USING LABVIEW TO DESIGN A PAYLOAD CONTROL SYSTEM

Stephen Horan
Klipsch School of Electrical and Computer Engineering
New Mexico State University
Las Cruces, NM 88003-8001

Abstract

As part of a project to develop small satellites, we have developed a combined ground station and flight computer control software package using LabVIEW. These computer systems are used to acquire data from sensors, control communications links, provide automatic data acquisition capabilities, and provide a user interface. In this paper, we will look at the state machines that describe both sets of software, the challenges for the flight computer development given the PC/104 format, and show how the final product was deployed.

Keywords

Instrumentation system, telemetry system, ground systems, data processing, and system management

INTRODUCTION

The faculty, staff, and students at New Mexico State University (NMSU) began the design and fabrication of a nanosatellite under the Air Force University Nanosatellite program in 2003 [1]. The basic science goal of the satellite is to measure the earth's background brightness level at 300 - 450 nm. This level is needed to design cosmic ray detectors that observe the interaction of cosmic rays with the earth's upper atmosphere. The science objectives are to measure the 300 - 450 nm intensity at least once per minute with associated position and timing information. The payload originally developed for this mission was not chosen by the Air Force to be launched into low-earth orbit. An alternative flight was obtained on a high-altitude research balloon. For this flight opportunity, the mission was renamed to *BalloonSat*. These research balloons fly at 100 kft for 24 hours. With this type of mission, the sensor design and the software design can be tested. Also, the operations of the payload can be fully tested. While the science and engineering environments do not exactly match the orbital cased, they are considerably more challenging than a bench test and can be used to validate the orbital requirements.

In this paper, we describe the design control software for the payload. This software is based on LabVIEW[1] and is run as an executable in both the groundstation and the flight computer. The

[1] LabVIEW is a trademark of National Instruments Corporation

software is designed as a state machine in both elements. The software also has the ability to initiate timing loops so that the operator does not need to explicitly command selected operations at each instance. Rather, the operator can set the flight computer to take actions autonomously. This includes both selected data acquisition and sequences of discrete commands.

SENSOR SUITE AND COMMUNICATIONS

The sensor suite for the payload was chosen to satisfy the science mission plus provide basic health & welfare sensing of the payload environment. Part of the payload design philosophy is to select sensors that can operate from the common 5V power supply and utilize a standard computer interface. The allowed computer interfaces are RS-232 and USB. The communications devices also needed to meet this same constraint of voltage and interfaces. Table 1 lists the sensors and communications devices in the payload.

Table 1 -- Sensors and communications interfaces.

Sensor/Communications	**Manufacturer**	**Purpose**	**Interface**
Photomultiplier Tube	Hamamatsu	Measure 300 - 450 nm light level	RS-232 96/8/N/1
GPS receiver	u-blox	Measure time and position	RS-232 96/8/N/1
Rate Gyro & Magnetometer	MicroStrain	Measure payload movement and magnetic field strength	RS-232 38.4/8/N/1
Communications String 1	Kenwood and Kantronics	Beacon position and quick-look telemetry	RS-232 96/8/N/1
Communications String 2	Kenwood and Kantronics	Command and telemetry communications	RS-232 96/8/N/1
Camera	Matrix Vision	Photography	USB
Analog-to-Digital Converters	National Instruments	Discrete voltage measurements of thermistors and earth sensors	USB

Each device in Table 1 uses a 5V supply voltage. A relay in the flight computer stack is used to switch each device on and off under software control. Only the Communications String 2 is kept powered up at all times the payload is active so that the operator can send commands and check on the payload's status.

In addition to the sensors, the payload has two resistive heaters for the photomultiplier tubes that are under operator control. These can be turned on and off via discrete commands.

COMPUTER SYSTEMS

There are two computer systems required in this system: one comprising the flight computer and one in the ground station. In this section, we will look at the individual computer systems and their associated requirements.

The flight computer is a Diamond Systems Athena II CPU. The computer has 256 MB of RAM and a 500 MHz clock speed. The computer has a 4 GB flash drive on the IDE bus acting as the hard drive. A conventional hard disk could not be used in the flight configuration because of temperature constraints. This particular computer system was chosen because of the ability to run on 5 V power supplies and it requires less than 1 A of current in most operating cases. Additionally, the CPU has on-board support for four RS-232 and four USB ports.

In addition to the CPU, the flight computer has three daughter boards in the PC/104 stack:

- One board with 16 switchable relays whose state can be set under software control,
- One board with 8 RS-232 ports to allow serial communications devices to interface; this board occupies COM5 through COM12 in the communications port array
- One board with power regulators that accepts 12 V battery supply inputs and provides regulated 5 V for the PC/104 stack and the peripherals; the peripheral power is switched through the relay board.

The operating system chosen for the system is Windows 2000. This provides the necessary support for development software and its size is compatible with the constraints of the 4 GB flash drive. A design goal is to have the operating system take up no more than 25% of the available drive.

The ground station is designed to operate completely with laptop computers. This need arises from the fact that to support a balloon flight, there may need to be two mobile ground stations in addition to the main ground station to support flight operations. Therefore, the entire ground station can be assembled with a laptop, portable radio, and VHF antenna. The software will need to run on any of the three laptops in the ground station locations. The laptops chosen are standard laptops running Windows XP.

SOFTWARE SELECTION

The software for the payload control needs to meet the following constraints:

- Can be used both for instrument check-out and for actual operations
- Does not require more than 2 GB of flash disk storage for any support environment for code development
- Can be made into an executable for more rapid execution
- Can work with the memory limitations of the CPU
- Can support multiple RS-232 and USB ports and

- Permits writing to the I/O space of the computer to control the relay board and RS-232 expansion board on the PC/104 stack.

Based on these constraints, the decision was made to choose LabVIEW as the programming development environment. In addition to meeting the constraints, the LabVIEW environment provides the following benefits in this programming

- LabVIEW has built-in tools to assist with the development of the state machine that forms the basis for the control program
- LabVIEW has libraries of analysis functions to make code development easier; for example, there is a library of routines for manipulating digital images including making JPEG formats and reducing the picture size to make thumbnail images
- LabVIEW has a library of widgets to assist in the construction of the user interfaced for the groundstation computers.

The following sections describe the state machines and the control loops used in the payload operations.

STATE MACHINE DEVELOPMENT

As mentioned above, the control programs in the flight computer and the groundstation are based on state machines. This method is used as a way to structure the program and it allows for easy code development because each function is compartmentalized and can be independently verified. This method also allows the code developer to build up the entire control program from basic functions to the final configuration by adding new states to the overall design. As the code is developed, modules for specific functions such as configuring communications ports are used in both the flight computer and groundstation code. Because the same development philosophy is used in both the flight computer and the groundstation, lessons learned in code development are easy to transfer between segments.

The state machine for the Flight Computer is illustrated in Figure 1. The basic control methodology is to first configure the operational parameters in the “Init” state, then initialize the command and telemetry communications RS-232 ports, radio, and modem in the “Port” state, and then go into a wait look for the groundstation to initiate communications in the “Connect” state. Once communications are established with the groundstation, the Flight Computer enters a continuous loop of states until a termination command is received. The states perform the following functions:

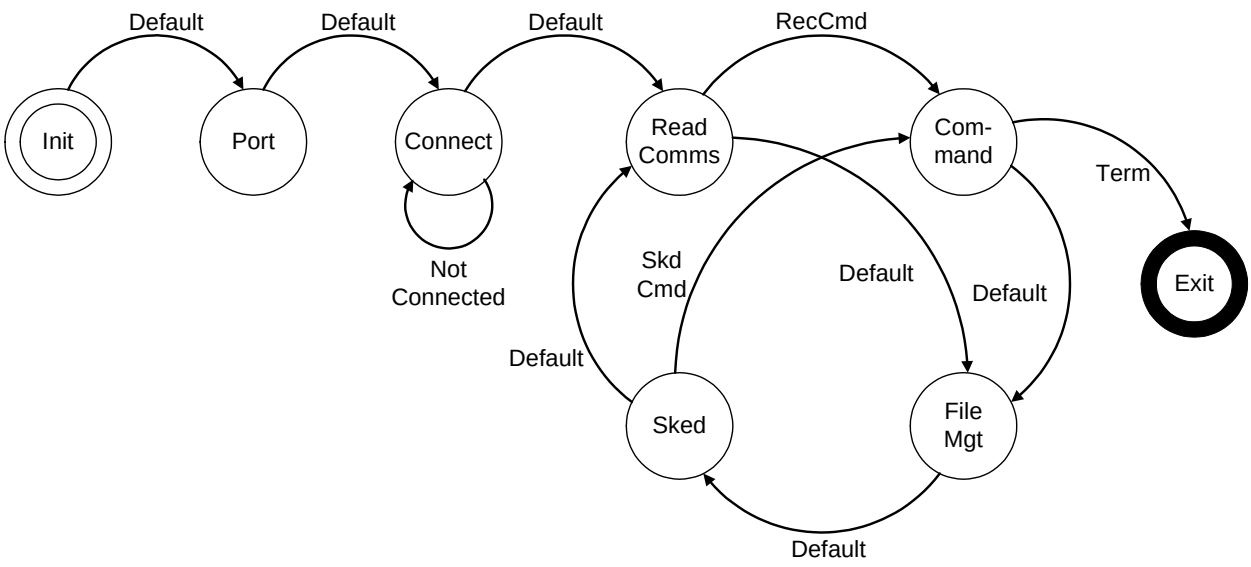

Figure 1 -- Flight computer control state machine.

- "Read Comms" determines if there is an operator command that has been sent; if so, transition to the command processor and if not, go to the "File Management" state,
- "Command" validates, parses, and executes valid commands from the operator or the scheduler; if a terminate command is received, transition to the "Exit" state so that the program can shut down, otherwise, transition to the "File Management" state,
- "File Management" determines if the hourly data files need to be closed and new ones opened,
- "Sked" determines if a command needs to be executed either from the operator's pre-loaded schedule of commands or from the periodic command loops that the operator may have enabled; If there is a scheduled command, transition to the "Command" state, otherwise go back to the "Read Comms" state.

Within the scheduled commands state, there is a hierarchy of command priority: data acquisition commands have highest priority, then position broadcasts, and finally operator commands.

The "Command" state is also configured as its own state machine as illustrated in Figure 2. The command syntax is

$$*SSFNdata**

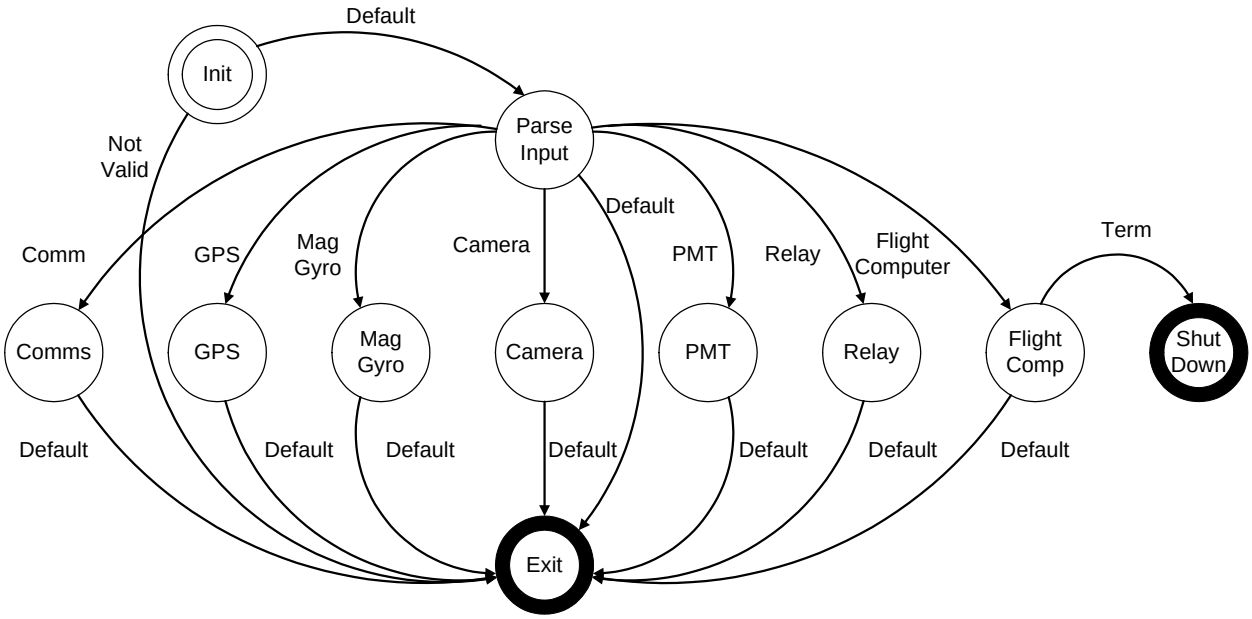

Figure 2 -- State machine for the command interpretation function.

where $$ is the start delimiter
SS is the subsystem mnemonic
FN is the subsystem function
Data is an optional data field if required by the function
* is the ending delimiter

The standard functions for the subsystem are:

- ON – turn on power
- OF – turn off power
- IN – initialize the device
- ST – obtain device status
- CD – send an explicit command as specified in the *data* field.

Each command is sent as an ASCII string. The processing for the Command state machine is:

- Check the command for proper syntax (starts with $$, ends with *, and is at least seven characters long) in the "Init" state
- Determine if the subsystem identifier is valid in the "Parse" state; if valid then the subsystem command function process is started, otherwise, the Command process terminates with an error message
- Check for a valid functions in each subsystem process and, if necessary, data; if the command is valid, then the process is executed with the results reported to the groundstation (this is true for both operator-initiated commands and scheduled commands), otherwise, the action terminates with an error message.

The full list of all commands is found in [2]. Most of the specific functions for each instrument and sensor were developed to initialize or check out the devices during initial development. Each function was programmed as a LabVIEW Virtual Instrument (VI). The VI's can be treated as callable modules from a higher-level routine. An example VI for reading the GPS unit is illustrated in Figure 3. The module accepts the communications port number and GPS message type (GGA, RMC, etc.) as the input. The module reads the GPS and reports out the standard navigation sentence. The other modules to read the other devices work in a similar manner.

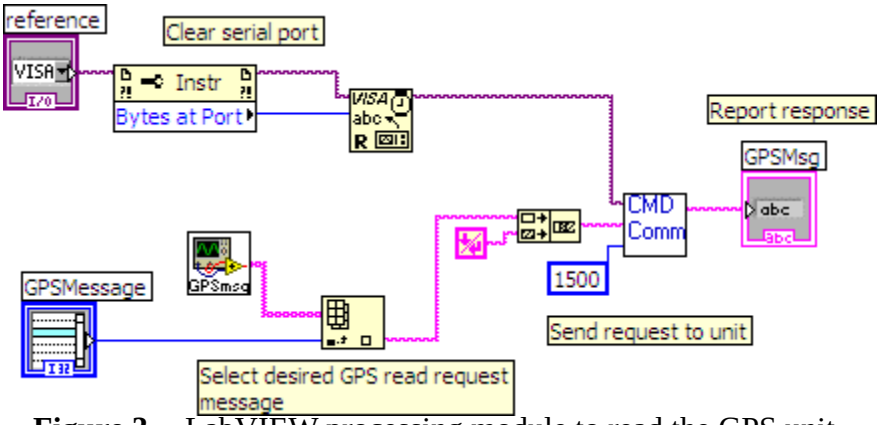

Figure 3 -- LabVIEW processing module to read the GPS unit.

As mentioned above, the Flight Computer can also operate under a schedule. There are three types of schedule operations available:

- Timed schedule from an operator-developed sequence
- Periodic (every ten seconds) reading of the photomultiplier tubes for science data
- Periodic (every five minutes) sending of position and quick-look data.

Each of these can be enabled or disabled via explicit operator command. Additionally, the two periodic schedules can be programmed as part of the operator-developed sequence. The operator-developed sequence is a listing of times and discrete commands. The times are in continuous seconds so there is no change-of-day problem.

The groundstation computer uses a similar control state machine as illustrated in Figure 4. The initialization sets up the communications port and local parameters. The program then goes into a repeating set of states until the program terminates. These states are

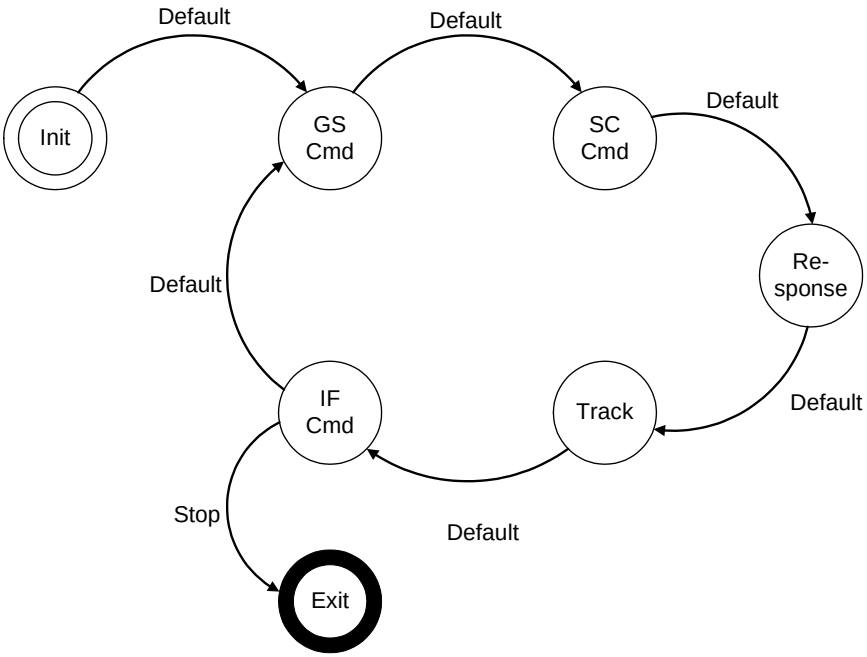

Figure 4 -- Groundstation control state machine.

- "GS Cmd" that looks for operator discrete commands to control the groundstation radios entered via the user interface,
- "SC Cmd" that looks for operator discrete commands to control the payload radios entered via the user interface and generates the timed, keep-alive command,
- "Response" that processes returned messages from either command source and displays them for the operator,
- "Track" that converts the payload GPS position into a pointing vector relative to the groundstation location to drive the radio antennas, if needed,
- "IF Cmd" processes the "quit" command from the user interface,
- "Exit" that performs an orderly shutdown of the program.

Whenever possible, specific buttons or menus on the user interface are used rather than operator data entry. This is done to minimize operator error and speed up training.

The groundstation control program has one autonomous operation programmed into it: keep-alive commanding. Once every three minutes, the flight computer status command is issued. This helps the operator monitor the link state and updates battery voltage, earth sensor, and the thermistor measurements that are displayed on the operator's terminal.

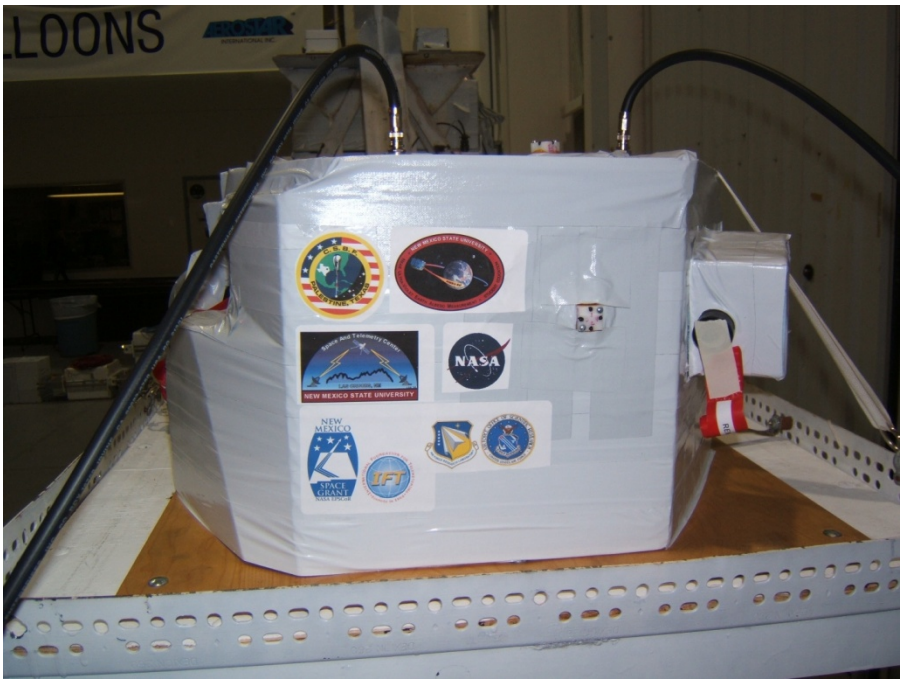

Figure 5 -- *BalloonSat* payload on the carrier awaiting launch.

RESULTS

The flight-ready payload is illustrated in Figure 5. The payload control program has been used in all phases of the pre-launch instrumentation check-out, mission simulations, and operator training. Additionally, it was exercised for approximately 20 hours during the *BalloonSat* mission flown earlier in 2008. During this time, it performed required mission functions.

The groundstation user interface is illustrated in Figure 6. As with the payload control program, the groundstation control program has been used in all phases of the pre-launch instrumentation check-out, mission simulations, and operator training. Additionally, the groundstation was exercised at two locations for approximately 20 hours during the *BalloonSat* mission flown earlier in 2008.

CONCLUSIONS

We were able to use the LabVIEW programming to develop the programs for the flight computer and groundstation and ground station to control a payload over a radio link. To date, these programs have been used in payload check-out and training. The state machine capabilities in LabVIEW greatly assist in the code development. The LabVIEW environment facilitates code sharing between the flight computer and groundstation programs. The environment also facilitates the development of the user interface.

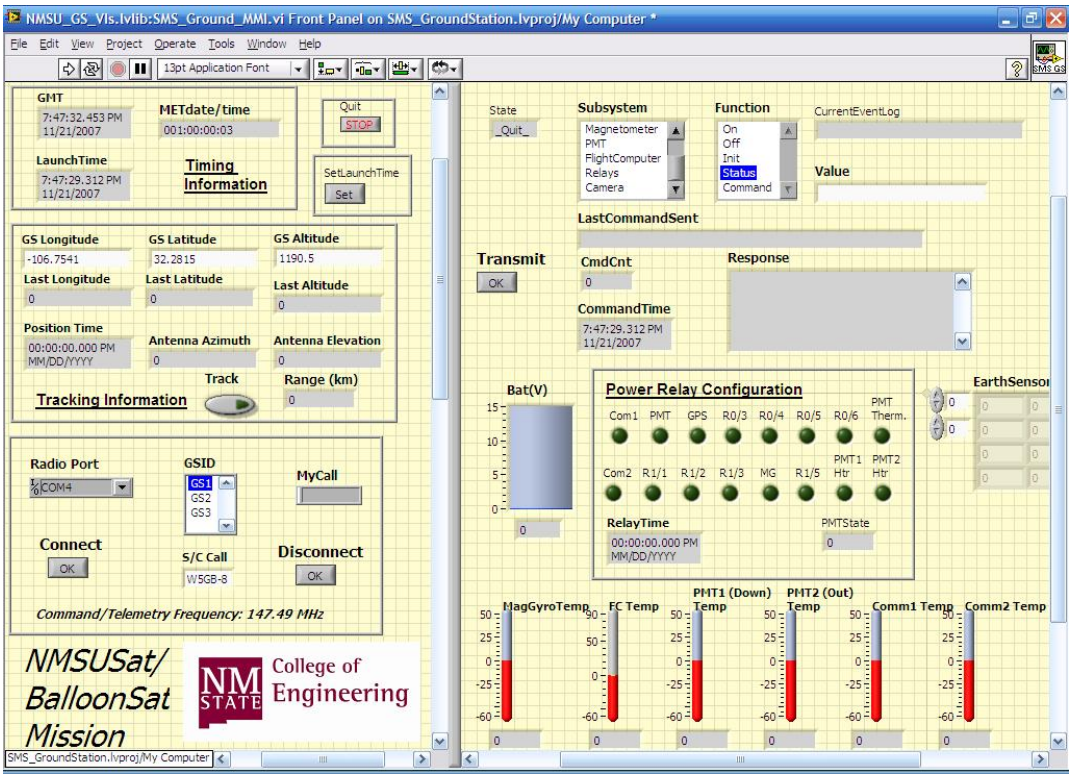

Figure 6 -- User interface for the groundstation computers.

ACKNOWLEDGEMENTS

The development of the software and the acquisition of the hardware were supported under Air Force Office of Sponsored Research grants FA49620-03-1-0168 and FA9550-05-1-0267. Support for the manufacturing of the payload structure was provided by New Mexico Space Grant Consortium. Support for the flight was provided by the International Foundation for Telemetering and NASA's Columbia Scientific Balloon Facility.

REFERENCES

[1] Horan, S., et al., "The New Mexico State University Satellite (NMSUSat) Mission," *Proc. 17th AIAA/USU Conference on Small Satellites*, SSC03-IX-5, Logan, UT, August 2003.

[2] Horan, S., "*BalloonSat* Operating Procedures," NMSU-ECE-08-002, New Mexico State University, Las Cruces, NM, May 2008.