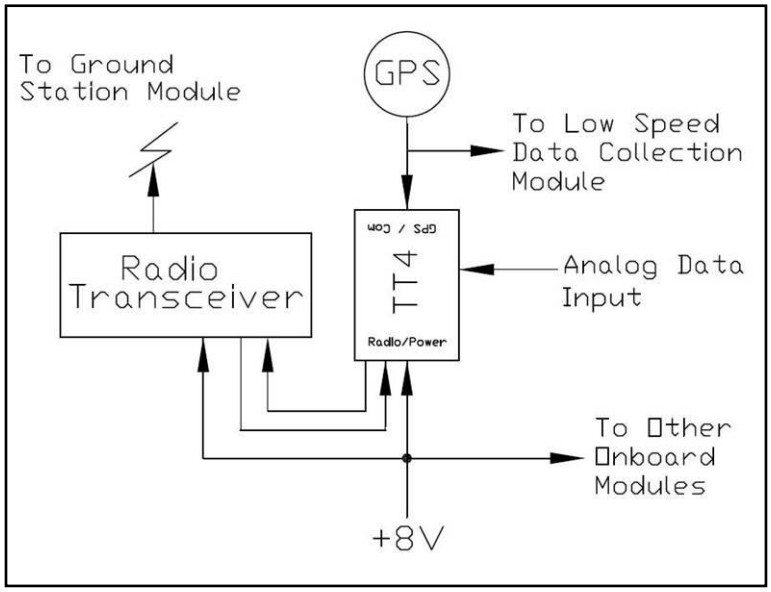

Figure 2.1 GPS/ Telemetry System Diagram

Power is provided to the entire system through the eight volt (7.2V nominal), Lithium Ion flight battery. Power for the GPS is regulated by the TinyTrak4 TNC to five volts. This system is the simplest module located in the balloon payload. The GPS receiver receives, decodes and sends GPS data to the TinyTrak4 and the Low Speed Data Collection Module. The GPS information is split by simply connecting it to both the LSM GPS input and the TinyTrak4 input. The TinyTrak4 also receives Analog Data from other sensors. This data is then formed into an APRS packet and transmitted to the ground station through the Radio Transceiver. This Module is quite simple to connect and wire. The complex aspect of this system lies in configuring the components to work together properly. The next three sections contain setup and connection information relating to the three main components of the GPS / Telemetry Module.

Garmin 18X LVC GPS Receiver Connections and Settings

There are some aspects of GPS systems and receivers that must be understood before the proper GPS receiver can be selected. GPS is a system that uses signals from multiple satellites to determine a three dimensional position on or above the earth. These signals are received by a GPS receiver that is mounted on the top of the balloon payload. The GPS receiver then decodes these signals and computes a position. For security reasons, GPS receivers that are available to the public can not report data if both the altitude is above 60,000 feet and the speed is above 1000 knots[13]. This restriction is only in effect if both the altitude and speed limits have been exceeded. Some GPS system manufacturers comply with these requirements by not allowing their GPS receivers to report position data if either the speed or altitude limit is exceeded. High altitude balloon flights usually exceed 60,000 feet. Therefore, it is important to verify that the GPS receiver used operates at altitudes greater than 60,000 feet. This is best done by contacting the GPS receiver manufacturer

The GPS receiver chosen for this balloon payload system is a Garmin 18X LVC. This GPS was chosen because of its small size, ease of configuration, and proven capability of operation at over 100,000 feet. Firmware of the 18X LVC earlier than version 2.80 contain a bug that would not allow position reporting of altitudes greater than 60,000 feet. This bug was corrected and all later firmware versions operate successfully above 60,000 feet. The LVC version was chosen because it operates on CMOS voltage levels (4-5.5 volts) and can communicate using either TLL or RS232 voltage level serial communication. The Garmin 18x LVC reports position data once

every second and reports data in standard National Marine Electronics Association (NMEA) sentences. NMEA is a standard communication protocol that is used by most GPS receiver manufacturers. This GPS receiver comes from the factory with a factory installed diagnostic connector. This connector will not work with the TinyTrak4 TNC. It must be removed, exposing the bare wires. If necessary, the wire length of the GPS can also be shortened at this time.

Connecting the Garmin 18X LVC to a computer or a TNC requires a female 9 pin serial connector, a soldering iron, and solder. Figure 2.2 shows the pin assignment for the female serial connector. This serial connector is soldered onto the Garmin 18X LVC leads. Table 2.1 describes the connections that must be made for the Garmin 18X LVC to be connected to either a TinyTrak4 or a computer. Wire color and pin assignments can change; it is best to use the product data sheet as a reference to ensure proper connection. The TinyTrak4 can be configured to supply +5V to the GPS through the J2 serial pin 4; however, when the GPS is connected to a computer, pin 4 should not be connected to the GPS and, instead, a +5V regulated power supply will need to be connected to the GPS input power pin.

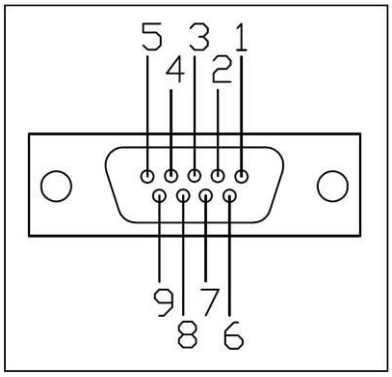

Figure 2.2 Pin Assignment as Seen When Looking into a Female Serial Connector

Table 2.1 Pinout for Connecting a Garmin 18X LVC to a Byonics TinyTrak4. Colors Correspond to Garmin 18X LVC Wire Colors.

Pin Number	**Connection**
2	GPS RX (White)
3	GPS TX (Green)
4	+5V (Red)
5	Ground (Black)
6,7,8,9	Not Connected

Once the serial connector has been attached to the GPS, without pin 4 being connected, the configuration settings can be altered by connecting the GPS to a computer. Configuration of the Garmin 18x LVC is done by first powering the GPS with a 5V regulated power supply and then connecting the GPS to a computer.

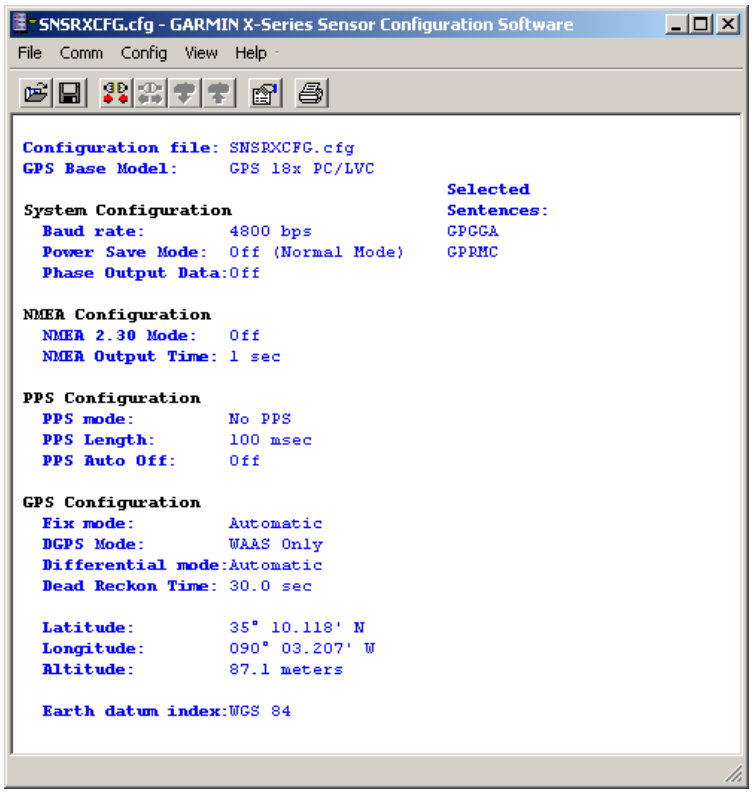

Figure 2.3 Configuration Software with the Settings Used.

Garmin X-Series Sensor Configuration Software is used to set the desired configuration. This software is available from the Garmin support website. A screen shot, including the settings used, is shown in Figure 2.3. Using the Garmin X-Series Configuration Software, the Garmin 18x LVC can be configured to send up to 12 different NMEA format sentences. The 12 NMEA sentences are unique, may contain different data and are usually ordered differently. Using the configuration software, baud rates between 480 and 11520 may be selected. It is very important that the GPS receiver is configured to send GPS data in a format and rate that are compatible with the devices that will receive the GPS data. The GPS used in this particular system was set to transmit

GPGGA and GPMRC NMEA sentences, at a rate of 4800bps. GPGGA and GPRMC format sentence examples appear below.

```
$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47
$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A
```

NMEA sentence fields are located between coma delimiters. Table 2.2 describes the fields located in the GPGGA and GPRMC sentence format NMEA, GPS sentences.

Table 2.2 GPGGA and GPRMC Sentence Fields and Labels

Field Number	**GPGGA**	**GPRMC**
0	NMEA sentence label	NMEA sentence label
1	Time HHMMSS	Time HHMMSS
2	Latitude	Status
3	Latitude Hemisphere N or S	Latitude
4	Longitude	Latitude Hemisphere N or S
5	Longitude Hemisphere E or W	Longitude
6	Fix Quality	Longitude Hemisphere E or W
7	Number of satellites being tracked	Speed over ground
8	Horizontal dilution of position	Course over ground
9	Altitude	Magnetic variation
10	Altitude units	Magnetic variation
11	Height of geoid	Checksum
12	Height of geoid units	
13	Age of DGPS data	
14	DGPS station ID	
15	Checksum	

At minimum, the GPGGA NMEA sentence must be sent because it contains altitude data, whereas the GPRMC NMEA sentence does not contain altitude data. In

addition to position, the GPRMC sentence contains ground speed and ground heading information, whereas the GPGGA sentence does not contain this information

GPS signals sent from a satellite are often skewed by the troposphere. This skew affects the accuracy of the GPS position information. To regain the lost accuracy, a second signal can be sent to correct the GPS position information. This second signal is referred to as a Differential GPS signal, or DGPS. The receiver is configured to use WAAS method DGPS, if WAAS is available. If a WAAS signal is not available, the receiver will not use other DGPS methods to increase the accuracy of the calculated position.

This Garmin 18X LVC GPS is capable of being configured to output a pulse every second. The pulse output timing is very accurate and is synched using GPS. This pulse is referred to as a PPS (pulse per second). Because the PPS signal is synched to the GPS satellite signal, it can be used as an external reference for timing of onboard devices. Figure 2.4 further explains the waveform of this pulse.

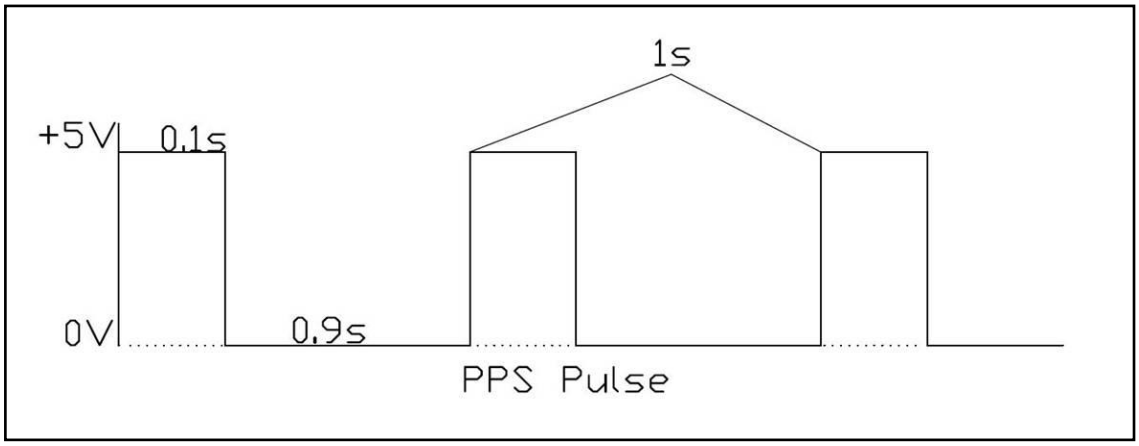

Figure 2.4 PPS Pulse Depiction

This pulse is sent from the GPS through the yellow wire. The length of the PPS pulse can be set using the Garmin X-Series Sensor Configuration Software. Further details of the PPS can be found in the Garmin 18X LVC datasheet.

Once configuration is complete, the GPS is disconnected from the computer and the external 5V power source. The GPS power lead is then connected to serial pin 4 on the female serial connector so that the GPS can receive its power from the TinyTrak4 TNC during flight operation.

TinyTrak4 TNC Device Overview and Tracker Settings

A TNC, or Terminal Node Controller, is a device that packs data so that it may be sent using a radio. A TNC is very similar to a radio modem. The TNC used in the GPS / Telemetry Module will receive GPS data from the Garmin 18X LVC and other analog data sources, package it and send it to the Ground Station Module, using the radio transceiver. The TinyTrak4 from Byonics was chosen because of its small size, reliability, ease of use, and its ability to relay analog data along with the GPS information. The TinyTrak4 is a TNC that is very small in size and is easily connected to a GPS receiver, radio, and power supply. The GPS receiver is connected to the TinyTrak4 through the serial connector that was described in the previous section. The TinyTrak4 is connected to the power supply and radio transceiver by using the Byonics HTK cable that is specially designed to connect the TinyTrak4 to radio transceivers made by Kenwood.

The TinyTrak4 requires a power supply between six and twelve volts. This supply is connected to the Byonics HTK cable through the included power pole style connector. However, because power pole connectors are not locking connectors these were replaced by Tamiya RC locking style connectors. The Tamiya connector is readily available from hobby dealers and is usually used in connecting remote controlled car batteries.

The TinyTrak4 is capable of supplying 5V, regulated, to the GPS. This eliminates the need of adding a 5V regulator to power the GPS receiver. The TinyTrak4 must be configured to supply power to a GPS by placing a jumper on the left side of JP6, located in the upper right corner of the TinyTrak4. Figure 2.5 shows a TinyTrak4 without the outer case and the locations of all connectors and jumpers. Table 2.3 provides a brief description of the pins and how they are configured or connected. The connectors and jumpers will be better explained throughout the chapter.

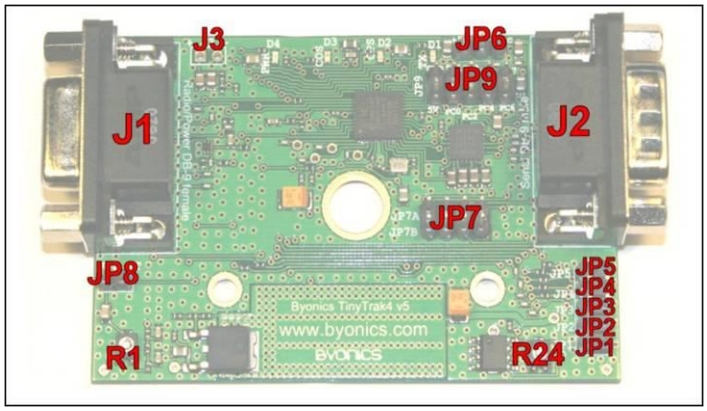

Figure 2.5 The TinyTrak4 without the Protective Enclosure. Photo from Reference 14.

Table 2.3 Listing and Brief Connection Explanation of all Connectors and Jumpers Located on the TinyTrak4

Connector / Jumper	**Description**	**Connection**
J1	Radio / Power Connector	Connected to radio and power using the Byonics HTK cable. Pin 4 Not Connected
J2	GPS / Computer Connector	Connected to the GPS using a created serial connector
J3	Alternate Power Input	Not Used
JP1	Used to select Tracker configuration (Primary or Secondary)	If grounded internally connected to J1 pin 4, Open
JP2	Analog Input	Not Used
JP3	Analog Input	Connected to LSM Chute release notification pin
JP4	Analog Input	Not Used
JP5	Analog Input	Internally connected to 2.2K pull up resistor, NC
JP6	Power Select for GPS	Left two pins should be connected for 5V
JP7	Secondary Serial Select Jumpers	Secondary Serial Port Not Used.
JP8	PTT mode select Jumper	Shorted, for use with Kenwood radios Open, for use with other radios
JP9	Digital Input Pins	Not Used
R1	Transmit Audio Level Pot.	Adjust as needed
R24	Temperature Sensor Calibration Pot.	Adjust as needed to calibrate TinyTrak4 on board temperature sensor.

The TinyTrak4 is a very versatile TNC. It can be configured for use as a ground station TNC or a Payload TNC. When received, the TinyTrak4 contains only diagnostic firmware. Its use as either a ground station TNC or Payload TNC is determined by the firmware downloaded to the TinyTrak4. The firmware used for the payload system is Tracker_v4.07. The Byonics Tracker firmware for the TinyTrak4 is very easy to use, because it is configured using the included Windows based interface. This interface allows two, completely independent, user determined, TinyTrak4 TNC configurations (Primary and Secondary) to be loaded to the TinyTrak4. Jumper JP1 located on the TinyTrak4 board is used to select between the two configurations. If the jumper is in place the TinyTrak4 uses the Secondary configuration. If the jumper is not in place it uses the Primary Configuration. Jumper JP1 will be used as an analog input, so the jumper will not be connected. The GPS / Telemetry Module TinyTrak4 TNC Primary and Secondary configurations will be set the same, because there is no need for the TNC to change configurations and it is very important that the settings are correct.

There are numerous settings that may be selected using the TT4 Tracker Config configuration program. This section will give a brief explanation of the settings chosen for the GPS / Telemetry Module. Figure 2.6 shows a screen shot of the TT4TrackerConfig.exe program. A more in depth description of the TinyTrak4 Tracker Configuration setting options is available from the TinyTrak4 Tracker documentation[13] The Alpha firmware, available from Byonics, may also be used to configure the TinyTrak4 for use as a payload TNC, but it must be configured using a command line interface through a terminal program, such as Tera Term Pro.

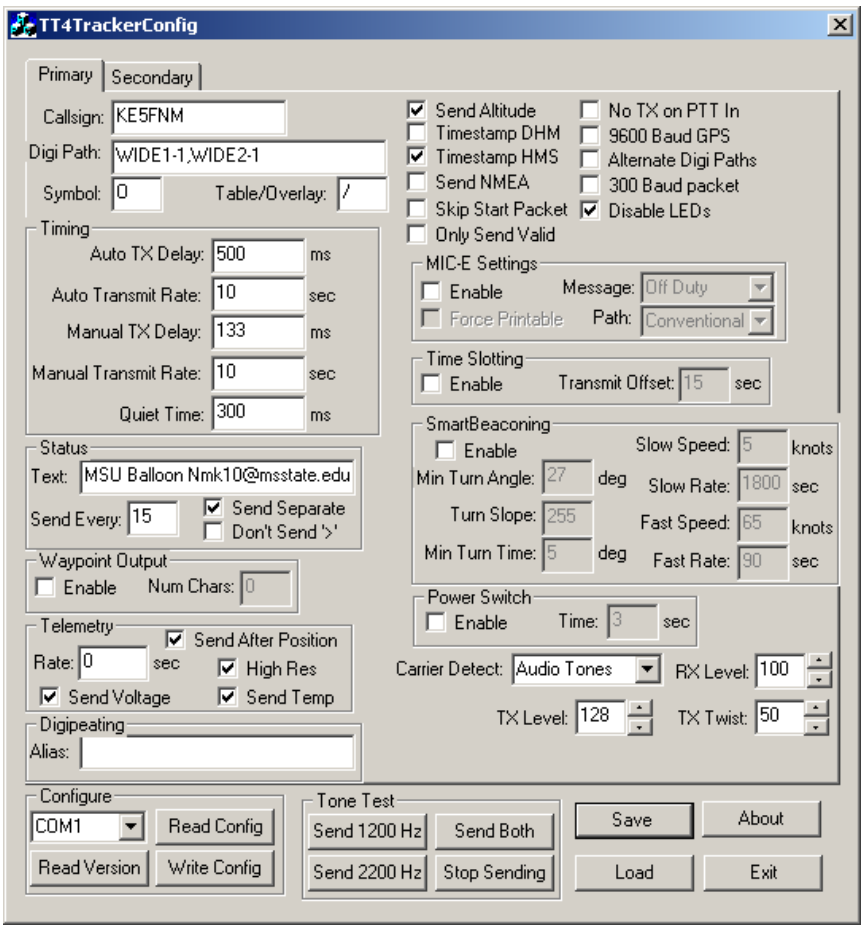

Figure 2.6 Screen shot of the TinyTrak4 Tracker Configuration Software, Including the Settings Used.

The settings outlined and shown in Figure 2.6 are preliminary settings for the tracker firmware and should be checked with the data sheet so that the TinyTrak4 is configured for desired operation. The first TinyTrak4 Tracker setting that must be chosen is the call sign. This is the user's amateur radio call sign. An amateur radio call sign is given by the FCC upon completion of the FCC technician license test. A call sign is unique, proprietary, and should not be used by anyone but the assigned user. Unauthorized use of amateur radio frequencies or an individual's call sign may lead to

fines and/or a prison sentence. The call sign is used to determine which station is sending a message. It is an important setting for the GPS / Telemetry Module because multiple stations may be sending data and the data needs to be identified, as coming from the payload.

The next setting is the Digi Path setting. This setting tells repeaters on the ground that might pick up the signal whether or not to re-send the signal. The recommended setting from Byonics is WIDE1-1, WIDE2-1. This setting will be used for the Balloon Payload. The Symbol and Table/Overlay settings determine what icon will be displayed by the ground station's APRS program. The O symbol and / table correspond to a hot air balloon. A high altitude balloon is a weather balloon not a hot air balloon, but there is not a weather balloon symbol and no one (or almost no one) would believe a car is at 100,000 feet.

The Timing section contains settings that relate to transmit timing. This section determines the rate that data is sent and other timing aspects. The Status portion of the TinyTrak4 Tracker settings allows a text string to be added to the end of a position transmission. This setting will send the text “MSU Balloon”, followed by an e-mail address. The e-mail address can be used after the flight to receive reports from other amateur radio operators that may have received transmissions from the balloon payload.

The Telemetry section of the TinyTrak4 Tracker software contains settings that relate to the telemetry data that will be sent by the TNC to the ground station. The ground station will receive a string similar to the following:

T#002,297,610,999,999,560,00000000

The first three digit number following the "#" is the reading number. This number is incremented for each telemetry transmission. The next number is the system voltage. To convert the received voltage number into an actual voltage, multiply the number by 0.0272. The supply voltage in the sample string is 8.08V. The next number "610" is the reading associated with the temperature. To convert this reading to a Celsius Temperature divide it by 2.04 and then subtract 273. The temperature that corresponds to the reading in the sample string is 26 degrees Celsius The fourth, fifth and sixth numbers are associated with the voltage readings of pins JP5, JP4 and JP3, respectively. It appears that pins JP4 and JP3 are connected, although this is not discussed in the TinyTrak4 manual. At this time only pin JP5 is connected to an external component. Other telemetry configurations can be used; it is best to determine the telemetry needs and use the TT4 Tracker manual to properly configure the Telemetry unit.

The field of check boxes at the upper right portion of the Windows interface is used for setting special settings. It is important that the Send Altitude and Timestamp HMS settings are checked. The LED disable option was chosen to save battery life. All other options on the interface should not be enabled. Not selecting the send NMEA allows the TinyTrack4 to format the data into a more readable format.

With the settings shown in Figure 2.6, the TinyTrack4 will transmit a string of the form shown below, to the ground every 10 seconds.

00:20:11R KE5FNM>APT407,WIDE1-1,WIDE2-1
/062408h3510.12N/09003.20WO000/000/A=000000

This string is transmitted using APRS standards and is decoded by the ground station to track and monitor the ground station. The first line consists of the relay information. The second line includes the time of the GPS reading (to the left of the h), the location of the payload (to the left of the N and W) and the altitude in feet (after the A=).

Kenwood TH-K2AT Transceiver and Settings

The radio used to transmit the APRS GPS/Telemetry Module data is the Kenwood TH-K2AT hand held transceiver. This transceiver was chosen because of its ease of operation. The Kenwood TH-K2AT operates on the 2 meter wavelength, or 144 MHz, band. This band is controlled by the FCC and requires a valid FCC license to use. This radio is fairly simple and requires only changing a few menu options to properly configure it for use with the TinyTrak4.

In addition to the TH-K2AT radio, it is recommended that the BT-14 alkaline battery holder be used. Alkaline batteries will not be used in the radio, but the BT-14 allows for soldering to the battery holder, rather than soldering directly to the radio. This is important because the radio may be used in another project later.

Connecting the flight battery to the transceiver is a matter of soldering leads onto the BT-14 battery case, connecting those leads (in the correct polarity) to the 8V Lithium

Ion flight battery, and installing the BT-14 case onto the TH-K2AT radio. To allow the TinyTrak4 TNC to work properly with the TH-K2AT a few menu options on the TH-K2AT must be changed. These menu options may change as other versions of this radio become available. It is best to check with the radio manual to ensure proper operation. Menu item twenty, named VOX, enables voice activation and sets the gain associated with the voice activation. This menu option should be set to five. This allows the TinyTrak4 to send a signal to the transmitter through the microphone port and for the transmitter to activate when the signal is received. If the VOX setting is not enabled, the transmitter will not activate when the TinyTrak4 sends a signal to the microphone input. With the VOX setting enabled any voltage signal present on the microphone port is transmitted. Menu option 21, VXB, should be turned on. This allows the transceiver to transmit a microphone input signal even if the transceiver is currently receiving data. All other settings on the TH-K2AT should remain at the factory defaults. Before the launch of the payload it is important to lock the keys on the TH-K2AT by holding down the function button. This prevents accidental changes to the settings during the balloon flight.

Once the entire Module is operational, it is tested using the ground station. If correct, the ground station should be able to accurately receive and record positions and other data sent from the GPS Module.

CHAPTER III

SERVO DRIVER MODULE (SDM)

The purpose of the Servo Driver Module, or SDM, is to receive commands from the ground station and the Low Speed Data Collection Module (LSM), and to drive the cut down servo. This module is capable of later being expanded to include other commanded functions, but is currently only used to trigger release of the balloon and the payload parachute. The SDM consists of the following main components:

- Ham Radio – Kenwood TH-K2A (Shared with the GPS/Tele Module)
- Microcontroller - PIC 18F1320
- DTMF Tone Decoder – TDK 75T204
- Metal Gear Mini Servo – Hitec HS-225MG

These components are connected as shown in Figure 3.1. Figure 3.1 shows the general flow of the system and all the main components. A complete circuit diagram for this system is located in Figure 3.3 and 3.4. A dotted line has been drawn around the components that are dedicated to the Servo Driver Module.

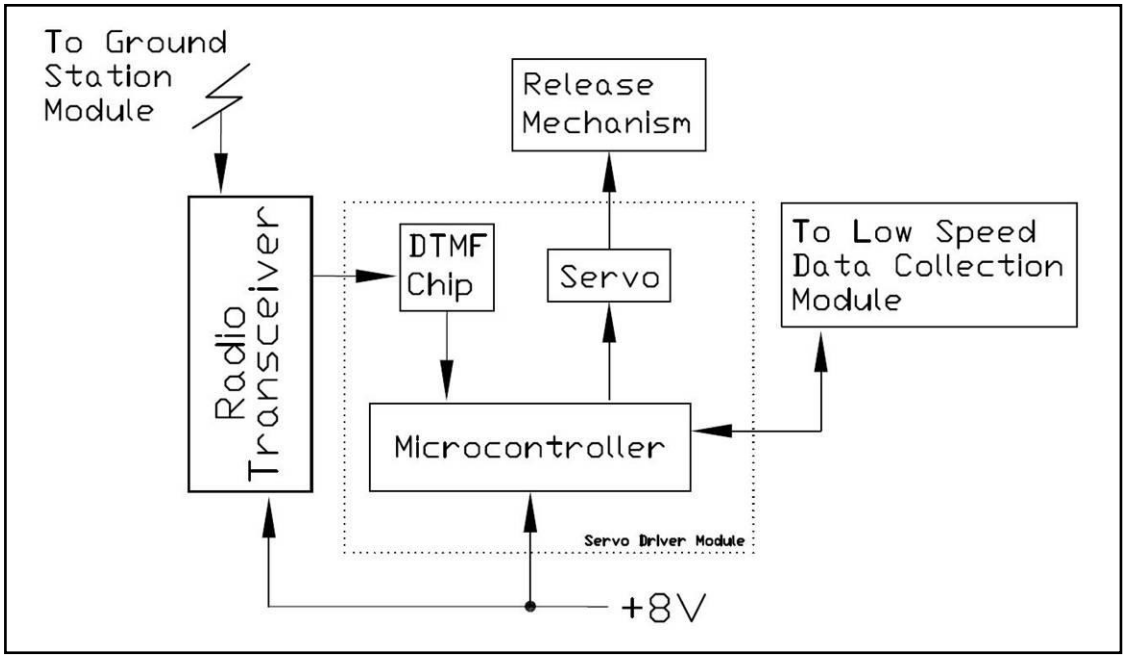

Figure 3.1 Servo Driver Module System Diagram.

Signals sent from the ground station are received by the radio transceiver and passed on to the Servo Driver Module through the speaker port of the radio. This signal is then decoded using a DTMF tone decoder and is read by a PIC 18F1320 microcontroller. The servo receives commands from the Microcontroller and operates the release mechanism. Servo movement may be triggered by commands from either the Ground Station via the Radio Transceiver or by the LSM. The Microcontroller relays the servo position to the LSM so that it may be recorded and stored for later analysis. The microcontroller receives power from the 8V Lithium Ion battery through a 5V regulator. This regulator also powers the DTMF chip. The servo is powered through a dedicated 5V regulator to allow for sustained current draws of greater than 1A by the servo. These components and their use are explained in the following sections.

Hitec HS-225MG Cut Down Servo and PWM Explanation

The servo used to operate the cut down release mechanism is a Hitec HS-225MG. The HS-225MG is a rotational servo that has a maximum rotation angle of 120deg. This type of servo is usually used in hobby related remote controlled applications such as airplanes or helicopters. The HS-225MG servo was chosen because of its shock resistant metal gear train, small size (32x60x30mm), light weight (31g), and relatively high torque rating (55in-oz). A metal gear train is important because the release mechanism might be subjected to large shocks throughout the flight. The HS-225MG operates using a three wire interface. The three wires are ground, usually black or brown, +5V (4.5-6V), usually red or orange (located in the center), and the signal wire, usually white or yellow. The ground wire should be connected directly to the system ground. The +5V should be supplied using a regulator dedicated to the servo. A separate regulator should be used because a servo can draw a large amount of current. The voltage output of multiple regulators should not be directly connected; doing this could cause damage or failure of the regulator. The servo position is determined using pulse width modulation, or PWM.

PWM is a communication method that uses a single wire and outputs a pulse along that wire. The wire voltage goes from 0V to +5V, stays there for a period of time, usually between 1000 and 2000 microseconds. The wire voltage then returns to 0V and remains there for approximately 17-20 milliseconds, until the next pulse is sent. The servo rotation angle is determined by the time that the signal wire voltage is at +5V. Figure 3.2 better shows how the servo angular position is related to the incoming pulse.

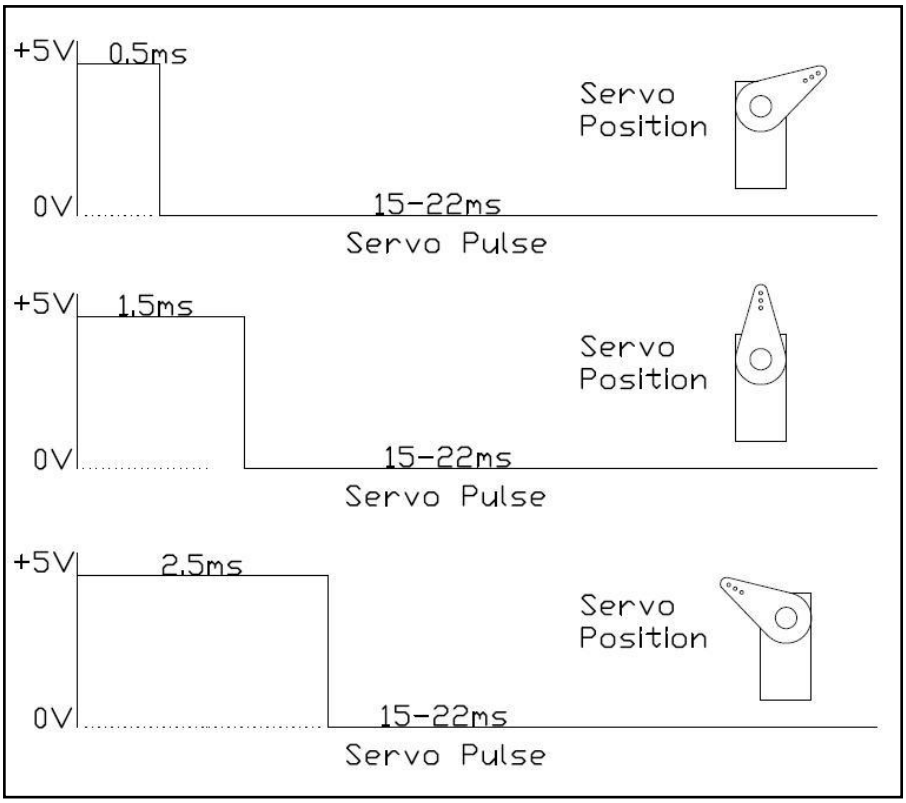

Figure 3.2 Servo Arm Position as it Relates to the Received Pulse Length.

One very important note, if an analog servo, such as the HS-225MG, does not receive pulses, the servo responds as if it is not powered and will not hold its position. Another detail about working with hobby servos, servos made by different companies will often rotate in opposite directions with respect to the servo pulse. For instance, if a servo made by Hitec rotates counterclockwise as the servo pulse length increases, a Futaba servo might rotate clockwise as the servo pulse length increases. Also, some servos may require a different pulse range to operate properly. For instance, some servos require an output

pulse between 1ms and 2ms instead of between 0.5ms and 2.5ms. It is best to fully test the settings calculated for proper operation with the servo used.

TDK 75T204 DTMF Tone Decoder Chip Use and Connections

The DTMF Tone Decoder receives the DTMF tone signal from the speaker port of the ham radio. The ham radio transceiver is shared with the GPS/Telemetry Module and is connected to both modules through the use of a 2.5mm (3/32") splitter (Radio Shack part number 274-948). The TDK 75T204 chip recognizes and decodes the DTMF tones into a digital number that corresponds to a DTMF character[15]. This chip was chosen because of its ease of use and CMOS compatibility. The DTMF chip transmits the digital number to other devices using 5 pins, DV, D1, D2, D4, and D8. All five pins have a low default state when the chip is powered. When a valid tone signal is detected by the DTMF chip, pin DV is high. Pins D1, D2, D4 and D8 are used to relay the specific tone that was received. The pins are assigned to the DTMF digits using standard binary.

The easiest way to understand how to read the DTMF chip pins is to view each pin as a value. D1=1, D2=2, D4=4, and D8=8. The pins values are added to determine the digital number. If all pins are high the digital number would be 1+2+4+8=15. If no pins are in the high state the digital number is 0. This number is then used to determine the DTMF character received. The pins are connected to the microcontroller and the digital number and corresponding DTMF character are decoded by the microcontroller.

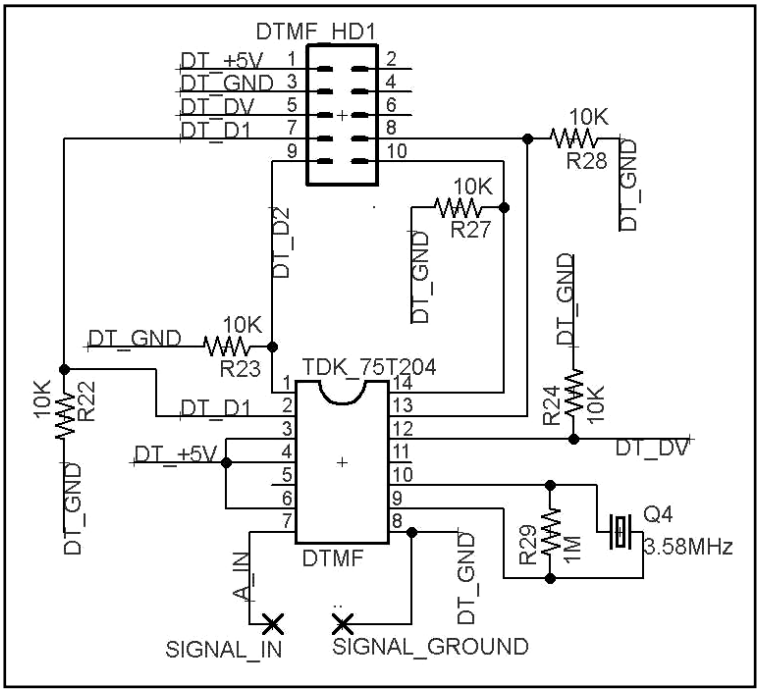

Figure 3.3 Servo Driver DTMF Chip Connection Schematic

Figure 3.3 shows the DTMF chip circuit schematic. Currently, the DTMF chip circuit is built on its own small board and connected to the microcontroller using a 5x2 male header and 5x2 ribbon cable. The 5x2 male header is labeled in Figure 3.3 as DTMF_HD1. Through this header the chip circuit receives power, and outputs to the microcontroller. Pins 1, 2, 12, 13 and 14 are signal pins that are read by the microcontroller. These pins are all connected to 10K pull down resistors. The pull down resistors keep the pins at the low state in the event that the DTMF chip loses power or fails. A standard color burst crystal (3.58MHz) is connected between pins 9 and 10 and is used by the chip to separate the two frequencies that make up the DTMF tone. Pins 3, 4 and 6 are connected to +5V. Pin 8 is connected to ground and to the incoming signal

ground. The signal from the transceiver speaker port is connected to the chip through pin 7. Table 3.1 further describes the pins and their uses.

Table 3.1 DTMF Chip Pin and Connection Descriptions

Pin #	Name	Description	Connection
1	D2	Digital output pin, second bit of binary number, low or high (0 or 1)	Connected to microcontroller, pulled low through 10KΩ resistor
2	D1	Digital output pin, first bit of binary number, low or high (0 or 1)	Connected to microcontroller, pulled low through 10KΩ resistor
3	EN	Output enable pin, if pulled high output pins are enabled	Connected to VP (+5V)
4	VP	Power pin, used to power the chip	Connected to +5V through header and ribbon cable
5	N/C	Not Used	Not Connected, Left Floating
6	XEN	Crystal enable pin, used to enable the crystal pins XIN and XOUT	Connected to VP (+5V)
7	Analog In	Analog Input	Connected to the positive signal coming from the transceiver speaker port
8	GND	Ground pin	Connected to system ground through header and ribbon cable
9	XOUT	Crystal oscillator output pin	Connected to XIN through crystal and 1MΩ
10	XIN	Crystal oscillator input pin	Connected to XOUT through crystal and 1MΩ
11	ATB	Oscillator Frequency Output	Not Connected, can be used to run multiple chips off of one crystal
12	DV	Digital output pin, used to signal that a valid tone is being received	Connected to microcontroller, pulled low through 10KΩ resistor
13	D8	Digital output pin, fourth bit of binary number, low or high (0 or 1)	Connected to microcontroller, pulled low through 10KΩ resistor
14	D4	Digital output pin, third bit of binary number, low or high (0 or 1)	Connected to microcontroller, pulled low through 10KΩ resistor

Pins 3 and 6 are used to disable or enable the output pins and the crystal oscillator to reduce the overall power consumption of the system; these two pins may be controlled. These pins are externally connected to VP (+5V) in the SDM because the current saved

by controlling them would be minute, compared to the current consumption of the entire system.

Microchip PIC 18F1320 and ServoDriver.bas Firmware

The Microcontroller chosen to receive the DTMF digital number and drive the servo is the PIC 18F1320. PIC microcontrollers were used because of their ease of programmability and their compatibility with the Swordfish Basic compiler. The microcontroller is the brain behind this module and performs tasks of decoding the DTMF number, driving the servo and relaying servo position to the LSM. These functions can best be understood by understating the SDM microcontroller firmware. The firmware program used by the SDM module is named "ServoDriver.bas". This firmware is written using the Swordfish Basic Compiler language. A very important detail to keep in mind when reviewing the firmware for this system is that the PIC 18F1320 does not have a typical CPU clock that can be sampled. Therefore, most timing in this program is achieved by knowing the approximate time it takes the program to run. Changing the clock speed (10MHz) will change the timing values located within the program. The ServoDriver.bas program is segmented into 3 sections--header, subroutines and main program.

The ServoDriver.bas header contains general information about the program, names of other programs that are included, dimensioning of variables, and aliasing of the microcontroller pins. The general information section contains notes on the firmware, the device name the firmware is meant for, the oscillator speed, and oscillator settings. The

“include” portion of the header contains information about other .bas program libraries that will be called -throughout the firmware. These included library programs were not written uniquely for this project. The next portion of the ServoDriver.bas header contains lines of code that dimensions the variables that will be used, and gives names (aliases) to the Input/Output pins.

The largest portion of the firmware contains the subroutines that will be called by the main program to perform tasks. There are nine subroutines that perform various tasks for the main program. One thing to note, subroutines in Swordfish Basic are not recursive. In other words, subroutines cannot be entered from subroutines. Subroutines can only be entered from the main program; however, functions contained in the included files can be called upon within subroutines.

The initialization routine is named “PinInit”. This subroutine sets all the pins to their initial states and activates them as either input or output pins. This subroutine also initializes some variables to their initial states.

The “Pointer” subroutine restores the last pointer and state variable to their last defined values. If these variables have not been defined since the firmware has been downloaded the values are set to 0. This subroutine guards against a momentary power loss resetting either the servo position or the password state.

The “TMR0_Initialize” subroutine contains settings for the PIC special function registers that enable and setup Timer0 in the PIC. Timer0 is an interrupt timer that can be used to interrupt the program at a specific time. The time the interrupt occurs is defined

by special function register values and the clock cycle. The Timer interrupt timer is used by the interrupt subroutine “Servo” to generate the servo pulse.

The subroutine “Servo” generates a PWM signal that is used to drive the cut down servo. This is done by using Timer0 to interrupt the program every 0.2 millisecond. There is also an offset associated with the timer. The formula that relates the count variable, in the Servo subroutine, to actual time values is the following:

$$Count = 5(millisecond)+2.5$$

Where Count is the value of the count variable in the Servo subroutine and millisecond is the number of milliseconds that the event should take place. An example of how this formula is used can be seen upon further investigation of the Servo subroutine. To output a servo pulse to send the servo to the initial (endpoint2) position of the servo, the servo pin must be high for precisely 2.5 milliseconds and low for approximately 20 milliseconds. Using the preceding formula the servo pin should be high for 15 counts and then low until count 100. At this point the Count variable should be reset so the next pulse can be sent. When the servo needs to change positions to the midpoint (balloon release position), the Pos variable changes to a value of 2, and the count when the pin goes high is reduced by the “if” structure to a value of 10. This outputs a high pulse that is 1.5 milliseconds long. When the Pos variable changes to 1 the pin stays high for 5 counts and a pulse length of 0.5 milliseconds is sent to the servo. This moves the servo to the release all position (endpoint1).

The PTPcom subroutine allows commands to be received from the LSM microcontroller. It also relays the servo position to the LSM, where it is recorded. The

input pins are connected to pull down resistors and held high by the LSM microcontroller. In the event that the LSM malfunctions or loses power the pins will be pulled low thorough the pull down resistors and the "cut down all" state will be triggered. The LSM can also intentionally change the servo position by pulling one or both of the input pins low.

There are four subroutines that are related to the DTMF input. They are GetDGT, Pswd, PasswordEXP, and Command. The GetDGT subroutine reads the four DTMF chip character output pins (D1, D2, D4, D8) and assigns the appropriate DTMF character to the Digit variable. The Digit variable is returned to the main program.

The Pswd subroutine sets up a DTMF character password that consists of 4 states. The base state for the password, state=0, indicates that no password characters have been detected in the current password cycle. The current DTMF password is "C", "*", "9". Each time the correct password digit is entered the state variable is increased. If state=3 all of the password characters have been received and the next DTMF digit received will be a command digit. If a wrong digit is detected, for instance "C", "A", "*", "6", "9" is received, the program will ignore the wrong digits and the state=3. The state variable can be reset by the reception of a command digit (which completes the sequence), or through the PasswordEXP subroutine.

The PasswordEXP subroutine is used to reset the password state variable after a period of time has been elapsed. This time period is set by the constant variable EXP. When the expiration time, EXP, has been reached the password state will be reset. The expiration time is reset each time a recognized password character is detected. This

means that for a command digit to be sent to the Servo Driver Module, the password must be entered, in order, followed by the command digit with no more than approximately 10 seconds (for EXP=10) between each character. If “C”, “*” is entered and is not followed by “9”, after approximately 10 seconds the PasswordEXP subroutine will reset the password state and the entire password string will have to be entered before a command digit may be accepted. For approximately the first 0.5 seconds after the password state is reset the status LED pin will be high and the status LED will be lit. The status LED is used to visually confirm that the Servo Driver Module is operational If no password digit is detected the PasswordEXP subroutine will continue resetting the state variable every 10 seconds. This means that every 10 seconds the LED will be illuminated for 0.5 seconds

The Command subroutine receives the command digit and performs the appropriate task. Currently, the only two tasks that have been programmed are moving the servo to two different positions, by changing the Pos variable used by the Servo subroutine.

The main program section of the firmware calls upon the subroutines in the appropriate order. The first subroutines to run are the initialization subroutines TMR0_Initialize and PinInit. Then the Pointer subroutine is run and the saved values of state and Pos are restored. When the pointer subroutine is run, or the Pos variable is updated, the Servo interrupt is disabled, the update is performed, and then the Servo interrupt is re-enabled. This prevents the Servo interrupt from interrupting a write to the

Pos variable. If this occurred, the Servo interrupt could use a bad value for Pos which could damage the Servo.

The DTMF subroutines are triggered when the DTMF chip tone detect pin DV goes to the high state and is read by the microcontroller. When the DV pin goes high, the GetDGT subroutine runs and returns the detected DTMF character. Then the password subroutine is run and the state variable is updated, if needed. If the state variable is equal to 3 and DV is high, then the command digit subroutine is called and the action corresponding to the command digit tone is performed.

To prevent multiple readings of a DTMF tone that is held down, a While loop in the Main program "traps" the program until the DV pin goes low, indicating that the detected tone has ended. If a tone is held down for approximately three seconds, the while loop will exit and the program will continue running. This exit prevents a constant tone from not allowing the LSM to update the servo position. In other words, this exit every three seconds keeps a constant DTMF tone from, effectively, freezing the system.

After the DTMF DV pin is checked, the pins that are used to communicate with the LSM are checked, by calling on the PTPcom subroutine. The PTPcom subroutine also updates the servo position, if needed. Finally, the PasswordEXP subroutine is run to update the time that has elapsed since the last password digit was detected. Most of the main program is contained with an infinite while loop. This means that the DV pin and input pins are constantly being polled.

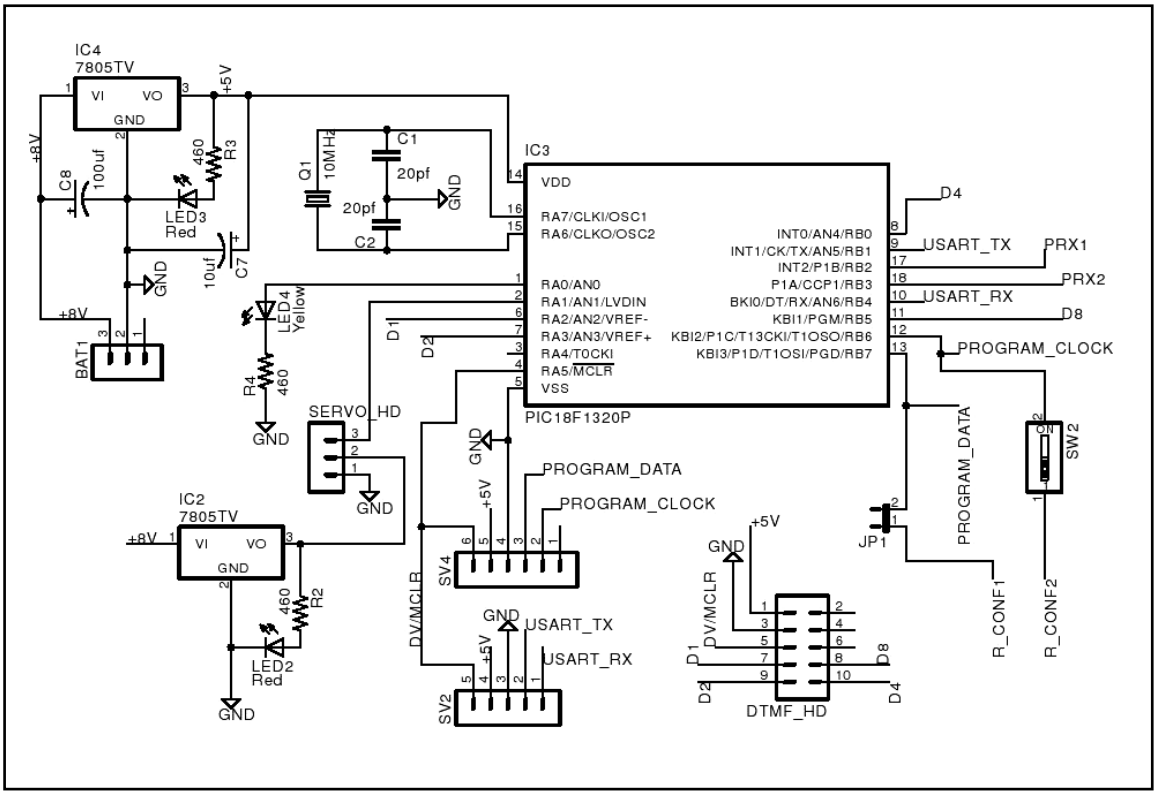

Figure 3.4 Complete Circuit Diagram for SDM

Figure 3.4 is the complete circuit diagram for the SDM. This diagram does not include the DTMF chip circuitry previously discussed, but does include a male 5x2 header that connects to the DTMF chip board through a 5x2 ribbon cable. Jumpers J1 and switch SW2 are used to disconnect the release output state wires from the LSM; this must be done so that programming may be accomplished. To program the SDM microcontroller requires disconnecting the DTMF cable, opening jumper JP1, and turning switch SW2 off. This disconnects the pins used for programming the microcontroller from their other connections. The red LED's connected to the SDM regulators are used to show when the system is being powered.

CHAPTER IV

HIGH SPEED DATA COLLECTION MODULE (HSM)

The purpose of the High Speed Data Collection Module, or HSM, is to record acceleration and angular rate data. This module is named the High Speed Data Collection Module because data is collected by this module several times a second. The HSM consists of the following main components:

- IMU – Sparkfun 6 degree of freedom IMU
- Microcontroller – PIC 18F2520
- EEPROM Array – 8 - 512 kilobit external EEPROM's

These components are connected as shown in Figure 4.1. Figure 4.1 shows the general flow of data throughout the system main components. The HSM is powered by the +8V flight battery. The battery voltage is reduced to +5V, for use with the HSM components, through the use of a 5V regulator. A complete circuit diagram for this system is located in Figures 4.2 and 4.3. A dotted line has been drawn around the components that are dedicated to the HSM.

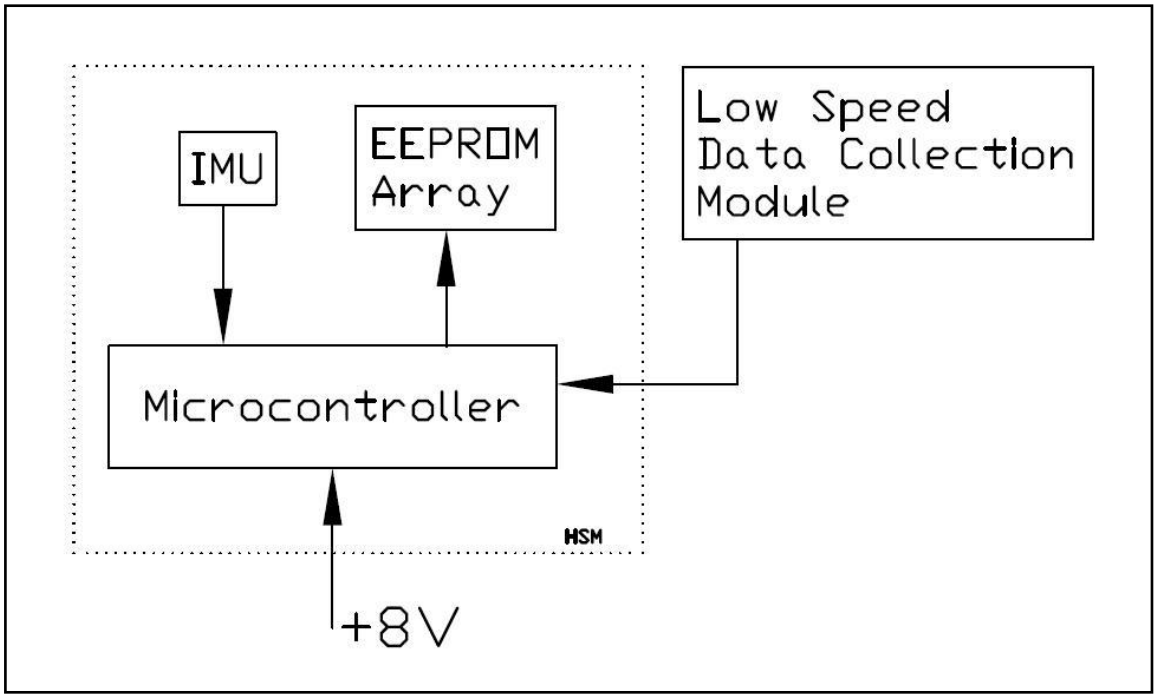

Figure 4.1 High Speed Data Collection Module System Diagram

The HSM collects acceleration and angular data through the use of a Sparkfun six degree of freedom IMU (Sparkfun part number SEN-09184). This data is then sent to the HSM microcontroller by way of UART serial. The data is then broken up by the microcontroller and written to an external EEPROM array. Timing information is provided to the HSM microcontroller by the Low Speed Data Collection Module. The stored data can later be retrieved and sent to a computer by using the HSM microcontroller read firmware, a PICkit 2, and the PICkit 2 software UART tool. By using the clear firmware program the HSM microcontroller can clear the EEPROM array. The three HSM microcontroller firmware programs and the use of the other components that make up the HSM will be discussed in the following sections.

HSM Timing

Timing information is provided to the HSM by the LSM. The LSM records a very accurate hhmmss time, received from the GPS every second. The LSM microcontroller code contains a variable, Tstat, that is incremented every minute (60 GPS readings) and recorded by the LSM every second. This variable is transmitted to the HSM by four pins. The four pins are used as a standard binary communication method. These pins are pulled high or low by the LSM and read in by the HSM. Pins Tin1 through Tin4 are bits 1 through 4 of the Tsat variable, respectively. Because the Tstat variable is recorded by both the HSM and the LSM, the time at which the Tstat variable changes will be recorded by both modules. This change point can be combined with the GPS time information, recorded by the LSM, to create a time basis for the HSM data.

EEPROM Array Connection and Use

The memory device used to store the values collected from the IMU is an external EEPROM array. This array consists of eight Microchip 24LC512 EEPROM's. This section discusses their use and connection. It should be understood that each EEPROM is a standalone device and does not require the other seven EEPROM's to function properly. Eight EEPROMS are used in order to meet the data storage requirements of the HSM. EEPROM's were chosen because of their stability and simplicity of use. A main drawback to using the 24LC512 external EEPROM is that it requires a delay of approximately 10 milliseconds between byte size writes to the same device. Also, writes

that are larger than a byte must be done as multiple byte size writes. It is recommended that the EEPROM array be replaced by a SD card or less complex data storage device for future versions of the HSM.

The 24LC512 EEPROM operates using the I2C communication protocol. A full description of the I2C communication protocol is beyond the scope of this project. However, it should be understood that I2C is a two wire communication protocol that is similar to that used in a typical LAN, or local area network. Each device has its own unique ID number that is set. Data is sent to the device by targeting the ID number and sending data. While all devices in the network, or array, are connected to the same two wires, only the device with the corresponding ID number recognizes and acts on the incoming data. Device ID numbers for the HSM EEPROM's are set by pulling three pins (A0, A1, A3) located on each EEPROM high or low. Figure 4.2 shows how the address pins A0, A1 and A2 are used to set a unique address for each EEPROM. If all the pins are low the I2C device address is set to 0. If A0=0, A1=1 and A2=1, the device address is set to 6. This means that a total of eight EEPROMS may be commanded using the I2C network. Figure 4.2 shows the wiring schematic for the HSM EEPROM Array.

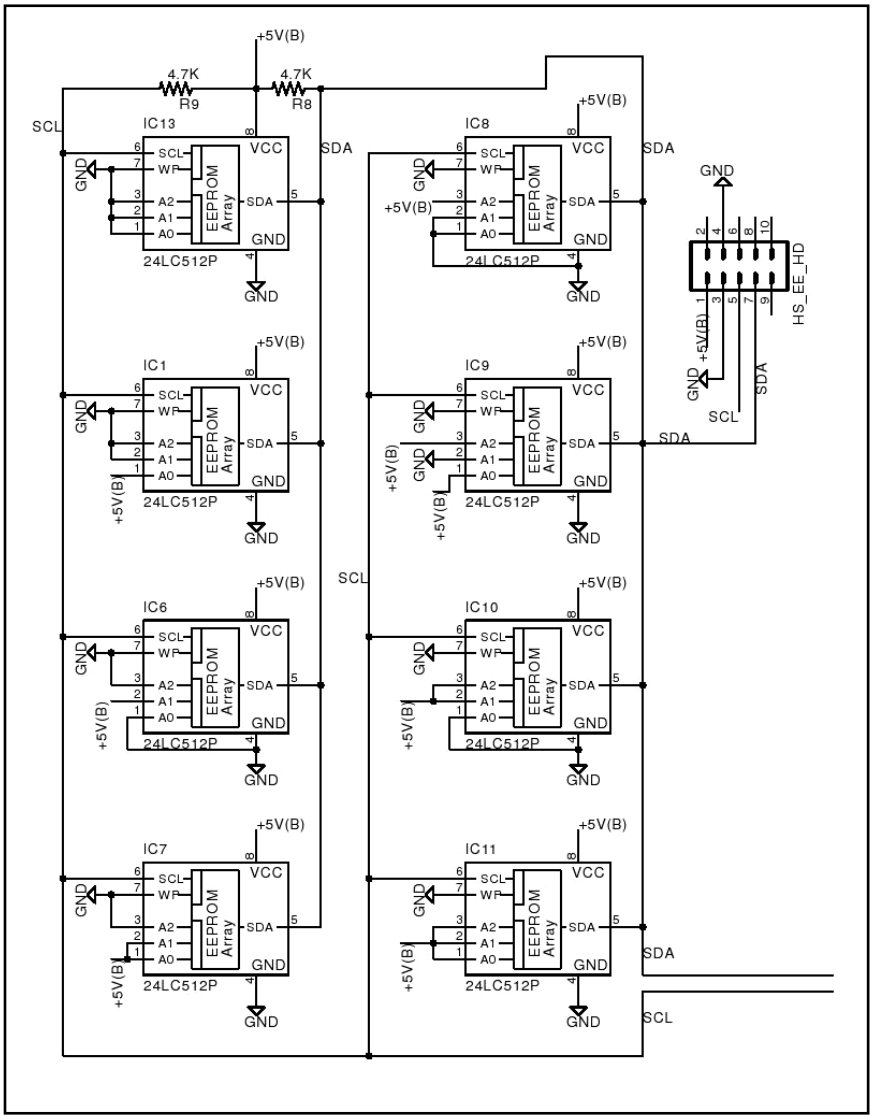

Figure 4.2 HSM EEPROM Array Wiring Schematic

Located at the top right of Figure 4.2 is a 5x2 male header that can be used by an external device to access the EEPROM array. The EEPROM Array is connected to the HSM microcontroller through four wires: ground, +5V, SDA, and SCL. SDA and SCL are wires that carry the I2C communications The SDA and SCL pins must be connected

to the +5V supply voltage through a 4.7KΩ pull up resistor. Table 4.1 further describes the 24LC512 EEPROM pins and their connections.

Table 4.1 EEPROM Pin and Connection Descriptions

Pin #	Name	Description	Connection
1	A0	Digital input pin, first bit of binary address, tied low or high (0 or 1)	Connected to either +5V or ground, depending on what I2C address is used for the EEPROM
2	A1	Digital input pin, second bit of binary address, tied low or high (0 or 1)	Connected to either +5V or ground, depending on what I2C address is used for the EEPROM
3	A2	Digital input pin, third bit of binary address, tied low or high (0 or 1)	Connected to either +5V or ground, depending on what I2C address is used for the EEPROM
4	GND	Power pin, used to power the chip	Connected to system ground
5	SDA	I2C Data input/output pin	Used to transfer data to the EEPROM
6	SCL	I2C Clock input pin	Used to synchronize transmissions to the EEPROM
7	WP	Write protect enable pin	Connected to Ground, to allow for writes to the EEPROM
8	VCC	Power pin, used to power the chip	Connected to +5V supplied by system battery, through HSM regulator

Additional information about I2C and the 24LC512 EEPROM is available in the 24LC512 Data Sheet[16].

Sparkfun IMU Description and Use

The Sparkfun IMU is a collection of 3 accelerometers and 3 gyros that are arranged on a single board. These sensors are sampled by the Sparkfun IMU microcontroller. The IMU microcontroller formats, scales, and applies temperature compensation to the data. The IMU microcontroller then sends the data to the SDM or other device using UART serial at a rate of 115,200 bits per second.

The Sparkfun IMU can be configured using a USB to 3.3V TTL Serial Cable. This cable is connected to a computer and a terminal program such as Tera Term Pro is used to interface with the IMU. For use with the HSM, the Sparkfun IMU factory default settings must be changed. The Sparkfun IMU can be configured to send data in either ASCII characters, able to be viewed on screen using the Tera Term Pro program, or in binary. For use with the HSM, the Sparkfun IMU must send the data in binary format. Binary transmissions read by the Tera Term Pro program are viewed as a string of gibberish, but binary format is easily read in correctly by the HSM microcontroller. The Sparkfun IMU can also be configured to output data at different rates. The minimum data output rate is 10Hz. The IMU maximum output rate is not limited. It should be noted that the recorded data frequency of the HSM will be half that of the IMU; this will be further discussed in the HSM microcontroller section. To be sure the IMU collects data within the desired range the accelerometer sensitivity should also be set. The IMU accelerometer sensitivity can be set to 1.5g, 2g, 4g or 6g. It is also important to enable all six data channels of the IMU. The final setting that should be changed is that the auto-run feature should be enabled. Enabling the auto-run feature means that when the IMU is powered it immediately begins sending out data. If this feature is not enabled, the IMU will be waiting for a start command, and will not function properly with the HSM.

The resolution of the IMU data readings is limited by the IMU microcontroller's internal analog to digital converter resolution. This is because the IMU gyro and accelerometer analog outputs are being read by the IMU microcontroller. The IMU currently uses a microcontroller that contains 10 bit analog to digital converters. This

means that all values for the accelerations and angular rates are 10 bit and have values of between 0 and 1023. Calibration of this unit should be performed, but is not included in the scope of this project.

The Sparkfun IMU outputs data using TTL serial. It is important to understand how the IMU sends out the data so that it can be properly read in by the HSM microcontroller. The first byte sent by the Sparkfun IMU is an ASCII "A" (decimal 65) character. This signifies the beginning of the IMU data output string. The last byte that is sent by the IMU is an ASCII "Z" (decimal 90) character. This character signifies the end of the transmitted string that contains IMU data. Even though the IMU is set up to run using binary transmission, the beginning and ending characters do not change format. Each value for acceleration and angular rate is transmitted as two bytes (MSB and LSB), because each value is a 10 bit (1 byte, 2 bit) number. The most significant byte of the value, or MSB, is transmitted first. Then the least significant byte for the value being transmitted, or LSB, is transmitted. The first value transmitted after the starting character "A" is the count. The count value is the number of the reading that is sent. This value is actually a 15 bit number and will change sequentially from 0 to 32767 each time a data string is transmitted. Once it reaches 32767 it will start over "counting" from 0. The first string sent (or data set) is count=0. The second string sent (or data set) is count=1, and so on.

Acceleration values for the X, Y, and Z axes are then sent in that order. Then the readings from the pitch, roll, and yaw gyros are sent. Repeating from above, each 10 bit value is transmitted as two bytes. This means that although there are only 10 bytes of data

(composed of 6 x 10 + 15 = 75 bits), they are transmitted as 7 x 2 = 14 bytes. The count variable is not recorded, meaning that only 60 bits (7 bytes, 4 bits) of data received from the IMU, are actually used by the HSM. Combined with the 4 bits of timing data received from the LSM, this makes 8 full bytes of data. A visual representation for the data arrangement is shown in the before package portion of Figure 4.4. The 8 bytes are written to the EEPROM array in parallel. This means that the first byte is written to the first EEPROM in the array, the second byte is written to the second EEPROM in the array, and so on. Writing to the EEPROM's this way allows for the 10ms delay to occur after writing an entire data string, rather than after each write. Theoretically, a write can occur as fast as 100 times per second. Obviously, because the program needs time to run, written data rates cannot occur at this frequency. Understanding how the data is structured becomes important in the HSM microcontroller section.

Microchip PIC 18F2520 Use and HSM Firmware

The brain of the HSM is the Microchip PIC 18F2520 microcontroller. This microcontroller receives input from both the LSM and the IMU and records those received values to the EEPROM array. Both the microcontroller circuitry and the firmware programs used to run the HSM will be discussed in this section. Figure 4.3 shows the complete circuit schematic for the HSM, excluding the previously discussed EEPROM array circuitry. The connections and components that are specific to the HSM will be discussed in this section.

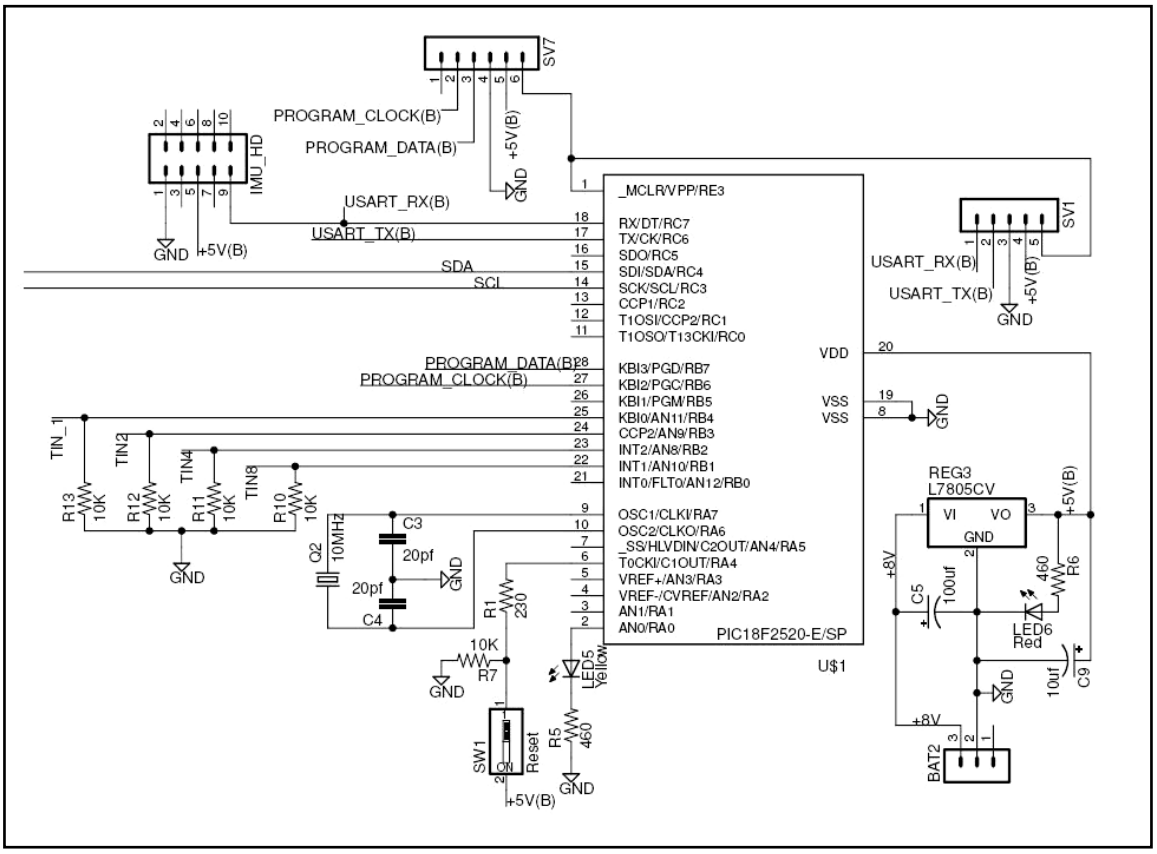

Figure 4.3 HSM Circuit Diagram, Excluding EEPROM Array

The power for the entire module is supplied through regulator Reg3, by connecting the +8V flight battery to male pin header Bat2. A red LED is used to show that the module is being powered. The six pin male header SV7 is used to program the microcontroller, by connecting it directly to a PICkit 2. The five pin header SV1 is used to receive UART transmissions from the microcontroller. This is done by using the PICkit 2 programmer and the PICkit 2 software UART tool. The IMU is connected to the 5x2 pin male header IMU_HD. The microcontroller is connected to the EEPROM array

through wire nets SDA and SCL. Nets TIN_1, TIN_2, TIN4, and TIN_8 are connected to the LSM and are pulled low through 10K resistors in the event of a LSM malfunction.

There are three firmware programs that have been written for the HSM. The first program, High_Speed_Record.bas, is used to record values collected throughout the balloon flight from the LSM and the IMU. The second program, High_Speed_Read.bas, is used to read the values collected in the EEPROM and to write these values to the screen or a file, via a PICkit 2 and the PICkit2 user software. The third program, EE_Clear_HS.bas, is used to write 0's to all the EEPROM addresses of the eight EEPROMS contained in the HSM EEPROM array. These firmware programs have been written using the Swordfish Basic compiler language. It is not possible to download the three programs to the 18F2520 at the same time. It may be possible, in the future, to combine the three HSM programs and to use a switch to determine which of the three should be run. However, to prevent confusion and to make certain the wrong program is not accidentally run during a balloon flight, the programs have been kept separate. All firmware programs used with the HSM output UART serial communicate at a rate of 38400 bits per second.

The High_Speed_Record.bas program is used to record data during the balloon flight. This program's purpose is to collect data from the IMU and the LSM then write this data to the EEPROM array. This program consists of the following three sections: header, subroutines, and main program.

The headers used for the HSM firmware programs are very similar to the headers of the other module firmware programs. The header contains general information about

the program, names of other programs that are included, dimensioning of variables, and aliasing of the microcontroller pins. The general information section contains notes on the firmware, the device name the firmware is meant for, the oscillator speed, and oscillator settings. The include portion of the header contains information about other .bas program libraries that will be called upon throughout the firmware. These included library programs were not written uniquely for this project. The next portion of the header contains lines of code that dimension the variables that will be used, and gives names (aliases) to the Input/Output pins.

There are 9 subroutines that are used by the High_Speed_Record.bas firmware program. These subroutines are labeled Header, Out, Package, Write, Read, GetData, GetData2, Avj, and ChkStat. These subroutines perform the major tasks the firmware is responsible for and are called, in the appropriate order, by the main program.

The Header subroutine is the first called by the main program. This subroutine outputs the data column header to the UART computer screen when the PIC is connected to a computer via the PICkit 2 and the PICkit 2 UART software. The Header subroutine also initializes the variables used for data collection and flashes the HSM status LED to signify the start of the program.

The GetData and GetData2 subroutines are almost identical. They are only differentiated by the fact that GetData returns a set of data stored in the primary set of variables (AccX, AccY, AccZ, Pitch, Roll, Yaw) and GetData2 returns a set of data stored in the secondary set of variables (AccX2, AccY2, AccZ2, Pitch2, Roll2, Yaw2). These two sets of data are averaged to form a single set of data that is written to the

EEPROM's. The purpose of the GetData subroutines is to collect the data string from the IMU and to write the received data to the appropriate variable. The GetData subroutines will actually wait for the lead character “A” sent by the IMU to signify that a data string is incoming. This means that if the IMU has been disconnected, or has malfunctioned the HSM will continue waiting to receive the start character and will be “stuck”. This also means that the data rate for the HSM is determined by the data rate of the IMU, and that it will ignore the transmission if it begins “listening” during the middle of a data string. If the HSM needs to record for a time period longer than 3 hours, the IMU data rate must be slowed below 10Hz. With two data collections being averaged to produce one written value, the written data frequency is half that of the IMU data rate setting.

The Avj subroutine averages the data sets collected from the two readings taken using the GetData and GetData2 subroutines. The returned values are stored in the GetData variables. Disabling the GetData2 and the Avj subroutine would change the HSM data write frequency from half the IMU data frequency to the IMU data frequency. The written frequency can be changed to 1/3 or less of the IMU data frequency by adding more GetData subroutines, and by modifying the Avj subroutine to include the other values collected.

The ChkStat subroutine sets the stat input pins to digital inputs and reads those pins. The time stat variable, sent from the LSM, is returned as a four bit number stored in the Stat variable.

The Out subroutine occasionally outputs the data collected by the microcontroller using UART. This data is received by a computer through the PICkit 2 and PICkit 2

software UART tool. The data is only output every few readings so that it does not clutter the UART tool screen. The Out subroutine is meant to be used in conjunction with the Read subroutine to check that the HSM is functioning correctly.

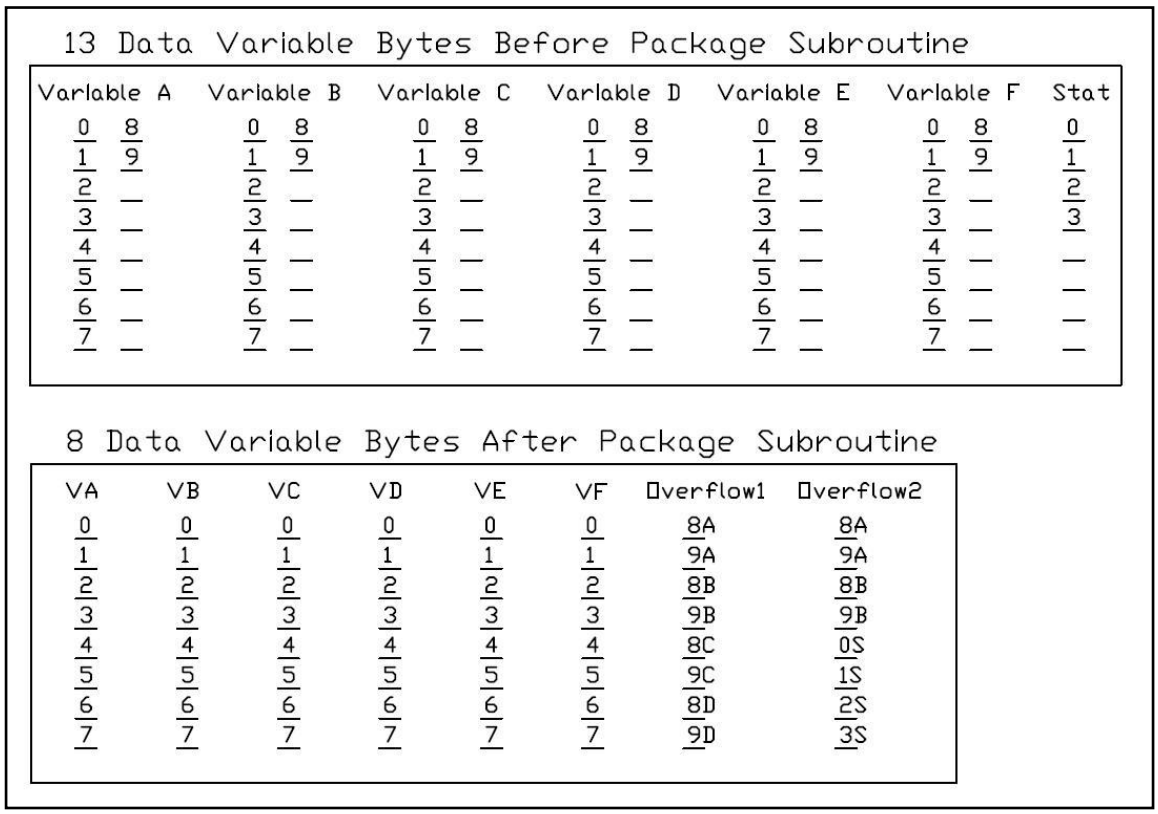

Figure 4.4 Visual Representation of Package Subroutine Function

The Package subroutine packages the data collected from the LSM and the IMU into 8 bytes that can be written to the EEPROM array. This subroutine takes the 6 – 10 bit values that were received by the IMU, and breaks them down into 6 byte length variables, the remaining 12 bits are written to two overflow byte size variables. The 4 bits of space left in the second overflow variable are used to store the 4 bit state variable. The

subroutine returns 8 byte size variables that are now ready to be written to the EEPROM's. Figure 4.4 shows how the data is arranged before and after the Package subroutine runs.

The Write subroutine is used to write the packaged variables to the EEPROM array. This subroutine also increases the memory address variable, and writes the new address to the Pointer location. The pointer keeps the program from overwriting data previously taken in the event of a loss of power or microcontroller malfunction.

The Read subroutine is meant to be used in conjunction with the Out subroutine to check the operation of the HSM. The Read subroutine occasionally reads, unpacks, and outputs the written data to a computer via the UART software tool.

The main program part of the High_Speed_Record.bas firmware calls on the subroutines in the correct order. On startup the main program checks the status of the reset pin. If the reset switch, SW1, is in the on position, the reset pin will be read as high (1) and the program will begin recording, starting at memory address 0. If the switch is in the off position the reset pin will be read as low (0) and the program will pull the starting write address from the Pointer memory location in the internal EEPROM. It is important for the reset pin to be in the off position during flight to prevent data from being overwritten due to a momentary power failure. Once the external EEPROM is filled the program will quit running and will enter a low power state. However, this will not prevent the IMU from continuing to run and draw power.

To calculate how long the HSM will record data, simply divide the total number of address bytes in an EEPROM (64000 bytes) by the written data frequency (5 writes

per second). This means that with a written frequency of 5Hz (IMU frequency of 10Hz), the HSM will record data for 12800 seconds, or 3.56 hours. The number of writes that can be performed is 64000 and not eight times that because each time data is taken a byte of data is written to each EEPROM (8 bytes of data, 8 EEPROMS). The IMU data rate can easily be increased to 20Hz without problems, but if write rates of faster than 5Hz are required, it is recommended that output subroutines in the High_Speed_Record.bas main program (Read and Out) be removed by commenting them out. This will allow for the fastest data write rates.

The High_Speed_Read.bas program is very similar to the Read subroutine located in the High_Speed_Record.bas program. The purpose of the High_Speed_Read.bas program is to read the EEPROM's, unpack the variables, and then to output those variables, along with the memory address the variables have been read from, to a file. The microcontroller sends this data using UART TTL serial. It is received by a computer through the PICkit 2 and the PICkit 2 software UART tool. This tool has the ability to log the communication it receives to a file. The data that is output from the microcontroller is in tab delimited format. Therefore, the file created by the PICkit 2 software will be in the same format and is easily read by a spreadsheet program. Before the data is sent by this program, a column header, labeling the data columns is sent.

The simplest program is the EE_Clear_HS.bas program. This program writes a 0 to the first address of all 8 EEPROM's, sequentially. The program delays 10 milliseconds to allow the EEPROM's to perform their writes. Then the address variable is incremented and the write is done again. Once all 64000 addresses of the EEPROM have been written

to a “done”, “All 8 EEPROMS set to 0” message is sent to the screen and the HSM status LED is flashed 5 times quickly every 2 seconds. To allow the program to run the fastest it possibly can, current memory addresses are output to the screen every 2048 addresses. To be able to tell that the program is running without connecting it to the PICkit 2 the HSM status LED changes state every 128 addresses. Clearing the EEPROM array takes approximately 30 minutes. If power is lost, or disconnected during the clear, the program resets and starts clearing from address 0.

CHAPTER V

LOW SPEED DATA COLLECTION MODULE (LSM)

The Low Speed Data Collection Module, or LSM, performs several tasks associated with the different modules. The primary task of the LSM is to sample and record pressure, temperature and GPS coordinates associated with the balloon payload. This module also sends cut down commands to the SDM, and provides timing information to the HSM. This module is named the Low Speed Data Collection Module because data is collected by this module once a second. The LSM consists of the following main components:

- GPS – Garmin 18XLVC (shared with GPS/Telemetry Module)
- Microcontroller – PIC 18F2520
- EEPROM Array – 8 - 512 kilobit external EEPROM's
- Temperature Sensors – 4 x Dallas DS18B20 one wire temperature sensor
- Pressure Sensors – 2 x Honeywell ASDX 015A24R

These components are connected as shown in Figure 5.1. Figure 5.1 shows the general flow of data throughout the system main components. A circuit diagram for this system is located in Figures 4.2, 5.2 and 5.3. The LSM is powered by the +8V flight battery. The battery voltage is reduced to +5V for use with the LSM components through the use of a

5V regulator. A dotted line has been drawn around the components that are dedicated to the LSM. A portion of the LSM has been created to operate within the high altitude balloon instead of the payload box. This subsystem is contained on its own circuit board, Balloon Board. The components have been outlined with a second dotted line.

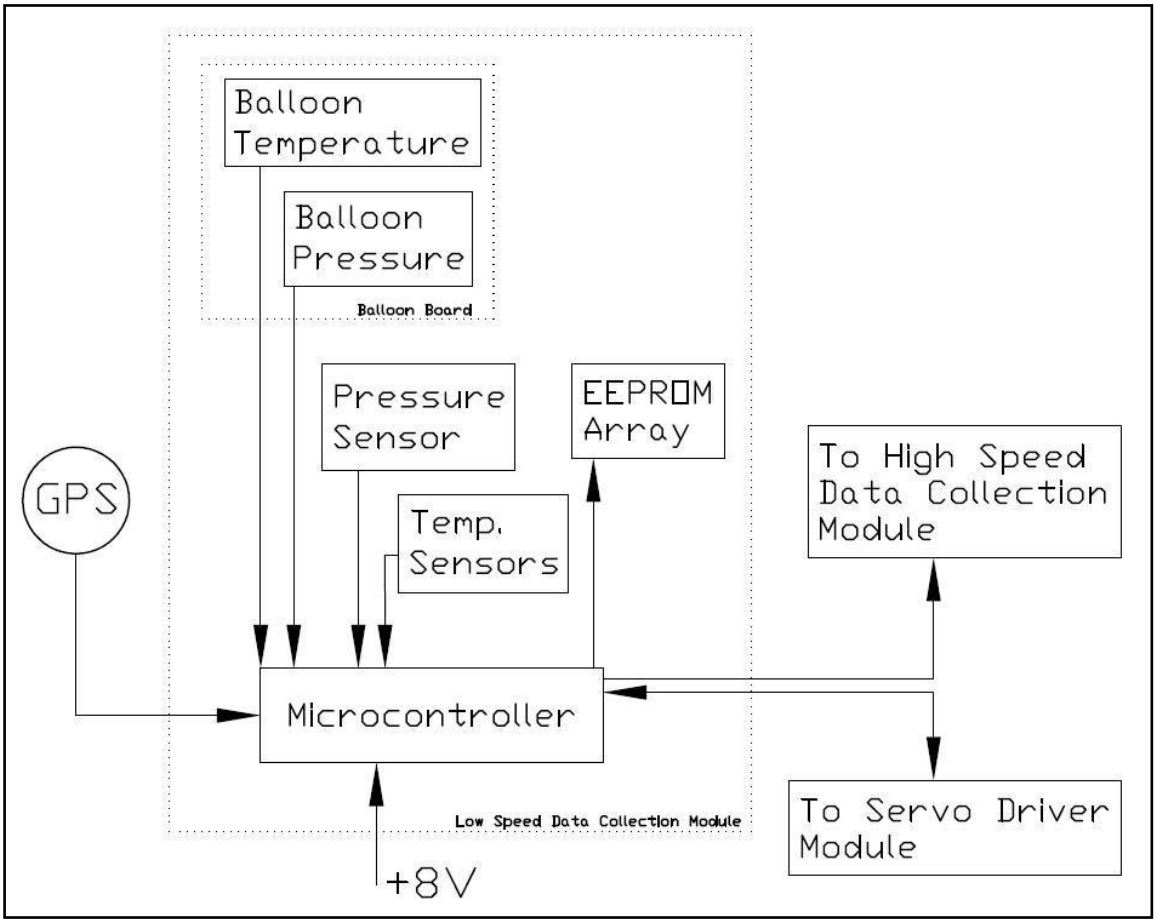

Figure 5.1 Low Speed Data Collection Module System Diagram.

The LSM receives data from four different sources that use different communication methods. The different source types and communication methods are GPS (serial), DS18B20 temperature sensors (one wire), pressure transducers (analog

voltages) and servo driver module (digital input pins). The data from these sources is collected and formatted by the LSM microcontroller. It is then written to the EEPROM array for storage.

Four pins connect the SDM to the LSM. Two of these pins are used to send the current servo position to the LSM. The other two pins are used to send cut down signals to the SDM. Four pins also connect the LSM to the HSM. These digital pins are used to transmit timing information to the HSM. The use of these four pins and the timing method is discussed in detail in the HSM Timing section of the HSM Chapter.

General Data Flow and LSM EEPROM Array Use

The GPS signal is used to trigger a data collection event. If the GPS signal is not present due to a malfunction, data is collected from the other components approximately every 3 seconds. The GPS signal provides three dimensional position, heading, ground speed, satellite tracking, and time information to the LSM. The LSM receives temperature data from four Dallas DS18B20 one wire temperature sensors. Balloon internal temperature and pressure are provided from the pressure transducer and temperature sensor mounted on a board that is placed inside the balloon. This board is connected to the main circuit board using two standard servo wires and connectors. Once the balloon is released the servo wire connectors are pulled apart and the Balloon Board is disconnected from the main payload. The Balloon Board is considered expendable and will not be recovered after the flight. Once the data is collected from the GPS,

temperature sensors, pressure transducers, and the SDM, it is written sequentially to the EEPROM array.

The EEPROM array circuitry used by the LSM is identical to that used by the HSM. As such, a circuit diagram and EEPROM use explanation will not appear in the LSM chapter. The slight differences in the use of the two EEPROM arrays will be covered in this section. For the complete EEPROM array circuit diagram and external EEPOM details, please refer to the EEPROM Array Connection and Use section of the High Speed Module chapter. It is recommended that both of these EEPROM arrays be replaced by another memory device, such as a SD card, in future versions of the balloon payload system.

The only difference between the HSM EEPROM array and the LSM EEPROM array is how they are written to. The HSM writes to each of the eight EEPROM's, then increments the write address, and then writes to all eight EEPROMS again. This method fills all the EEPROM's equally with each write. The LSM writes all 30 bytes of the LSM data to the first EEPROM, and continues to write all data sets to the first EEPROM. Once the first EEPROM is full, the LSM begins to fill the second EEPROM. This sequence continues until all the EEPROMS are full, at which point the LSM microcontroller terminates the program and enters a low power state. One detail to note is that because the writes are being performed as individual byte writes, there is no need to consider the page boundaries of the external EEPROM. The difference in the writing methods can be attributed to the amount of data collected by each module and the speed requirements of the module. It would be best if the HSM method of writing was modified to be used with

the LSM in future versions. Using the HSM method of writing would allow compatibility of the LSM with 5Hz GPS receivers. With the current circuitry and programming the LSM is only compatible with the Garmin 18XLVC 1Hz GPS receiver. With the current LSM recording firmware program settings and EEPROM array size the LSM can record data for 4.74 hours. This time was found using the following formula:

$$\frac{64000\ Bytes/EEPROM \times 8\ EEPROM's}{30\ Bytes/sec \times 3600\ sec/hr} = 4.74\text{hrs}$$

LSM GPS Connection and Use

The GPS unit is shared by both the LSM and the GPS/Telemetry module and must be connected to both. The LSM GPS is used as the basis for the main, payload triggered, cut down sequence. This cut down sequence is a responsibility of the LSM and will be further described in the microcontroller firmware section of this chapter.

The GPS outputs RS-232 serial communication at 4800 baud. The GPS output signal voltage is not standard RS-232 (+-12V). The signal output voltage is equal to the power input voltage of the GPS (+5V). It is also at RS-232 polarity, meaning the voltage output is ± 5V. The LSM microcontroller cannot read RS-232 serial, so it must be converted into TTL (0 to 5V) polarity before it is connected to the microcontroller. The circuit in Figure 5.2 converts the RS-232 GPS signal into a TTL polarity signal that can be received by the microcontroller. The following portion of this section will describe the signal conversion circuitry and its use.

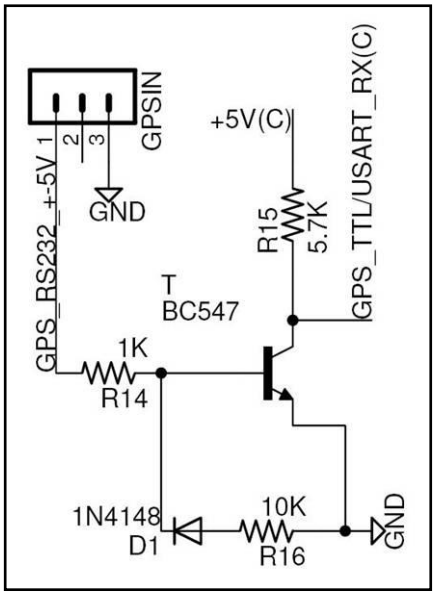

Figure 5.2 Circuit Used to Convert GPS Serial Signal to TTL

The GPS signal is connected to the main board through a 3 pin male header, and a male servo connector. The RS-232 GPS signal comes in on pin 1 of the header, and is grounded through pin 3. The grounding connection is optional, because the GPS is grounded to the system battery through the TinyTrak4. The signal comes in to the circuit through a 1KΩ resistor. This resistor is only needed if the GPS signal is needed by the TinyTrak4 and may need to be changed if the TinyTrak4 internal circuitry changes. The Diode D1 and resistor R16 are used to ground the negative portion of the GPS signal. The transistor, T, is an NPN transistor. It is used to invert the remaining 0-5V GPS signal. When the positive portion of the GPS signal is present at the base, the transistor is active and the output to the microcontroller, GPS_TTL/USART_RX(C), is grounded. When the 0V portion of the GPS signal is present at the base, the transistor is not active and the output to the microcontroller is pulled high through the 5.7K, R15, resistor. The value of

the 5.7K resistor is very important, because it must be large enough so that the microcontroller sees a low state when the transistor is active, but not so large that the voltage at the microcontroller pin is not pulled high when the transistor is not active. It is best to test different resistor values for R14 and R15, and select resistors that operate well with the entire system.

Dallas DS18B20 One Wire Temperature Sensor Connection and Use

The DS18B20 Temperature Sensor, by Dallas Semiconductor (MAXIM), is a temperature sensor that communicates the temperature at the sensor, in Celsius, using a digital communication known as one wire communication. A full explanation of one wire communication protocols is beyond the scope of this paper. However, it should be understood that each one wire device is programmed by the manufacturer to have a unique, 64 bit, Rom ID. This allows many one wire devices to be connected to the microcontroller through the same wire. To receive data from a specific one wire device, the receiving device targets the desired device Rom ID and then retrieves the data associated with that device. The name one wire comes from the fact that the device uses only one wire to communicate and power the device. Actually, despite its name, the device uses two wires, a power/data line and a ground.

Although one wire devices can communicate using only one wire, sometimes it is best to use more than one wire because some complications can arise when using long leads and one wire circuitry. The DS18B20 temperature sensor can be connected in either a one wire configuration, or it can be connected using two (three) wires. The three wires

are ground, power (+5V) and the signal wire. This means that power is supplied to the device through a pin other than the signal pin. With this configuration, there are fewer concerns to running multiple devices on the same data line, or having long distances between the devices and the microcontroller. The only unique connection that must be made for the device to operate properly is a 4.7KΩ resistor between the data and power line.

Currently the LSM microcontroller firmware is configured for use with four DS18B20 devices. One sensor is mounted on the main board. One sensor is mounted on the Balloon Board that is placed inside the balloon during flight. The other two sensors have been mounted to a typical servo wire and male servo connector so that standard servo extensions may be added and the temperature sensors may be placed wherever needed. If a temperature sensor is not connected, the LSM microcontroller firmware program will operate correctly, except that the value recorded for the not connected temperature sensor will always be -1. The DS18B20 temperature sensor is specified to have an accuracy of 0.5˚C between -10˚C and +85˚C. The default temperature resolution is 12 bit, which corresponds to 0.0625˚C. The microcontroller firmware program has been set up to ignore the decimal values because the temperature accuracy is reduced at temperatures below -10˚C. The temperature values are recorded by the LSM microcontroller as short integers. This means that the recorded values are integers between -128 and +128.

Honeywell ASDX 015A24R Pressure Transducer Connection and Use

The Honeywell ASDX 015A24R Pressure Transducer is a fairly standard analog pressure transducer. This device is connected using three wires, Power, Ground, and Vout. The pressure transducer is powered by the LSM system voltage of +5V. The output of the pressure transducer, Vout, is proportional to the input voltage and varies with changes in pressure. The LSM uses two ASDX pressure transducers. One of the pressure transducers is mounted on the Balloon Board and is used to sense the pressure inside the balloon. This transducer has a relatively long lead that is used to connect the transducer to the main circuit board. Therefore, it is very important that this pressure sensor be calibrated with the lead in place. Ideally, the resistance change of the wire due to changes in temperature, at the current output of the transducer, would be taken into account. However, changes in the voltage due to these factors will probably be small compared to the 10 bit resolution of the microcontroller analog to digital converter. So, calibration at room temperature with the lead in place should yield sufficient accuracy.

The other pressure transducer is mounted on the main circuit board. This pressure transducer is used to sense the local atmospheric pressure around the balloon payload. This pressure transducer receives the atmospheric pressure through tubing that runs to ports that are located on all sides of the balloon payload. The pressure on each side of the balloon payload is physically averaged by connecting all of the port tubes to a small reservoir. The pressure transducer is also connected to this reservoir. It is important that the pressure be an average of all sides of the balloon payload because when the balloon is falling the pressure on the side leading the descent will be much greater than the pressure

on the trailing side. Therefore, measuring the pressure on only one side of the balloon payload would, most likely, not give pressure readings that could accurately be use for determining altitude.

In the event that the GPS does not have a valid fix on the payload location, the main board mounted pressure transducer is used as the basis for a secondary, pressure related, cut down sequence. These cut down sequences will be discussed in the microcontroller firmware portion of this chapter.

Microchip PIC 18F2520 Use and LSM Firmware

This section will discuss the circuitry and the three firmware programs associated with the LSM microcontroller. The LSM microcontroller receives input from the temperature sensors, pressure transducers, GPS receiver and Servo Driver Module. The data is received, formatted, and then written to the LSM external EEPROM array. Both the microcontroller circuitry and the firmware programs used to operate the LSM will be discussed in this section. Figure 5.3 shows the complete circuit schematic for the LSM, excluding the LSM EEPROM array and the GPS connection and conversion circuitry. The LSM EEPROM array circuit diagram can be found in the EEPROM Array Connection and Use section of the High Speed Module chapter. The GPS connection and conversion circuit diagram can be found in the LSM GPS section of this chapter. The other connections and components, specific to the LSM, will be discussed in this section.

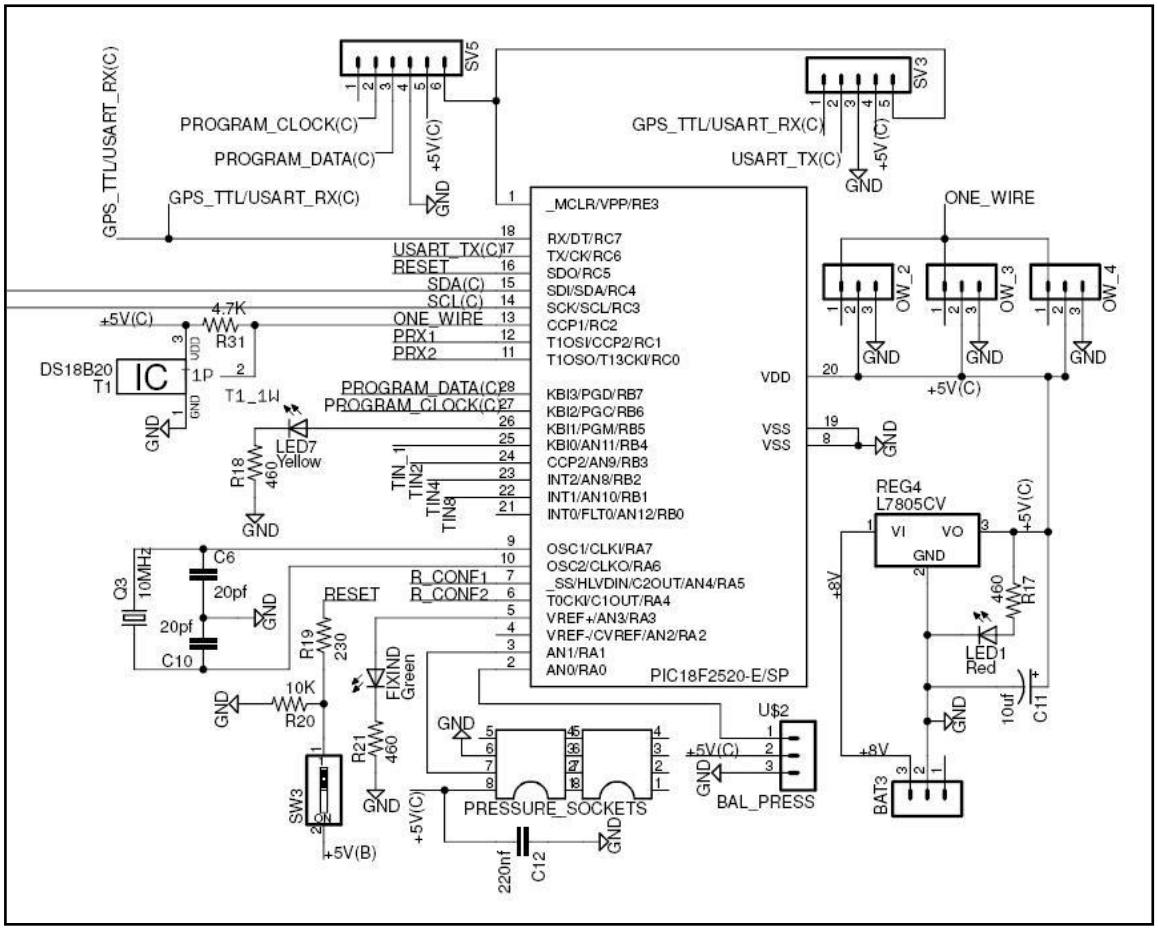

Figure 5.3 Low Speed Module Microcontroller Circuit Diagram.

The power for the entire module is supplied through regulator Reg4, by connecting the +8V flight battery to male pin header Bat3. A red LED, located next to the regulator, is used to show that the module is being powered. The six pin male header SV5 is used to program the microcontroller by connecting it directly to a PICkit 2. The five pin header SV3 is used to receive UART transmissions from the microcontroller. This is done by using the PICkit 2 programmer and the PICkit 2 software UART tool. Three pin male headers OW_2, OW_3 and OW_4 are used to connect the one wire temperature

sensors. The placement of these sensors on the three pin headers does not matter. The other one wire temperature sensor is soldered to the board and located in the upper left quadrant of the schematic. The main board mounted pressure transducer is connected using two 8 pin IC sockets placed side by side (PRESSURE_SOCKETS). A small amount of low temp hot glue may be used to better secure the transducer to the socket. The Balloon Board pressure transducer is connected using the 3 pin male header BAL_PRESS. The microcontroller is connected to the EEPROM array through wire nets SDA and SCL. Nets TIN_1, TIN_2, TIN4, and TIN_8 are connected to the HSM and are pulled low through 10K resistors located on the HSM side of the schematic. Microcontroller pin 4 can be used later to connect an additional analog or digital device. The current microcontroller firmware would have to be changed to include this expansion.

Three programs have been written for the LSM. The first program, Low_Speed_Record.bas, is used to record values collected throughout the balloon flight from the various LSM devices. The second program, Low_Speed_Read.bas, is used to read the values collected in the EEPROM and to write these values to the screen or a file, via a PICkit 2 and the PICkit2 user software. The third program, EE_Clear_LS.bas, is used to write 0's to all the EEPROM addresses of the eight EEPROMS contained in the LSM EEPROM array. These firmware programs have been written using the Swordfish Basic compiler language.

It is not possible to download the three programs to the LSM 18F2520 microcontroller at the same time. The 18F2520 must be loaded with the program that is to

be used. All firmware programs used with the LSM output UART serial communication at a rate of 4800 bits per second.

The Low_Speed_Record.bas program is used to record data during the balloon flight. This program's purpose is to collect data from the various LSM data sources and then write this data to the LSM EEPROM array. This program consists of the following three sections: header, subroutines, and main program.

The headers used for the three LSM firmware programs are very similar to the headers of the other module firmware programs. The header contains general information about the program, names of other programs that are included, dimensioning of variables, and aliasing of the microcontroller pins. The general information section of the header contains notes on the firmware, the device name the firmware is meant for, the oscillator speed, and oscillator settings. The include portion of the header contains information about other .bas program libraries that will be called upon throughout the firmware. These included library programs were not written uniquely for this project. The next portion of the header contains lines of code that dimension the variables that will be used, and gives names (aliases) to the Input/Output pins.

Thirteen subroutines are used by the Low_Speed_Record.bas firmware program. These subroutines are labeled Timer, Release, Parse, Header, Print, GetGPS, FixInd, SenCHK, Pointer, Init, Record, GetData, and CheckK. These subroutines perform the major tasks the firmware is responsible for and are called, in the appropriate order, by the main program.

The Init subroutine is used to initialize all variables to 0. This is very important, especially when performing bit writes to variable locations. When the microcontroller is programmed or when the program begins, the flash memory variable locations may not necessarily be cleared. The Init subroutine clears these addresses, making them ready to use.

The SenCHK subroutine is used to check for the one wire sensors, check the initial output voltage of the pressure transducers, check the reset pin, and initialize the digital input/output pins to either input or output. The results of each check are output to the screen using the UART interface and can be used to determine if the module components have been connected correctly and are operational. The SenCHK subroutine also checks the EEPROM array connection and operation by writing a value to address 0 of each EEPROM and reading the address. If the check determines that an EEPROM is not working, a STOP message is sent to the screen along with the EEPROM number that is not functioning properly. Detecting a faulty EEPROM will not stop the program from continuing.

The Pointer subroutine checks the state of the reset pin connected to LSM memory reset switch SW3. The LSM reset switch performs the same function for the LSM as the HSM reset switch performs for the HSM. If the switch is in the off position, the LSM microcontroller begins writing data to the first EEPROM at the first address and the release variable Rstat is set to 0. If the switch is in the on position the last written memory address, EEPROM, and Rstat values are restored from the internal EEPROM and the microcontroller begins recording data at the last saved data point. This portion of

the program also has an if statement that detects the initial “FF” state of the internal EEPROM and sets the restored variable values to 0, if this condition is detected. With the Reset switch in the off position the system is protected from writing over already stored data in the event of a system reset due to a momentary loss of power or malfunction.

The Header subroutine outputs column header labels to the screen. These column headers label the data output to the screen by the Print subroutine. The Print subroutine is used to output the data collected to the screen in tab delimited format. The values output to the screen are used to check proper function of the LSM sensors. The Print subroutine should be commented out before the actual flight to allow for maximum program efficiency and to make certain that the program has ample time to perform all tasks in between receiving GPS strings.

The GetGPS subroutine is used to determine when a new GPS string has been received. If a GPS string has been received, the GetGPS subroutine returns the string variables associated with the data that was received. If a string has not been received then no action is taken. The received string data cannot be immediately written to the EEPROM because a string variable is larger than a decimal variable. To be written to the EEPROM the collected GPS string variables must be converted into decimal variables.

The Parse subroutine converts the string variables returned from the GetGPS subroutine into byte, word, or long word type variables. Once converted, the variables are ready to be broken down into byte size variables and written to the EEPROM array.

The GetData subroutine is used to collect new values from the pressure transducers, temperature sensors and digital pins. The values received are in byte or word

format and only need to be broken into byte size values to be written to the EEPROM array. A 15 millisecond delay is used in between readings of the one wire temperature sensors to improve the chances of receiving accurate data from these sensors.

The Record subroutine writes both the converted GetGPS variables and variables returned from the GetData subroutine to the EEPROM array. The function ExtEEPROM.Write2 function is used to perform each write. Unlike the ExtEEPROM.Write function, the ExtEEPROM.Write2 function contains a 10 millisecond delay to allow time for the EEPROM to make the write. This delay is mandatory when writing to an external EEPROM. Variables that are larger than one byte are broken up by calling on the specific bytes within that larger variable. The bytes that make up the variables are written to sequential addresses. These variables are later reassembled by the LS_Read.bas firmware program. Other details about this subroutine are explained in the EEPROM array section of this chapter.

The FixInd subroutine is quite short and simply checks to see if the GPS fix variable returned by the GetData subroutine is greater than 0. If this value is greater than 0, the FixInd sets the LED2 pin high. This pin is connected to a green LED that can be checked to be sure the GPS has a fix before the balloon payload is launched. If the GPS does not have a fix, the returned fix value is equal to 0 and the green LED is not lit.

The Release subroutine is used by the LSM main program to sense when to send release commands to the SDM. Two release states can be commanded by the LSM. These states can be detected by the LSM through either the GPS data (if fix>1) or by reading

the board mounted pressure transducer in the event that the GPS does not have a fix (fix=0).

The GPS trigger portion of this subroutine first checks to determine if, according to the GPS altitude, the balloon has fallen 10 meters since the last reading was taken. If this has occurred, the subroutine does two things. First, it updates the Fall variable. The Fall variable is used to determine how far the balloon has fallen in total. When this variable gets to 300, signifying a fallen distance of 300 meters the balloon release servo position is commanded. The second action that takes place, once the balloon has fallen 10 meters, is that the address associated with the decrease in altitude is reset and stored in the FallStart variable. If the altitude continues to decrease at a rate of at least 10 meters per second, the FallStart variable is constantly reset. If the GPS altitude begins to increase, the FallStart variable is not reset. There is a second if loop in the Release subroutine that waits for a period of 5 seconds without detecting a fall and then resets the Fall variable. This means that in order for the balloon release to be triggered from the LSM GPS data, the GPS must have a fix and the balloon must have fallen at least 300 meters in 10 seconds. If a fall is detected, but the payload does not fall 300 meters in 10 seconds, the balloon release is not triggered. The GPS triggers a parachute release if the balloon release has been triggered, and the altitude is below 21,336 meters (70,000 ft).

If the GPS does not have a valid position fix, the pressure release portion of the Release subroutine is activated. This portion of the subroutine uses the readings taken from the on board pressure transducer to detect rises in pressure, signaling that the balloon is falling. If the pressure has risen continually for 10 seconds the balloon release

is triggered. If a pressure decrease is detected during that time, one second is added to the time for which continually increasing pressure readings must be received. The parachute release is triggered by the pressure portion of the Release subroutine when the balloon release has been triggered and 6 pressure readings corresponding to altitudes less than 70,000 feet are detected. The pressure portion of the Release subroutine is currently commented out because inaccuracies are suspected in the calibration of the on board pressure sensor.

The Timer subroutine uses the incoming GPS data to count one minute and then increase the Tstat variable and change the state of the Tstat pins used to synchronize timing with the HSM.

The CheckK subroutine is used to trigger a data collection sequence if data has not been received from the GPS after checking for it 88752 times (3 seconds without data). When triggered, this subroutine places a GPS error bit in the WRStat variable that is recorded and can later be read. This subroutine also alternates both LED's very rapidly to signify that no data from the GPS is being detected, possibly due to a loose or missing connection.

The Main portion of the Low_Speed_Record.bas firmware program calls on the subroutines in the appropriate order. It also performs a few system initialization tasks. First, this portion of the program disables the MCLRE pin, so that the microcontroller cannot be reset by this pin. Then the pins used for analog reception, and the pins used for digital reception/transmission are set (ADCON1). Values are then written to the special function register ADCON2 to set microcontroller timing associated with the internal A/D

converter. Setting the ADCON1 and ADCON2 registers is quite tricky and is microcontroller model specific. The microcontroller data sheet should be used to ensure that the proper values are written to these registers.

Once the general system setup is complete the main program flashes both the GPS fix LED and the status LED, and outputs a UART transmission that signifies the program is running. Then the variables are initialized and the sensors are checked using the Init and SenCHK subroutines. The recorded variables are then restored, or not, using the pointer subroutine and the Header subroutine is run to output data column headers to the screen.

The next portion of the program runs inside an infinite loop. The GetGPS subroutine is constantly run. When a new data set is detected, signifying one second has passed, the program calls on the Parse, GetData, Print (if not commented out), Record, FixInd, Release, and Timer subroutines, in that order. If new GPS data is not detected (valid or not) the K timeout will trigger and data will be collected approximately every 3 seconds. When the K timeout is triggered the Tstat pins will also be updated, but at a slower rate.

The Low_Speed_Read.bas LSM microcontroller firmware reads the external EEPROM array and outputs this data to the screen or file using 4800bps TTL serial, PICkit 2, and the PICkit 2 software UART tool. This program also unpacks and reassembles all variables that had been modified before the values are sent to the screen, or file. The columns output by this program are shown and explained in Table 5.1.

Table 5.1 Variables Recorded By and Output from the LSM.

Variable	Description
Fix	Variable used to signify the type of fix associated with the GPS position data
Time	GPS time received from GPS receiver
Sats	Number of Satellites the GPS is tracking (max 12)
Lat	Non decimal portion of the GPS Latitude
LatDec	Decimal portion of the GPS Latitude
Long	Non decimal portion of the GPS Longitude
LongDec	Decimal portion of the GPS Longitude
Alt	Altitude reading received from the GPS (in meters)
Speed	Ground speed received from the GPS (in knots)
Heading	Heading received from the GPS
Press	Analog voltage reading taken from the pressure transducer located on the main board
BPress	Analog voltage reading taken from the pressure transducer located on the balloon board
BoardT	Temperature reading taken from the temperature sensor mounted on the main board
BallT	Temperature reading taken from the temperature sensor mounted to the balloon board
T2A	Temperature reading taken from the T2 sensor
T3A	Temperature reading taken from the T3 sensor
LSMstat	Release stat variable determined by LSM
SDMstat	Release stat variable received from the SDM
Tstat	Timing variable used to synchronize the LSM and HSM
GPSerr	Value of 1 signifies that GPS data was not received and the K timeout triggered collection of the data set

The EE_Clear_LS.bas LSM microcontroller firmware writes 0's to every address in the EEPROM array. This clears all data that is held in the array. It is recommended that this program be run before each flight to clear the EEPROM and keep from improperly assigning data sets.

CHAPTER VI

GROUND STATION MODULE (GSM)

The Ground Station Module, or GSM, is used to receive information sent from the payload by the GPS/Telemetry Module and to send commands to the balloon payload. The Ground Station Module consists of the following components:

- Laptop Computer
- UIView software (requires ham radio license)
- TinyTrak4 – used as receiving KISS TNC
- Power Inverter – used to supply power to laptop during tracking.
- Radio Transceiver – Kenwood TH-K2 – Receives/Sends Data From/To Payload

These components are connected as shown in Figure 6.1. Figure 6.1 shows the general flow of data throughout the system main components. The system is designed to run off of a chase vehicle 12V power supply. This enables the payload to be stalked on the ground by a recovery team. The Power inverter component is used to convert the 12V DC chase vehicle voltage to 120AC voltage for use with the computer and radio power supply. It is best to check that the chase vehicle alternator is capable of running these devices and that the port supplying the 12V supply (cigarette lighter) is capable of

handling this level of current draw. It is always best to have a few spare fuses for the chase vehicle 12V port circuit during balloon payload tracking. Power for the TinyTrak4 TNC can be supplied from a separate 7-12V DC battery, or it may be connected to the chase vehicle 12V port.

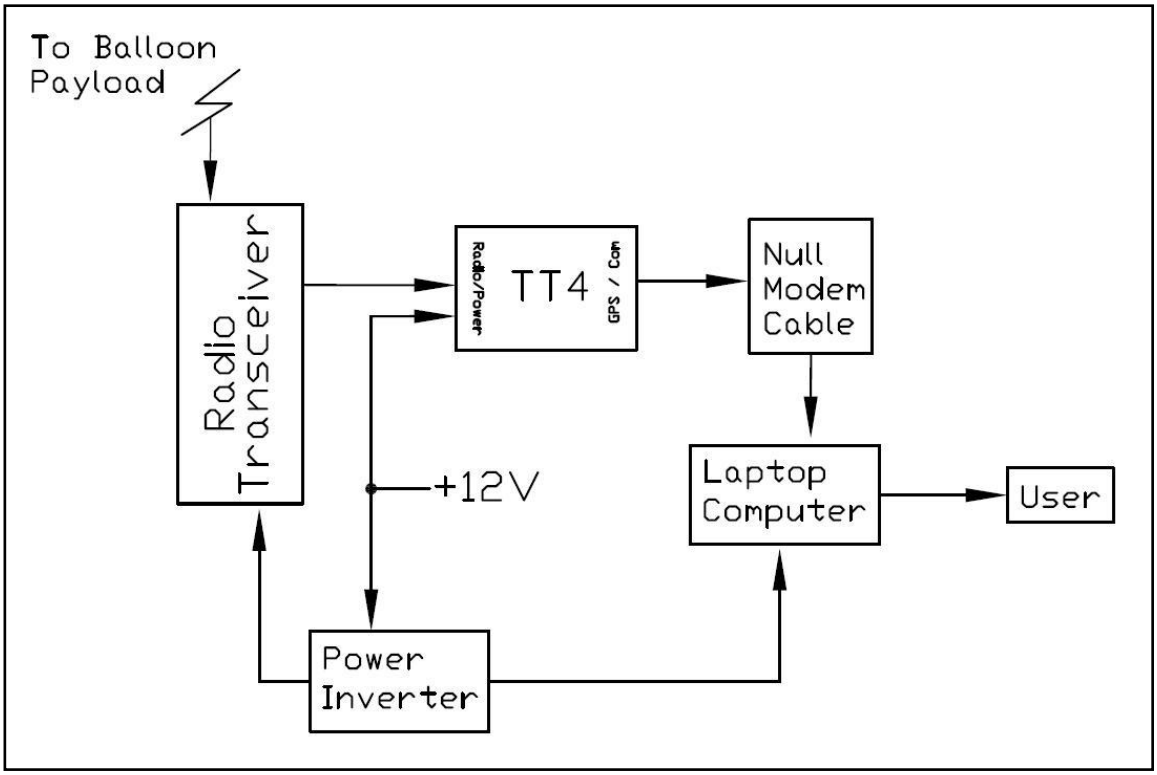

Figure 6.1 Ground Station Module System Diagram

The GSM receives radio transmissions that are packaged by the TinyTrak4 payload TNC. These data transmissions are received by the ground station radio transceiver and unpackaged by the ground station TinyTrak4. The TinyTrak4 sends the data to the GSM computer through a serial port and a standard null modem RS-232 cable.

The UIView program is run on the computer to log and display the data received and to display the location of the balloon payload on a map. The data collected can be used to assess the condition of the balloon payload. The position information is used by the ground team to stalk the payload during flight, and to retrieve the payload after the flight. In the event that the data transmitted by the balloon payload indicates a payload system malfunction the ground station radio transceiver may be used to trigger release of the payload balloon or parachute, by transmitting the appropriate DTMF tone sequence to the payload. The next section of this chapter contains a more detailed setup discussion of the GSM components.

GSM Component Setup and Configuration

The GSM is fairly easy to setup. Unlike the setup and configuration of the payload modules, the setup of the GSM can be done with minimum circuitry and programming knowledge. Setup of the GSM Transceiver involves changing only one of the transceiver default settings. This is done by accessing the transceiver menu and changing the default state of menu item 17 (APO, Automatic Power Off) to Off. This setting disables the transceiver power saving automatic shut off function. If this setting is not changed the transceiver will shut off after a period of time.

Setup of the TinyTrak4 consists of simply downloading the KISS TinyTrak4 firmware to the GSM TinyTrak4. This firmware is used to receive KISS TNC transmissions and to send them to the connected computer serial port. A standard female to female 9-pin null modem adapter (Radio Shack Part No. 55010600) is used to connect

the TinyTrak4 to the GSM laptop computer serial port. If the GSM laptop does not have a serial port, a serial to USB adapter will need to be used in conjunction with the null modem adapter

Setup of the GSM Laptop Computer involves installing and configuring the UIView software. The UIView software license is free, but is only granted to people with a valid Ham radio license At this point it is a good idea to capture a few maps that cover the projected flight path of the balloon payload. Because the UIView program is constantly being updated it is best to consult the UIView website or forum for instructions on map capturing.

To communicate properly with the TinyTrak4 the program must be "pointed" at the TNC. This can be done by opening the UIView program and choosing Setup->Comms Setup. This will open a window used to setup the communication port use to communicate with the TinyTrack4. The proper settings for use with the TinyTrack4 with the KISS firmware are located in Figure 6.2.

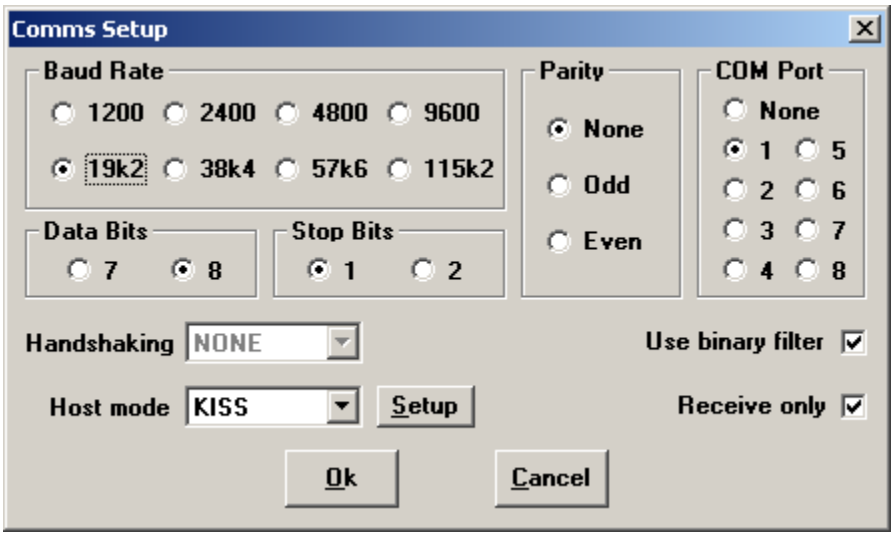

Figure 6.2 Comms Setup Settings for UIView and TinyTrack4 TNC

Once the Comm Setup is complete, it can be checked by first opening the UIView terminal window by selecting Terminal from the menu UIView menu bar. Then connecting the GSM and powering up the GPS/Telemetry module. Upon power up the GPS/Telemetry TNC will send a complete set of transmissions to the Ground Station. This transmission should be received by the Ground Station and appear in the UIView Terminal window as shown in Figure 6.3.

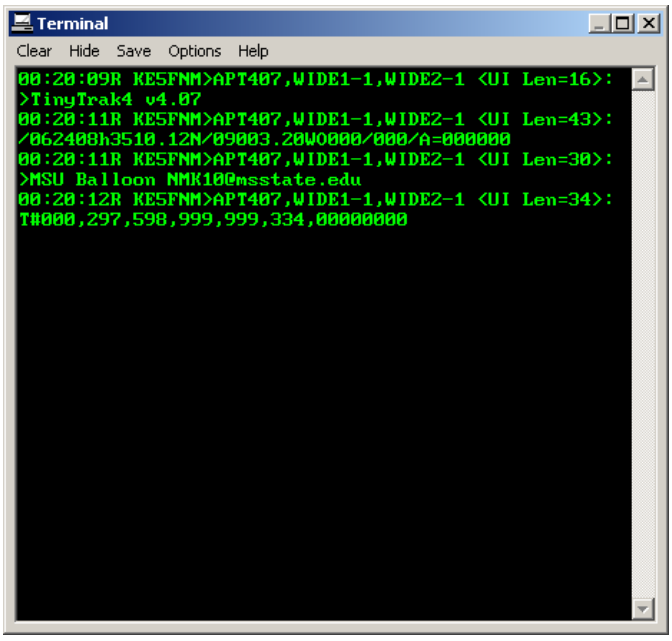

Figure 6.3 View from the UIView Terminal of Data Sent by Payload TNC

The terminal window is a useful tool to use when tracking the balloon payload as well. From this tool the GPS coordinates may be captured and transferred to a mapping program that contains more detailed road maps.

Before a flight begins it is very important to start logging the data collected by the GSM TNC. This is done by choosing Logs->Start a Log. This setting creates a text file containing all the information collected by the TNC. When the flight has finished the log may be stopped and retrieved from within the UIView program folder. The location for the log file can best be found by performing a file search for the file name associated with the log file. It is usually found at the address C:\Program Files\Peak Systems\UI-

View32\LOGS. Once found, the log file may be imported into a spread sheet program for analysis.

In the event that the Ground Station must trigger a cut down, the push to talk button must be pressed on the radio transceiver and the appropriate tones buttons must be pressed on the front of the transceiver. Currently, the DTMF cut down key is (C, *, 9) followed by either a 6, to release the balloon, or a "#" to release the balloon and chute. Once the parachute is released, the Servo Driver will set the chute release pin high and the last A/D value sent by the payload TNC will change from 000 to 999 signifying that the release all command has been received.

CHAPTER VII

CONCLUSION

This system has been designed as a solution for the data collection, control and telemetry needs of a high altitude balloon payload. This system combines the flexibility of microcontrollers with the availability of commercial off the shelf components to create a modular system capable of being modified, simplified or expanded upon. The flexibility and functionality of this design creates the foundation for development of further High altitude testing capabilities.

The modular structure of this design allows for components and capabilities to be tailored to fit the needs of individual mission requirements. Separate microcontrollers were used for each module to allow for increased computational function and to allow for modules to have ample capability to be modified for future system iterations.

The payload box design and construction techniques create a buffer from the harsh environment. This buffer allows the system components to operate properly even when subjected to the low temperature and high shock loads that are associated with high altitude balloon flight. This and other parts of the system design were driven by the system requirements outlined in Chapter I. These system requirements are presented again in Table 7.1 and are matched with the system module or component designed to fulfill these requirements.

Table 7.1 System Requirements and Designed Solutions

System Requirements	Designed Solutions
Data Collection	I2C external EEPROM array
Reduced Cost	Modular System, PIC Microcontrollers
Deployable Chute	SDM, Release Mechanism
Increased Reliability	Altitude Chamber Testing
IMU Data Acquisition	High Speed Module
GPS Data Acquisition	Low Speed Module
Temp/Pressure Acquisition	Low Speed Module
Automatic Balloon/Chute Release	Low Speed Module
Commanded Emergency Release	SDM (DTMF tones sent from GSM)
Ground Tracking	GPS/Telemetry Module, GSM
Telemetry	GPS/Telemetry Module, GSM

Table 7.1 shows how the system design aspects (modules) were directly driven by the outlined system requirements. Additional functionality was added because it did not interfere with the fulfillment of the targeted system requirements.

Future Possibilities and System Improvements

The system that has been discussed and explained is a baseline system that was built and tested. This system is an initial design iteration prototype and is intended to be improved upon as new technology, requirements, and better solutions are found. A few recommended system changes will be outlined in this section.

Changing a few basic components of the LSM, HSM, and Servo Driver module circuitry could greatly improve the stability of the circuitry. One change is to replace the

four +5V regulators to LDO (low drop out) regulators. This change will provide greater stability to the system voltages throughout the voltage range of the flight battery. This change is highly recommended, because it is quite inexpensive and prevents possible drift of the analog sensor voltages as the flight battery becomes discharged.

It is also recommended that decoupling capacitors be placed between the power and ground traces at each integrated circuit component. This will better filter the supply voltage and allow for a more stable supply for the different boards and components. A better supply voltage filtering system, possibly consisting of a capacitor array or possibly adding inductors, would greatly reduce the susceptibility of the system to outside noise and interference, such as that generated by the payload transceiver. Proper supply voltage filtering will also increase the longevity and reliability of all of the associated system components.

It is possible that greater in-flight telemetry range can be achieved through proper antenna selection. A no ground plane (NGP) antenna should improve the in-flight performance of the system without raising cost significantly. However, the possible loss of range, once the payload has landed should also be considered when deciding what antenna to use for the balloon payload transceiver.

The system presented is a stepping stone that is to be used as the basis for future development of the high altitude ballooning capabilities of the program. It has been built and the baseline functionality of the system has been proven. Future testing and design iterations are recommended to allow for further development of the components and implementation of better technology. The system, in its current form, fully fulfills the

outlined requirements, and has been tested to perform all the outlined and described tasks.

REFERENCES

1 Braun, R., Wright, H., Croom, H., Levine, J., Spencer, D., "Design of the ARES Mars Airplane and Mission Architecture", *Journal of Spacecraft and Rockets*, vol.43 no.5, 2006, pp. 1026-1034.

2 http://www.grc.nasa.gov/WWW/K-12/airplane/atmosmrm.html, accessed April 6, 2010.

3 Shearer, C., Cesnik, C., "Nonlinear Flight Dynamics of Very Flexible Aircraft", *Journal of Aircraft*, vol.44 no.5, 2007, pp.1528-1545.

4 http://www.eoss.org, accessed April 6, 2010.

5 Electronic Code of Federal Regulations, Title 14, Part 101-Moored Balloons, Kites, Amateur Rockets and Unmanned Free Balloons, http://ecfr.gpoaccess.gov/cgi/t/text/text-idx?c=ecfr&rgn=div5&view=text&node=14:2.0.1.3.15&idno=14, accessed April 6, 2010.

6 Brown, C. D., *Elements of Spacecraft Design*, AIAA Education Series, Reston, VA, 2002, p. 381.

7 Anderson, J. D., Jr., *Introduction to Flight*, 6th ed., McGraw-Hill, Boston, 2008.

8 http://www.kaymont.com/pages/sounding-balloons.cfm, accessed April 6, 2010.

9 Conner, J. P., Jr. and Arena, A. S., Jr., "Near Space Balloon Performance Predictions," AIAA 2010-37, 48th AIAA Aerospace Sciences Meeting, Orlando, Jan. 4–7, 2010.

10 Kendziora, C., Marinelli, M. and Ruschman, M., "TPG and PGS Thermal Conductivity," BTeV-doc-1801-v3, Fermi National Accelerator Laboratory, June 9, 2003.

11 http://www.eoss.org/ansrecap/ar_160/recap138_139.htm, accessed April 8, 2010.

12 Griffin, M. D. and French, J. R., *Space Vehicle Design*, 2nd ed., AIAA Education Series, Reston, VA, 2004, p. 275.

13 Wallio, R., "GPS Receivers vs 60kft," http://showcase.netins.net/web/wallio/, (homepage), accessed April 6, 2010.

14 "TinyTrak4 Built Hardware Manual V0.63," http://www.byonics.com/tinytrak4, accessed April 7, 2010.

15 "75T204 5V Low-Power Subscriber DTMF Receiver," TDK Semiconductor Corp., April 2000, http://www.datasheetcatalog.com, accessed April 7, 2010.

16 "24AA512/24LC512/24FC512 512K I2C™ Serial EEPROM," Microchip Technology Inc., 2009, http://www.microchip.com/, accessed April 7, 2010.

APPENDIX A

BATTERY CURRENT/DUTY CYCLE ANALYSIS

Table A.1 Current Draw of the Various Components During a Nominal Balloon Flight

Device	Description	Consumption (A)	Duty Cycle	Number of Devices	Amp Hours
LED Red	LED Regulator	0.0150	1.00	3	0.0450
LED Yellow	LED Status	0.0150	0.10	3	0.0045
LED Green	LED GPS Fix	0.0150	1.00	1	0.0150
18F1320 (p.247)	SDM Micro	0.0150	1.00	1	0.0150
18F2520 (p.332)	LSM, HSM Micro	0.0250	1.00	2	0.0500
24LC512 Active	EEPROM (HSM, LSM)	0.0004	0.05	16	0.0003
24LC512 Passive	EEPROM (HSM, LSM)	0.0001	0.95	16	0.0015
015A24R	Pressure Transducer	0.0060	1.00	2	0.0120
DS18B20	Temperature Sensor	0.0040	1.00	4	0.0160
TDK 75T204	DTMF Chip	0.0160	1.00	1	0.0160
18X LVC	GPS	0.0900	1.00	1	0.0900
TH-K2A	TX Transmit	2.0000	0.05	1	0.1000
TH-K2A	RX (Standby)	0.1000	0.95	1	0.0950
HS-225MG	Servo Active (No Load)	0.3000	1.00	2	0.6000
Sparkfun IMU	IMU	0.0240	1.00	1	0.0240

From this Table, the total current draw is 1.08 A-hr. The battery selected for this system has a capacity of 5.2 A-hr and an estimated cold capacity of 75% of that value, or 3.9 A-hr. Using the cold capacity gives an estimated battery life of 3.6 hr.