

Flight Computers

Larry Lynch-Freshner
Software Partner, G-Wiz Partners
<http://www.gwiz-partners.com>

Copyright © 2001 by G-Wiz Partners, All Rights Reserved, permission granted to reprint for non-profit reasons.

Overview

Electronics of all kinds have become common high-power rocket payloads in recent years. Everything from timers, to data collection, to video and telemetry. But the most popular reason for putting a computer in a rocket is to figure out how high it went, and maybe pop a parachute when it gets there. That is what I will be focusing on in this paper.

While there are many things a computer can do while being flown in a rocket, I will concentrate on altitude determination, and in-flight events. I believe that these are the two things most of us care about when we put one of these in our rockets, whether its called a flight computer, an altimeter, or simply avionics.

I will also only be talking about flight computers that are used in our hobby. Most military and commercial rockets have similar electronics payloads as well, but often use very different sensor systems.

Who am I?

I'm Larry Lynch-Freshner, the Software Partner with G-Wiz Partners. I've been developing software professionally for over 15 years, and as a hobbyist 10 years before that. Also, I was in the Air force reserves for several years, where I was trained in aircraft avionics, and spent my weekends testing and fixing instruments, including altimeters, airspeed indicators, and the computers driving them, on several cargo aircraft.

I started playing with flight computers several years ago, after discovering Paul Campbell flight computer web page (<http://www.taniwha.com/~paul/fc>). At the time, he was selling a simple flight computer kit, based on the Intel 8051 Micro controller. It was set up so you could use any sensors, and ADCs, etc. and it was easy to program. You had to write your own code for this. I went through a couple of designs, learned many tough lessons and posted my code for others to use (<http://www.albireo.org/rockets>). Doug Steinfeld has taken things even further at: <http://www.cmass.org/member/Doug.Steinfeld/taniwha.html>. If you want to start experimenting with your own flight computers, I highly recommend these 3 web sites as good places to start.

How do they work?

In general, all flight computers work the same way. They have a sensor on board that can measure some physical quantity that changes during the flight. The computer periodically samples this sensor, and uses this quantity, possibly in conjunction with past quantities, to determine if an event has occurred that is significant (like apogee), and either records that event or performs some action, like igniting a pyrotechnic charge.

Some flight computers have more than one sensor. Why? Because some sensors are better for one type of event while other sensors are better for other event types. When we discuss how each of the sensors work, we'll see why they might be better for one thing than another.

Accelerometers

Accelerometers are sensors that can measure acceleration and output a signal that represents that measurement. The type of accelerometer used most commonly for flight computers is a micro-machined weight-and-spring arrangement on a chip. These were initially developed for the auto industry for use as collision sensors to trigger airbags. They are also used in various industrial settings for vibration sensing. Because of the auto industry connection, they have been manufactured in massive enough quantities to bring the price down to something most of us can find affordable.

The actual sensor is a pair of capacitors formed by a moveable beam placed between two plates. Acceleration along the measurement axis causes the beam to move toward one plate or the other (depending on the direction of acceleration), changing their relative capacitance. The beam deflection, therefore the amount of capacitance change, is relative to the amount of acceleration. Conversion circuits on the sensor chip change this signal from varying capacitance to a varying voltage – which is a lot more useful. A special circuit called an *Analog to Digital Converter (ADC)* is used to convert from a voltage to a binary number that the computer can read.

The computer samples the ADC many times a second, reading how the acceleration changes. When the rocket is on the launch pad, just sitting there, the computer reads a constant acceleration of one 'Gravity' or 'G' – the amount the earth is pulling on it. When the acceleration increases, the computer knows that the rocket has launched. In reality, it is harder than that. We need some way of determining that it's a real launch, and not something silly like the rocket being dropped, rotated, etc. Different manufacturers have different ways of doing this. Under acceleration, the sensor will read changes that, not surprisingly, follow the thrust curve for the motor being used very closely. When thrust stops however, drag starts slowing the rocket down. The sensor reads this as a negative acceleration. This change from positive to negative acceleration is very sharp, and makes it easy for the computer to know that the motor has burned out. Very useful information if you have a second motor you might want to start!

So, how do you determine apogee and altitude when you have an accelerometer?

*** MATH WARNING ***

I'll keep it simple. What is acceleration? It is a change of velocity. What is velocity? Velocity is a change of position. Think about it, it really is that simple. We can calculate the rockets speed by adding up all those speed changes it makes, and adjust for amount of time they changed in. So, if in the rocket first second of flight, we measure 4 G's – a G is 32.2 feet per second (a speed) change in 1 second. So we have measured our rocket changing its speed from 0ft/sec to 4×32.2 or 128.8ft/sec in one second. So it must be going 128.8 ft/sec after one second. If we measure the same over the next second, then we add another 128.8 ft/sec to the total, so we know our rocket is going 257.6 ft/sec. Now, if our rocket starts at 0 ft above the ground, then moves at 128.8 feet per second for 1 second, then it must now be 128.8 ft above the ground. If it then moves at 257.6 ft/sec for 1 second, it now must be $128.8 + 257.6 = 386.4$ ft. above the ground. Now, if you are paying particularly close attention, you have detected a problem with my math. You know that the rocket's acceleration is changing over the seconds measured, not remaining constant. The problem is that we can't tell how its changing, so we have to calculate based on the assumption that the amount it changes from one reading to another averages out between them. We can make it more accurate by sampling more often. In fact, the more samples we take, the more accurate the calculation. This process is called integration.

OK, so we can determine velocity and altitude. There's a neat trick we use here for apogee determination. Take a look at the ideal rocket flight. It accelerated straight up until burnout, then decelerates. When its speed reaches 0, it starts coming back down. So we can say that when the rockets velocity goes from positive to negative (sound familiar?), then we are at apogee. Why positive to negative, and not just to 0? Well, remember we are sampling (and calculating) at discrete intervals. What happens if 0 is reached between 2 intervals? That's right, from plus to minus.

Now, how many times have you actually seen a rocket fly like that? (I have twice). Does that invalidate our assumptions? Not this one. I won't go into the math, but think of it this way: Even if the thrust doesn't go totally into vertical movement, then neither does the pull of gravity subtract totally from forward motion. This means that as the rocket arcs over, the compute will calculate 0 airspeed at the top of the arc, even though the rocket doesn't stop. This does mean the calculated altitude is incorrect though. So obviously, the accelerometer isn't the best way to determine altitude.

Barometric Sensors

Barometric sensors measure atmospheric pressure and output a signal. The type of sensor used most in our hobby consists of two chambers with a diaphragm between them. There is a special type of sensor on the diaphragm called a *Strain Gauge*. Strain Gauges work by resistance changing when they are flexed. This change is amplified, and converted to a voltage change. Strain Gauges are used in many types of sensors, not just barometers, as they are very good sensors of flexure in the surface they are attached to. So, by measuring the flex of a diaphragm, they are essentially measuring the pressure difference between the two sides of the diaphragm. Put a vacuum on one side, and vent the other to open air, and we are detecting absolute barometric air pressure. Again, these things are

used in cars to control fuel injection, emission systems etc, so they are relatively inexpensive.

We all know that the higher you go, the less air, and therefore air pressure, you have. This implies that just knowing the pressure tells you the altitude. Well, if you make your measurements on a 'Standard Day', it will. The big problem is that the weather will also change the air pressure. Aircraft use this principle, but must call their destination airport to get the altimeter 'offset'. This number is dialed in to the altimeter to compensate for the local air pressure (and I know of one flight computer that works this same way). Also, you need very accurate sensors for this to work right. We do something tricky here. When the rocket is on the pad, we calculate altitude from air pressure. This altitude will be incorrect because of local pressure differences, sensor errors, etc. That's OK. We do the same thing at apogee, so the apogee altitude contains all the same errors. Now, when we subtract the pad altitude from the apogee altitude, the errors cancel out, and we get an accurate altitude above ground.

OK, so it is now apparent how apogee altitude is detected, and should be apparent how low altitude for second stage recovery deployment is calculated, but how do you determine other things – lift off, burnout, etc? With great difficulty. Ever notice that most barometer-only flight computers have some other way of determining liftoff? Either a G-switch or a trip wire? That because the only way, using just the barometric sensor, is to look for an altitude change. OK, you can get the same data you get with the accelerometer, by using the same math – only backwards. Like this:

*** MATH WARNING ***

If you take the difference between two altitude samples, adjust for the time between those samples, you get how fast the rocket was going (velocity). If you take the difference between two velocities, you have acceleration. No problem! Well, yes problem. We'll talk about noise later, so suffice to say that this calculated data has problems. Integration causes variations in the data to average out. This process, called *Differentiation*, does the opposite – it exaggerates these differences, increasing the errors in the data.

Magnetic and Other

There are other sensors that can be used to detect in flight events. They are not in widespread use, so I will not say much about them other than to list ones I know about.

- Magnetic (Hall Effect) sensors – These detect the orientation of the earth's magnetic field lines. By watching them rotate, apogee can be detected. There has been a couple of interesting articles about flight computers using this method lately.
- GPS (Global Positioning System) – This is a cool one that hobbyists are just starting to play with. They tend to have problems during the acceleration phase of the flight, and lose satellite lock, but they regain it quickly. GPS will give you direct altitude readout, as well as (differentiated) velocity. There does seem to be

some doubt about the altitude accuracy of them however, and many commercial GPS units have barometric altimeters built in. Look for more products using GPS in the future.

What else goes on?

Remember that these are all computer based, so they can do other things as well. They can initiate timers, look for patterns in the data, record all the samples in memory for later download, form the samples into data packets and transmit them. It can also test for continuity in the pyrotechnic outputs, keep an eye on battery voltage, etc.

What can they tell me?

Now we know how quantities like altitude and accelerate are measured and converted for use by the computer. Is it really that simple? Of course not. There are many factors that come into play to obscure our data.

The conversion process – accuracy, precision, and noise

Simply put, the three biggest contributors of bad data are: the sensor, the Analog to Digital Converter (ADC), and the calculations used – the main parts of the device. Lets start with the sensor. If you were to look at the technical write up (called the *Data Sheet*) for a sensor (any sensor), you would find details like its range of measurement, how accurate that measurement is, how the accuracy changes in different circumstances (and there are many – over the range of the device, by temperature, even by how many times you sample it per second). Sometimes the device will have electrical noise caused by its own internal circuitry mixed with the signal. These factors all contribute to the overall *accuracy* of the measurement. The range of the device also puts operating limitations on the flight computer. If an acceleration sensor only measures +/-50g, then that's all the computer can handle. A minimum pressure puts a cap on measurable altitude.

The next link in the chain is the ADC. These chips will take an analog signal, generally a voltage, within some range, 0-5v in our case, and convert that voltage to a number. Now, a given ADC generates numbers of a certain size – like 8 bit numbers, 10, 12, 14, or 16 bit numbers. They all divide up the signal range equally with their numeric range, so that (in our example) 0v will read 0, and 5v will read the biggest number (255, 1023, 4095 for 8, 10, and 12 bit ADCs respectively). From this example, you can see that the more bits the ADC uses, the smaller a voltage change it can measure. The number of bits in the ADC then determines the *precision* of our measurement. For example, an 8 bit ADC used with a common barometer yields pressure data in increments that are the equivalent of about 100ft. That is the smallest change that can be read is 100 ft of altitude! A common accelerometer on that same ADC yields changes of .5g ! These are big, chunky numbers. Guess what? Most flight computers out there use 8 bit ADCs (including ours). Note that if your precision is higher than your accuracy, then your measurements will contain the noise limiting the sensor's accuracy. Noise can come from many sources, but

this is the largest. The next largest is electrical noise from the flight computer board. A well-designed flight computer isolates the analog circuits from the digital circuits, and employs a large ground plane. The digital portion of a flight computer can actually induce false signals into the analog portion, so isolation is important.

Next is the calculations used in the computer. Most of us are using inexpensive micro controllers to drive our flight computers. A micro controller is a microcomputer with program memory and input / output interfaces all on one chip. Now, almost any calculation done in a computer suffers something called *Rounding Error*. There are exceptions, but they are specialized, so I won't deal with them here. Rounding error occurs with decimal number are converted to binary. When they are converted back, and especially when they are normalized to fit in a certain machine number size. So the best thing for use is to minimize these conversions. I find this relatively easy for accelerometer calculations. Because I don't care about units for event detection, I can integrate the accelerometer signal directly to determine events like liftoff, burnout, and apogee. Only when I need altitude in some recognizable units that I need to do any such conversion. Combined with the fact that integrated altitude isn't very accurate anyway due to the rocket's angle of attack, you can understand why we don't really recommend trusting these numbers. Barometer calculations are more difficult, because we need to get the actual altitude for each pressure reading to eliminate the sensor and weather variations. To make matters worse, the equation to calculate altitude from pressure is a hairy one¹:

$$A = \left(\frac{T_0}{k} \right) \cdot \left[1 - \left(\frac{P_A}{P_0} \right)^{k \cdot \frac{R}{g}} \right]$$

Where:

A = Altitude

T₀ = Temp at Sea Level

k = Local Temperature Gradient

P_A = Pressure at Altitude

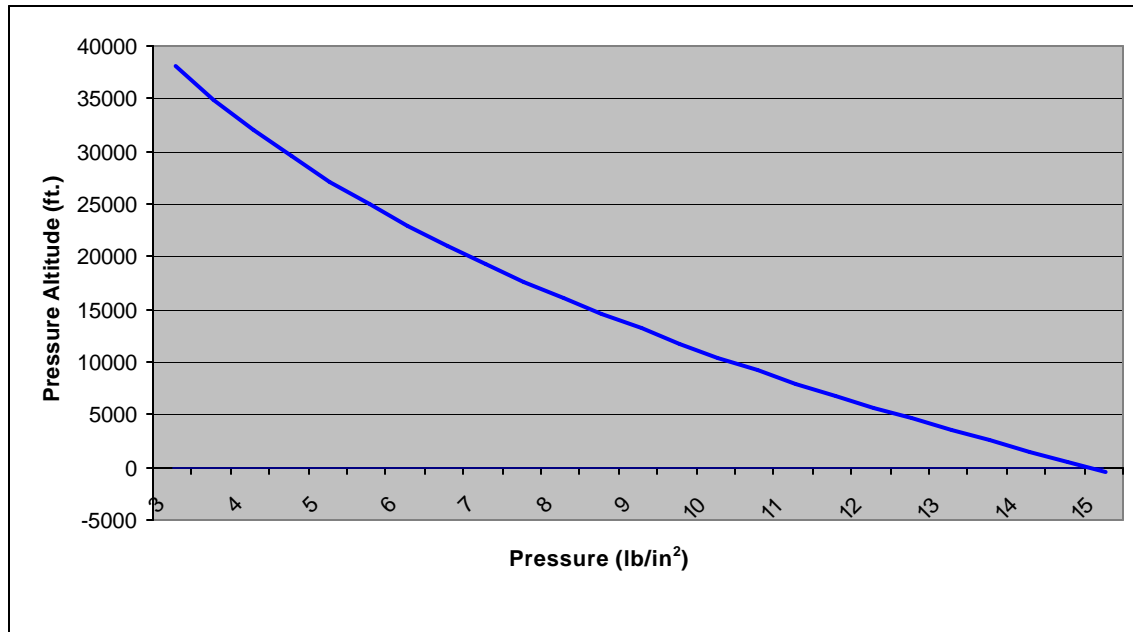
P₀ = Pressure at Sea Level

R = Gas constant for Air

g = Acceleration due to Gravity

Temperatures and gradients used are from the U.S. Standard Atmosphere definition, and vary with altitude².

This type of equation is not only difficult to program, very prone to rounding error, but also takes a long time to execute. Now, the graph of this equation looks like this:



Notice that for low altitudes, the curve is almost straight? Many articles I've read encourage the developer to make use of this fact to perform a *linear approximation* of the altitude. That is, instead of calculating altitude using the equation above, instead they derive the equation for the line forming the slope of the lower altitudes, and use that for altitude. I use a different approach that is still an approximation, but follows this curve very closely. In fact, the calculated deviation from the real altitude is less than is caused by the precision of measurement. For example, in a 10 bit ADC unit, the error caused by ADC precision is +/- 13 ft. (approx., at low altitudes), with my calculation causing less than a foot of error.

Integration vs. Differentiation

Noise is a random variation in the signal. If the signal were to remain the same, we would see a series of small changes that would all average out to the signal value. Obviously, the signal rarely remains the same, but the random behavior of noise still affects it. Now, even though a divide operation is never performed, you can see that the summation process involved in Integration will tend to average out noise. A little too much one measurement, a little too little the next. This is good for accelerometer based computers, as it will tend to make computed (integrated) values more accurate. However, during differentiation, the opposite occurs. Because of the noise, the speed can appear to change abruptly, and randomly between samples. In turn, this causes wild variations in acceleration. To get calculated velocity and acceleration data from a barometric unit that even looks good, let alone has any validity, requires careful *filtering* of the barometric data. Filtering smoothes out the data curve, reducing the random variations. It will also tend to smooth out any real variations in the data, so it has its costs.

How events are detected

Some events are best detected one way, some another. Here's a list of the significant flight events:

1. **Lift-off** – Detect using G-Switch or Accelerometer. Using the accelerometer, look for several consecutive significant acceleration readings. What do I mean by significant? Larger than noise for your system. If you must use a barometric sensor only, look for at least a higher reading than noise to start with, then increasing altitude for a couple of consecutive readings (decreasing pressure).
2. **Engine burn out** – Using an accelerometer is preferred for accuracy. When the measured acceleration goes from positive to negative, burn out has occurred. I would not feel comfortable doing this with a barometric only system without a lot of experimentation.
3. **Apogee** – Again, I prefer using the accelerometer. When calculated airspeed drops to 0 or less, apogee has occurred. Using a barometer, you can look for the pressure to reach a low point, then start increasing (i.e. dropping) – which would give you apogee a bit late, or you could look for several constant pressure readings – which could be early depending on trajectory.
4. **Maximum altitude** – This recorded value is best attained with a barometer. While pressure altitude has many problems, it is a standard method for reading altitude, and I rank it above integrated altitude for accuracy on average.
5. **Low Altitude** – Two stage recovery has become a popular method for recovering those high-altitude flights. Accelerometers are difficult to use past apogee because it may be nose up, nose down, sideways, swinging, etc. we can't tell. The barometric sensor is the only way to go here.
6. **Landing** – Again, I favor the barometer for landing detection. Several seconds of no average pressure change is a good indication that your rocket has stopped coming down.

What can't they tell me?

We've already discussed some of the problems that get in the way of very accurate measurements. Sensor limits, noise, limited precision. Etc. Environmental issues can also get in the way. Temperature causes the '0g' point drift on an accelerometer, though slowly, and affects barometers even more. In fact, we spend a bit extra in our computers, and use a temperature compensated barometer.

Placement can even be a problem. A unit with an accelerometer depends on the measurement axis of the accelerometer chip to be parallel with the thrust axis of the rocket. Many units want a certain end up even. And when the rocket starts to lean over in flight? Well, there will be an error proportional to the angle, even under thrust. This is because on the pad, the Earth's pull was recorded and allowed for. As the rocket tilts, it sees less of the earth's gravity, so it will incorrectly compensate for this. For that matter, the value of the earth's pull changes just a bit from place to place.

For a barometer to read pressure, it needs to be placed in a location vented to atmosphere. Which brings us to turbulence. Turbulence is a localized chaotic movement of air –

which can also mean local changes in air pressure. This is an extremely complex area, and one with a lot of supercomputers devoted to. So let's just say that there will be turbulence. You can minimize its effect on your flight computer by proper placement of vent ports³, 3-4 calibers behind the nose cone is good. Mach transition is a major source of turbulence that affects flight computers. Starting at about Mach 0.8, and culminating with a shockwave leaving a momentary vacuum behind it at Mach. If unprotected, and not allowed for by the developers, this can cause a barometric sensor to think it has reached apogee. Many units on the market use timers to allow the barometer to bypass Mach. Some barometric units have special programming, and special mounting directions that must be followed. Units (like G-Wiz) that are accelerometer based, do not have a problem with this.

What else? Would you believe light? Light hitting the open hole on the face of a barometric sensor can cause false low-pressure readings. How about mountains? Yup – if you launch from within a valley, when the rocket goes higher than the mountains forming the valley, it will read a faster pressure drop than is caused by altitude alone. This is because the mountains keep air in the valley, like soup in a bowl, causes the air pressure in it to be just a bit higher than 'normal'.

The important thing here is to look at what sensors are used. They determine which quantities are being directly measured. Then look at the quantities that must be calculated from those measurements and how those calculations must be done. This will tell you where it is likely to do well, and where it isn't. Also, look at the resolution of the ADC. 8 bits is very low resolution (but it does work). 16 bit is very high – in fact, much higher than the accuracy of the commonly used sensors. 10 to 12 bits would probably be ideal.

How to use them Safely and Effectively

We all try our best to make electronics easy to setup and use. The reality is that all avionics are more complicated than the motor's ejection charge. And some people mess that up. I'm not perfect. I've flown my own avionics, and found that I had done things wrong. So it's important to know how to use your computer, and to take things slow. I highly recommend creating a checklist, and following it carefully.

Learn how to use your flight computer

Read the manual. Twice. If you don't understand something, talk to somebody – a friend who uses one just like it, or call up (or email) the vendor. It's important to know what every jumper does, what headers do what, and what each flash of an LED or beep means. Many flight computer makers include templates for drilling mounting holes. Make several copies of this, cut it out, and glue it to where you will mount it. If it matters (it does with G-Wiz), show which end is nose up. I also recommend wasting a battery. Hook one up, and let the flight computer indicate ready until it dies. This gives you some idea how long a battery lasts. I hate to see people use a fresh 9v for every flight, when they're good (in our units, at least) for several. Even when left on the pad

for a hour each time. Those little 12V jobs, one the other hand, and we use them too, in one unit, don't last long at all. Change them often.

At this point, you should know how to properly mount your computer, what the switches / LEDs / jumpers all do, how and where to connect your ejection charges, and how to judge battery life.

Mounting

Secure mounting is important. I prefer to mount directly to the airframe on thinner rockets, or use a sheet of fiberglass or plywood in the center of the tube on larger diameters. Not only should the board be secure, but the batteries need to be secure as well. Battery holders will generally not hold a battery well under high acceleration unless additional security is provided. I recommend a cable tie, or electrical tape. When the battery is separate, as is the case with G-Wiz, I like to mount the battery so it is resting against a firm surface while under acceleration, then secure it firmly with cable ties and / or tape.

I like to use 3 vent holes, equally spaced around the airframe, and a good 4 calibers behind the nose cone. There was a very good article in the May / June, 1996 issue of *Sport Rocketry* about proper ventilation. I highly recommend reading it. In short, the coefficient of pressure forward of this point deviates enough from 0 to cause incorrect altitude readings. Admittedly, this is only important for a good record, or barometric only units, as the rocket is slow enough at apogee (generally) to give a good maximum altitude reading.

Another important mounting consideration is to make sure that the compartment holding the computer is sealed from ejection gasses. This may sound obvious to many of you, but we have seen way too many boards that have been toasted by ejection. Ejection gasses are highly corrosive, and the over-pressure might damage the barometric sensor.

Finally, make sure all wiring, especially that going to the ejection charges is secure against acceleration. I've seen many connector arrangements that just won't deal with more than 2Gs. If the wires pull out, the charge won't blow. No fun.

Arming jacks

I don't like arming jacks. I think they are more trouble than they're worth. However, many people feel safer with them, so if you are going to use them, find a variety that doesn't depend on a spring while in flight. I have had several failures all traceable to phone jacks. In flight vibration can cause these to break connection, and cause computer resets (if computer power is on one), or pyrotechnic charge failures, if a break occurs at the wrong time. I've found some small plastic key switches that I've had the best luck with. In any case, before you use something, think about how it might behave in flight.

Ejection charges

Well, what can I say – I’m an AeroPac member, so I learned the ‘Walby ejection charge’ direct from the source. I swear by these things. Take a 5ml centrifuge capsule, drill a 1/8” hole in the end, and feed a ‘B’ type Davey-Fire through. Add the powder, and close the lid. But whatever you use, make sure the battery on your flight computer is capable of firing it. Rob Briody (the Hardware Partner, at G-Wiz Partners), has written a paper on how much current is drawn by various electric matches. This paper is available on the IgniterMan web site (<http://www.uniquerocketry.com/>). Most flight computers can only deliver about 1 amp of current. This is sufficient for a ‘B’ type Davey-Fire and flash bulbs. It will fire some other electric matches, but not most hand-dipped igniters. G-Wiz can deliver up to 8 amps continuous, which is why we require 2 batteries – one for the computer, one for the charges.

Checklists

Setting up electronics can be complex. The best way to manage that complexity is to write everything down ahead of time, and practice setting up your rocket. Double check everything, and add in anything missed. Then follow this list when you get to the flight line. If you make sure every line has a check, then it should all work out OK. Here is a sample checklist:

- | | |
|--------------------------|--|
| <input type="checkbox"/> | Prepare and Insert motor |
| <input type="checkbox"/> | Prepare ejection charges |
| <input type="checkbox"/> | Place ejection charges |
| <input type="checkbox"/> | Load buffer and parachute(s) |
| <input type="checkbox"/> | Power on computer |
| <input type="checkbox"/> | Check for proper operation |
| <input type="checkbox"/> | power off computer. |
| <input type="checkbox"/> | (Next two may be reversed) |
| <input type="checkbox"/> | Mount computer securely in payload bay |
| <input type="checkbox"/> | Connect ejection chargers to computer |
| <input type="checkbox"/> | Power on, and check continuity. Power off |
| <input type="checkbox"/> | Insert arming plugs, if any. |
| <input type="checkbox"/> | Assemble Rocket |
| <input type="checkbox"/> | Fill out flight card |
| <input type="checkbox"/> | Have inspected by RSO |
| <input type="checkbox"/> | At Pad: |
| <input type="checkbox"/> | Insert igniter, check continuity |
| <input type="checkbox"/> | Power on computer |
| <input type="checkbox"/> | Verify ejection charge continuity, and ready state of computer |
| <input type="checkbox"/> | Secure rocket. |
| <input type="checkbox"/> | Arm (if needed) |
| <input type="checkbox"/> | Exit flight line |

Sources

- 1: U.S. Standard Atmosphere, 1976, U.S. Government Printing Office, Washington, D.C., 1976
- 2: Kevin Brown's Math Pages. Geophysical Altitudes, <http://www.seanet.com/~ksbrown/kmath054.htm>
- 3: Sport Rocketry, May / June 1996, pp18, 'Vent holes for Altimeters'. Tom Beach with Guppy Youngren.
- 4: High Power Rocketry, Oct 1999. pp46, 'Why Altimeters Fail'. John Fleischer and Chris Pearson.