Project Number: MQP SMJ 1A06

Avionics Electronics Suite

A Major Qualifying Project Report:

submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

Ryan D. Meador

Date: April 26, 2007

Approved:

Professor Susan M. Jarvis

Abstract

This MQP consists of designing and building the avionics systems for the WARRIORS II rocket. The avionics are comprised of a flight data recorder (FDR), flight computer, telemetry, power supply, and support modules. The FDR possesses a sensor suite including an altimeter and accelerometer, and logs the data over the entire flight. The flight computer is a backup for a commercial system and is capable of orchestrating all rocket functions. Telemetry relays real-time FDR data to the ground.

Table of Contents

Introduction

WARRIORS I Overview

The WPI AIAA Research Rocket for the Investigation and Observation of Recovery and Staging (WARRIORS) was constructed and launched during the 2005-2006 school year. It used an airframe with a diameter of 2.63" and an overall length of about 7'. The design featured a fairly unique staging system. In a traditional system, the rearmost section of the rocket detaches when the motors are exhausted; WARRIORS I sported three boosters instead. It also had one larger motor in the core that was ignited after booster separation. The fuselage was composed of a phenolic tube, broken into many segments with bulkheads that attached to each other. This provided some degree of modularity in the design, but wires from the various subsystems had to converge on the avionics bay, which contained an on-board flight computer. Many problems were encountered running wires between compartments because of the need for connectors that could be unplugged.

One of the design goals of this rocket was to avoid the use of pyrotechnic charges. As a result, the jettisoning of the boosters was accomplished by electromagnets. A hinge at the tail bore the thrust load of the booster while the electromagnet held it against the fuselage. When the electromagnet reversed to repulse the booster, it pivoted away from the fuselage until the hinge disengaged and separated. The large repulsive force requirement entailed the use of a high-capacitance, high-voltage capacitor power supply and triggering circuitry duplicated for each booster. This circuitry remained inside the core of the rocket even after the boosters were jettisoned.

The flight computer was a commercial model, the MC2 from G-Wiz Partners. It controlled all events that occurred after liftoff. It consisted of a sensor suite, a processor, and four outputs designed to drive pyrotechnic devices, since that is the norm for staging and recovery deployment in the model rocket world. These four outputs are "user programmable" in the sense that they can be programmed to trigger a specified amount of time after launch, burnout, apogee or low altitude being detected. The barometric sensor and accelerometer are read at 33Hz and can store up to 18 minutes worth of data. There is an expansion header on the MC2, but G-Wiz hasn't released a telemetry module. However, the data can be downloaded via USB when the rocket is recovered. [G-Wiz MC2]

WARRIORS I had several fatal problems, most notably (and finally) the failure of the second stage to ignite that resulted in the loss of the rocket. It would have been much easier to diagnose the failure(s) had the flight data survived the crash, but the flight computer's data was unrecoverable even by the engineers at G-Wiz. It is believed that the failure to ignite the second stage was caused by insufficient voltage in the battery powering the pyrotechnics. Combined with a failure of the mechanical springs to deploy the parachute, the avionics were completely ineffective after booster separation. Because the flight was so much shorter than expected, the backup computer (actually just a simple timer) didn't fire the backup parachute (via pyrotechnics) before impact with the ground. The impact broke the rocket into many pieces. Many of the breaks occurred at the bulkheads. The avionics bay, being in the middle of the rocket, was one of the most intact modules, but even so, the data on board the flight computer was lost.

WARRIORS II Overview

The second generation of the WARRIORS project combines innovative new designs with improving the reliability of the old system. The mechanical design of WARRIORS II was done by Nicholas Behlman, Ryan Caron, Sara Dempsey, and Robert Niles. The volume of the rocket is close to that of WARRIORS I. An important difference is the body diameter is being increased to 4” and the body length is slightly shorter. It has an internal frame composed of carbon fiber rods that will take the majority of the stress on the structure, whereas the older model had only the phenolic tube as structure. There are bulkheads internally once again, but instead of using modules, the entire frame is be removed from the skin as a unit, allowing greater and easier access to the internal systems. This also facilitated the routing of wires, since they don't need connectors at the edge of each module.

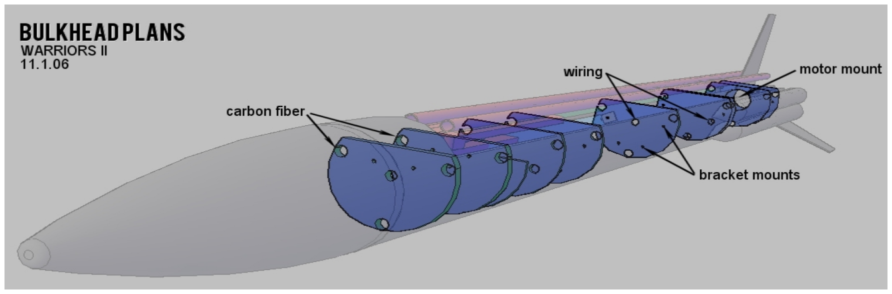


Figure 1: Internal Bulkhead Design

The new rocket has four booster rockets, with each one being taller to accommodate the electronics for separation that were formerly in the core. This will allow the jettisoning of more weight when the boosters separate. The boosters are also grouped asymmetrically in pairs (WARRIORS I was radially symmetric) pattern due to the unique recovery mechanism. Initially it was thought that this would require that the boosters not all separate at once to prevent them from becoming entangled in one another, but testing proved it was better for them to all separate at once.

The unique recovery mechanism alluded to above is wings. The rocket will not fall back to earth with a parachute, but will instead glide much like an airplane. The wings deploy via a servo that swings the two leading edges (actually part of the rocket skin, and the primary reason for the internal frame) away from the body, stretching nylon fabric out to form a wing. Initial deployment will be under computer control at apogee, but the rocket's control system supports handing off control to a human operator if desired, though this isn't the planned mode of operation.

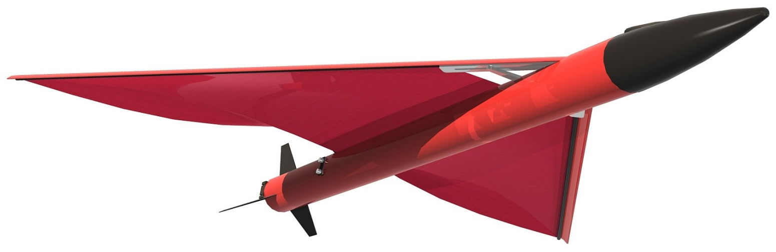


Figure 2: WARRIORS II In Flight (CAD Rendering)

Requirements

Such an ambitious mechanical design requires ambitious electronics to control it. These electrical and electronic systems have been divided into five groups: flight data recorder (FDR), flight computer, telemetry, power supply, and support modules. The final system also contains circuitry inside the boosters to respond to the signal to separate and a module dubbed the "command receiver" that permits release of an emergency parachute from the ground – neither of those are part of this MQP. Over the course of the project, several changes to the electronics design requirements were received from the mechanical team. These changes, all part of working with in an interdisciplinary team, were the source of multiple delays and challenges. The requirements as laid out in this section are essentially the final version. Some of the major changes from the initial specifications are highlighted in the end of this section. To provide an easy way to refer to the flight data recorder, flight computer, and telemetry modules as a single entity (mostly comparable to the MC2), they are known as the Meador Avionics System, or MAS.

The overall design must be robust enough that the failure of any one component won't jeopardize the rocket. During the design, size and weight were considerations; while no maximum size was specified, a significant effort to make the system as small and light as possible was made. Also, as this is a rocket, G-forces must be considered as well. The expected take-off acceleration of WARRIORS II is around 15G's, so all the avionics must be able to withstand that acceleration. The deceleration of a controlled landing is unknown, but an uncontrolled landing (a crash) will likely encounter many hundreds or thousands of G's.

The requirements for the flight data recorder were that it perform at least at the level of the MC2 in terms of what data is recorded (altitude and acceleration) and how fast it is recorded (33Hz). It should be a "black box" in the sense that the data contained in it will be able to survive a catastrophic failure of the vehicle. There are two ways this could be accomplished: by the physical device surviving, which is preferred, or by transmitting the data to the ground via the telemetry module before being destroyed.

The flight computer should be able to fly solo as a a drop-in replacement for the MC2, but primarily geared towards being a redundant system used alongside the MC2. This arrangement is quite complicated because the rocket's control requirements have surpassed that of what the MC2 can provide. To provide a common bridge between the MC2 and the MAS, the support modules perform all the advanced control functions based on simple logic-level triggers from either the MAS or MC2, meaning nearly all the complexity of the flight computer is in the software.

Requirements for the telemetry module are straightforward: transmit to the ground in real time all the data recorded by the FDR. The primary motivation for this capability is to safeguard against the destruction of the data recorder by keeping a second copy of the data outside the rocket. The range of the radio must exceed the maximum altitude of the rocket and also the distance the glider will fly away from the launch site. The minimum altitude for a launch deemed successful is 1500'. The maximum altitude is still unknown, it is not likely to exceed 3000'. The distance the rocket will travel from the launch pad depends on the glider wing configuration and flight path. A spiral flight path will be preferred to minimize the distance the rocket travels from the launch pad, but in the final mechanical design, there was no provision for this – it is expected that minor imperfections in the wing and the balance of the rocket will cause it to fly in a circle.

The requirements changed fairly dramatically over the course of this project. The mechanical team was in the position of being the customer, and I was trying to design to meet their needs. Some of the requirements changes were due to imperfect communication, but just as many if not more were because of changes to the mechanical design or to the philosophy of the group. For instance, in the earliest sets of requirements, it was thought that the MAS would be the backup computer on the first flight and then be the only computer (to save weight) on subsequent flights. This decision was later changed to have the MAS be a permanent backup for the MC2. This necessitated a number of software changes. More diverse sets of changes to the requirements were experienced with the support modules, and eventually an entirely new module (the Nichrome Controller) was added to the list only a week or so prior to the first scheduled launch date.

Design

The design consists of three main components, a “black box” flight data recorder, a flight computer, and a telemetry module. The FDR will have to have a suite of sensors to assess the status of the rocket. The other two modules will both require the same data that is recorded by the FDR. The flight computer uses this data to make decisions about when to eject the boosters, light the second stage motor, or deploy the wings. The FDR data stream also contains information about the actions the MC2 is taking, so the MAS flight computer can monitor the status of the MC2 and step in if needed. The telemetry module obviously needs access to this information because it must transmit it all to the ground.

The power supply module provides power to all parts of the rocket. Because of the diverse set of circuits that are on board, the power supply must provide 3.3V, 5V, ~6V and ~12V. The 3.3V and 5V are for most of the MAS circuitry, the servo requires a voltage near 6V, and the igniters and MC2 desire something near 12V.

The block diagram below depicts the MAS, the power supply, and the three original support modules.

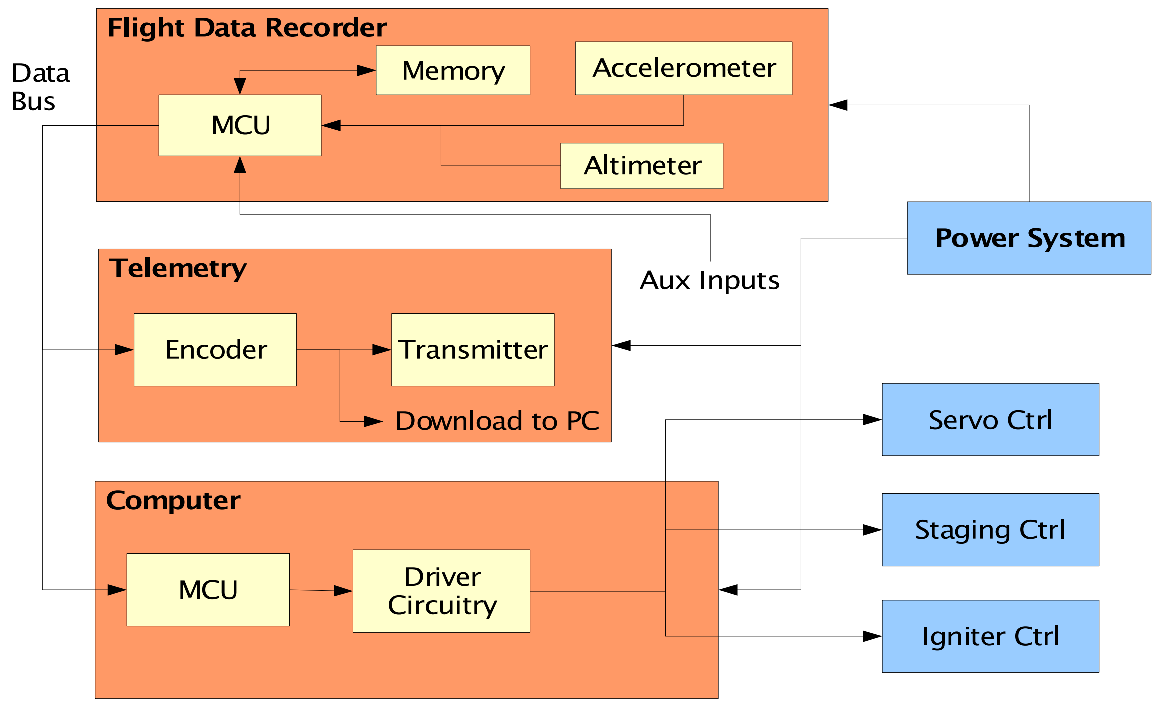


Figure 3: System Block Diagram

Flight Data Recorder

The core of the FDR is a PIC microcontroller. It was chosen due to ease-of-use considerations, cost, and a wide array of on-board peripherals (such as A/D conversion). This processor orchestrates the sensor polling and data storage, as well as driving a bus that sends the data to the flight computer and telemetry modules. Factors in the ease-of-use decision included WPI's facilities for programming various models of PICs as well as classes teaching their use. In addition to these places to turn for assistance if something goes wrong, my extensive personal experience with these chips made the decision easy. The PIC16F785 in particular was chosen for this role because it is the newest and most full-featured of the PIC16 line, which is a line of mid-range MCUs with moderate (~18) pin counts. It has 12 channels of 10-bit A/D, thus removing the need for a separate A/D chip. The internal oscillator and timer makes it easy to poll the sensors on a regular basis, as well as reducing the overall component count and size of the PCB. [PIC16F785 Data Sheet]

Sensors

The accelerometer is an ADXL321 from Analog Devices. This is a MEMS device. MEMS devices are extremely small and light-weight, while also being accurate and reasonably priced. It measures ±18G on two axes and can withstand impacts of up to 10,000G, making it well suited to this rocket. By powering it off the 3.3V supply, it will have an output swing of approximately 1.65V, corresponding to a resolution (with 10-bit A/D) of 0.05G. [ADXL321 Data Sheet]

The altimeter, a pressure sensor, was originally intended to be a Motorola MPX2100. This particular chip has altitude sensing as one if its intended purposes. It also has a port for connecting a pipe that will extend outside the fuselage of the rocket to get a more accurate pressure reading during flight, which is important since engine exhaust and other factors may cause the rocket's internal pressure to be different than that outside. [MPX2100 Data Sheet] Unfortunately, this part was out of stock at the time the parts were being ordered, and the backorder date was late April (the time of this writing); waiting for the part to come in was impossible. The simplest replacement was the MPXM2053, which is of the same family. This part was available through the WPI ECE Shop because it is used in the demo boards for the TI MSP430, taught in ECE2801. The pinout and voltage range is the same as the MPX2100, and it is also ported. The drawback to this part is that the pressure gradient it expects is the opposite of the MPX2100, so while it should do the job, it probably won't be as accurate.

In addition to the sensors that monitor the rocket's external status, the FDR also has a number of inputs that monitor the health of the on-board systems. These include battery voltage monitors and digital status indicators for both the MC2 and MAS control signals.

Memory and Data Bus

A 1MB FLASH memory was chosen as the storage medium because it is nonvolatile and can be interfaced serially. The part chosen was the M25P80. It supports writing data up to 256 bytes at a time, which are then written as a single page. This helps lower the overhead on the processor of initiating the write cycles. It would have been preferably to have a memory device that could be written continuously from start to end, but it was not possible to find such a device. The 1MB size was deemed more than sufficient because it can store more than half an hour's worth of data even at a very conservatively-estimated record size. When the final record size was learned, it was only 8 data bytes plus a one byte checksum (just an XOR of the other bytes), so it was decided to do the sensor sampling at 50Hz. At this speed, the memory would store approximately 114,000 records and record for 38 minutes. The storage scheme puts 28 records on each 256 byte page to prevent the additional record keeping required for splitting records between sectors in the memory.

To save I/O pins and further lessen the overhead on the PIC, the same bus that provides data to the flight computer and telemetry module also shares some pins with the FLASH memory, permitting the data to only be output once and go to both the memory and the other modules. The bus is a high-speed synchronous serial modified SPI bus. It has three lines instead of the usual two: serial data, serial clock, and packet sync. The packet sync line goes high when a packet is being sent so that if a bad packet is ever received by the other modules, they won't have to rely on timing or sync bytes to figure out when the next packet begins. It also permits them to spend more processing time doing their assigned duties and only go into receive mode when the packet sync line goes high. The other two lines in the bus function as in traditional SPI, with the data line being shared with the FLASH memory. The FLASH memory has a separate clock to facilitate read back of the data after the rocket has landed. The read data must be pulled out of the FLASH memory and put on the data bus so that the telemetry module can encode it into RS232 for download to a PC. For both the memory and the other modules on the bus, data is latched on the rising edge of the clock.

Flight Computer

The flight computer (or flight controller) does not need to adhere to the MC2's standard of providing high-current outputs as it is not intended to directly drive pyrotechnics. All the output signals are TTL to the support modules. In addition to interfacing with the support modules, the MCU also needs to read the data being sent by the FDR. The total number of I/O pins required is quite low, even after adding two LEDs for debugging purposes. The PIC16F684, which is of the same family as the one in the FDR, was considered for this role, but it was decided that the insignificant savings in cost and space would be sacrificed to have fewer different chips in the final design. The MCU in the flight computer is the same model PIC16F785 as in the FDR.

Software

The software of the flight computer is the most complex component of this module. The program must interpret the sensor data from the FDR and react to certain events. The system should arm itself when it notices launch by reading the accelerometer. At any time during the flight, it will determine if an action is required, and then permit the MC2 a few seconds to come to the same realization and act. If the MC2 doesn't act within the specified period of time, the flight computer will step in and initiate the event. When detecting booster burnout (the acceleration should drop), it will initiate booster separation. A short time after booster separation, it will ignite the core motor. Apogee can be determined with the altimeter or accelerometer, but because the accelerometer is estimated to be much more accurate, the software uses that as the input and integrates the acceleration to get velocity; apogee is when the velocity hits zero. At apogee, the wings must be deployed. Ideally, throughout all these states, the system would maintain internal timing information so that if the data from the FDR is cut off during flight or indicates a problem (the second stage not igniting, for instance), the timer can be used to base subsequent events and hopefully will permit some degree of failure recovery. The software to do the timing based events was never written because it was decided by the mechanical team that the MAS would operate in record-only mode, effectively nullifying the job of the flight computer. The code to perform monitoring and backup of the MC2 was written but not fully tested.

Telemetry

Early on in the design phase, it became clear that implementing a radio link with long range and low cost was not going to be easy. Investigation of components had shown only one likely solution. Using a pair of Lynx RF modules, it was possible to make a telemetry unit with an estimated range of 3000+ ft for a parts cost of $40 and minimal size and weight. The specific solution is the TXLC-434-LR transmitter module and its matched receiver. It is capable of data rates up to 10Kbps at maximum range. Other transmitter/receiver combinations can still be investigated if a longer range is desired, however it seems unlikely that a cheaper setup will be discovered considering that many individual modules (either a transmitter or receiver, not a pair) discovered during research were on the order of $30. The TXLC-434-LR uses CPCA modulation (carrier-present, carrier absent) to send data, so it is innately a digital device and well-suited to sending serial data. The protocol that was implemented, for ease of coding and debugging, was simply 9600 baud 8N1 RS232. The data on the FDR's bus is synchronous serial, so that data is converted by a PIC12F675 before being broadcast via the radio. Before choosing that model PIC, it was investigated whether or not it would be possible to use a UART or some combination of shift registers and combinational logic. The simplicity, small size, and cheap cost of the PIC12F675, which has only 8 pins, made it a good choice for this role, despite the overkill in having a fully programmable 16MHz processor here (an external 16MHz ceramic resonator was used to give the clock the stability and high speed necessary to produce 9600 baud output). Using the PIC12F675 in situations where it is over powered for the role it needs to fill was to become a theme of this MQP, as the constantly-changing requirements made its flexibility increasingly important.

Power Supply

As mentioned above, the MC2 requires 12V for operation, whereas all MAS systems operate on

3.3V or 5V. The igniters also need a voltage near 12V at approximately 8A for at least 2sec. At full current draw, the servo is likely to require 2A at 6V. These power requirements are quite extreme, especially considering the weight, size and cost constraints of the project. It is impractical to regulate the 12V source down to ~6V for the servo's use, especially if it is drawing 2A. This necessitates a ~6V source for the servo. In consultation with the mechanical team, it was decided to use two Lithium Polymer batteries. One would be 7.4V, which would power the servo directly and also be regulated down to 3.3V and 5V to power the MAS circuitry. The other battery is 11.1V and powers the igniters and other high-current, high-voltage devices. The MC2, however, for safety reasons, shouldn't be powered off the same battery as the pyrotechnics, so it was decided to build a step-up regulator to produce 12V from the 7.4V battery.

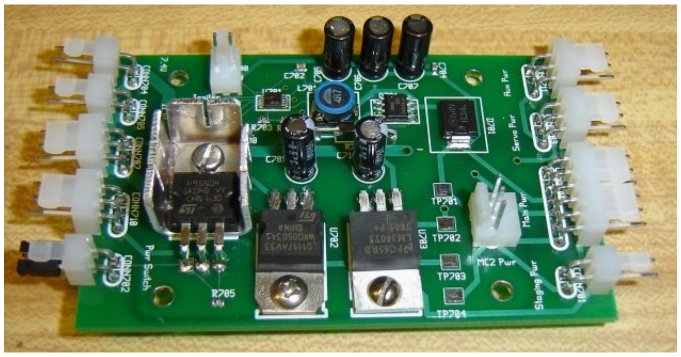


Figure 4: Fully Assembled Power PCB

Lithium Polymer batteries are the best choice from a weight standpoint. They have a very high energy density. Also, by being entirely solid-state, they are more able to resist the acceleration of the rocket during launch. Unfortunately, these are also some of the most expensive batteries on the market. The two packs each ended up costing approximately $30.

WARRIORS I had a compartmentalized design with the avionics bay in the middle, so charging the batteries and turn the rocket on and off required major disassembly. The power supply for WARRIORS II was designed with these lessons learned in mind. An important request from the mechanical team was the ability to charge the batteries while they are in the rocket and also provide an external power source so that the batteries wouldn't be drained while on the launch pad for extended periods of time. It was also desirable to have a switch located in an accessible location on the rocket that could turn the entire system on and off. An early design for this system was quite complicated and involved a large number of high-current FETs that were used to switch the power around. I observed, however, that by dividing the ground paths into two sets, it was possible to reduce all that circuitry to a single FET with the only cost being that it was impossible to charge and operate the control system at the same time. All the ground power connections, such as external power and charging, would be directly connected to ground, and thus active whenever they were plugged in. The single FET would be controlled by the main power switch, and when on, would connect all the ground paths through the MAS an other avionics circuitry through to the battery's ground. This solution seemed too good to be true, so I spent quite a long time making sure that all the current paths through the circuit wouldn't conflict with each other, and I was eventually satisfied that that was the case.

Servo Controller

The servo controller's purpose is to move the servo that controls wing deployment to the deployed state upon receiving a signal from the MC2 or MAS. The servo, though much larger than a standard hobby servo, has the same pulse-width-modulated interface. The capability to generate the required 1-2ms pulse every 20ms is well within the means of a PIC12F675 using its internal 4MHz oscillator, so that part was chosen to be the core of this module. To preserve the ability to run the servo off of a standard hobby radio receiver, a 2:1 data selector was used to the servo's control signal between the PIC's output and a port for a radio to connect externally. The selector's choice is controlled by one of the extra PIC pins. The board also includes a button to manually trigger the servo and an LED to provide feedback. The software is very simple, using a timer interrupt to schedule the pulse every 20ms and a compiler-provided delay function to control the pulse width with a resolution of 10µs.

Staging Controller

In a traditional rocket, the stage separation is accomplished via pyrotechnics that literally blast the stages apart. One of the requirements for the mechanical/aerospace MQP (also a requirement for WARRIORS I) was to use a non-pyrotechnic staging solution. The solution invented for WARRIORS I was a series of electromagnets in the core powered by large capacitors that repulsed the boosters on command. As an evolution of this design, the staging system of WARRIORS II moved the electromagnets and associated control and power circuitry from the core to the boosters. This permits the dropping of more weight when the boosters are jettisoned, which is even more important for WARRIORS II than it was for WARRIORS I because it has one more booster. Ryan Caron was responsible for the boosters, so the in-booster circuitry is not part of this MQP. He decided that having electrical contacts between the boosters and the core would be problematic because of vibration during flight, and might result in erroneous booster separation. The interface that was agreed upon for triggering was a 36kHz modulated IR beam. It is the staging controller's responsibility to generate that signal upon triggering by either the MC2 or the MAS.

One of the original requirements from the mechanical team was to separate the boosters in two sets, opposite boosters being considered a set. This was because it was feared that with the asymmetric configuration of the boosters, if both boosters on the same side were jettisoned at the same time, they would hit each other and jam. The delay between the sets was specified as 200ms. A PIC12F675 was chosen for this role because it was much smaller and simpler than a timer circuit to generate these two sets of 36kHz pulses. It turned out to be a fortuitous decision, since during testing, it was discovered the system actually worked best when all the boosters jettisoned at the same time, so a minor software change was necessary. Running at 4MHz, producing a 36kHz wave is not a lot of work for the processor, but it is fast enough that the internal timer can't make interrupts trigger at that frequency.

The compiler also refused to generate a delay function with the required resolution for the same reason. The solution was to write a delay loop in assembly language (the only use of assembly in this entire project).

Igniter Controller

The igniter controller is responsible for igniting the second stage (core) motor after the boosters have been jettisoned. Because this module is just a glorified switch, I designed it to be two FETs in parallel, each with their gate connected to the MC2 and MAS respectively. That configuration permits either computer system to initiate the second stage burn. The initial requirements from the mechanical team were that the igniters would draw around 4A, so I designed the module using FETs rated at 8A. It was later discovered that the custom-built igniters had a much lower resistance than expected, and the current draw was 8A or more. No design changes were made as this was a non-critical module for our first flight, which wasn't scheduled to use the core motor.

Nichrome Controller

Nichrome wire is a special alloy that is designed to be heated and/or melted by application of electricity. It is commonly used as the “hot wire” in the shaping of Styrofoam forms. When given even more electricity, it melts cleanly and easily, making it an ideal semi-pyrotechnic way of holding parts of the rocket together that needed to come apart in flight. The reflex on the trailing edge of the rocket's wings was added fairly late in the mechanical design to guarantee an acceptable angle of attack during the glide phase of the rocket's flight. The original plan was to have it be spring-loaded and held in place by the wing struts when the wings were closed and spring open when they deployed, but it was found that the required spring strength was enough to push the wing struts away from the body. A last-minute solution was to tie the reflex down with a piece of nichrome wire and electrically melt the wire at the same time as wing deployment to free the reflex. Because this circuit was so last-minute, it was designed non-optimally using leftover and spare parts from the other modules, such as a DIP-sized PIC12F675 used for prototyping and a spare power supply switch FET. The board was fabricated by milling a copper clad board on a CNC machine.

Design and Programming Environment

Schematic and PCB Layout

Nearly a year ago, I switched the last of my Windows computers to Fedora Core Linux because

I discovered I was much more productive when using Linux than I had been using Windows. When starting this MQP, I had never attempted to do any electronics schematic capture or PCB layout under Linux. I decided that if I was going to be able to effectively do the work required for this MQP, I needed to be able to work on my own time using my own computers, so the search was on for an open-source tool suite. I found the impressively mature gEDA project (GPL'ed Electronic Design Automation). The primary tools in this package are *gschem* and the aptly-named *pcb*. [gEDA]

Both programs were available as RPMs, so installation was very easy. They were also both very easy to use, with an intuitive interface that I had never before encountered in any sort of EDA product. I didn't encounter any of the annoying “you can't do that” limitations imposed arbitrarily by packages such as Orcad and Xilinx. Among the nice features were a large library of preexisting components and an easy way to add custom components (I had to create a number of PCB footprints myself). There were a few minor bugs, as can be expected of any project, especially an open source one, so I joined the developers group and started submitting bug fixes whenever I encountered something that prevented my work from getting done; the most significant was adding the ability to one of the tools to handle filenames with spaces in them. The design and PCB layout phases of the project went very smoothly with these tools, probably much more so than it would have if I had used any of the tools provided through WPI. Interoperability with other products was also not an issue, as both *gschem* and *pcb* can export their data as images or standard file formats, such as standard netlists from *gschem* and Gerber files from *pcb*, the only real limitation being that *gschem* cannot export schematic files in a format understood by other schematic capture programs. I had no problem getting my boards fabricated.

C Programming

In keeping with my decision to use Linux tools as much as possible for the project, I looked for solutions to program the PIC chips under Linux. I had a USB PIC programmer (the PICkit 1) that I had bought a long time ago when I was still a Windows user. This model is very popular as it is quite cheap and geared towards hobbyists, so I didn't have to spend long searching the Internet to find a Linux program capable of communicating with it.

The only piece of the puzzle remaining was to find a C compiler that would run under Linux. I searched far and wide, and found some open source projects that were working on the problem, but they were either for the wrong series of PIC chip or were too immature to be useful. I eventually discovered SourceBoost, a very inexpensive Windows IDE and C compiler for the series of PIC chips I was using. [SourceBoost] When I contacted the company about which license I needed to buy to use it for an academic project, they told me that if I could get by with the code and RAM size limitations of the demo version, it was fine if I just used that. The code and RAM size limits were quite generous; the family of PICs that I'm using are entirely under the limits, so it wasn't a problem. I also discovered that this IDE and compiler will run without a problem under Wine (a Windows API layer written for Linux), so while it isn't actually a Linux tool, I was able to use it from the comfort of my home computers.

In my past programming experience, I have found it invaluable to use a version control system. Such a system is a piece of software that keeps track of all the source code used in a project and all the changes that are made to it. It can permit the recovery of deleted files and any past version of any file can be restored at any time, so if undesired changes are made, they can be rolled back. It also stores a log message with each new change that is committed, so there is a record of why the files were changed and when. As a final benefit, since the software works using a client-server model, all the code is stored centrally on a server where it can be accessed from anywhere (provided you have the password) and can be easily backed up. Due to favorable experience with it in the past, the version control solution that I selected was Subversion, an open source program available for all major operating systems. [Subversion] I already had a Subversion server set up on my home server, so it was just a matter of creating a new source code repository for the MQP. I ended up creating a folder in the repository for each one of the programs I had to write for the MQP (each PIC chip being its own program), and common aspects (such as the bus interface) were stored in common files, so changes to them on one PIC would be automatically reflected in the code for the other PICs.

Parts Ordering

The electronic components used in this MQP were largely bought from Digi-Key. A few components, most notably the flash memory, had to be ordered from Jameco because Digi-Key didn't stock the parts or was prohibitively expensive. There were at least three Digi-Key orders over the course of the project, the later ones largely for replacement parts, but also for a few things that either had been changed or forgotten. Some of the PIC chips were also procured as samples directly from Microchip. The accelerometer breakout board was bought from Sparkfun Electronics. [Sparkfun Electronics]

The printed circuit boards were ordered in two sets. The first set was the three original support modules, and was ordered from BatchPCB, a spin off of Sparkfun. [BatchPCB] They have very cheap prices compared to other companies in the industry, but their lead time can vary from 10 days to significantly longer. In this case, it was 18 days, and it put work on the project quite behind schedule. The second set was ordered at significantly greater expense from Advanced Circuits and consisted of all the remaining circuit boards except the Nichrome Controller, including the radio receiver for receiving telemetry on the ground. [Advanced Circuits]

Assembly, Testing, and Debugging

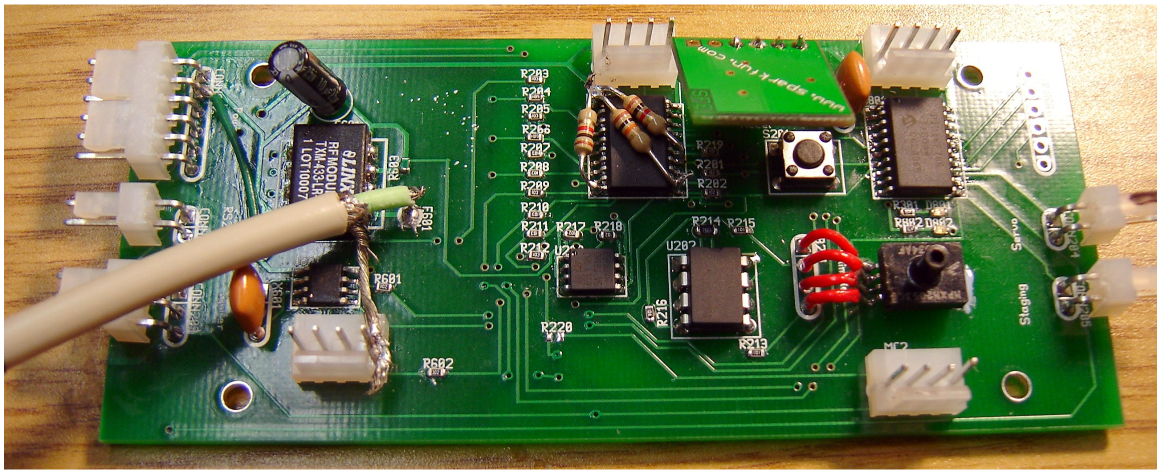


Figure 5: Fully Assembled MAS PCB

When the first order of PCBs came in (the one with the three original support modules), a terrible problem was discovered: the diameter of the holes for the connectors were too small to fit the pins through. This was an error introduced by not checking the default size of the footprint library the *pcb* software uses. I had mistakenly assumed that a footprint marked as SIP would be suitable for any reasonable SIP component. This error was corrected in the second PCB order, but for cost reasons the first set of PCBs wasn't reordered with the larger hole size. Faced with the prospect of filing down dozens of pins to fit into the holes, I instead opted to drill out the holes in the PCBs. This resulted in losing the metal plating inside the holes, which I didn't consider a major problem because the pin could still be soldered to both sides. However, it became clear quite quickly that this had been a very big mistake. Many of the holes had also lost the ring of copper around the hole, leaving little space to solder to except for the trace, which thus had to have the solder mask scraped off. These poor mechanical attachment points resulted in many of the connections being broken when the pins were stressed. Luckily, almost all of the pins that were fragile were limited to the programming connectors, which are non-critical once the PICs are programmed.

The MAS also had a brush with disaster right before one of our scheduled launch dates. By that point in time, the avionics had already been mounted in the fuselage. One of the members of the mechanical team was working inside the rocket and accidentally allowed the BNC connector on the telemetry module's antenna to touch the Nichrome Controller's power input. An unknown amount of current was shorted directly through the MAS from the 11.1V battery. The Nichrome Controller was unharmed, however, nearly all of the active components on the MAS were damaged or destroyed. Even several of the traces on the PCB were burned through (see Figure 6), and jumpers had to be installed across the burned sections to repair it. All of the PICs on the MAS and the flash memory had to be replaced. The status of the altimeter was unknown, and since it had never worked properly anyways, it was disconnected.

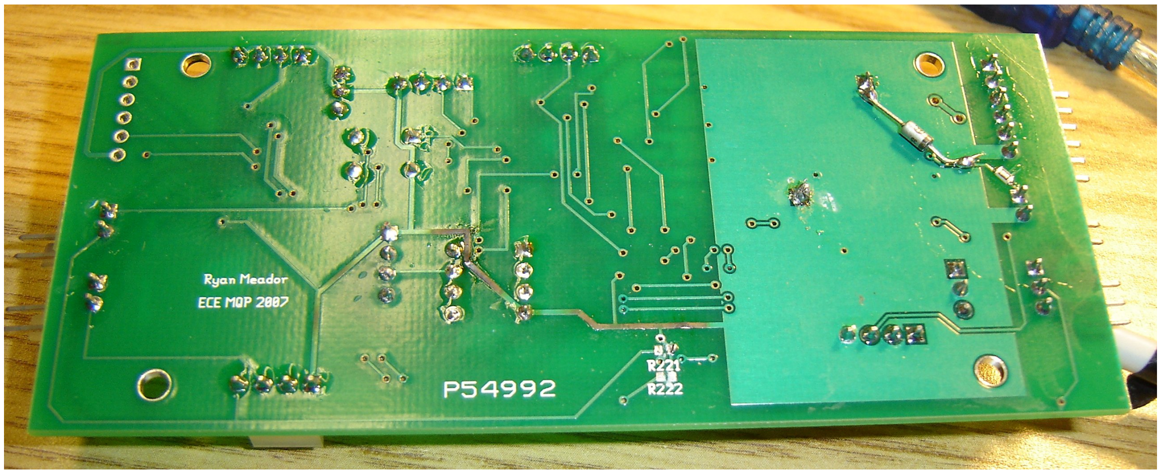


Figure 6: Burned Traces on the MAS

Igniter Controller

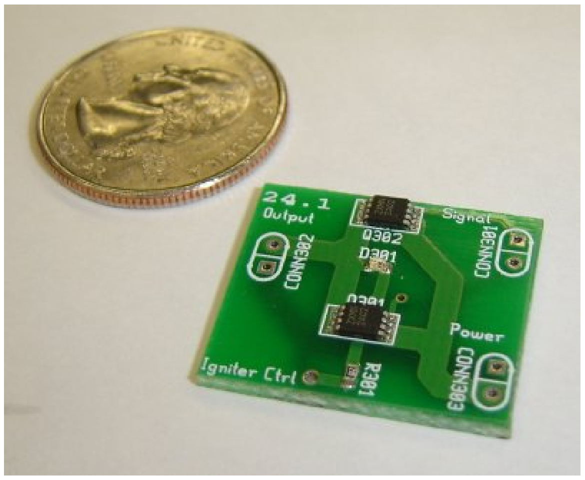


Figure 7: Igniter Controller Prior to Installation of Connectors

The igniter controller is simply a pair of FETs used to trigger core motor ignition mid-flight. After soldering the components onto this simple board, it was ready to be tested as it is the only board in the system without a PIC.

- I applied 11.1V to the power connector and the LED lit up, indicating that it was installed with the correct polarity.
- By pulling both of the signal lines to ground, the LED was extinguished. If either of the signal lines went high, the LED would come on, indicating that the igniters would be energized if either the MC2 or the MAS were to attempt to activate them.
- I checked the output voltage of the circuit and got 0.06V when it was turned off and the full 11.1V when it was turned on.
- As a test, I shorted the output to simulate an igniter being attached and ran it off the bench power supply with the current limit set at 2A. It worked exactly as designed and the FETs only got slightly warm.

Upon testing with the actual igniters, they had significantly lower resistance than the requirements I was given during the design phase, so they couldn't handle the necessary current. I was not made aware of this prior to the actual testing, and it was only brought to my attention when the Igniter Controller failed to turn off when the inputs were brought low. Apparently, the high current draw had somehow damaged the FETs. A change in the design will be necessary before the igniter controller can be used in the rocket. There is also the issue with the MC2 having inverted logic outputs, so some kind of inversion circuitry will need to be added to one of the inputs in the new version. This module is not at all critical for the initial launch because the core motor isn't going to be used for the first flight, so repairing this module is an extremely low priority.

Staging Controller

The staging controller provides a 36kHz IR beam to trigger the boosters to separate from the core. After the first assembly, two of the pins on the programming connector were not properly connected. I re-soldered the programming connector using just pins that weren't encased in nylon. The connection was sufficient to program the device several times, after which it stopped working – continuity was lost between one of the programming wires and the PIC, but upon further testing it appeared that the path was intact between the programming connector and the PIC, possibly indicating a problem with the programming cable.

- The device was verified to be functioning by testing the output with an oscilloscope – it was oscillating at the expected 36kHz. This process was impeded by having a broken scope, which made me think the PIC was broken. The peak-peak voltage of the waveform wasn't the full 0-5V that I expected, and I conjectured this was because the scope probe wasn't connecting across the output as the LED would in actual use. Upon connecting across where the LED should go directly with the probe, I got the expected 5V square wave.
- The connection between the PIC and the second set of outputs wasn't properly soldered at the pad on the PIC, so that had to be repaired after much confusion about why the output wasn't working.
- Once both sets of outputs were working at the requisite frequency, as determined by the oscilloscope, I modified the program to read the button and only output the pulse train for approximately 1sec after the button was pressed. To avoid intermittent or accidental button presses, I added a timeout on the button of approximately 1/2sec. A similar but shorter timeout was added to the signals that will come from the flight computers to prevent noise from triggering the system.
- The oscilloscope easily showed this was working by not having a pulse train until I pressed the button. Now that it could be triggered, I could move on to the final stage which was making the the second set of outputs start about 200ms after the first set. An oscilloscope probe connected to each output verified that both were still working and the delay was correct.
- I activated the brownout detect, power-up timer, and watchdog timer on the PIC, adding a call to clear the watchdog timer to the loops in the code and repeated this test. It passed, so I concluded the board was complete.

- A requirements change from the mechanical team meant I had to go back into the software and remove the 200ms delay between the two sets of boosters. This was a simple matter of commenting out the delay code.

The staging controller was integrated with the rest of the avionics systems after the accident where the MAS was short-circuited. The only modification required before it worked reliably with the rest of the system was a slight lengthening of the amount of time the outputs were active.

Servo Controller

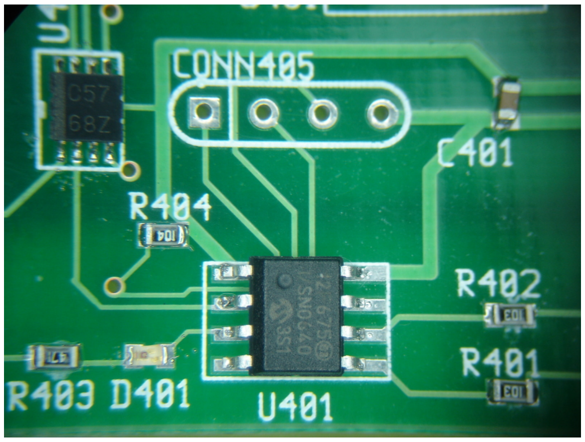


Figure 8: Closeup of Servo Controller

The Servo Controller is designed to cause the wing deployment servo to move to the specified deployed position when triggered at apogee. This board had programming connector problems similar to the staging controller. I re-soldered the connections in the same manner and achieved reliable programming.

- The first program I wrote was a very simple one to toggle the LED on and off just to prove a) the LED was installed with the correct polarity, and b) the PIC was alive. This worked perfectly, so I copied the prototype servo controller program into the PIC and modified it to work with the data selector.
- The oscilloscope showed me that the output of the PIC was as expected, but I wasn't getting anything out of the servo output. It turns out the data selector IC was suffering from two pins not being properly soldered to their respective pads. During the re-soldering of those I also redid a few other connections on the board that looked questionable or may have been broken. The circuit and existing program worked reliably after that.
- I modified the program, as a test, to tell the data selector to use the radio input and connected said input to the probe calibration connector on the scope just to see if it would pass the wave through, which it did, so I concluded the data selector was working as designed and restored the PIC's program.
- I modified the program to perform timeout operations on the button and signal lines just as the staging controller does. Upon verification with the oscilloscope that pressing the button caused the pulse width to change back and forth between the two desired servo positions, I activated the brownout detect, power-up timer and watchdog timer on the PIC and tested it once again. In the absence of a servo to test the circuit with, I had to settle for the waveform on the scope as proof that it would work.

Upon connecting to a standard servo, it worked well, but didn't work with the actual servo used in the rocket. Consulting with the tech support for the servo manufacturer, I was told the rocket's servo's data sheet contains an error and I had to alter the pulse width of the signal. That resolved the problem and I was able to calibrate the servo position. Early testing was cut short due to an apparent short of the servo that rendered it inoperable. As a last-ditch effort to find out what was wrong, I disassembled the servo and discovered the potentiometer had skipped past its stop. The pot's wiper was shorting across both ends of the resistive element. Twisting the pot back into position and re-securing the stop solved the problem. During further testing, the wings got caught during the closing of the servo and the key on the shaft was bent. The repairs conducted by the mechanical team caused the calibration to be rendered inaccurate. New calibration was carried out after the repairs to the MAS from the short circuit accident, and presently the servo controller works perfectly.

Flight Computer

The flight computer module is very simple, consisting of only a single PIC chip and a few external components. It is used to trigger the various support modules if the MC2 fails to do so. The only problem with the circuit discovered during testing was the lack of a pull-up resistor on one of the I/O pins, which the FDR also suffers from. A resistor was added and it was verified as fully functional. The software was written easily to interface with the FDR and receive the data stream, using a common set of bus interface code. Using the skeleton code that had been written prior to the assembly of the main board, I quickly implemented software that should be capable of being the backup of the MC2, but due to time constraints it was not fully tested. After working well for many days, for unknown reasons, a short developed through the PIC directly from power to ground. After desoldering the defective chip, the diagnosis was confirmed; the board itself is undamaged. A new PIC was installed and the board was fully functional again. This new PIC was destroyed during the MAS shorting incident and was again replaced. Due to the lost time from this accident, the code to perform checking and backup of the MC2 was never fully tested and the mechanical team had me reprogram it to do nothing.

Flight Data Recorder

The flight data recorder is the heart of the MAS, recording and relaying sensor data to the other modules on the board.

- The FDR was assembled and the first code that was written for it only used the bus to send data to the other PIC modules on the main board.
- The data transfer protocol was further developed and debugged once a pull-up resistor was added to the necessary bus line, similarly to what was discovered with the flight computer. When that design flaw was revealed, it also became apparent that two other pull-up resistors needed to be added, which they were. At this point, the telemetry module was not functional, so bus debugging was accomplished by verifying receipt of packets by an LED attached to the flight computer.

- The software was expanded to sample all the analog and digital channels on the board. All inputs appeared fully functional except for the barometric sensor, which due to its status as noncritical has not been debugged further.
- The code to log the data to the flash memory was written, but an off-by-one bug was found in it the day before our first scheduled launch, which I spent a couple hours debugging and fixing with a logic analyzer. The bug had manifested as all values being stored to the flash memory as two times their actual value, caused by a missing clock pulse that effectively was a bit shift. It was a simple software change to resolve.
- The 11.1V analog channel stopped working properly the night of our first scheduled launch for an unknown reason; it seems likely that a resistor on that circuit was defective in some way. There was never time to repair this because of the accident where the MAS was shorted to the 11.1V battery.

Presently, the MAS samples and records all data necessary for flight. It is ready for launch.

Telemetry

The telemetry module converts the data on the FDR's synchronous serial bus into 9600 baud 8N1 RS232, which is fed into the radio transmitter.

- After the telemetry module was fully assembled, I attempted to program it and the download failed. It didn't take me very long to realize that the problem was that the entire telemetry module ran on 3V (by design) and my programming device can only program at 5V. I modified the circuit so that the PIC runs on 5V and uses as step-down circuit to make the data line to the radio be 3V.
- The revised circuit worked perfectly and software development went forward swiftly. I wrote a quick algorithm using the timer interrupt to send data in RS232 format at 9600 baud and tested it by sending a few known characters.
- Once that worked, I added the same data receiving code as in the flight computer and had it send that data back out via RS232. During testing, no problems were observed and it functions 100% as intended.
- Range testing was conducted in Elm Park and a line of sight distance of over 1200' was within acceptable error limits. It is believed that the radio link will function over longer distances, but we were unable to find a longer distance that was line of sight to test it.

Nichrome Controller

The nichrome controller was a late addition to the rocket with the purpose of releasing the wing's reflex mechanism. The signal is driven off the servo controller's signal line. I drew up the circuit quickly and handed the design off to one of the volunteers on the project, who had a personal CNC machine. He redrew the PCB layout and fabricated it with the CNC machine by milling a copper clad board.

- The resistance of the nichrome wire was found to be so small in comparison to the resistance of the copper wire driving it that the length of it had to be increased and the diameter decreased to guarantee melting.
- During testing, a logic error was discovered that caused the PIC to interpret the “don't fire” state as the fire command, but this was resolved with a minuscule change to the code. The board appears to work as designed.

Conclusion

When I first considered taking on this project as my MQP, the vague notion of the project was that it would be only the parts that eventually became the Flight Data Recorder and Flight Computer, quite possibly rolled into a single PIC. I was worried this wouldn't present enough challenge or workload to qualify as an MQP. I couldn't have been more wrong. By putting the mechanical team in the position of customers and myself as the engineer responsible for meeting their requirements, we were all provided with an MQP that was at times difficult, challenging, and a reasonable approximation of a real-life situation in industry.

The dialog with the mechanical team resulted not just in constantly-changing requirements, but new friends. There were a number of moments on both sides that are best captured by “wait, you can do that?” -- I learned a lot about mechanical and aerospace engineering from them and I believe they learned a few things about electrical and computer engineering from me. The back-and-forth nature of discovering and refining requirements caused a lot of “feature creep” that made my MQP far more challenging and interesting than my original concept of it. The design phase stretched for nearly two full terms and changes were still made once the project entered the assembly and testing phase. Each individual module isn't terribly complicated, but each of them required me to go out and learn new things to produce an effective design. In the end, the sprawl of requirements caused me to learn far more over the course of this project than I have in any other collection of courses at WPI over a similar time period.

As of this writing, the rocket has never launched. Everyone involved with WARRIORS II is saddened by this fact. Due to the arrangement with WPI and requirements from the FAA, the rocket can only be launched at sanctioned club launch locations and dates. The first two of these were canceled due to weather and the third was scrubbed due to problems with the Command Receiver (the backup parachute deployment system, which isn't just a good idea for safely testing the rocket, but is mandated by the rocket clubs we would be launching with). The fourth launch date was also canceled due to weather, but we probably wouldn't have been able to fly anyways due to more problems with the Command Receiver, problems we though we had resolved after the first scrub. All the tests that we can perform on the bench indicate that the rocket is ready to fly and I expect that if we ever do get the opportunity to launch it, it will work perfectly.

Acknowledgments

This project would not have been possible without:

- The members of the mechanical team, Nicholas Behlman, Ryan Caron, Sara Dempsey, and Rob Niles, without whom the rocket would never have taken shape.

- Professor Susan Jarvis, who advised this MQP and provided significant motivation in coping with the long design cycle and short fabrication cycle, as well as being a sounding board for ideas.
- The AIAA volunteers who were indispensable when it came to fabricating some components of both the mechanical and electrical designs.
- Professor John Blandino, advisor to the mechanical/aerospace MQP.
- Mr. Tom Angelotti and Mr. Pat Morrison from the ECE shop who helped immensely by teaching me surface mount soldering, repair techniques, and supplying badly needed parts on short notice.

Appendix A: Project Proposal

Abstract

This MQP will consist of designing and building the avionics systems for the second-generation WARRIORS rocket. The avionics are broken down into five subsystems: flight data recorder, flight computer, telemetry, power supply, and booster separation. The flight data recorder will possess a full sensor suite including altitude, acceleration, and gyroscopic sensors and the ability to record the data from those sensors and other inputs over the duration of the flight. The flight computer will orchestrate all of the rocket's functions after launch, primarily booster separation, second stage ignition, and deployment of the glider recovery mechanism. Telemetry will relay real-time data from the flight data recorder to a receiver on the ground. The power supply will have to be designed to meet high voltage and current requirements as well as being light-weight. An evolution of the first-generation WARRIORS booster separation system will be used to detach the boosters from the rocket with electromagnetic repulsion.

WARRIORS I Overview

The WARRIORS I rocket was constructed and launched during the 2005-2006 school year. It used an airframe with a diameter of 2.63" and an overall length of about 7'. The design featured a fairly unique staging system. In a traditional system, the rearmost section of the rocket detaches when the motors are exhausted; WARRIORS I sported three boosters instead. It also had one larger motor in the core that was ignited after booster separation. The fuselage was composed of a phenolic tube, broken into many segments with bulkheads that attached to each other. This provided some degree of modularity in the design, but wires from the various subsystems had to converge on the avionics bay, which contained an on-board flight computer. Many problems were encountered running wires between compartments because of the need for connectors that could be unplugged.

One of the design goals of this rocket was to avoid the use of pyrotechnic charges. As a result, the jettisoning of the boosters was accomplished by electromagnets. A hinge at the tail bore the thrust load of the booster while the electromagnet held it against the fuselage. When the electromagnet reversed to repulse the booster, it pivoted away from the fuselage until the hinge disengaged and separated. The large repulsive force requirement entailed the use of a high-capacitance, high-voltage capacitor power supply and triggering circuitry duplicated for each booster. This circuitry remained inside the core of the rocket even after the boosters were jettisoned.

The flight computer was a commercial model, the MC2 from G-Wiz Partners. It controlled all events that occurred after liftoff. It consisted of a sensor suite, a processor, and four outputs designed to drive pyrotechnic devices, since that is the norm for staging and recovery deployment in the model rocket world. These four outputs are "user programmable" in the sense that they can be programmed to trigger a specified amount of time after launch, burnout, apogee or low altitude being detected. The barometric sensor and accelerometer are read at 33Hz and can store up to 18 minutes worth of data. There is an expansion header on the MC2, but G-Wiz hasn't released a telemetry module. However, the data can be downloaded via USB when the rocket is recovered. [G-Wiz MC2]

WARRIORS I had several fatal problems, most notably (and finally) the failure of the second stage to ignite that resulted in the loss of the rocket. It would have been much easier to diagnose the failure(s) had the flight data survived the crash, but the flight computer's data was unrecoverable even by the engineers at G-Wiz. It is believed that the failure to ignite the second stage was caused by insufficient voltage in the battery powering the pyrotechnics. Combined with a failure of the mechanical springs to deploy the parachute, the avionics were completely ineffective after booster separation. Because the flight was so much shorter than expected, the backup computer (actually just a simple timer) didn't fire the backup parachute (via pyrotechnics) before impact with the ground. The impact broke the rocket into many pieces. Many of the breaks occurred at the bulkheads. The avionics bay, being in the middle of the rocket, was one of the most intact modules, but even so, the data on board the flight computer was lost. The computer itself was repaired by G-Wiz and may see flight again aboard WARRIORS II.

WARRIORS II Overview

The second generation of the WARRIORS project hopes to combine innovative new designs with improving the reliability of the old system. The volume of the rocket is close to that of WARRIORS I. An important difference is the body diameter is being increased to 4” and the body length is slightly shorter. It will have an internal frame composed of carbon fiber rods that will take the majority of the stress on the structure, whereas the older model had only the phenolic tube as structure. There will be bulkheads internally once again, but instead of using modules, the entire frame will be removed from the skin as a unit, allowing much more access to the internal systems as well as facilitating the routing of wires.

The new rocket has four booster rockets, with each one being taller to accommodate the electronics for separation that were formerly in the core. This will allow the jettisoning of more weight when the boosters separate. The boosters are also grouped asymmetrically in pairs (WARRIORS I was radially symmetric) pattern due to the unique recovery mechanism. This will likely require that the boosters not all separate at once to prevent them from becoming entangled in one another. If they were all to separate at once, when they pivot back on the hinge, due to the asymmetric formation, their tails could hit each other. Introducing a delay of only a few hundred milliseconds should be sufficient to prevent this from being an issue.

The unique recovery mechanism is wings. The rocket will not fall back to earth, but will instead glide much like an airplane. Current plans call for the wing geometry to be adjustable in-flight, giving control over the flight path. The wings will deploy via the same mechanism as the adjustable wing geometry – a servo will swing the two leading edges (actually part of the rocket skin, and the primary reason for the internal frame) away from the body, stretching some kind of fabric out to form a wing. Initial deployment will be under computer control at apogee, but subsequent control will either be handed off to a human controller or, in more ambitious designs, performed by the on-board computer. If the team opts for the latter, the flight computer will probably only be responsible for maintaining a slow downward spiral to prevent the rocket from traveling too far away from the launch site.

Avionics Requirements

Obviously, such an ambitious mechanical design will require ambitious electronics to control it. These electrical and electronic systems have been tentatively divided into five groups: flight data recorder, flight computer, telemetry, power supply, and booster separation. Only the first four of them are the subject of this MQP. It is also likely that some non-flying components such as battery chargers and telemetry receivers will be required. The overall design should be robust enough that the failure of any one component won't jeopardize the rocket. During the design, size and weight will be considerations; while no maximum size has been specified, a significant effort to make the system as small and light as possible will be made. Also, as this is a rocket, G-forces must be considered as well. The expected take-off acceleration of WARRIORS II is around 15G's, so all the avionics must be able to withstand that acceleration. The deceleration of a controlled landing is unknown at this time pending a more detailed design and test of the recovery mechanism. An uncontrolled landing (a crash) will likely encounter many hundreds or thousands of Gs.

The requirements for the flight data recorder from the mechanical team are that it perform at least at the level of the MC2 in terms of what data is recorded (altitude and acceleration) and how fast it is recorded (33Hz). It should be a “black box” in the sense that the data contained in it will be able to survive a catastrophic failure of the vehicle. There are two ways this could be accomplished: by the physical device surviving, which is preferred, or by transmitting the data to the ground via the telemetry module before being destroyed.

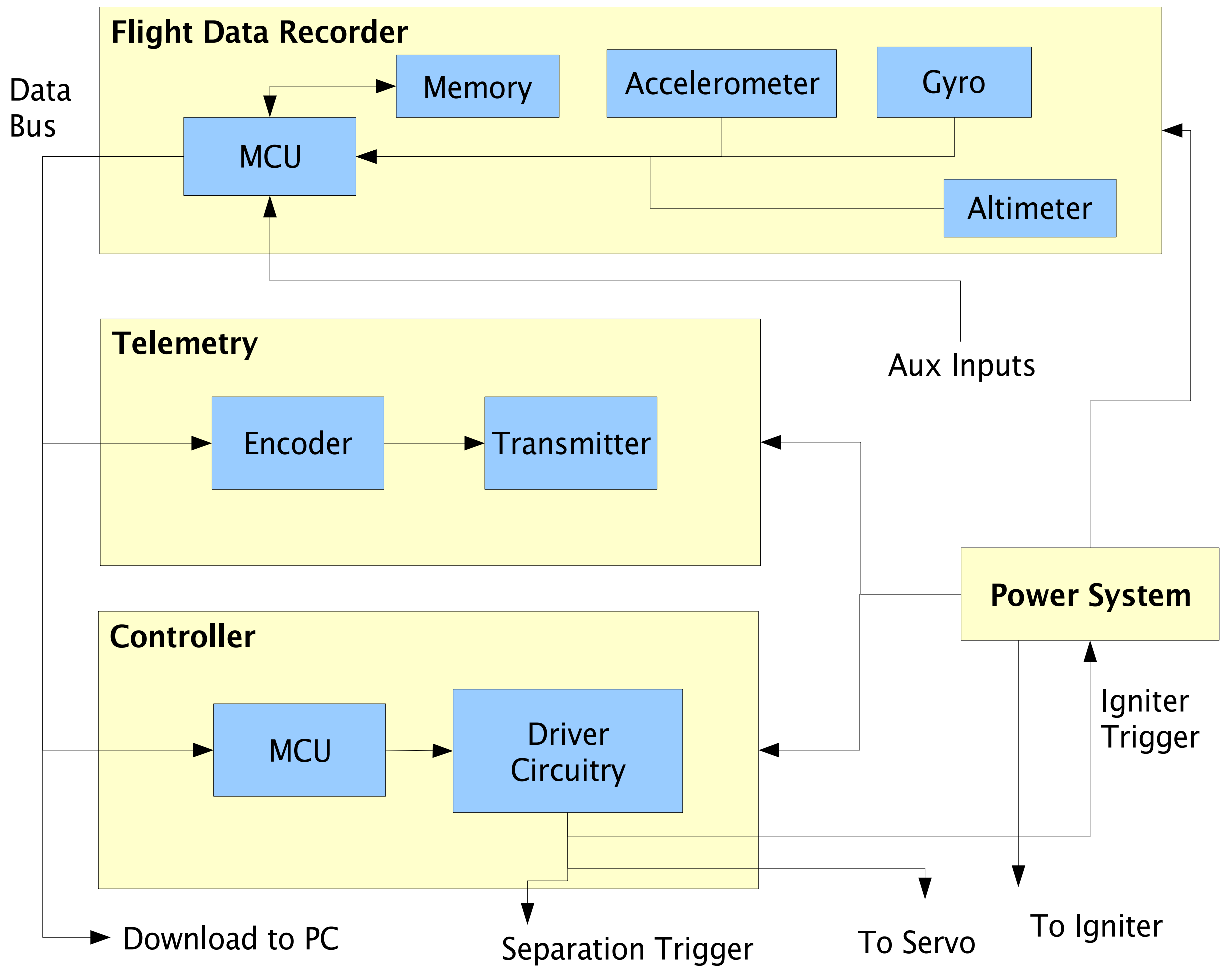
The flight computer should be a drop-in replacement for the MC2, but also capable of delayed separation of the four boosters as well as operation of the wing deployment mechanism. These latter two requirements will likely be stand-alone modules that can also be triggered by the MC2, should the rocket wish to fly with that controller. The MC2 is intended as the main computer and the newly designed flight computer as a backup, at least until the new system has proven itself in flight. This arrangement has become complicated because the rocket's control requirements have surpassed that of what the MC2 can provide.

Requirements for the telemetry module are straightforward: transmit to the ground in real time all the data recorded by the data recorder. The primary motivation for this capability is to safeguard against the destruction of the data recorder by keeping a second copy of the data outside the rocket. It could also be used by a human pilot for the descent/glider phase of the flight. Being able to know the angle of attack even when visibility of the rocket is poor could assist the operator in landing safely. The range of the radio must exceed the maximum altitude of the rocket and also the distance the glider will fly away from the launch site. The minimum altitude for a launch deemed successful is 1500'. The maximum altitude is still unknown, pending a more precise mass estimate and motor selection for the rocket, but it is not likely to exceed 3000'. The distance the rocket will travel from the launch pad depends on the glider wing configuration and flight path. A spiral flight path will be preferred to minimize the distance the rocket travels from the launch pad.

Proposed Design

The design consists of three main components, a “black box” flight data recorder (FDR), a flight computer, and a telemetry module. The FDR will have to have a suite of sensors to assess the status of the rocket. The other two modules will both require the same data as recorded by the FDR or a subset thereof. For weight and cost reasons, all three modules will use data from a single sensor suite. Sharing the data, the only dependencies between these modules will be that the FDR must be functioning for the telemetry to function. The flight computer will also use the sensor data for timing decisions for various activities. The process will be fully programmable, unlike the MC2. If the FDR is either not present or fails during flight, the flight computer would resort to an internal timer for basing events, much like the backup system from last year's rocket. The lone flight computer module could easily fulfill the backup role if it (or any other module) is not wanted as the primary flight computer.

High-Level Block Diagram



Flight Data Recorder

The accelerometer and gyro sensors for the black box are very likely to be some kind of MEMS device. MEMS devices are extremely small and light-weight, while also being accurate and reasonably priced. The leading accelerometer candidate is the ADXL321 produced by Analog Devices. It measures ±18g on two axes and can withstand impacts of up to 10,000g. [ADXL321 Data Sheet] A possible gyro is the ADXRS300, also from Analog Devices. It can measure angular rates up to 300°/sec. [ADXRS300 Data Sheet]

Possible memory solutions include serial and parallel FLASH memories and EEPROMs, as well as the potential to use a removable SD card. A synchronous serial FLASH or EEPROM that supports continuous writing (begin writing at the beginning and it will continue to fill up memory addresses sequentially without the need to give it a change-address command) would probably be the best choice because it is very easy to use and the same bus that transports the data to the memory could be used to send the data to the other modules in the avionics system. An SD card is also attractive in that a variant of the I^2C protocol is used to communicate with it, which could also be used to communicate with the other modules. An SD card is also more economical for large memory sizes. As a very conservative estimate, recording 12 channels of 16-bit data at 33Hz for half an hour only requires 1.5MB, which probably doesn't put it into the realm of an SD card being more economical. The final decision will likely rest on what is available for a fair price from Digi-Key or another major supplier as well as settling concerns that a removable memory module might come unseated during flight.

The altitude sensor could be a Motorola MPX2100 or similar chip that retails for around $12 at Digi-Key. This particular chip has altitude sensing as one if its intended purposes. It also has a port for connecting a pipe that will extend outside the fuselage of the rocket to get a more accurate pressure reading during flight, which is important since engine exhaust and other factors may cause the rocket's internal pressure to be different than that outside. [MPX2100 Data Sheet]

Due to ease-of-use considerations, cost, and a wide array of on-board peripherals (such as A/D conversion), a PIC microcontroller seems like the best choice for the processor that orchestrates the sensor polling and data storage, as well as driving a bus that sends the data to the flight computer and telemetry modules. WPI has both facilities for programming various models of PICs as well as classes teaching their use. In addition to these places to turn for assistance if something goes wrong, extensive personal experience with these chips makes the decision virtually certain. The PIC16F785 in particular is attractive because it is the newest and most full-featured of the PIC16 line, which is a line of mid-range MCUs with moderate (~18) pin counts. It has 12 channels of 10-bit A/D, thus removing the need for a separate A/D chip. It also possess three timers which would be useful for polling the sensors regularly and also has an internal oscillator to reduce the overall component count and size of the PCB. Other PICs in this family may be equally good choices, though none have as much RAM as this one. [PIC16F785 Data Sheet]

Flight Computer

The flight computer (or flight controller) does not need to adhere to the MC2's standard of providing high-current outputs as it won't be directly driving pyrotechnics. All the output signals will be either logic (booster separation, 2nd stage ignition) or PWM waveforms (servo control for wing deployment), which nearly any modern MCU can provide. The MCU will also need to read the data being sent by the flight data recorder. The total number of I/O pins required is likely to be quite low; to save weight and space, the use of an MCU smaller than the candidate for the flight data recorder may be desired. A more appropriate MCU might be the PIC16F684, which is of the same family as that proposed for the FDR, but has only 14 pins. To protect the MCU from over-current or reverse-voltage damage, buffer or driver circuitry will likely be needed on the outputs.

The software of the flight computer is going to be the most complex component of this module. The program must interpret the sensor data from the FDR and react to certain events. The system should arm itself when it notices launch by reading the accelerometer. When detecting booster burnout (the acceleration should drop), it will initiate booster separation (with the delay being provided either internally or by an external circuit). A short time after booster separation, it will ignite the core motor. Apogee can be determined with the altimeter or accelerometer, or a combination of the two. At apogee, the wings must be deployed. Throughout all these states, the system must maintain internal timing information. If the data from the FDR is cut off during flight or indicates a problem (the second stage not igniting, for instance), the timer can be used to base subsequent events and hopefully will permit some degree of failure recovery.

Telemetry

An early investigation of components has shown that it should be possible to make a telemetry unit with a range of 3000+ ft for a parts cost of $40 or less and minimal size and weight. The specific solution is the TXLC-434-LR transmitter module from Reynolds Electronics, which measures only 5/8" x 11/16". It is capable of data rates up to 10Kbps at maximum range. [Reynolds Electronics] Other transmitter/receiver combinations can still be investigated if a longer range is desired, however it seems unlikely that a cheaper setup will be discovered considering that many individual modules (either a transmitter or receiver, not a pair) discovered during research were on the order of $30. The TXLC-434-LR uses CPCA modulation (carrier-present, carrier absent) to send data, so it is innately a digital device and well-suited to sending serial data.

If a synchronous serial data bus is used in the FDR, that data will have to be converted to asynchronous serial before being broadcast via the radio. This conversion may be possible using a shift register or other form of queue to form packets from the synchronous source and then pass the packet to the radio as a unit based on a clock signal. If, upon detailed design, this collection of logic circuitry proves to be too complex or too large (due to many components) to be practical, a PIC MCU could likely fill the role, though special care would have to be taken to ensure its internal clock is accurate.

Power Supply

The MC2 requires 12V for operation, whereas all the systems outlined in this proposal will operate on 5V. The MC2 isn't the only subsystem needing 12V, however. The igniters also need 12V at 4A for at least 2sec. At full draw, the servo is likely to require 2A at 6V. These power requirements are quite extreme, especially considering the weight, size and cost constraints of the project. It is impractical to regulate the 12V source down to 6V for the servo's use, especially if it is drawing 2A. This necessitates a 6V source for the servo. To save as much weight as possible, it makes sense to use two 6V battery packs connected in series to power the igniters and other 12V devices. Only a single battery pack would be connected to the devices requiring 6V. The draws on the packs would be uneven, but this scheme minimizes power loss due to voltage regulation while keeping mass to a minimum with fewer batteries. Whatever the final design of the power system turns out to be, it is almost certain that it will require several high-capacity capacitors to help cope with surges, especially from the igniters. The individual modules of the avionics system will probably also want capacitors to help insulate them from fluctuations in the main power source.

Lithium Polymer batteries are the best choice from a weight standpoint. They have a very high energy density. Also, by being entirely solid-state, they are more able to resist the acceleration of the rocket during launch. Unfortunately, these are also some of the most expensive batteries on the market. The only source discovered thus far for Li-Poly batteries that seems reputable and also has the necessary selection is Ultralife Batteries. They seem to be an industry leader in small and light weight batteries with high power ratings. The cost of their products isn't listed on their website; they do mostly OEM work. It is known that a large (16V, 10Ah) Li-Ion battery they sell is about $320, but it is not known if that scales at all to the smaller batteries. Using Ultralife Li-Poly batteries, it may be possible to get 12V for as little weight as 125g. [Ultralife Batteries]

Nickel-Metal Hydride batteries are another viable alternative, but they are heavier than either Li-Poly or Li-Ion. Digi-Key can make custom packs for very reasonable prices (~$20), but they weigh several hundred grams. [Digi-Key Catalog] Tower Hobbies can supply battery packs designed for use in model airplanes, so they are significantly lighter. However, the voltages don't match well with out requirements. A solution of this type would probably cost around $120 and weigh about 160g. [Tower Hobbies]

Testing

Unit Tests

These tests are stepping-stones to mark progress in the construction of the individual modules.

Primarily, they represent verification of the smallest blocks in the block diagram.

- Flight Data Recorder

  1. Microcontroller operation can be verified by attaching an LED to one of the output pins and writing a simple program to blink it. To verify proper timer setup, the LED can be blinked at timed intervals, and if it looks approximately correct, it can be verified exactly with an oscilloscope. The analog to digital converter can be tested by writing a program that reads it and turns on and off two different LEDs at different sensed voltage levels, then feeding the system a varying voltage and watching the LEDs. More advanced debugging can be accomplished by attaching an LCD panel (if necessary) or using WPI's in-circuit debugging equipment (if it is compatible with the model of microcontroller chosen). Once the data bus is functional, reading the data from that source will probably be the easiest way to debug. When moving on to other modules of the avionics, if the proper operation of the FDR is assured, the data bus can also be a debugging tool for them by knowing what is going into the other modules.

  2. Operation of the accelerometer and gyro can be tested by looking at the outputs with an oscilloscope while moving the circuit board. If an exact calibration of the output is required, WPI possesses the facilities to test these MEMS devices with precise accelerations or rotations. The Mechanical Engineering department has machines that can be set to move in a very carefully controlled way. By attaching the gyro or accelerometer to these machines, it is possible to watch the output under controlled circumstances and thus perform detailed calibration.

  3. Depending on the type of memory used, a logic analyzer may be suitable for testing the memory functionality. Alternatively, the MCU could write and then read back some data, either passing it to a human for verification or verifying it internally and lighting an LED to indicate status. If the storage medium is an SD card, it may also be able to have the MCU write data to the card and then read it with a PC.

  4. It seems likely that the only way to test the altimeter is to produce a partial vacuum around the circuit board while watching the output with a voltmeter. The WARRIORS lab has the equipment to do this. If the sensor is sufficiently sensitive, it might also be possible to get a change in the output by walking up and down in one of the taller buildings on campus.

  5. The data bus output can be tested with a logic analyzer.

- Flight Computer

  1. Microcontroller function can be verified in a manner similar to that listed under Flight Data Recorder above.

  2. Software function can be verified by feeding the flight computer artificial or pre-recorded

data through the bus and checking the outputs. The computer should be presented with various scenarios of data and the reactions observed and compared to the expected functioning of the system. Once it has been demonstrated to work reliably, the failsafe functionality can be checked by starting similar tests, but disconnecting the bus in the middle of “flight” to simulate a failure of the FDR.

- Telemetry
    1. Nearly all radios suitable for the telemetry purpose, including the leading candidate parts use CPCA (carrier present, carrier absent) modulation to carry the data. To test the transmitter/receiver pair, the output of the receiver can be looked at with an oscilloscope or logic analyzer while toggling the transmitter.
    2. Sending known serial data over the radio link at a slow baud would be the next step, reading it first with a computer or logic analyzer, and then when that works, a MCU. Advancing upwards in baud rate to the rate desired in the rocket, the system will be in a position to perform range tests. The transmitter and receiver should be separated with increasing distance until the error rate at the receiver is unacceptably high to determine the maximum range.
- Power System
    1. Verify booster separation is possible with the triggering circuitry.
    2. Verify second-stage igniter performs as required.
    3. Perform stress tests consisting of drawing a conservatively-estimated load from the power system and measuring battery life.

Flight-worthiness Tests

Tests in this section are intended to demonstrate the successful integration of the different high-level modules of the avionics system. They are intended to be performed more or less in the order below. When the final test is completed successfully (perhaps more than once), the system should be deemed flight worthy.

1. Verify reception with acceptable error rates of the actual telemetry radio signal at a range in excess of 1500'. Document actual maximum reliable range, which hopefully will be the same as that found in the unit test for the telemetry module. To verify correct data, the received data can be compared against the data stored in the FDR.
2. Verify booster separation by triggering with flight computer. The computer can be fed false data much as in the unit test and should successfully cause the boosters to separate at the simulated booster burnout portion of the flight.

3. Verify second-stage ignition when triggered by flight computer in a manner similar to the booster separation test.
4. Verify control of the wing deployment servo as described for booster separation and second-stage ignition. At some point in the testing regimen, it will be desirable to perform these three tests simultaneously.
5. Simulate a flight with only the flight data recorder by throwing the module up into the air and/or dropping it off a building, then verify the stored data is as expected. Exact measurements probably won't be possible for comparison, but simple calculations and estimations should suffice to show that the stored data is reasonable.
6. Install the complete avionics system in the rocket mock up and throw it off something tall to simulate an entire flight. If all the rocket's systems are activated during the appropriate phase of flight and the FDR and telemetry data match reasonable expectations (and each other), declare it a success.

Budget

Below is a table of expected budget items and estimated costs. If the projected cost is different from the extension, it means that there is a way to procure the indicated part(s) at below-market price.

Item	*Quantity*	*Cost*	*Extension*	*Projected Cost*
PIC MCU	2	$6.00	$12.00	$0.00
Accelerometer	1	$8.00	$8.00	$0.00
Altimeter	1	$12.00	$12.00	$12.00
Gyro	1	$50.00	$50.00	$0.00
Radio (TX/RX pair)	1	$40.00	$40.00	$40.00
PCB(s)	1	$60.00	$60.00	$60.00
Misc.	1	$10.00	$10.00	$10.00
TOTAL				$122.00

Appendix B: Support Module Pinouts

The pins on all connectors are counted from the one with the square solder pad (also indicated by being sectioned off from the others with a line on the silkscreen). All of the connectors are labeled on the silkscreen with "CONN###", and many are also labeled with a more friendly description.

Igniter Controller

The igniter controller accepts signals from both the MC2 and MAS and energizes the core motor igniter if either signal is present. The LED on the board will light up when the igniter port is energized.

Connector	***Pin***	***Purpose***	***Voltage/Current***	***Notes***
CONN301 *Signal*	1	MAS signal	0 - 12V, high-impedance	Active high
	2	MC2 signal	0 - 12V, high-impedance	Active high
CONN302 *Igniter*	1	Igniter Output	11.1V, 4A	Igniter should be connected across these two pins
	2	Ground		
CONN303 *Power*	1	Power	11.1V	
	2	Ground	0V	

Servo Controller

The servo controller accepts logic signals from the MC2 and MAS to move the wing servo from open to closed. It can also switch a data line selector to permit an external hobby radio receiver to control the servo once the wing has opened. There is a button on the board which is used to toggle the wing position from open to closed for reset or testing purposes.

Connector	***Pin***	***Purpose***	***Voltage/Current***	***Notes***
CONN401 *Servo*	1	Ground	0V	
	2	Servo Power Out	7.4V	
	3	Servo Signal Out	0 - 5V	1-2ms pulse approx. every 20ms

Connector	***Pin***	***Purpose***	***Voltage/Current***	***Notes***
CONN402 *Power*	1	Logic Power In	5V	
	2	Servo Power In	7.4V	
	3	Ground	0V	
CONN403 *Radio*	1	Ground	0V	
	2	Radio Power Out	7.4V	
	3	Radio Signal In	-0.5 - 5.5V	Beware that most radios probably output a voltage close to their supply, which in this case is 7.4V – too high!
CONN404 *Signal*	1	MAS Signal	0 – 12V, high impedance	Active high
	2	MC2 Signal	0 – 12V, high impedance	Active low
CONN405 *Programming*	1	V_{pp}		
	2	DAT		
	3	CLK		
	4	Ground		

Staging Controller

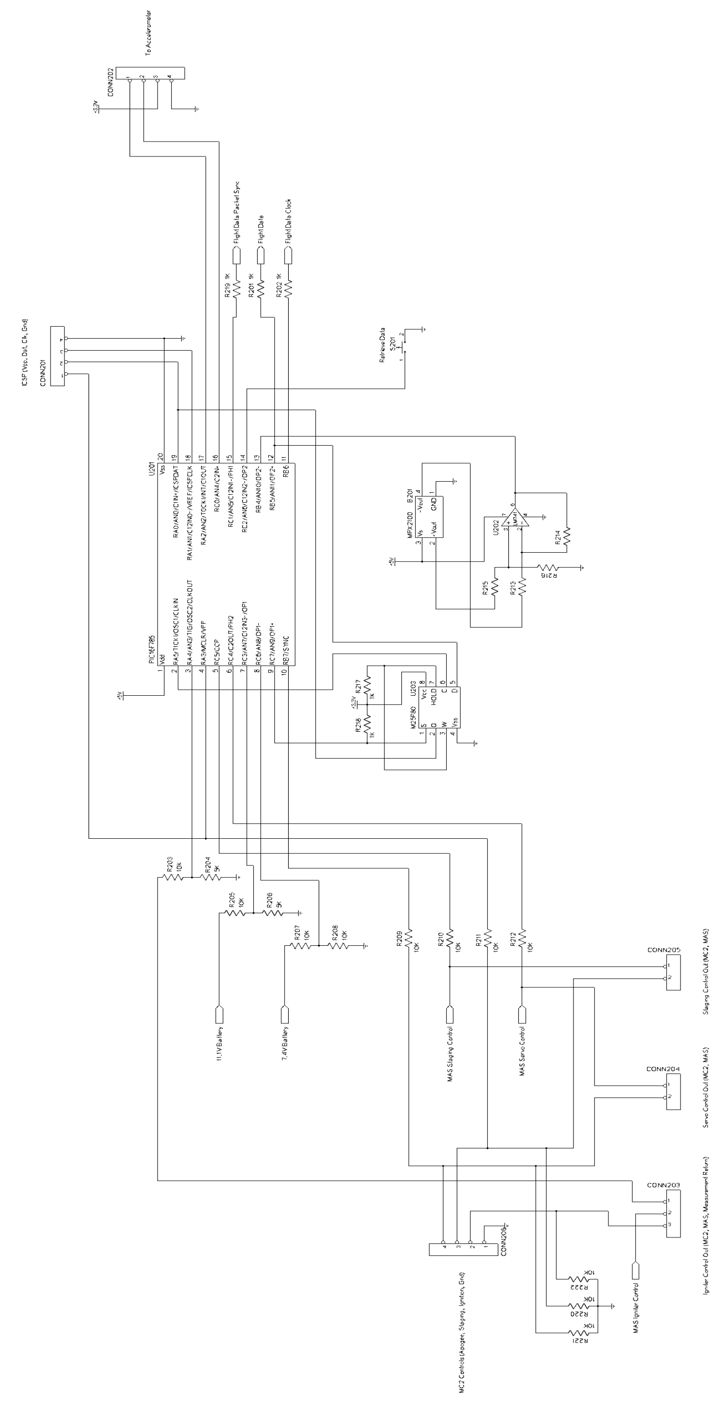
The staging controller provides two sets of two outputs each for the IR LEDs that trigger the boosters to separate. With the present software, both sets activate at the same time. The outputs stay on for a total of about 1.5sec before the module resets. A button is provided to manually trigger the outputs for testing or demonstration purposes.

Connector	***Pin***	***Purpose***	***Voltage/Current***	***Notes***
CONN501 *Power*	1	Power In	5V	
	2	Ground	0V	
CONN502 *Programming*	1	V_{pp}		
	2	DAT		

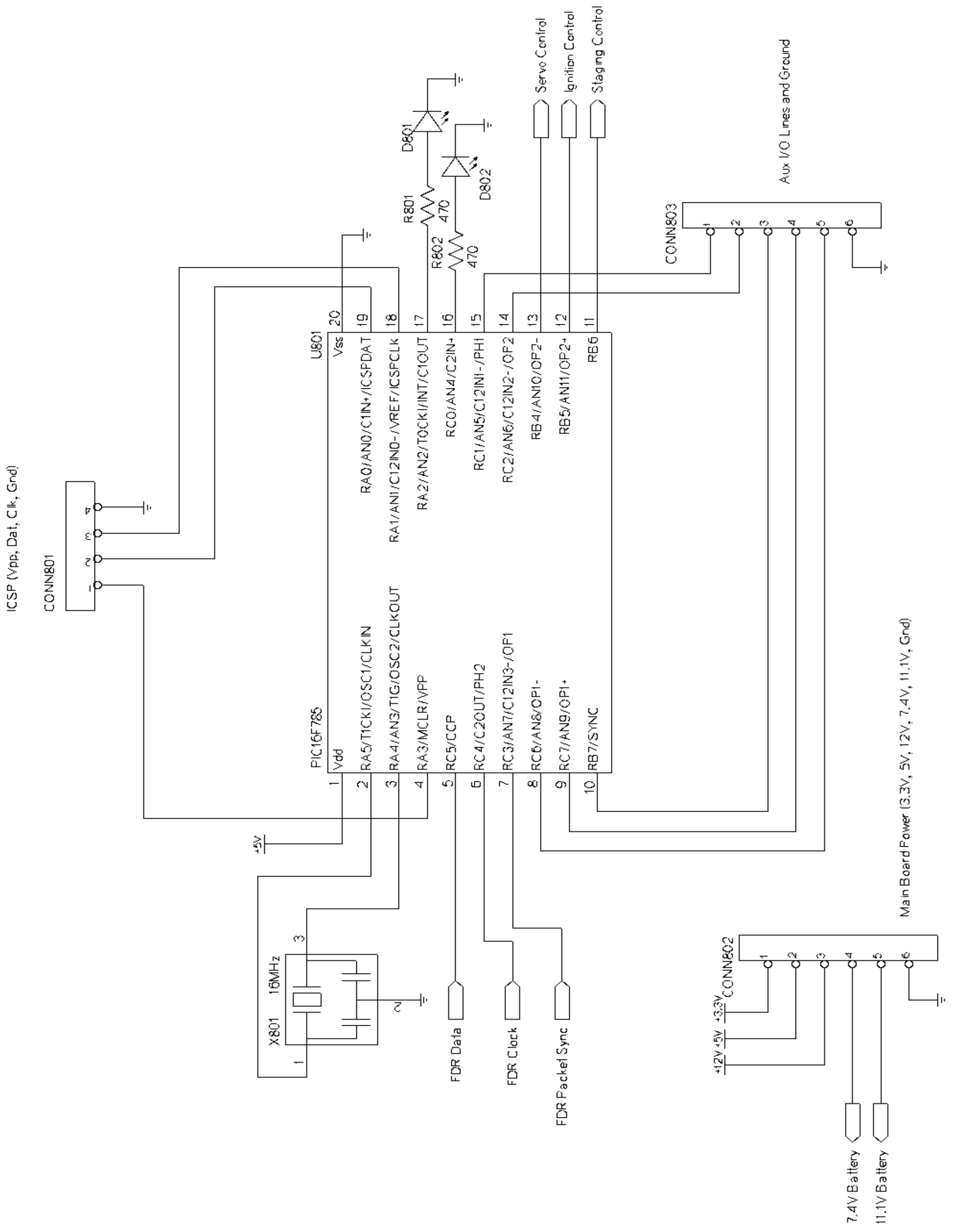
Connector	***Pin***	***Purpose***	***Voltage/Current***	***Notes***
CONN503 *Signal*	1	MAS Signal	0 – 12V, high impedance	Active high
	2	MC2 Signal	0 – 12V, high impedance	Active low
CONN504 *LED 1A*	1	LED Output	5V, 100mA	36kHz square wave, LED should be connected across these two pins
	2	Ground		
CONN505 *LED 1B*	1	LED Output	5V, 100mA	36kHz square wave, LED should be connected across these two pins
	2	Ground		
CONN506 *LED 2A*	1	LED Output	5V, 100mA	36kHz square wave, LED should be connected across these two pins
	2	Ground		
CONN507 *LED 2B*	1	LED Output	5V, 100mA	36kHz square wave, LED should be connected across these two pins
	2	Ground		

Appendix C: Schematics

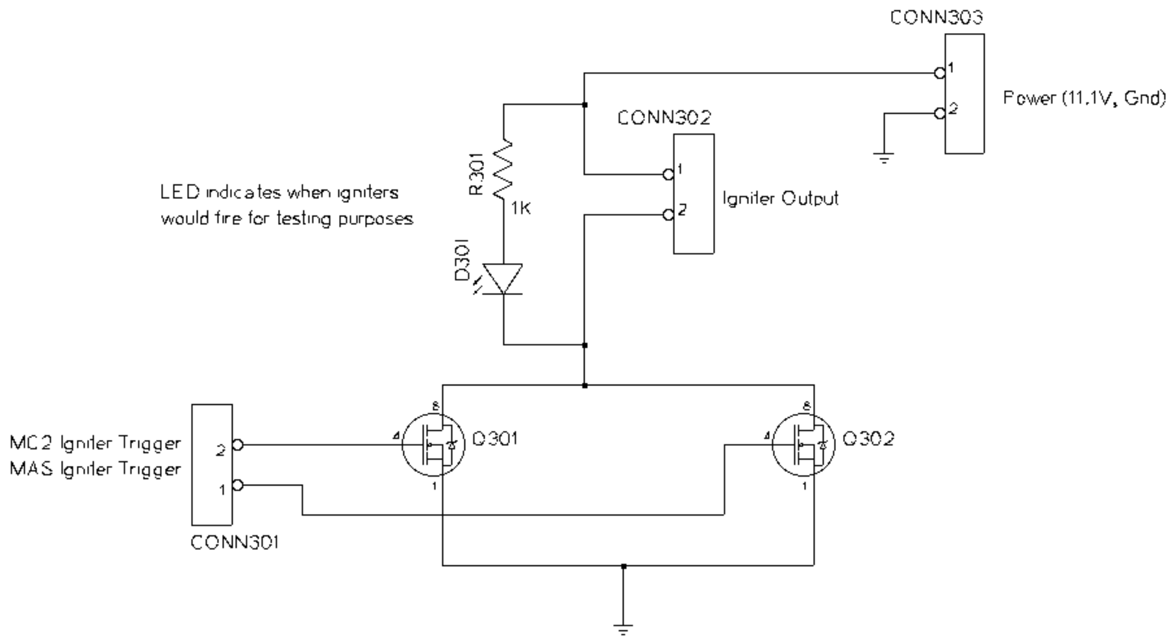
Flight Data Recorder



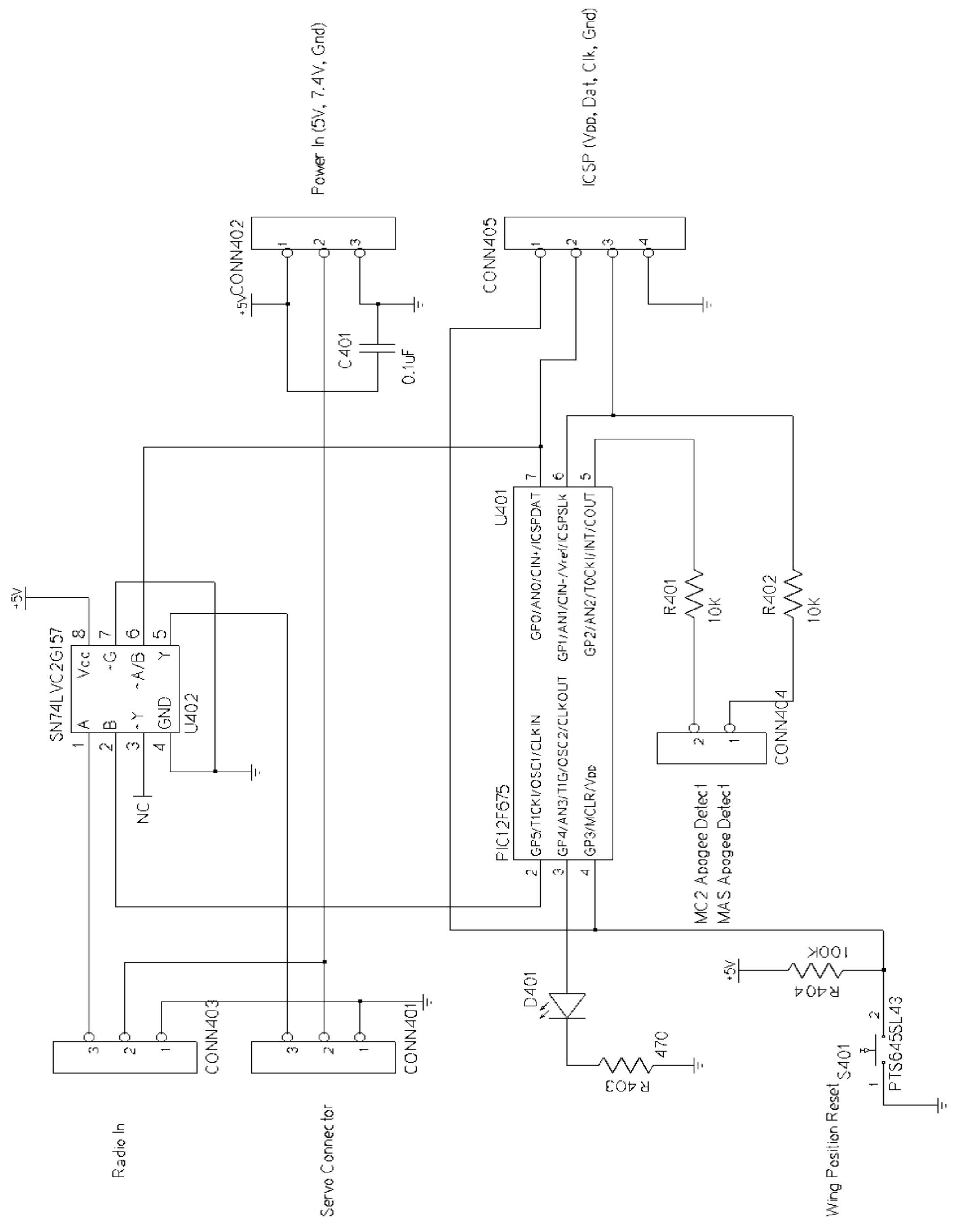
Flight Computer



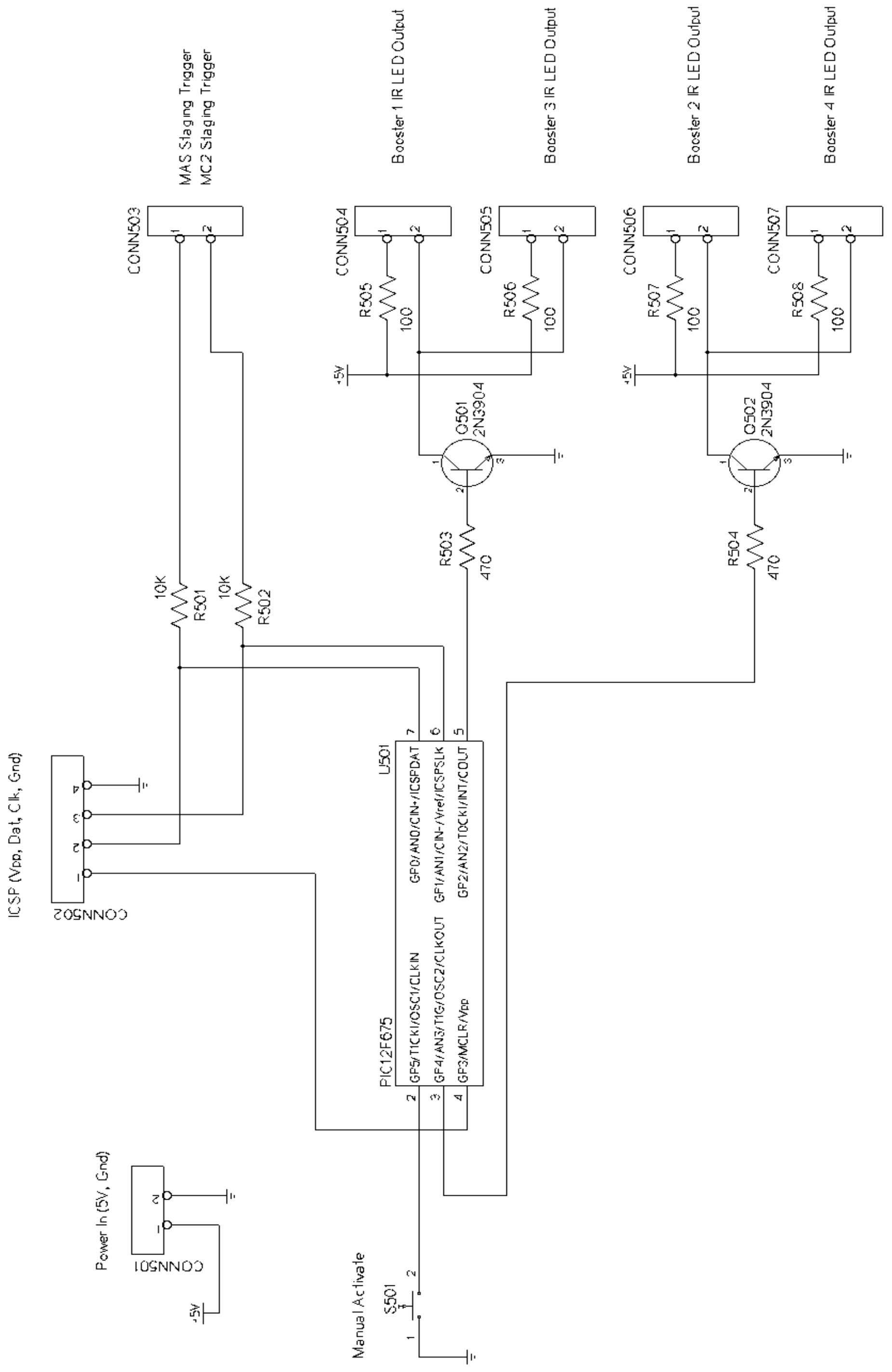
Igniter Controller



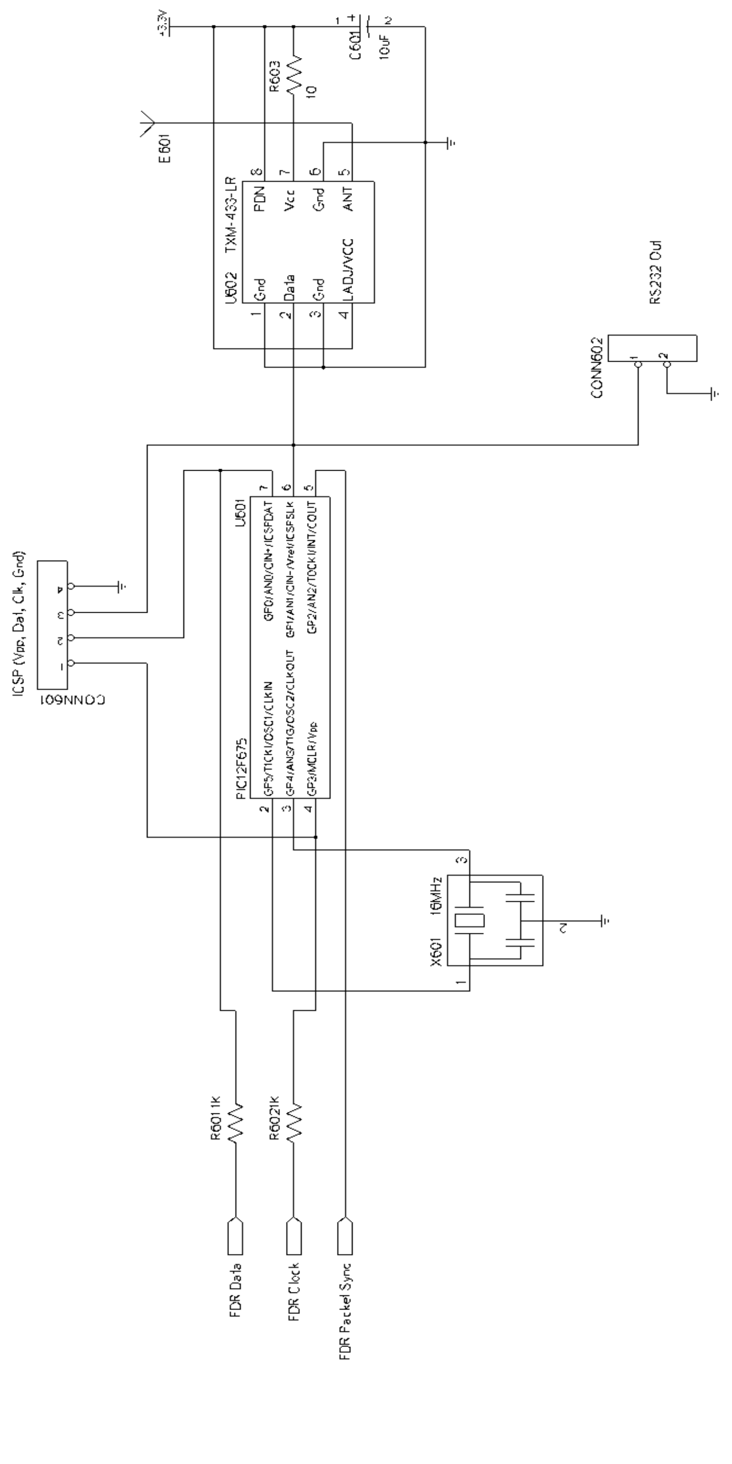
Servo Controller



Staging Controller



Telemetry


+3.3V
C601
10uF
R603
10
E601
U602
TXM-433-LR
PDN
Vcc
Gnd
ANT
Gnd
Data
Gnd
LADJ/VCC
RS232 Out
CONN602
U601
PIC12F675
GP0/AN0/CIN+/ICSPDAT
GP1/AN1/CIN-/Vref/ICSPCLK
GP2/AN2/T0CKI/INT/COUT
GP5/T1CKI/OSC1/CLKIN
GP4/AN3/T1G/OSC2/CLKOUT
GP3/MCLR/Vpp
ICSP (Vpp, Dat, Clk, Gnd)
CONN601
X601
16MHz
R601 1K
R602 1K
FDR Data
FDR Clock
FDR Packet Sync

Bibliography

Advanced Circuits. Available at: http://www.33each.com

ADXL321 Data Sheet

ADXRS300 Data Sheet

BatchPCB. Available at: http://www.batchpcb.com/

Digi-Key Catalog. 2006 Available at: http://dkc3.digikey.com/PDF/T063/1775.pdf

G-Wiz MC2. Available at: http://www.gwiz-partners.com/html/g-wiz_mc2.html

G-Wiz MC2. 2006 Available at: http://www.gwiz-partners.com/html/g-wiz_mc2.html

gEDA Website. Available at: http://geda.seul.org/

MPX2100 Data Sheet

PIC16F785 Data Sheet

Reynolds Electronics. 2006 Available at: http://www.rentron.com/remote_control/TXLC-XXX-LR.htm

SourceBoost Technologies. Available at: http://www.sourceboost.com/

Sparkfun Electronics. Available at: http://www.sparkfun.com

Subversion Version Control System. Available at: http://subversion.tigris.org/

Tower Hobbies. 2006 Available at: www.towerhobbies.com

Ultralife Batteries. 2006 Available at: http://www.ultralifebatteries.com/subcategory.php?ID=2