

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

**APPLICATIONS OF RAPID PROTOTYPING
TO THE DESIGN AND TESTING OF
UAV FLIGHT CONTROL SYSTEMS**

by

John A. Komlosy III

March 1998

Thesis Advisor:

Isaac I. Kaminer

Approved for public release; distribution is unlimited.

DTIC QUALITY INSPECTED 2

19980527 074

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE March 1998		3. REPORT TYPE AND DATES COVERED Master's Thesis
4. TITLE AND SUBTITLE APPLICATIONS OF RAPID PROTOTYPING TO THE DESIGN AND TESTING OF UAV FLIGHT CONTROL SYSTEMS				5. FUNDING NUMBERS
6. AUTHOR(S) Komlosy, John A., III				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000				8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.				12b. DISTRIBUTION CODE
ABSTRACT (maximum 200 words) The modern engineer has a myriad of new tools to assist in the design and implementation of ever increasingly complex control systems. A promising emerging technology is rapid prototyping. By totally integrating the development process, a Rapid Prototyping System (RPS) takes the designer from initial concept to testing on actual hardware in a systematic, logical sequence. At the Naval Postgraduate School (NPS), we have applied the concept of rapid prototyping to the discipline of flight control. The NPS RPS consists of a commercially available rapid prototyping software suite and open architecture hardware to permit the greatest possible range of control and navigation projects. The RPS is crucial in that it allows students to participate in projects from the initial concept to the flight testing phase of the design process. This thesis will describe in detail two of these projects; the development of an Airspeed Controller using the RPS tools; and the integration of a Voice Control System developed by ViA, Inc. of Northfield, Minnesota. Both projects demonstrate the inherent flexibility and risk reduction of the rapid prototyping approach to system design.				
14. SUBJECT TERMS Rapid Prototyping, Unmanned Aerial Vehicles, Flight Control Systems				15. NUMBER OF PAGES 108
				16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

Approved for public release; distribution is unlimited

**APPLICATIONS OF RAPID PROTOTYPING TO
THE DESIGN AND TESTING OF UAV FLIGHT CONTROL SYSTEMS**

John A. Komlosy III
Lieutenant Commander, United States Navy
B.A.E., Georgia Institute of Technology, 1985

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

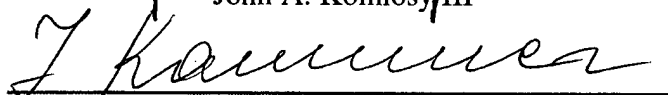
from the

**NAVAL POSTGRADUATE SCHOOL
March 1998**

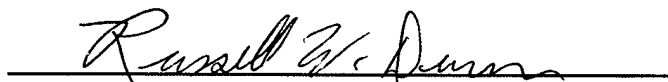
Author:


John A. Komlosy III

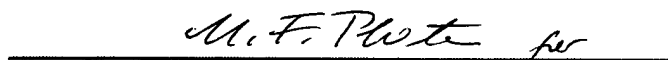
Approved by:



Isaac I. Kaminer, Thesis Advisor



Russ W. Duren, Second Reader



G. H. Lindsey, Chairman
Department of Aeronautics and Astronautics

ABSTRACT

The modern engineer has a myriad of new tools to assist in the design and implementation of ever increasingly complex control systems. A promising emerging technology is rapid prototyping. By totally integrating the development process, a Rapid Prototyping System (RPS) takes the designer from initial concept to testing on actual hardware in a systematic, logical sequence. At the Naval Postgraduate School (NPS), we have applied the concept of rapid prototyping to the discipline of flight control.

The NPS RPS consists of a commercially available rapid prototyping software suite and open architecture hardware to permit the greatest possible range of control and navigation projects. The RPS is crucial in that it allows students to participate in projects from the initial concept to the flight testing phase of the design process. This thesis will describe in detail two of these projects; the development of an Airspeed Controller using the RPS tools; and the integration of a Voice Control System developed by ViA, Inc. of Northfield, Minnesota. Both projects demonstrate the inherent flexibility and risk reduction of the rapid prototyping approach to system design.

TABLE OF CONTENTS

I. INTRODUCTION	1
II. RAPID PROTOTYPING SYSTEM	3
A. SOFTWARE TOOLS	3
1. Realsim GUI	3
2. Xmath/SystemBuild	5
3. AutoCode/Compile and Link	7
4. Interactive Animation Editor/Hardware Connection Editor/Realsim Drivers	7
5. Download and Run	9
B. HARDWARE	10
1. FROG UAV	10
2. Ground Station	13
III. AIRSPEED CONTROLLER	19
A. DESIGN APPROACH AND REQUIREMENTS	19
1. Control Signal Path	20
2. Airspeed Controller Requirements	21
B. DESIGNING THE CONTROLLER IN THE XMATH/SYSTEMBUILD ENVIRONMENT	21
1. FROG Controller Overview	22
2. Throttle Control SuperBlock Overview	23
3. Current Conditions Hold SuperBlock	23
4. Open Loop Controller	24
5. Closed Loop Controller	25
6. Control Loop Analysis	27
7. Sensor Loop Analysis	27
8. State Analysis	30
C. AIRSPEED CONTROLLER IMPLEMENTATION	30
1. Volts to Airspeed Conversion	30
2. Airspeed to PWM Conversion	32
3. PWM to Volts Conversion	33
4. User Interface	34
D. SIMULATION AND TESTING IN THE XMATH/SYSTEMBUILD ENVIRONMENT	36
1. Simulation Runs	36
E. HARDWARE IN THE LOOP TESTING	39
F. FLIGHT TESTING	41

1. Transition Tests Results.....	41
2. Airspeed Tracking Tests Results.....	42
3. Summary of Flight Testing	45
IV. VOICE CONTROL	47
A. BACKGROUND.....	47
B. VOICE CONTROL HARDWARE AND SOFTWARE.....	47
1. Interfacing with the RPS.....	47
2. ViA Wearable Computer	51
3. ViA Hand Held Display/Audio Headset.....	52
4. Wearable Software.....	54
5. Real-time Video Capture System.....	55
C. HARDWARE IN THE LOOP TESTING.....	57
D. FLIGHT TESTING.....	58
1. Flight Test Results	58
2. Control Gains	59
3. Turn Performance	61
4. Image Capture.....	61
5. Summary	61
V. CONCLUSIONS AND RECOMMENDATIONS	63
A. IMPROVEMENTS TO THE RPS	63
1. Sun Workstation/AC100.....	63
2. Comm Box/Modified Futaba Transmitter.....	64
3. Airfield.....	64
B. AIRSPEED CONTROLLER	64
1. Integrated Flight Management Page	64
2. Control Groundspeed.....	65
C. VOICE CONTROL SYSTEM.....	65
1. ViA Touchscreen	65
2. Airspeed Control.....	65
D. CONCLUSION.....	66
APPENDIX A. BLOCKSCRIPT SOURCE CODE	67
APPENDIX B. RESULTS OF AIRSPEED CONTROLLER SIMULATIONS IN XMATH/SYSTEMBUILD.....	69
APPENDIX C. FLIGHT TEST EQUIPMENT SETUP AND OPERATIONAL PROCEDURES.....	83
LIST OF REFERENCES	89

INITIAL DISTRIBUTION LIST.....	91
--------------------------------	----

LIST OF FIGURES

Figure 1. MATRIX _x Product Family [Ref. 4]	4
Figure 2. Realsim GUI	4
Figure 3. Xmath/SystemBuild Architecture [Ref. 4]	5
Figure 4. SystemBuild Graphical Design Environment	6
Figure 5. Example IA Interface	8
Figure 6. Real-time Control Windows	9
Figure 7. RPS Hardware Configuration	10
Figure 8. FROG UAV	11
Figure 9. Center Payload Bay	12
Figure 10. RPS Ground Station	13
Figure 11. AC100 Luggable Computer	14
Figure 12. Comm Box	16
Figure 13. Antenna Array	17
Figure 14. Control Signal Path	20
Figure 15. Upper Level of FROG Control Model	22
Figure 16. Flight Management SuperBlock	23
Figure 17. Throttle Control SuperBlock	24
Figure 18. Current Conditions Hold SuperBlock	24
Figure 19. Open Loop Controller Configuration	25
Figure 20. Closed Loop Controller Configuration	26
Figure 21. Control Loop	28
Figure 22. Control Loop Bode Plot	28
Figure 23. Sensor Loop	29
Figure 24. Sensor Loop Bode Plot	29
Figure 25. Airspeed Conversions SuperBlock	31
Figure 26. Airspeed Filtering	32
Figure 27. Knots to PWM Conversion	33
Figure 28. PWM to Volts Conversion	34
Figure 29. Airspeed Controller IA Interface Page	35
Figure 30. Simulation Setup	37
Figure 31. Simulation Results	38
Figure 32. IAS/GS Comparison	42
Figure 33. PWM Transmitted	43
Figure 34. Run #1	44
Figure 35. Run #2	44
Figure 36. VCS Overview	48
Figure 37. PC_S_C30 Laptop Display Window	50
Figure 38. Voice Control IA Interface	50

Figure 39. Voice Commands Variable Gain Block	51
Figure 40. ViA Wearable Computer [Ref. 7].....	52
Figure 41. ViA Hand Held Display [Ref. 7].....	53
Figure 42. FROG Voice Control System.....	53
Figure 43. Touchscreen Display	54
Figure 44. FROG Camera Installation.....	56
Figure 45. Notebook TV External Tuner [Ref. 8].....	56
Figure 46. Lateral Voice Commands.....	60
Figure 47. Longitudinal Voice Commands.....	60

LIST OF TABLES

Table 1. AC100 I/O Configuration	15
Table 2. Airspeed Controller State Analysis.....	30
Table 3. Laptop Control Keys.....	49
Table 4. ViA Wearable PC Card Socket Configuration	52
Table 5. Flight Control Voice Commands.....	55

ACKNOWLEDGMENT

I would like to thank Steve Case and Jennifer Wightman of ViA, Inc. for their technical assistance and the cool toys to play with. ViA's outstanding support was instrumental in the realization of an operational Voice Control System.

I would also like to thank my advisor Dr. Isaac Kaminer for his guidance throughout this project.

Most of all, I would like to thank my wife Leslee; and my sons Anthony and Michael; for their unwavering support and understanding over the past two years.

I. INTRODUCTION

The modern engineer has a myriad of new tools to assist in the design and implementation of ever increasingly complex control systems. One promising emerging technology is rapid prototyping. Rapid prototyping allows the user to make the most of limited time whether using classical or modern control techniques. By totally integrating the development process, a Rapid Prototyping System (RPS) takes the designer from initial concept to testing on actual hardware in a systematic, logical sequence. A RPS uses the capacity of the digital computer to ease the design process by providing a graphical, interactive environment. The results are tested in simulation and coded in a high level language using rapid prototyping software tools. The real power of the RPS is in the coding ability of the software tools. RPS coding allows the designer to create a functional prototype early in the development process and facilitates system changes and improvements by removing the laborious task of writing processor code by hand. Additional RPS tools allow hardware in the loop testing of the system's actual equipment.

At the Naval Postgraduate School's Aeronautical Engineering Department (NPS Aero), we have applied the concept of rapid prototyping to the discipline of flight control. Traditionally, flight control systems have been designed using classical design methods. The advent of digital flight control systems and their associated real-time processors has begun to change this fact. However, engineers are reluctant to utilize computer generated code for safety critical systems such as flight control. It is our aim at NPS to demonstrate that RPS designed systems are safe and are the future of flight control design.

Once widely accepted as safe and effective, the practice of rapid prototyping could revolutionize the flight control design process. This is especially true for applications to the operational flight programs (OFP) of military aircraft. Currently, most

military aircraft utilize unique proprietary programming languages; resulting in OFPs that are extremely cumbersome and difficult to update. Most platforms only have a few engineers that have the knowledge to write code for that system. The incorporation of rapid prototyping would greatly speed up the update process as well as greatly decrease the cost.

The NPS RPS consists of a commercially available rapid prototyping software suite and open architecture hardware to permit the greatest possible range of control and navigation projects. The NPS RPS takes rapid prototyping one step further by using the software tools to perform actual operational flight testing of the control systems. The RPS is crucial in that it allows the Postgraduate School's unique brand of officer/student to participate in projects from the initial concept to the flight testing phases of the design process. This thesis will describe in detail two of these projects; the development of an Airspeed Controller using the RPS tools; and the integration of a Voice Control System developed by ViA, Inc. of Northfield, Minnesota. Both projects demonstrate the inherent flexibility and risk reduction of the rapid prototyping approach to system design.

II. RAPID PROTOTYPING SYSTEM

The purpose of a Rapid Prototyping System (RPS) is to aid the systems engineering process by providing a set of integrated tools that allow the engineer to quickly design, test and implement a control concept. The RPS developed by the Naval Postgraduate School's Aero Department utilizes the MATRIX_X Product Family of software tools developed by Integrated Systems, Inc. (ISI) of Sunnyvale, California. This software collection along with the ground station and FROG unmanned aerial vehicle (UAV) allow control projects to be developed from initial concept to final flight testing. The iterative system design process is greatly accelerated since changes to the project do not require extensive manual reworking of the real-time controller computer code. This is very important in the NPS academic setting as it allows the officer/students to complete projects from the initial design, to the final testing phase in the limited amount of time available. Refs. [1-3] provide additional information about the RPS.

A. SOFTWARE TOOLS

The MATRIX_X Product Family provides an integrated set of software tools for the development of control systems. Figure 1 illustrates how the different MATRIX_X tools are integrated and the functionality each provides. The functions of each component of the rapid prototyping software set are described below.

1. Realsim GUI

The Realsim Graphical User Interface (GUI) provides overall control of the MATRIX_X rapid prototyping tools. The GUI (Figure 2) is intended to step the user through the design process from initial formulation to actual real-time implementation of

the control system. The GUI also controls data acquisition with the Data Acquisition Editor function that allows the recording of any system input or output.

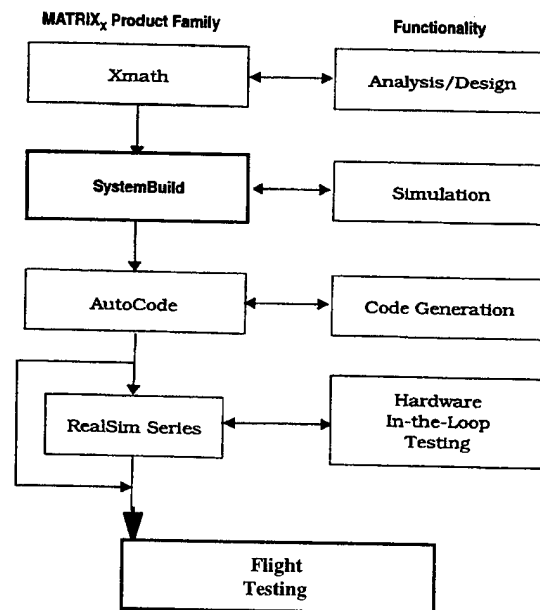


Figure 1. MATRIX_x Product Family [Ref. 4]

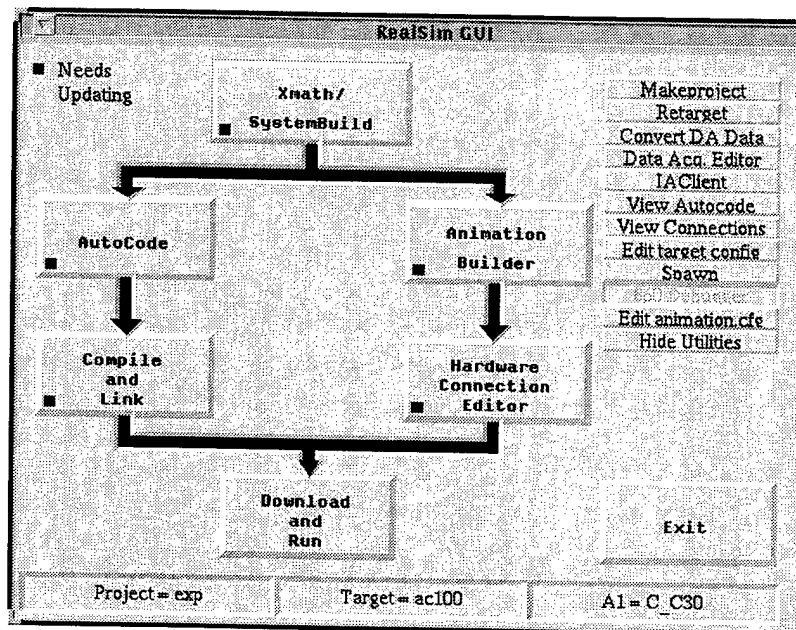


Figure 2. Realsim GUI

2. Xmath/SystemBuild

Xmath/SystemBuild is an interrelated program set very similar to the more familiar Matlab/Simulink products produced by MathWorks Inc. Xmath is the computational engine of the set and provides matrix manipulation and many built-in functions to aid in the design and analysis of control systems. SystemBuild is a graphical, interactive program that uses both ready-made and user defined blocks as the primary modeling element. SystemBuild also provides extensive simulation services. The Xmath/SystemBuild programs utilize Interprocess Communication (IPC) to synchronize functions and share data, allowing several processes to operate simultaneously. Figure 3 illustrates the interrelationships between the Xmath/SystemBuild components.

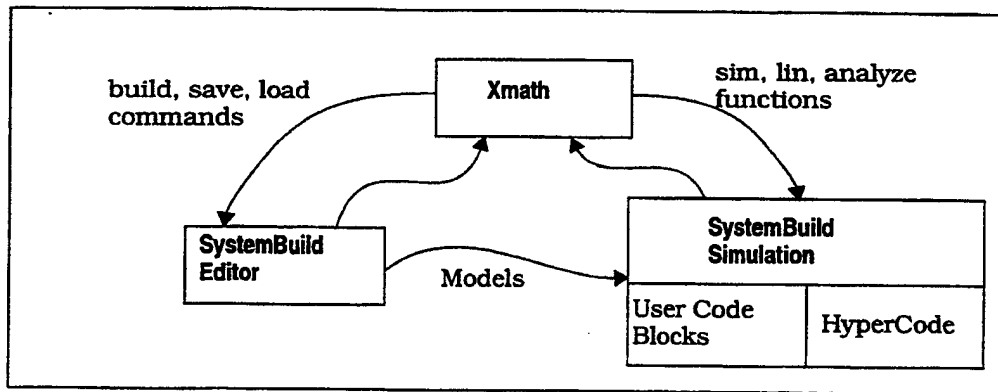


Figure 3. Xmath/SystemBuild Architecture [Ref. 4]

SystemBuild (Figure 4) utilizes a hierarchical "SuperBlock" concept to aid in the construction and understanding of control systems. SuperBlocks are made up of function blocks that come from the library provided with the SystemBuild program; and of user defined blocks. One of the program's strengths is that it allows the user to label and display each individual variable. The connection editor allows the user to pass the variables up and down the hierarchical structure while preserving the variable names.

Connections can also be made to the “outside world”, with inputs and outputs utilized by the Interactive Animation Editor and/or Hardware Connection Editor described later in this section. SystemBuild also includes a “Transform Block” function that converts continuous time systems to discrete time systems and vice versa. Figure 4 shows two equivalent systems. A time delay must be added to the discrete system to prevent algebraic loop errors when code is generated.

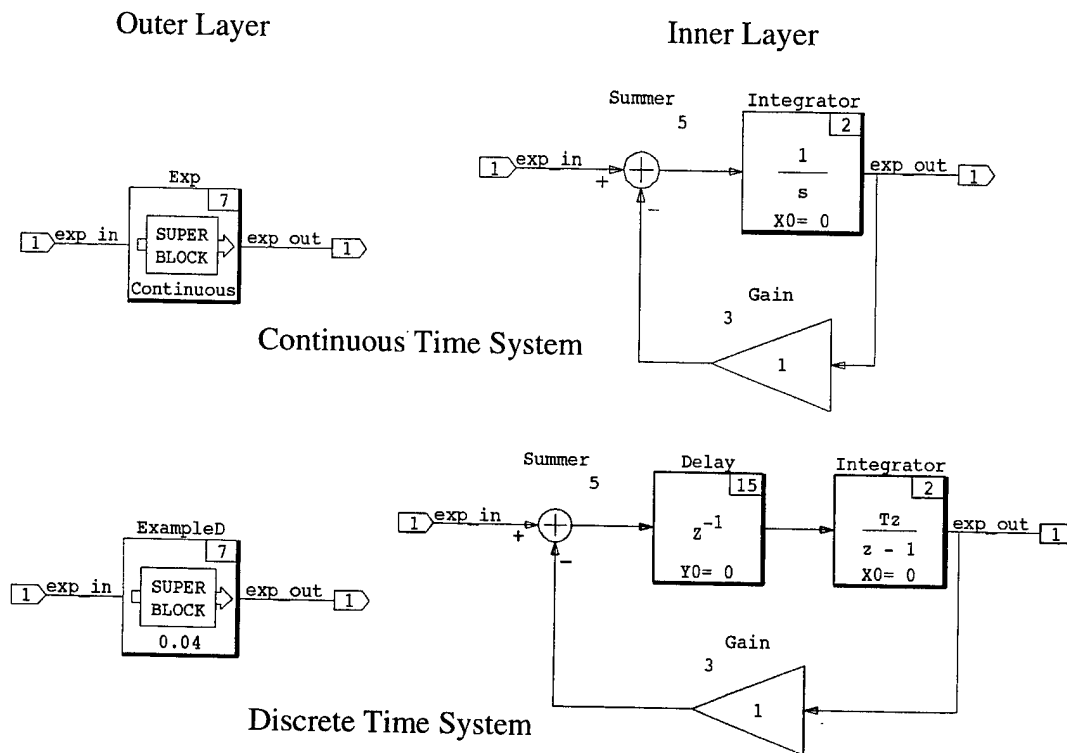


Figure 4. SystemBuild Graphical Design Environment

The Xmath/SystemBuild program set includes an extensive array of simulation functions that allow the engineer to test a new control system and determine whether or not it meets design requirements. The use of these functions will be covered in depth in Chapter III. SystemBuild allows the user to save the system as a real-time code file (.rtf) that is used by the other Realsim components to generate code in a higher level language

such as C or ADA. This code can be run on a real-time processor and used for hardware in the loop testing or actual real-time system control.

3. AutoCode/Compile and Link

One of the most powerful and time saving elements of the MATRIX_X Product Family is AutoCode. AutoCode, on the left-hand path of the Realsim GUI (Figure 2), uses the .rtf file generated by SystemBuild as an input to generate high level code; in the C programming language in the case of the NPS RPS. In order to generate code for the proper real-time processor, the target_config.cfg file must be updated via the retarget utility available in the Realsim GUI. The lower middle and right-hand blocks of the Realsim GUI (Figure 2) show the user the currently selected host computer and target real-time processor.

Once the code has been generated it is sent to a host computer via the "Compile and Link" function. The host computer, described in full in the next section, also contains the real-time processor and input/output boards. Selecting Compile and Link in the Realsim GUI transfers the C files to the host computer via FTP. The host computer compiler generates the object code and the link produces the executable code for the target processor.

4. Interactive Animation Editor/Hardware Connection Editor/Realsim Drivers

The right-hand side of the Realsim GUI steps the engineer through the design and hook-up of the input/output (I/O) interfaces for the control system. The Interactive Animation (IA) Editor enables the user to design and build a graphical interface with the control system that allows real-time user inputs as well as the display of system outputs during testing and operational use of the controller. The connection editor allows the

input and output display icons to be connected to the system in the same manner as connections are made in SystemBuild. Several interconnected IA interfaces may be created for a control system. Figure 5 shows an example of a user IA interface created for the system pictured in Figure 4.

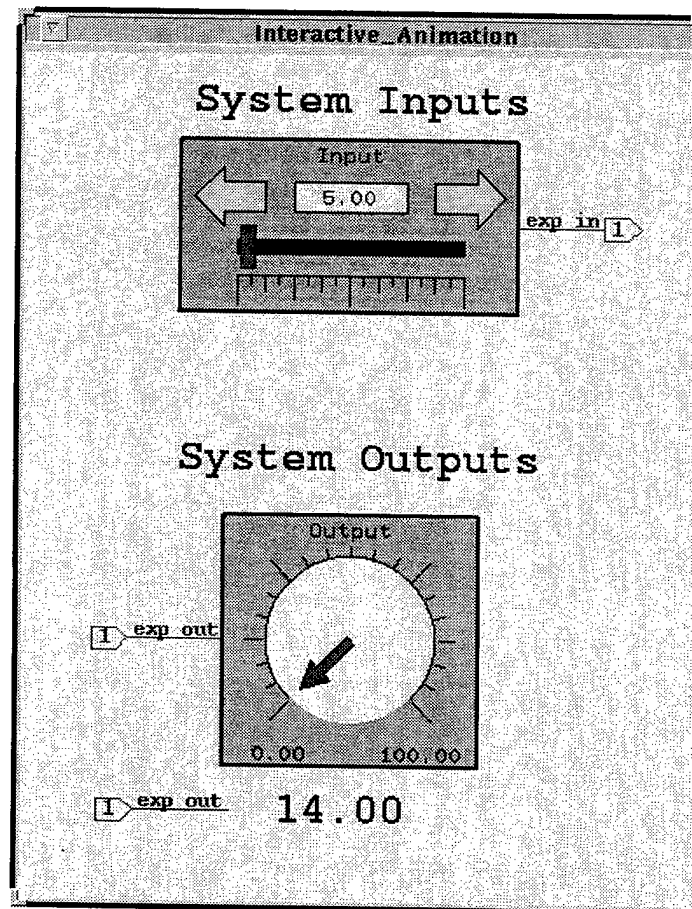


Figure 5. Example IA Interface

The Hardware Connection Editor (HCE) is used to associate system inputs and outputs with external I/O hardware. Many different external I/O devices are available from ISI and are provided complete with Realsim compatible drivers. The HCE is configured to recognize the available I/O boards and allows only functions associated with those boards to be selected. For a complete explanation of the HCE, see Ref. [4].

5. Download and Run

The final selection available on the Realsim GUI is "Download and Run". Selecting this function will load the executable code into the target processor and prepare it for real-time operation. The IA Client Control window and the upper level user IA interface will appear on the workstation screen (Figure 6). The Client Control window enables the computer operator to start and stop the real-time controller. Once "Start Controller" is selected, the IA interface windows are active and allow the computer operator to input commands to, and observe reactions of, the control system. The Client Control window also allows the operator to record the system variables selected with the Data Acquisition Editor.

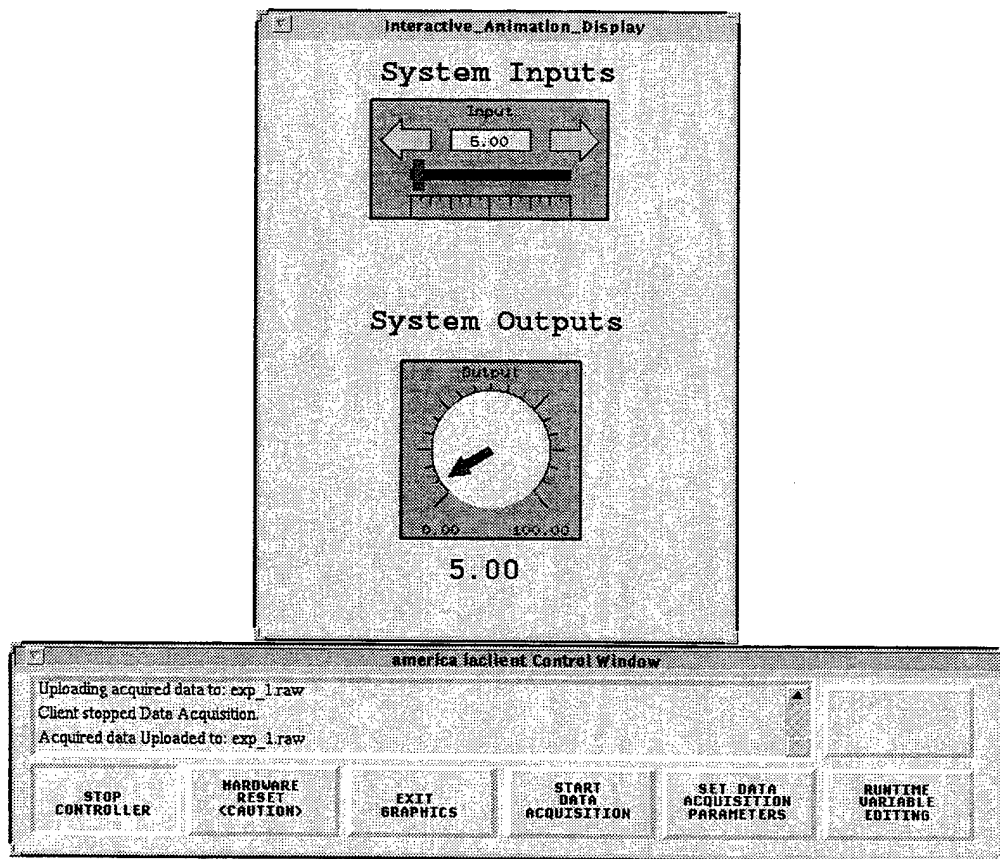


Figure 6. Real-time Control Windows

B. HARDWARE

The hardware portion of the RPS is designed around readily available commercial equipment in order to be as flexible as possible. This open architecture allows for the widest spectrum of possible applications and design projects. An overview of the hardware portion of the RPS is provided in Figure 7. Please refer to it often as you read the next few sections.

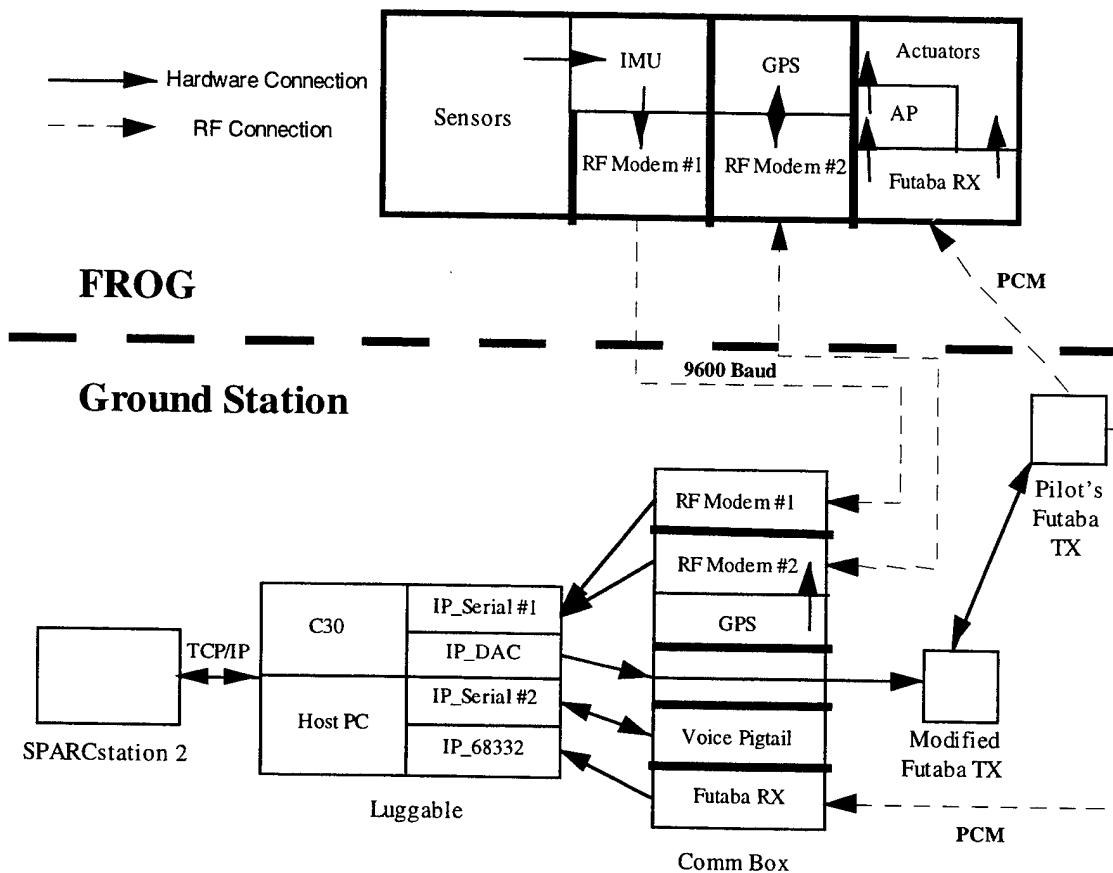


Figure 7. RPS Hardware Configuration

1. FROG UAV

The FOG-R flight vehicle (Figure 8) was obtained from the U. S. Army's TEXCOM Experimentation Center at Fort Hunter Liggett, California. Originally wire

guided, the aircraft has been converted to utilize standard Futaba radio control equipment common to R/C enthusiasts. Nicknamed the FROG, the aircraft is a high wing monoplane with the engine mounted on a pylon atop the twelve foot span. It features two payload bays that can hold a total of twenty pounds of equipment. The aircraft is equipped with an avionics and sensor suite that enables it to be controlled from the ground by computer. This same suite will eventually allow for autonomous operation. The main components are described below.

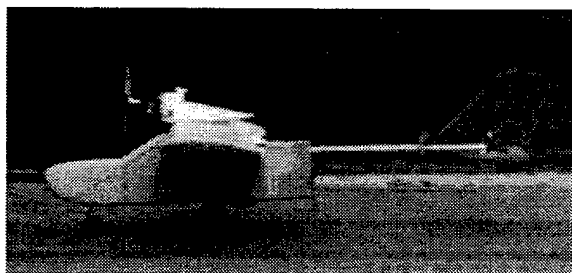


Figure 8. FROG UAV

a. Sensors

The FROG has a full pitot/static system consisting of separate static and total pressure sensors located in the forward payload bay. These sensors output analog voltage signals to the IMU. These signals are sampled and transmitted to the ground station as described below.

Displacement of the aileron and elevator are measured by potential sensors which output analog voltage signals to the IMU. These signals are sampled and transmitted to the ground station as described below.

b. Watson Inertial Measurement Unit (IMU-600AD)

Located in the center payload bay, Figure 9, the IMU measures aircraft linear accelerations and angular rates. It samples the data at a 25 Hz rate and outputs it in

the digital RS-232 format. The IMU also has the capability to sample and transmit five additional channels of data, four of which are used by the RPS. The first two channels, called word1_imu and word2_imu in the SystemBuild Model, sample the analog signals from the total and static pressure sensors. Channels three and four, word3_imu and word4_imu sample and output the signals from the control surface displacement potential sensors.

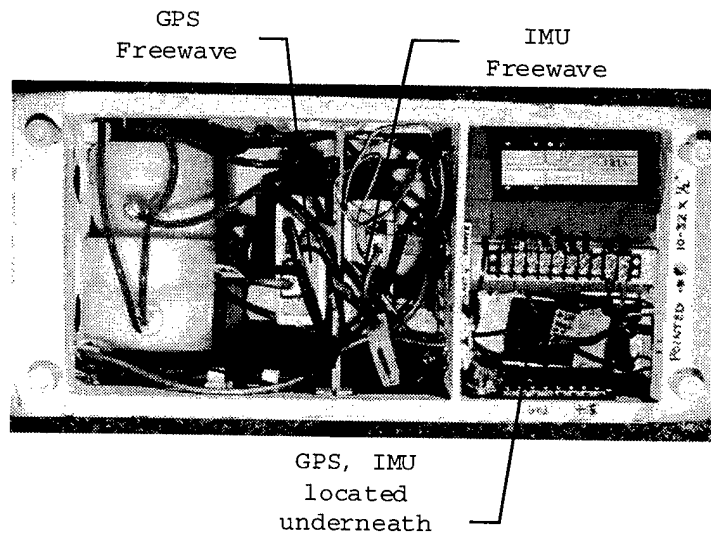


Figure 9. Center Payload Bay

c. *Motorola Encore Global Positioning System (GPS)*

Also located in the center payload bay, the GPS (Figure 9) operates in a differential mode utilizing corrections from an identical GPS located in the base station. The GPS receive antenna is located on the tail boom of the aircraft just forward of the empennage.

d. *DGR-115 Spread Spectrum RF Modems*

Digital data from the IMU and GPS is transmitted to the ground station by two spread spectrum modems located in the center payload bay (Figure 9). Produced by

Freewave Inc., they are capable of the transmission of data at rates of up to 116K baud at distances up to twenty miles. Freewave #1 normally operates in a broadcast mode and sends the IMU data down to the ground station. Freewave #2 both sends aircraft GPS data down to the ground station, and receives GPS differential corrections from the ground station in a full duplex mode. Both links operate at a 9600 baud data rate.

2. Ground Station

Due to the high replacement cost of the computing equipment and the tenuous nature of UAV operations, the ground station concept is employed to reduce overall risk of the program. The ground station provides all the computational power used to perform the flight management and data collection functions of the RPS. The "portable" ground station portion of the RPS, Figure 10, consists of the three major components described below.

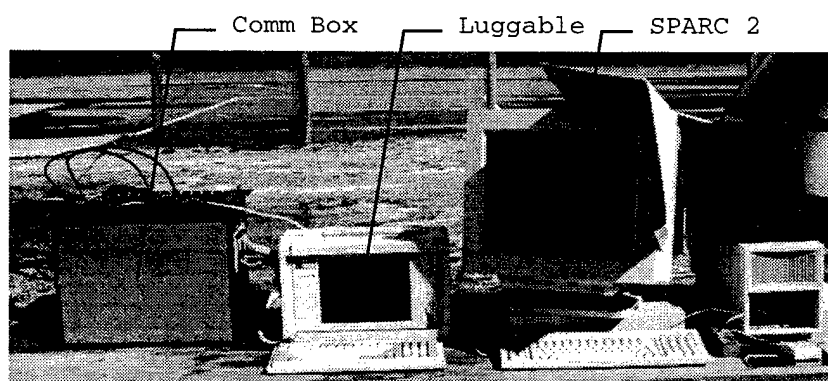


Figure 10. RPS Ground Station

a. *SPARC 2*

The SPARCStation 2 workstation named "Intrepid" executes all of the MATRIX_x software tools described in Section A of this chapter. Everything from initial development, to the actual control of the aircraft during flight test, is performed utilizing

this computer. During both hardware in the loop and flight testing, control inputs to the aircraft are entered utilizing IA screens displayed on this computer. All data recording and subsequent data analysis is also performed on this workstation. In the interest of portability and cost, the workstation is slated to be replaced by a Pentium notebook computer running the MATRIX_X software tools.

b. Luggable PC/IP Modules

The Luggable PC unit, called AC100 (Figure 11), contains the host processor and the AC-100 Model C30 real-time hardware controller. The host computer handles FTP, compile, link and download functions of the system as described in Section A of this chapter. The C30 board contains a Texas Instruments' TMS320C30 floating-point DSP. The C30 board works in conjunction with the "DSP_FLEX" board that can hold up to four I/O boards called "IP" modules. Once "Download and Run" is selected on the Realsim GUI, the C30 DSP executes the controller code and provides commands to the IP modules on the DSP_FLEX board and also to the IA screens by a TCP/IP connection to the SPARC workstation. AC100 contains one DSP_FLEX board with the following IP modules connected to the C30 board by ribbon cables. Table 1 summarizes the configuration of the IP_modules.

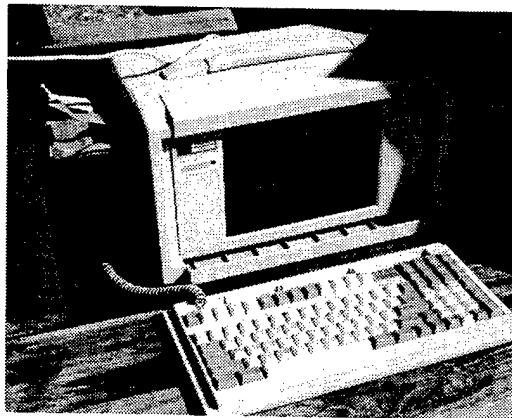


Figure 11. AC100 Luggable Computer

(1) **IP_Serial Module:** AC100 contains two of these boards. Each provides two channels, labeled A and B, of serial asynchronous or synchronous communications protocols with RS-232-C and RS-422 capability. Serial #1 handles inputs from the Aircraft's IMU and the Differential GPS (DGPS) system. Serial #2 is used in conjunction with the Voice Control System.

(2) **IP_DAC Module:** This board provides six channels of digital to analog conversion. It is used to provide flight control command voltages to the modified Futaba RC transmitter described below.

(3) **IP_68332 Module:** This board has sixteen channels of user configurable input or output. It is used to measure the PWM signals being transmitted by the Futaba controllers.

Module	DSP_FLEX Position	Input		Output	
		A	B	A	B
IP_Serial #1	1	IMU RF Modem	GPS RF Modem		
IP_Serial #2	3	Voice from Laptop		Voice to Laptop	
IP_DAC	2			Volts to Futaba TX	
IP_68332	4	Futaba PWM RX			

Table 1. AC100 I/O Configuration

c. Communications Box/Antennas

The communications box (Figure 12) contains all the equipment necessary to send and receive data from the FROG UAV. The four IP boards are connected to the communications box by 50 pin SCSI ribbon cables. The main components of the comm box/antenna array are described below.

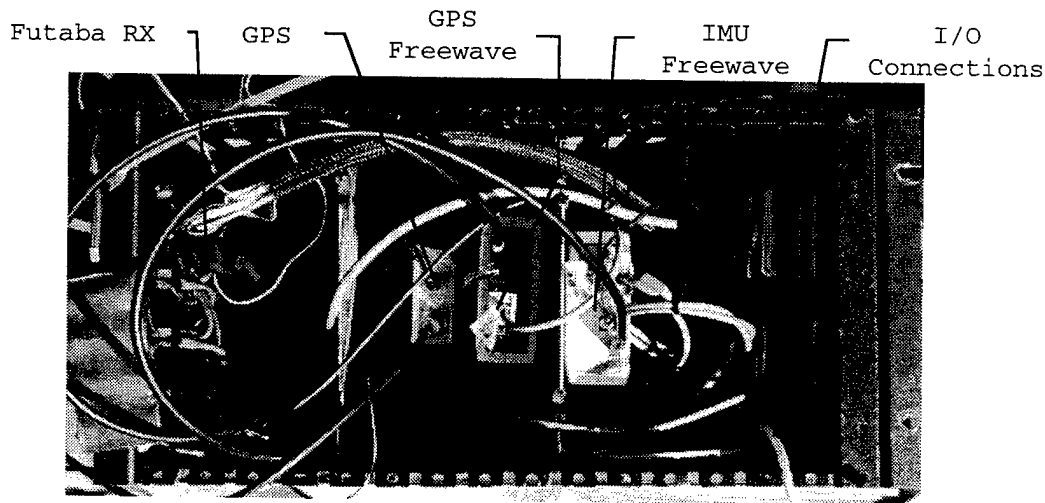


Figure 12. Comm Box

(1) **DGR-115 Spread Spectrum RF Modems:** The communications box holds two of these, identical to the ones in the aircraft. They receive the IMU and GPS information from the aircraft and pass it to the C30 via the IP_Serial board #1.

(2) **Motorola Encore GPS:** Identical to the GPS in the aircraft, this receiver is used to provide differential corrections to the aircraft. The position of the ground station GPS antenna has been surveyed and is at a known latitude and longitude. Use of the differential mode of the GPS system improves position accuracy from on the order of 10 meters to on the order of 1 meter. The ground station GPS is connected to the GPS Freewave via serial cable. For an in depth discussion of the use of DGPS with the FROG aircraft, see [Ref. 3].

(3) **Futaba PWM Receiver:** This receiver, identical to the one in the aircraft, is used to record the actual PWM commands being transmitted to the FROG. It is connected to the IP_68332 module.

(4) **Modified Futaba Transmitter:** A rewired R/C controller that takes input from the C30 via the IP_DAC board and allows computer control of the UAV with standard Futaba PWM signals. It is connected to the Comm Box by a standard 9 pin cable. The transmitter is also connected by cable to the UAV pilot's Futaba transmitter and operates as a slave in a trainer mode. This mode, developed to train novice R/C pilots, allows the slave transmitter to command the aircraft only while the UAV pilot holds the Trainer Switch engaged. This setup enhances safety of flight by allowing the UAV pilot to instantly take command of the aircraft.

(5) **Voice Pigtail:** Provides connection point for Voice Control System detailed in Chapter IV. It is connected to the "A" side of IP_Serial board #2.

(6) **Antenna Array:** Provides two helix antennas, one for each RF modem, and a "puck" antenna for the DGPS station (Figure 13). Connected to the communication box by coaxial cables.

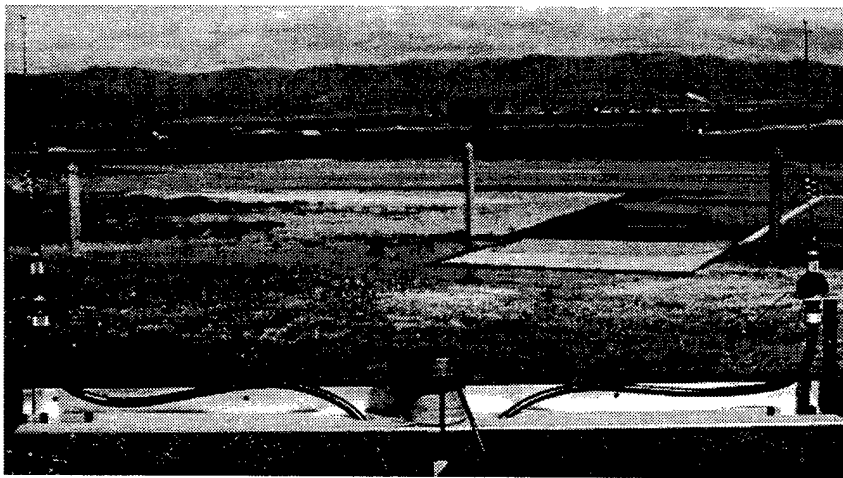


Figure 13. Antenna Array

III. AIRSPEED CONTROLLER

This chapter details the design and implementation of an airspeed controller for the FROG UAV utilizing the RPS discussed in the previous chapters. The need for an Airspeed Controller became evident during this voice control and other FROG projects. Earlier versions of the FROG control algorithms allowed lateral and/or longitudinal control of the aircraft by the ground station in a manual or an autonomous mode; however, the throttle channel, and thus airspeed control, was left under the control of the UAV pilot. In order to build a control system capable of truly autonomous flight and landing, manipulation of all four of the basic inputs for controlled flight, changes in aileron (δ_a), elevator (δ_e), rudder (δ_r), and throttle (δ_t), is necessary. Control of the first three has been addressed during previous projects. The success of this effort will allow follow on projects to address much more complicated flight management problems.

The design of the Airspeed Controller will follow the systematic approach offered by utilization of the RPS. First, the basic problem and the requirements the controller must meet are identified. Alternative solutions are discussed. Once the most attractive solution is chosen, the basic layout of the controller is designed. This layout is transferred to the Xmath/SystemBuild environment and the completed system tested in simulation. The controller is then tested with hardware in the loop using the other Realsim tools and finally flight tested. As stated earlier, the power of this approach is that the same real-time code that performs the simulation, actually flies the vehicle during flight test.

A. DESIGN APPROACH AND REQUIREMENTS

In order to control the airspeed of the FROG vehicle it is necessary to control the throttle movement of the engine. While it is possible to affect airspeed through elevator

and to a lesser degree aileron and rudder deflections, the most efficient method is by varying throttle position to maintain desired airspeed. This is especially true of the benign flight regime in which the FROG usually flies. The effect of the gentle longitudinal and lateral control changes on airspeed can be completely negated by a properly designed Airspeed Controller.

1. Control Signal Path

The throttle in the FROG's engine is actuated by a Futaba servo. Unlike the control surfaces, the throttle servo is not controlled through the autopilot, so direct control of the throttle is possible. In order to emulate pilot control, a feedback controller that compares actual aircraft speed as measured by the pitot/static system to commanded airspeed was chosen as the design approach. The path the control signals will follow is detailed in Figure 14. For our purposes, design of the airspeed controller deals with the blocks inside the dashed line and is implemented using the rapid prototyping software tools.

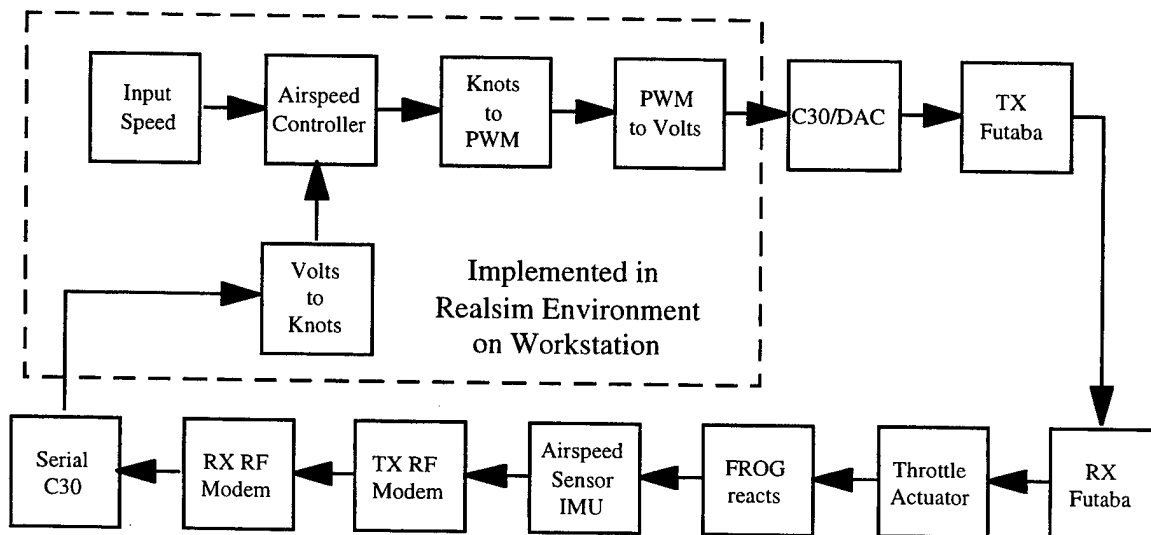


Figure 14. Control Signal Path

2. Airspeed Controller Requirements

Determining just what performance is expected from a controller is a big part of the design process. The Airspeed Controller was designed to meet the following requirements:

a. Seamless Transitions

Controller should not cause a large fluctuation in throttle movement when initially selected or deselected.

b. Zero Steady State Error

Controller should achieve zero steady state tracking error in airspeed in the presence of constant and light variable winds.

c. Bandwidth

The bandwidths for the throttle channel should be wide enough for tight tracking of airspeed, but narrow enough to prevent throttle controller from causing engine stalls.

d. Stability Margins

Gain and Phase Margins for each loop should be greater than 6 dB and 45° respectively.

B. DESIGNING THE CONTROLLER IN THE XMATH/SYSTEMBUILD ENVIRONMENT

The main effort of this project was to design the Airspeed Controller block depicted in Figure 14. The other blocks that complete the overall controller

implementation are mainly for the conversion of inputs or outputs into quantities that are in the proper form for the controller, in the case of inputs, or the hardware, in the case of outputs, to utilize.

1. FROG Controller Overview

The FROG UAV control system is quite complicated. The system provides a myriad of flight management services including navigation and flight control. The upper layer of the SystemBuild model (Figure 15) details the external inputs and outputs to the flight management system. As specified earlier, the inputs come from either the IP I/O modules or are entered via IA interface windows. Outputs are either sent to an IP module or displayed on IA interface windows.

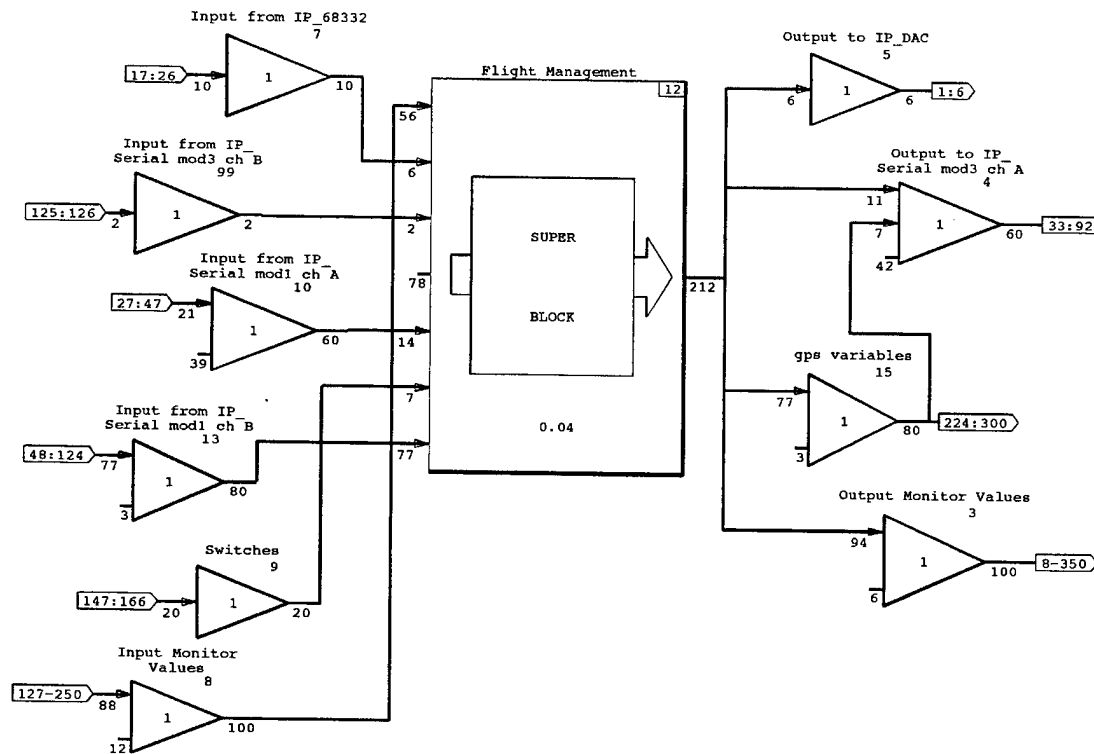


Figure 15. Upper Level of FROG Control Model

Inside the Flight Management SuperBlock (Figure 16) reside the SuperBlocks that perform the main services of the flight management system.

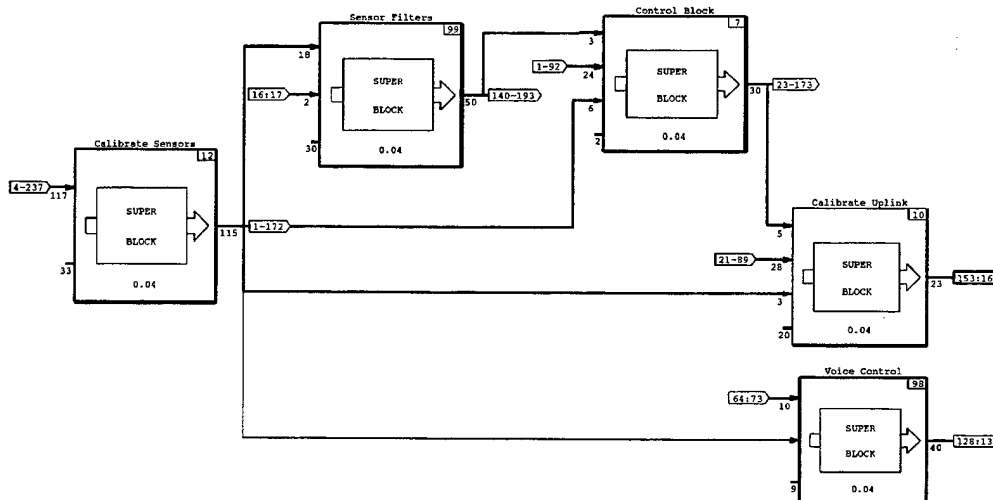


Figure 16. Flight Management SuperBlock

2. Throttle Control SuperBlock Overview

The Throttle Control SuperBlock, located inside the Control SuperBlock, consists of three main parts: a block that freezes the value of certain variables when various switches are engaged; an open loop controller and a closed loop controller. The operator selectable open loop/closed loop (OL/CL) switch allows the choice of either as the primary Airspeed Controller. A limiter prevents the output of either controller from commanding airspeeds less than stall or greater than V_{max} . The implementation of the upper level blocks of the Airspeed Controller are depicted in Figure 17.

3. Current Conditions Hold SuperBlock

The first component of this SuperBlock (Figure 18), records the value of PWM being transmitted to the throttle by the UAV pilot's controller at the moment the Trainer Switch is selected. The frozen PWM value is used as a reference by the Open Loop

Controller for all commanded speed changes during that run. The second block records the value of airspeed at the moment the OL/CL switch is selected “ON”. This value is used as a reference by the Closed Loop Controller. The BlockScript code for these user defined blocks is detailed in Appendix A.

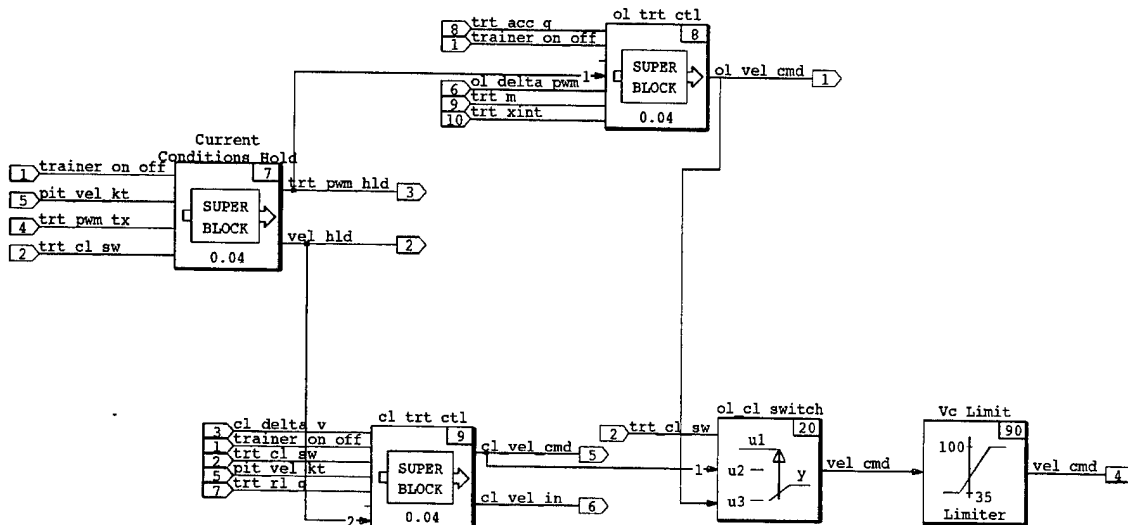


Figure 17. Throttle Control SuperBlock

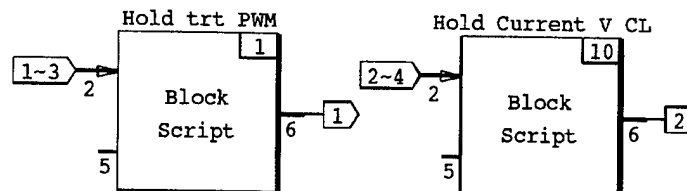


Figure 18. Current Conditions Hold SuperBlock

4. Open Loop Controller

The Open Loop Controller (Figure 19) provides airspeed control by directly manipulating the throttle position. At transition to computer control, the Current Conditions Hold SuperBlock provides a reference PWM to the Open Loop Controller which is used as the initial commanded PWM. This provides for a seamless transition

assuming that initially the change commanded from the IA page (Δ PWM) is set to zero. The commanded PWM, and thus airspeed, is increased or decreased by adjusting the Δ PWM. The rate at which Δ PWM can change is controlled by the limited integrator feedback loop. This rate can be adjusted utilizing the variable acceleration gain. It is important to limit the maximum rate of throttle movement in order to prevent the controller from causing an engine stall while airborne. The maximum safe value of the acceleration gain will be determined during the hardware in the loop testing phase.

User control of Δ PWM and the acceleration gain is provided through the Throttle Control IA window described below. The commanded PWM is converted to open loop velocity commanded using the same inputs for slope and intercept as the Conv to PWM SuperBlock detailed later in this section. The controller outputs open loop velocity commanded directly to the OL/CL Switch. With the switch in the "OFF" position, the open loop velocity commanded becomes the velocity commanded.

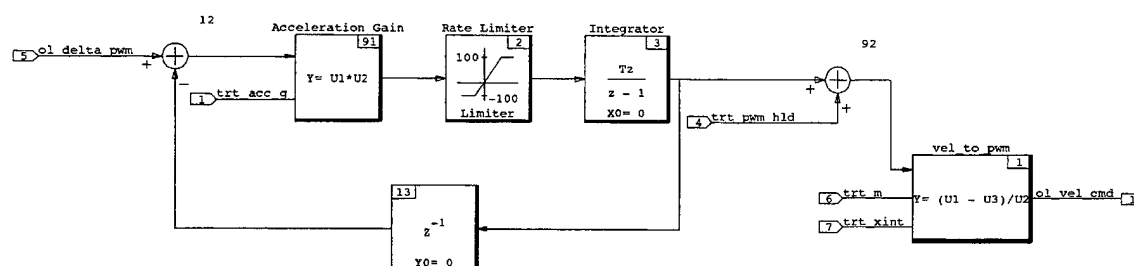


Figure 19. Open Loop Controller Configuration

5. Closed Loop Controller

Unlike the Open Loop Controller, the Closed Loop Controller (Figure 20) utilizes a feedback loop that compares commanded velocity to actual aircraft speed. Think of it as cruise control for the FROG. In order to meet the requirements specified in section A-2 of this chapter, a feedback Proportional/Integral (PI) controller was chosen. The

Closed Loop Controller produces a velocity commanded signal and uses the actual aircraft speed as measured by the pitot system as the feedback signal.

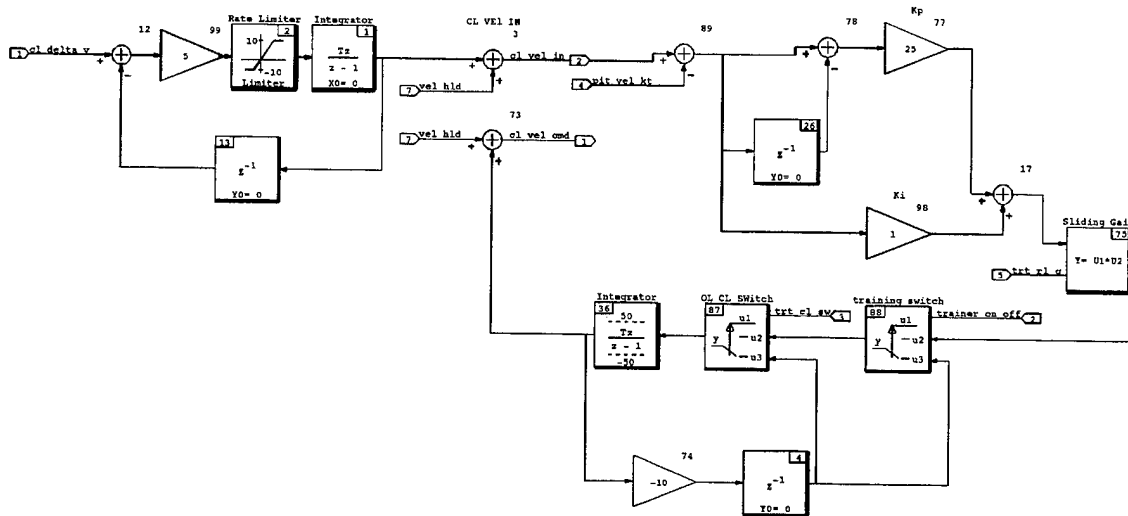


Figure 20. Closed Loop Controller Configuration

The Closed Loop Controller initially commands the velocity captured by the Current Conditions Hold SuperBlock at the time of selection of the OL/CL switch to the “ON” position. This provides for a fairly seamless transition depending on wind velocity. The user increases or decreases the selected speed by adjusting the rate limited ΔV value as described below. The held velocity and commanded delta are added to produce the user commanded velocity input which is displayed on the Throttle IA page. The actual aircraft speed is subtracted from this value to create the error signal on which the controller acts. The controller used was a basic PI controller, where the proportional gain, K_p , was set to one; and the integral gain, K_i , to one as well. The output is sent through a variable throttle gain that allows the operator to control the reaction time of the controller. A limited integrator is used to control the magnitude of the velocity commanded. The 50 knot bound is large enough to allow the entire range of airspeeds which the FROG is capable of obtaining to be commanded.

The two switches are part of the Wind Down Loop. If either or both switches are in the "OFF" position, the integrator is forced back to its initial value. This is done to prevent initiation of the closed loop controller at a previous state. The error signal is added to the reference velocity provided by the Current Conditions Hold SuperBlock. The output of the Closed Loop Controller is sent to the OL/CL switch and becomes the velocity commanded when the switch is selected to the "ON" position.

6. Control Loop Analysis

In order to determine if the controller meets the bandwidth and stability margin requirements, the feedback system must be "broken" at control and sensor loops. The first look is at the control loop as pictured in Figure 21. The system was linearized and a Bode plot generated using the appropriate Xmath functions. Figure 22 is the Bode representation of the control loop and shows a gain margin of 12 dB, which meets the design requirement of 6 dB. Unfortunately it also shows a phase margin of 22° which is below the 45° requirement detailed earlier. Further testing showed that a Throttle Gain of .3 produced the required phase margin, but unacceptably sluggish performance. It was decided to leave the Throttle Gain at one and accept the reduced phase margin.

Figure 22 also shows a bandwidth of .35 radians per second (rad/sec). This is not surprising since the CL Controller was designed to operate at low frequencies in order not to cause engine stalls by moving the throttle too quickly.

7. Sensor Loop Analysis

The sensor loop analysis (Figure 23) produced results very similar to the control loop. Figure 24 shows the sensor loop produces the same gain and phase margins as well as bandwidth. The gain margin and bandwidth requirements are met and give adequate

response times for aircraft reaction to throttle changes. Once again the phase margin is below the 45° that the requirements outlined earlier specified.

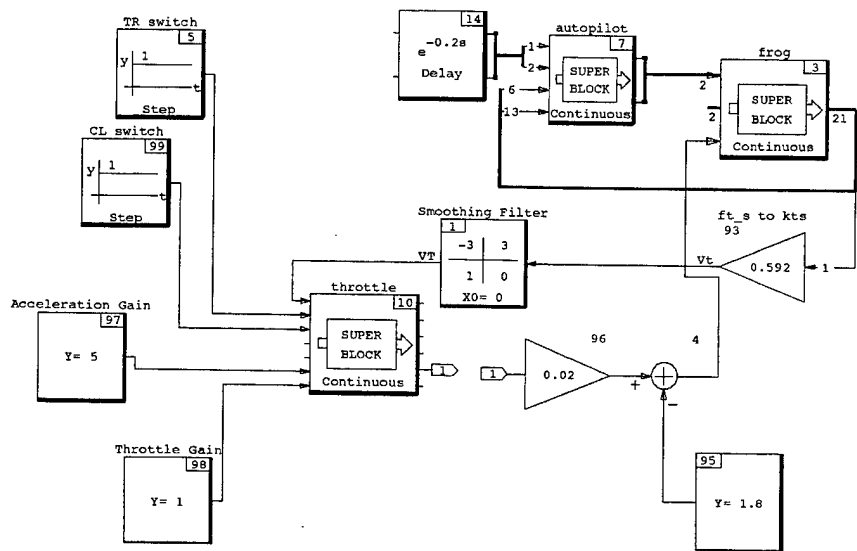


Figure 21. Control Loop

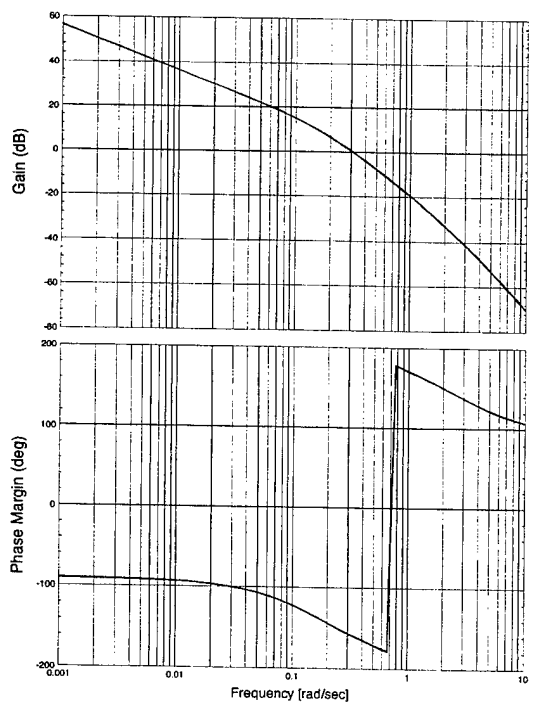


Figure 22. Control Loop Bode Plot

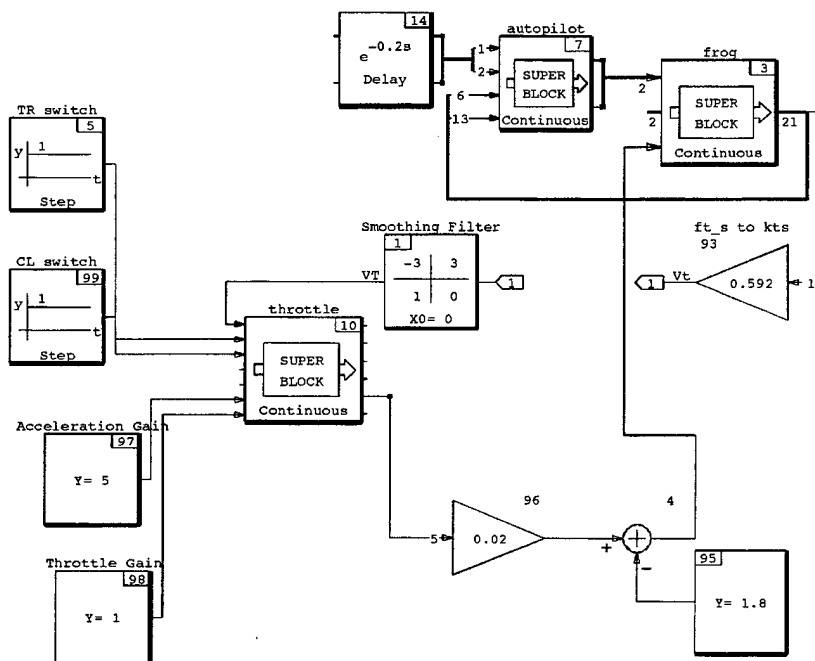


Figure 23. Sensor Loop

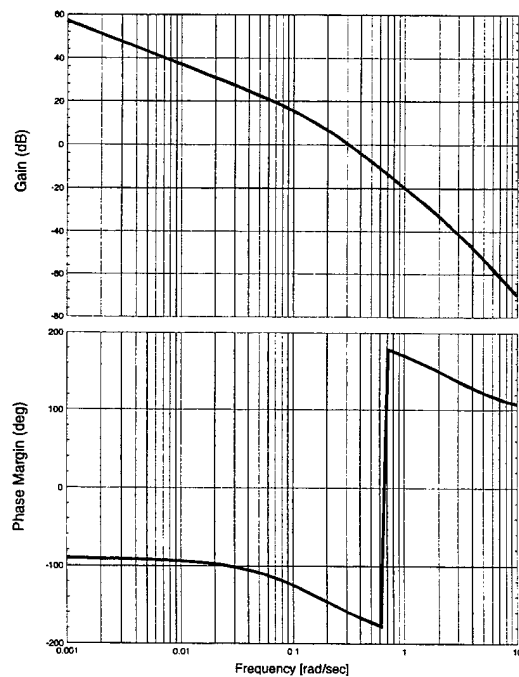


Figure 24. Sensor Loop Bode Plot

8. State Analysis

Whenever one designs a control system it is important to analyze the effects of all the possible states of that system. The Airspeed Controller, with its two switches, has a total of four possible states. The states and resulting actions of the controller are summarized in Table 2.

State	TR Switch Position	OL/CL Switch Position	Throttle CMD Source
0	OFF	OFF	Pilot's TX
1	OFF	ON	Pilot's TX
2	ON	OFF	OL Controller
3	ON	ON	CL Controller

Table 2. Airspeed Controller State Analysis

C. AIRSPEED CONTROLLER IMPLEMENTATION

Once the basic Airspeed Controller design was finalized, it was integrated into the FROG Flight Management SystemBuild model. In order to accomplish this, several additional SuperBlocks that convert inputs or outputs into quantities that are in the proper form for the controller, in the case of inputs, or the hardware, in the case of outputs, to utilize were added.

1. Volts to Airspeed Conversion

The Airspeed Conversions SuperBlock (Figure 25) is located inside the Calibrate Sensors SuperBlock of the FROG SystemBuild model. The external input word1_imu, the sampled voltage output of the aircraft's pitot system, is converted to indicated airspeed in nautical miles per hour (knots). Additionally, GPS measured velocity is converted from meters per second to knots. Knots are used because this measure of vehicle speed is most familiar to the officer/students involved in the various RPS

projects. The 6th order conversion approximation from volts to airspeed was obtained by Evangelos Papageorgiou during his development of a dynamic model of the FROG UAV [Ref. 5]. The output of this block is the pitot velocity input to the Current Conditions Hold SuperBlock and is also used as the feedback signal in the Closed Loop Controller. Both pitot velocity and GPS velocity are displayed on the Throttle Command IA page and recorded for further analysis.

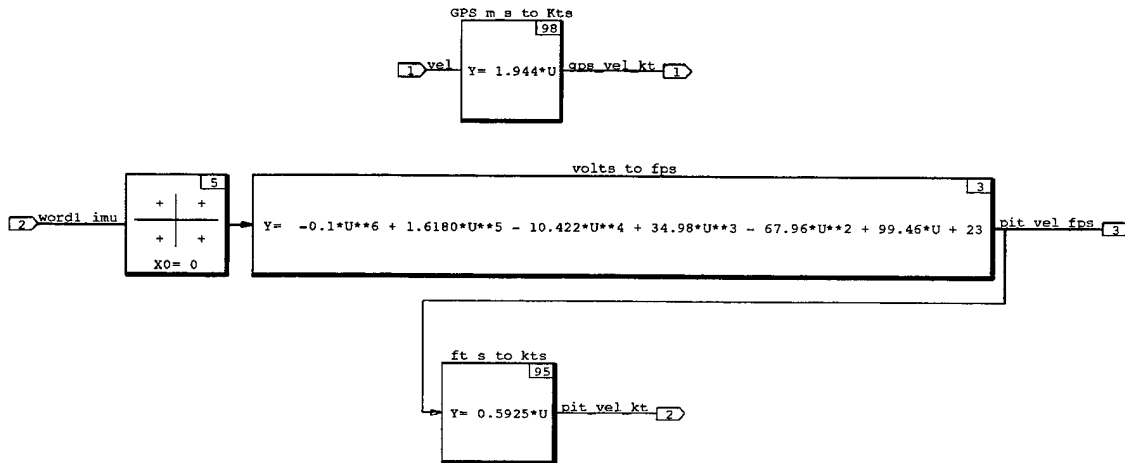


Figure 25. Airspeed Conversions SuperBlock

Due to the excessive noise associated with the pitot sensor, smoothing of the word1_imu data is necessary before conversion to airspeed. Word1_imu data recorded during a previous flight test (Figure 26A) was used to design the smoothing filter. A FFT was performed on the data in Xmath to determine the frequency content of the signal to pin point the noise contribution. A state space representation of a lowpass filter was loaded into SystemBuild and a simulation performed using the actual pitot sensor data as input. By trail and error, a filter with the following transfer function was determined to be sufficient to provide the required suppression of the high frequency noise:

$$T_{VK}(s) = \frac{s}{s + 3}$$

Figure 26B illustrates the smoothing effect the filter has on the word1_imu input signal. The output of the Airspeed Conversions SuperBlock, aircraft velocity measured in knots, is shown in Figure 26C.

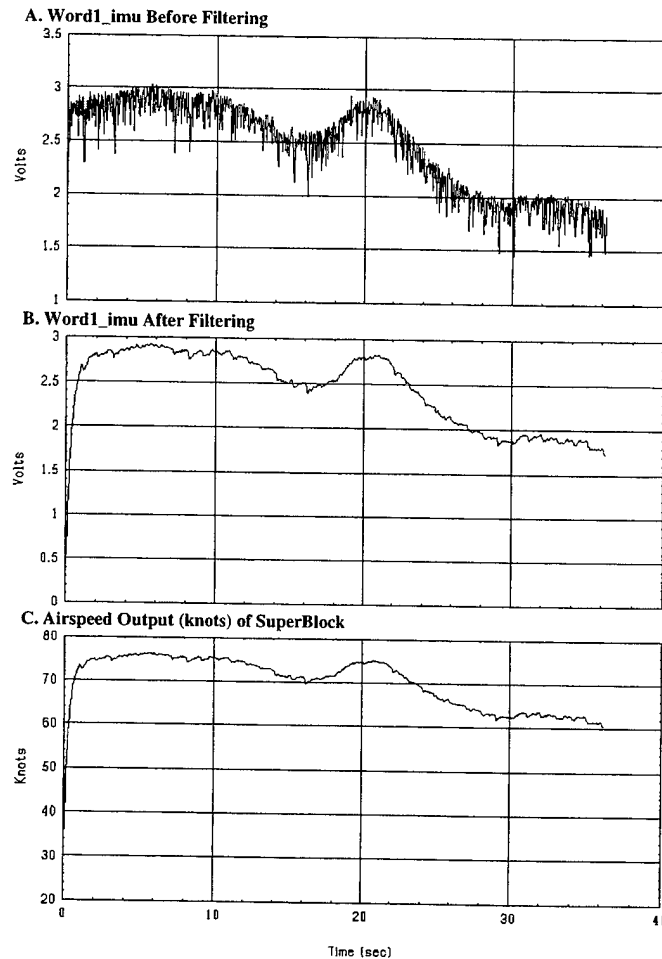


Figure 26. Airspeed Filtering

2. Airspeed to PWM Conversion

The velocity commanded output from the Throttle Control SuperBlock, whether open loop or closed loop, must be converted into a form that can be used by the hardware portion of the RPS. As explained in an earlier section, the aircraft accepts PWM control signals from the computer via a modified Futaba transmitter that accepts a voltage signal

supplied by the C30/IP_DAC as an input. Earlier projects that addressed control of the elevator and ailerons also faced this problem [Ref. 2]. The conversion from velocity commanded to throttle voltage commanded is performed using a two step process. Both conversions occur within the Calibrate Uplink SuperBlock. The first step is to convert the velocity commanded to PWM commanded. This is done by the Conv to PWM SuperBlock; a portion of which is illustrated in Figure 27. The conversion to PWM is a linear approximation, the parameters of which are entered by the user in an IA interface, the applicable part of which is also depicted in Figure 27. The values entered for the conversion were determined by analyzing data from several flight tests; comparing steady state velocity as measured by the aircraft's pitot/static system to controller PWM commanded. The output is passed through a limiter to prevent the controller from commanding a PWM that is too high or low for safe flight.

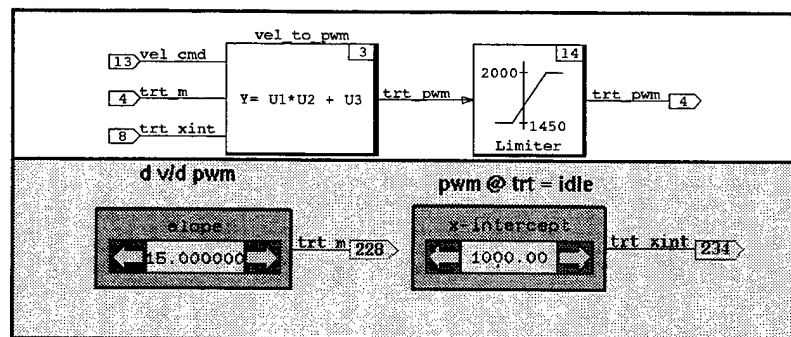


Figure 27. Knots to PWM Conversion

3. PWM to Volts Conversion

The PWM to Volts SuperBlock completes the conversion of commanded airspeed to volts. Also located inside the Calibrate Uplink SuperBlock, this block uses a linear approximation to convert PWM to Volts (Figure 28). This conversion is obtained prior to each flight during a calibration process using the Calibrate DAC IA interface window. With the Trainer and Cal Mode Switches engaged, the commanded voltage on the dial

input shown in Figure 28 is sent directly to the modified Futaba transmitter where it is transmitted as a PWM command. The ground station Futaba receiver receives this command and it is sent to the IA window via the IP_68332 board. The user sets the PWM values for 2.4 and 2.7 volts commanded into the sliders on the IA interface and the calibration is complete. The process for converting the velocity commanded signal to a volts commanded signal is now finished. Once converted, the throttle voltage commanded is sent to the modified Futaba transmitter through the IP_DAC module.

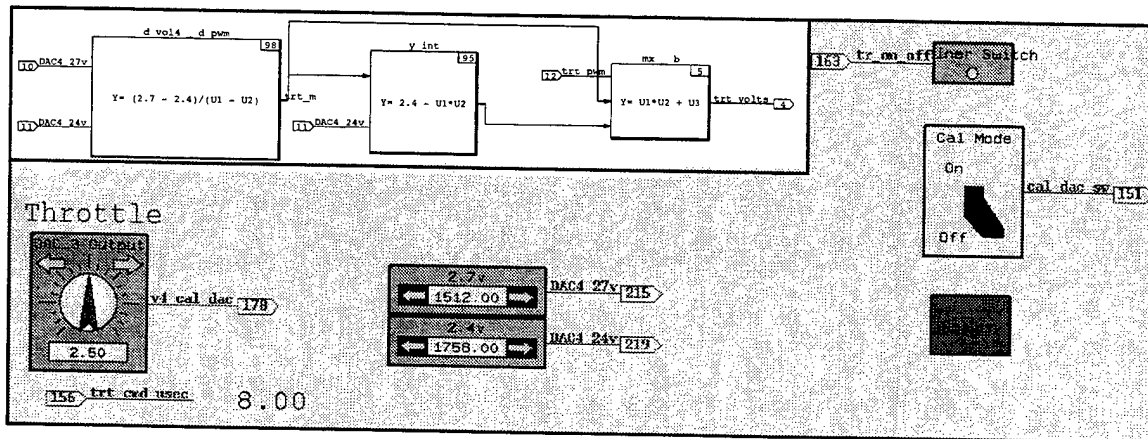


Figure 28. PWM to Volts Conversion

4. User Interface

Now that an Airspeed Controller has been realized in the SystemBuild environment, a means to provide inputs to the controller and observe its operation must be created. As described earlier, the MATRIX_x Product Family provides an extremely powerful tool for creating interactive interfaces with SystemBuild control models. The Airspeed Controller IA window or Throttle Control page is shown in Figure 29.

This page was created specifically for flight testing and displays many values neither desired or required on an operational flight display. The right side of the display contains the user input to the Airspeed Controller. This includes the input deltas in PWM (OL) and knots (CL), the CL airspeed input display, the acceleration and throttle gains,

and the OL/CL Switch. Also displayed is the airspeed of the aircraft as measured by both the GPS and pitot/static systems. Once testing is complete, these controls and indications will be incorporated onto the main flight display page. The left side of the IA window displays several values that will be indispensable in determining whether or not the controller is performing correctly during hardware in the loop and flight testing. These values include the velocity and PWM values held; controller and actual Futaba commanded PWM; and the commanded speed of each controller. By observing and comparing certain values, great insight into controller operation can be obtained. For example, the controller and actual Futaba transmitter PWM commands would be equal if the conversion blocks inside the SystemBuild models were exact. By analyzing these values, the user can make a judgment about the quality of the system calibration.

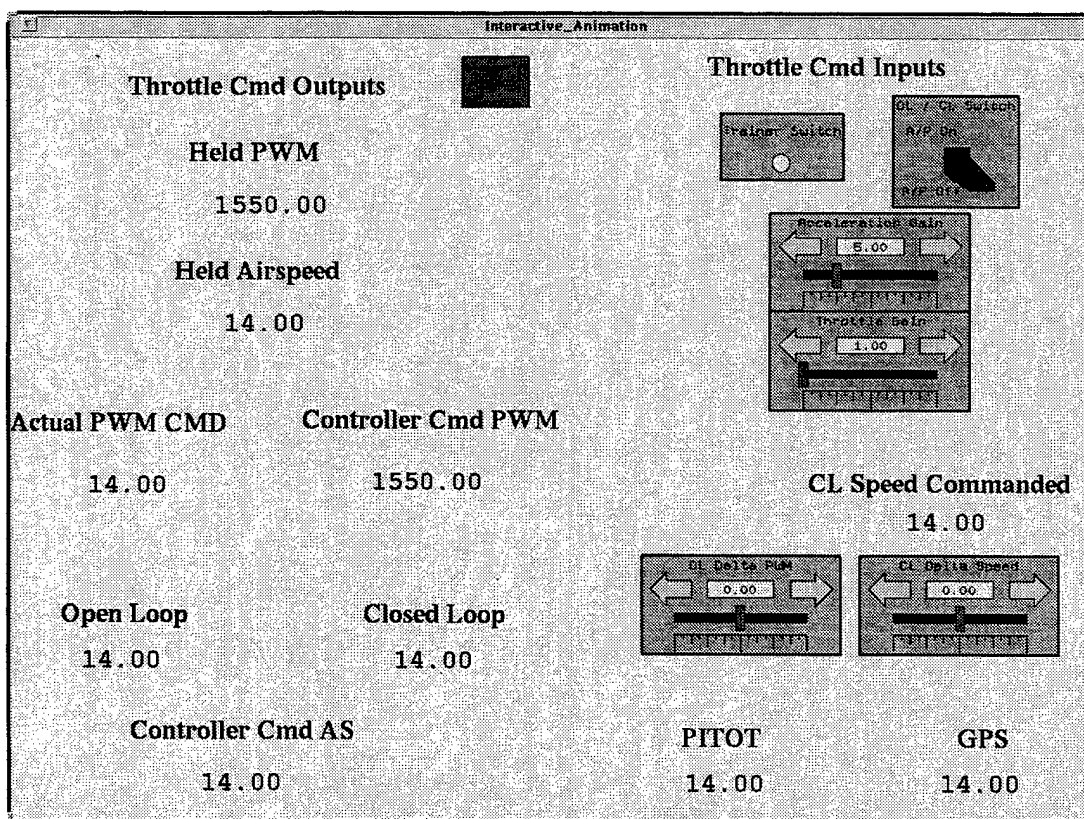


Figure 29. Airspeed Controller IA Interface Page

D. SIMULATION AND TESTING IN THE XMATH/SYSTEMBUILD ENVIRONMENT

The MATRIX_X Product Family provides extensive simulation and analysis tools in the Xmath/SystemBuild design environment. These tools can be used to determine whether or not a controller design meets the performance requirements the engineer set out to fulfill. Several tests were performed on the Airspeed Controller to ensure that each of the requirements defined earlier were satisfied by this design. For all tests, a SystemBuild model of the FROG aircraft and autopilot system developed by Professor Isaac Kaminer was used. This simulation models the FROG UAV in trimmed flight at 88 feet per sec (ft/s). The Airspeed Controller was integrated into the simulation.

1. Simulation Runs

The requirements for seamless transitions and zero tracking error in steady state were investigated by operating the controller in simulation. The Airspeed Controller/FROG model system were configured as shown in Figure 30. All Airspeed Controller inputs normally provided from the IA screen were configured to accept the output from SystemBuild blocks. The parameters entered into these blocks were changed to produce the desired simulation. The SystemBuild simulation window was set to record and graph the values of FROG Velocity, OL Controller Velocity Commanded, CL Velocity Inputted, CL Controller Velocity Commanded and Controller Velocity Commanded. Figure 30 represents a configuration with a 10 feet per second (ft/s) headwind and a .2 second actuator delay added to the simulation to explore the disturbance rejection and robustness of the controller. The first set of runs were performed without the delay and with the wind block disconnected. The results of all of the simulation runs are presented in Appendix B. For all runs, the Trainer Switch was engaged at the 20 second point; allowing the system to get "airborne" and settle into

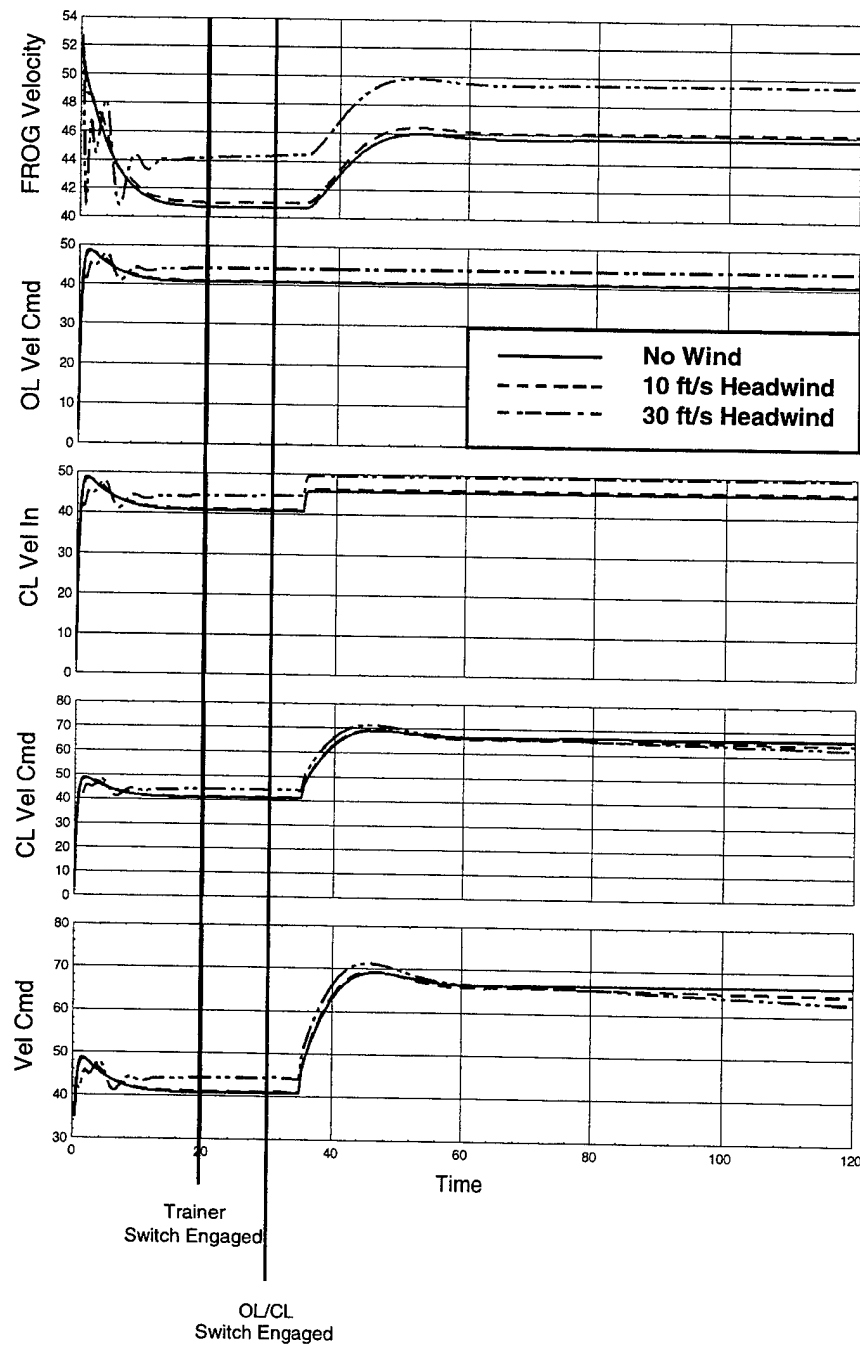


Figure 31. Simulation Results

For all three runs in Figure 31, the Trainer Switch was engaged at 20 seconds and the OL/CL Switch at 30 seconds. The commanded velocity was held constant until the

35 second point when it was increased by five knots. In all three cases the CL Controller reacts properly and drives the aircraft's airspeed to the user commanded velocity and holds it there.

E. HARDWARE IN THE LOOP TESTING

Hardware in the loop testing of the Airspeed Controller was performed using the actual FROG UAV. The tests were performed at the NPS Aero Department's Laboratory facilities located on the grounds of the Navy Golf Course. The facilities include the UAV Lab where both testing and routine maintenance of the NPS Aero Department's fleet of unmanned aerial flight vehicles takes place; and the Blockhouse, where all tests and maintenance requiring the engine to be running take place. For the hardware in the loop testing, the base station equipment was set up in the engine test Blockhouse as it would be out at the airfield for a flight test. The FROG UAV was secured just outside the Blockhouses' open garage style doors. It is important to note that this was not true hardware in the loop testing since we did not measure the actual throttle movement; but rather the PWM commanded. The first step was to calibrate the throttle in PWM vs. volts. The controller was started and the CAL DAC IA page selected. To enable the calibration mode, both the Trainer Switch and the Cal Switch must be selected to the ON position. The one difference in throttle calibration is that because the throttle does not use the autopilot, the PWM commanded by the pilot's controller at selection of the Trainer Switch affects the calibration values. In order to circumvent this problem, throttle calibration is always initiated with the pilot commanding 1650 PWM, the approximate value that will be commanded by the pilot at control transition.

In order to provide an airspeed indication for the Airspeed Controller, a differential pressure unit nicknamed the "Schmidter" was connected to the pitot probe of the aircraft. With the controller running on the base station, the pressure supplied by the

Schmidter was increased until the airspeed indicated 50 knots on the IA screen readout. With the engine running, the UAV pilot selected the Trainer Switch, passing control of the throttle to the computer operator. The Open Loop Controller was tested by selecting several different values for the Δ PWM input and observing reaction of the engine throttle and RPM. The acceleration gain was raised and throttle slams (rapidly advancing the throttle from minimum to maximum and vice versa) were performed to determine the maximum safe rate of throttle movement. This test is important to prevent the Airspeed Controller from causing an engine stall airborne.

Testing continued with the selection of the Closed Loop Controller. With the Schmidter providing a constant apparent airspeed of 50 knots, the throttle controller either drove the throttle to the maximum, if the commanded airspeed was above 50 knots, or the minimum, if the commanded airspeed was below 50 knots. The controller was also tested by commanding 50 knots and varying the pressure supplied to the pitot tube by adjusting the Schmidter. In each case, the Closed Loop Controller performed as expected, driving the throttle to one limit or the other. The controller gain was increased, again to determine the maximum safe rate of throttle movement. Before returning control to the OL controller, the inputted airspeed delta was set to zero. This was done to prevent a jump in throttle position during the transition from CL to OL control. The same procedure is followed when returning control to the UAV pilot. A test was performed to determine the effect of mismatched PWM during computer to pilot transition, in case it was necessary for the pilot to unexpectedly resume control during the flight test. It was determined that the mismatch and resulting PWM jump would not cause an engine stall.

The hardware in the loop testing was invaluable in determining that the Airspeed Controller was operating properly, and just as importantly, that it was safe to flight test. A value of 5 for the OL acceleration gain and a value of 1 for the CL throttle gain resulted in a good performance with no engine stalls. Hardware in the loop testing also

gave the UAV pilot and computer operator a chance to practice and coordinate control transfers. A major advantage of the RPS was evident, namely that the exact equipment and code utilized for the hardware in the loop testing would perform the flight test.

F. FLIGHT TESTING

Flight testing of the Airspeed Controller on the FROG UAV was conducted at the Salinas R/C Modelers Club airfield located near Chualar, California. The airfield features a 300' asphalt strip positioned in a north/south orientation. The entire RPS was packed up, moved and reassembled in accordance with the procedures in Appendix C. For the test flights, the pilot's Futaba controller was programmed to relinquish control of the throttle channel only with selection of the Trainer Switch. The pilot was instructed to fly a race track pattern while the airspeed was under control of the real-time processor. The Data Acquisition Editor was set up to record the same variables that were recorded during simulation with the addition of the Trainer and OL/CL Switch positions and PWM transmitted.

The weather conditions were good with the exception of a strong northerly wind. A plot of the aircraft's indicated airspeed (IAS) as measured by the pitot system and the ground speed (GS) as measured by the GPS (Figure 32) shows a wind varying between fifteen and twenty knots at the test altitude.

1. Transition Tests Results

Two four minute test runs were performed and recorded. The first item tested was the seamlessness of the transitions between pilot and OL control; and between OL and CL control. From the ground the pilot/OL transition was not detectable by listening to the engine. The transition from OL to CL was accompanied by a detectable throttle movement; although, not a drastic one. This is to be expected, especially on a windy day,

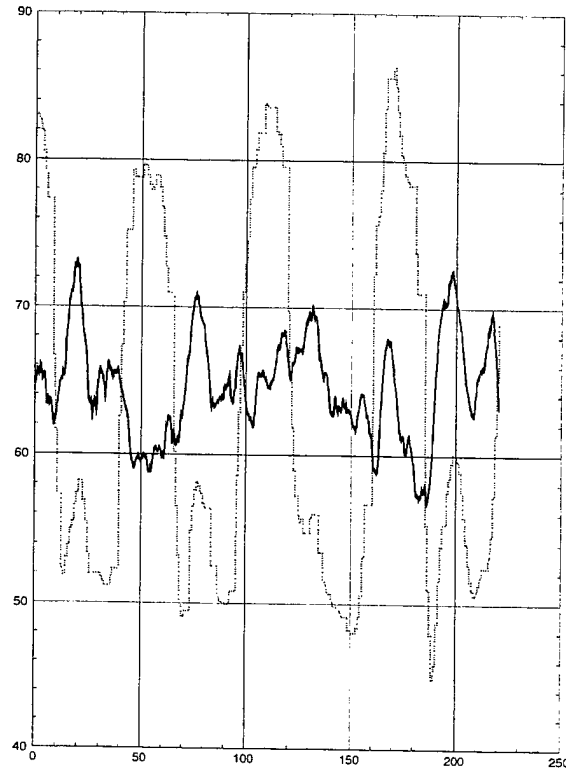


Figure 32. IAS/GS Comparison

due to the fact that airspeed and not PWM is captured and used to produce the initial controller command. Figure 33 details the actual magnitude of the PWM jumps for test Run #1. In all cases the engine responded to the throttle commands without hesitation.

2. Airspeed Tracking Tests Results

The main point of the flight test was to determine how well the Airspeed Controller tracks commanded speed inputs. Once the CL Controller was selected, the aircraft's reaction to speed inputs was observed and recorded. From the ground station, it appeared the controller did a good job keeping the aircraft at the desired airspeed despite the windy conditions. Figure 34 details the results of test Run #1. As designed, the Closed Loop Controller freezes the current aircraft speed at the selection of the OL/CL Switch to the "ON" position. The initial commanded input speed is 67 knots. The

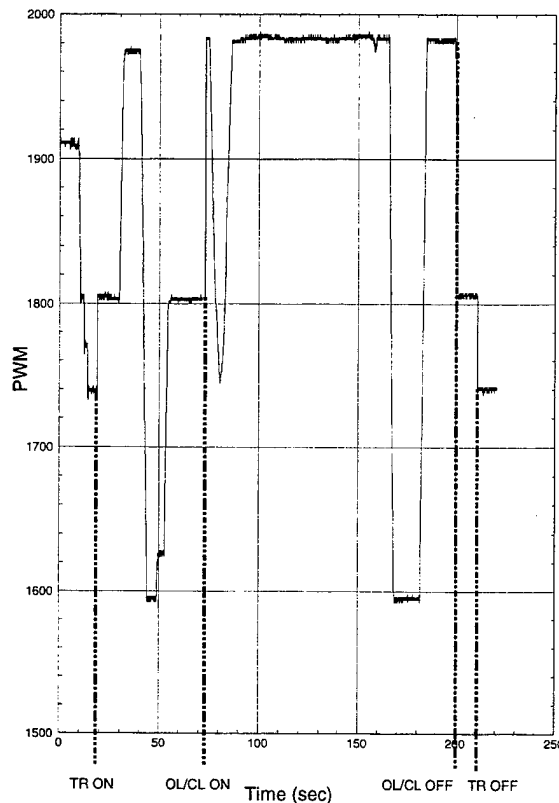


Figure 33. PWM Transmitted

controller attempted to hold this speed as the FROG flew in a circle, turning the headwind into a tailwind and back again. At the 70 second point, the commanded airspeed was reduced by five knots to 62 knots via the Throttle Control IA page. The Airspeed Controller reacted properly by reducing aircraft speed. The controller responded correctly to all further commands during this run.

The results of the Run #2 (Figure 35) confirm the findings of Run #1. However, this run highlights one of the shortcomings of the FROG. During the period between 80 and 120 seconds, the aircraft was flying downwind with a 18 knot tailwind and despite the fact the controller was commanding full throttle, the aircraft could not attain the commanded indicated airspeed of 77 knots. This is not a fault in the controller design; but rather, a result of the limited performance of the FROG aircraft with respect to the range of airspeeds it is able to achieve.

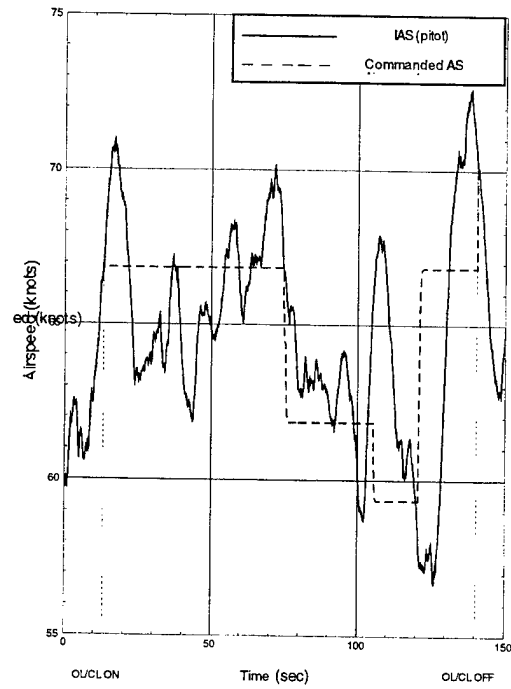


Figure 34. Run #1

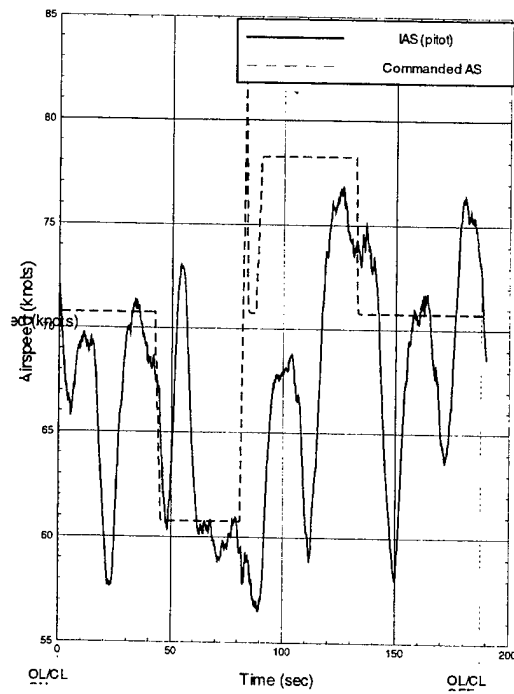


Figure 35. Run #2

3. Summary of Flight Testing

The Airspeed Controller design proved to be very successful at maintaining a commanded aircraft speed. The controller showed good disturbance (wind) rejection up to the limits of the aircraft. The design demonstrated good transition qualities between states. With the values for acceleration and throttle gains determined during hardware in the loop testing, the design exhibited smooth yet quick control of throttle movement at rates that did not cause any adverse affects to engine performance.

It was determined that the input to the Open Loop Controller should be changed to accept a ΔV vice a ΔPWM . This change was implemented and successfully tested on a later flight. It also permits the controller to be used as the inner loop on a more complex trajectory management system.

IV. VOICE CONTROL

A. BACKGROUND

The NPS Aero Department got involved in the Voice Control Project through the Navy's Maritime Avionics Subsystem Technology (MAST) Program. After a presentation on the capabilities of the ViA Wearable Computer system and voice recognition software, Professor Isaac Kaminer offered the FROG UAV as a possible candidate for integration with that system. The project proposed that the ViA system be used to control the FROG in flight. During FY 98, a Cooperative Research and Development Agreement (CRADA) was reached between ViA and NPS.

B. VOICE CONTROL HARDWARE AND SOFTWARE

The Voice Control System (VCS) is comprised principally of commercially available components. In addition to the standard RPS, the VCS requires the use of a laptop computer to translate commands between the ViA Wearable and the Sun workstation. The VCS software programs that run on the laptop and wearable computers are the only custom made components of the system. Figure 36 provides an overview of the RPS including the additional equipment associated with the VCS.

1. Interfacing with the RPS

In order to allow an external computer access to the RPS real-time controller, a method for communication between the ViA Wearable and the Realsim software running on the Sun workstation was necessary. The solution selected involved connecting a standard laptop computer running special software to an I/O module on the DSP_FLEX board.

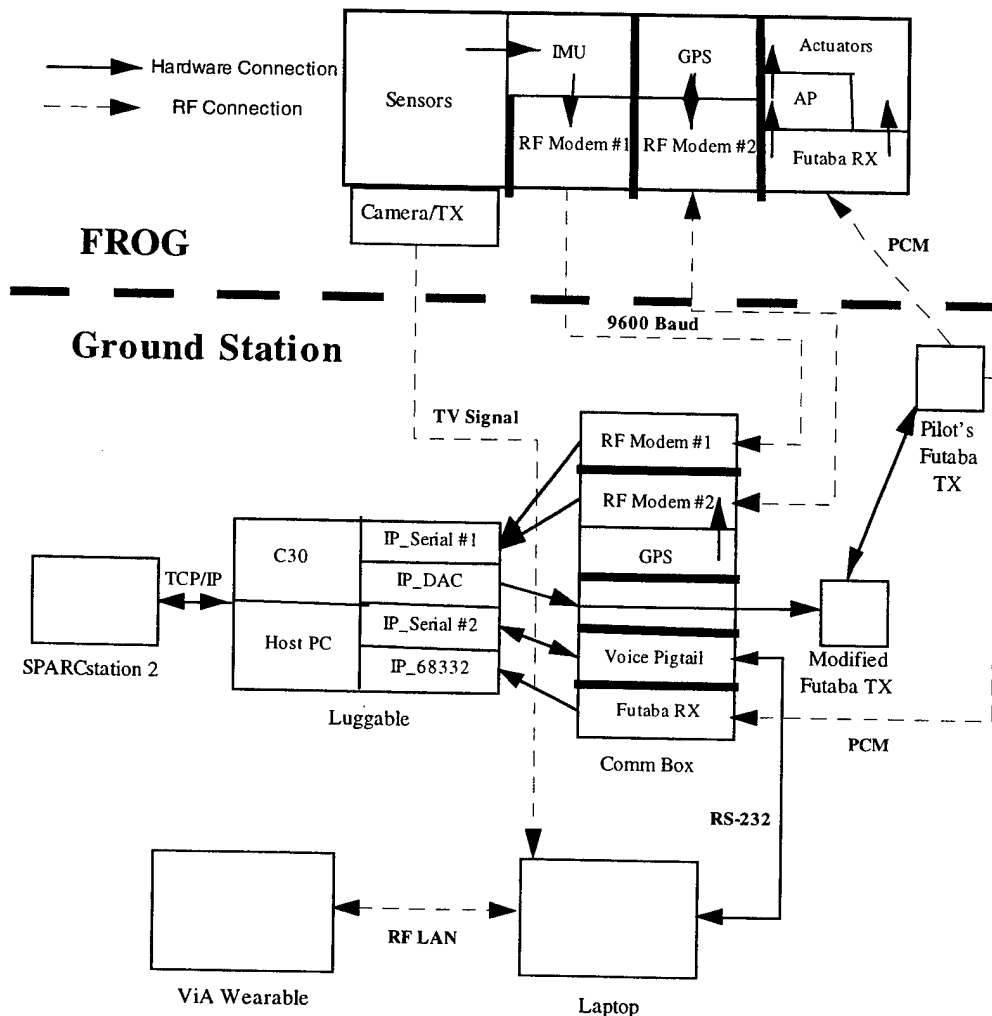


Figure 36. VCS Overview

The original version of the software that allowed commands to be entered via the laptop was written by Luis Sebastiao, a visiting student from Portugal. His program is called PC-serial-C30 (PC_S_C30) [Ref. 6]. The program requires that the laptop be connected to the "A" side of the IP_Serial #2 module via a null modem serial cable. PC_S_C30 runs under Windows 95 and uses a full duplex serial link. Lateral (aileron) and longitudinal (elevator) commands from the laptop are sent via into the IP_Serial module for use by the standard FROG controllers running on the C30 real-time processor. PC_S_C30 receives Trainer Switch status, GPS local tangent plane (LTP) position

information and 10 user assignable float variables for display on the program window. The controls provided by PC_S_C30 are summarized in Table 3. The basic program was tested in July 1997 and was successful in controlling the FROG aircraft from the laptop keyboard.

KEY	COMMAND
↑	Climb
↓	Descend
→	Right Turn
←	Left Turn
L	Level (Constant Altitude)
C	Center (Constant Heading)
Space	Level & Center
Q	Standard Turn Rate
A	Half Turn Rate
W	Standard Climb Rate
S	Half Climb Rate

Table 3. Laptop Control Keys

In order to allow the ViA Wearable Computer to interface with the RPS, the basic PC_S_C30 program was modified by Jennifer Wightman of ViA. The wearable communicates with the laptop by way of a Netware Wireless LAN card inserted into one of the PCMCIA (PC card) slots of the laptop. The Windows 95 display of the current version of PC_S_C30 is shown in Figure 37.

On the workstation side of the connection with the laptop, an IA interface screen was created to allow the user to monitor the commands received by the FROG flight management program. This IA page is illustrated in Figure 38. Besides displaying the direction and magnitude of the laptop inputted command, the IA page shows what, if any, data is being sent for display on the laptop using the 10 float variables built into the

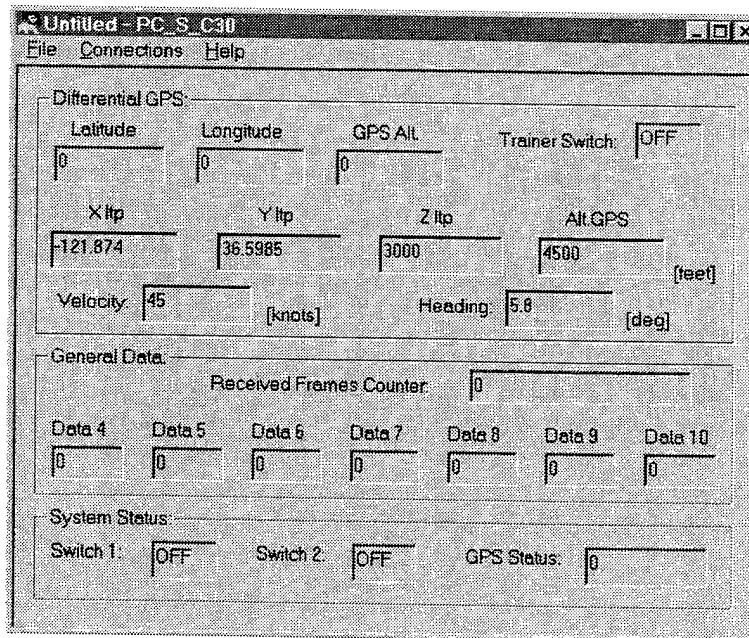


Figure 37. PC_S_C30 Laptop Display Window

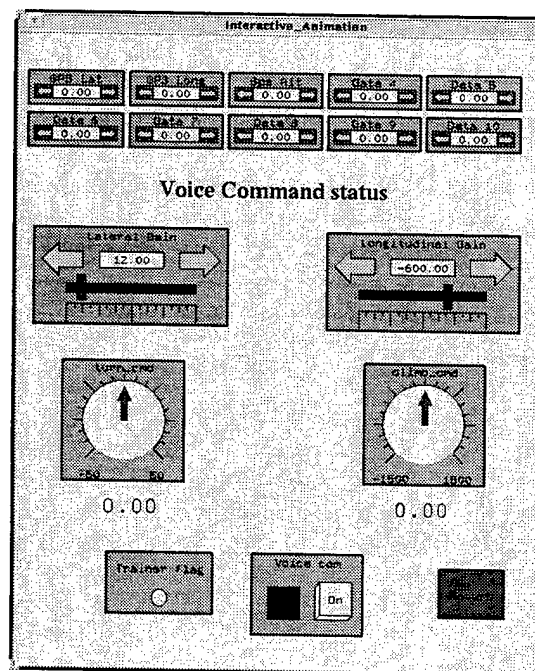


Figure 38. Voice Control IA Interface

PC_S_C30 program. During lab testing it was determined that GPS latitude, longitude and altitude should be displayed, thus variables 1-3 are now used for this information. This page also features the switch that allows the laptop computer to control the FROG aircraft. Two variable gains were added to the system to magnify the incoming lateral and longitudinal commands. This is done with a simple SystemBuild algebraic block as depicted in Figure 39. These gain controls allow the user to set the amount of control surface deflection that standard and half standard rate commands produce. The default values for these gains were determined during flight test.

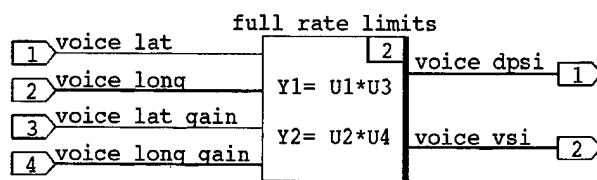


Figure 39. Voice Commands Variable Gain Block

2. ViA Wearable Computer

The ViA Wearable Computer (Figure 40) is a fully functional microprocessor based computer packaged in a highly mobile, physically flexible form. It is capable of running software based on the DOS, Windows 3.11 and Windows 95 operating systems. The system currently used with the VCS consists of three pods, the center of which contains the CPU. The wearable features a AMD 586/133 MHz processor with 24 Mb of DRAM. Like a standard laptop computer, the wearable can be configured for a wide variety of applications through the use of PC cards. The two outer pods each contain 2 PC Card sockets configured as shown in Table 4. One socket is reserved for the system disk which is always a Type-III 340 Mb hard disk card. The system used in conjunction with the VCS has two of these cards. The VCS system also features a Netware Wireless

LAN PC card used to communicate with the laptop computer. The three pod system is worn around the waist in a special belt assembly.

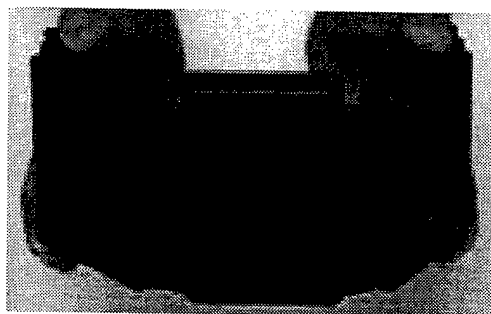


Figure 40. ViA Wearable Computer [Ref. 7]

Socket	PC Card Type	VCS Configuration
0	I or II	Netware Wireless LAN
1	I, II or III	340 Mb HD
2	I or II	Not Used
System Disk	III	340 Mb HD

Table 4. ViA Wearable PC Card Socket Configuration

The VCS wearable is powered by a single Duracell DR-36 NiMH battery that is housed in a pouch worn on the waist belt. The batteries are rechargeable and hot-swappable with the use of two battery pouches or a temporary AC power source.

3. ViA Hand Held Display/Audio Headset

The ViA Wearable system features two methods of entering commands during normal operation. Commands may be entered in the traditional manual method utilizing the touchscreen or with voice via the Audio Headset and installed recognition software. The ViA Hand-Held Display (Figure 41) is a 640 x 480 active matrix LCD panel capable of displaying 256 colors. It features a touch overlay that allows the screen to be used as an input device. The Audio Headset operates with the voice recognition portion of the

wearable software to send commands to the computer and receive back programmed audio responses. Figure 42 illustrates the ViA Wearable being used as a portion of the VCS.

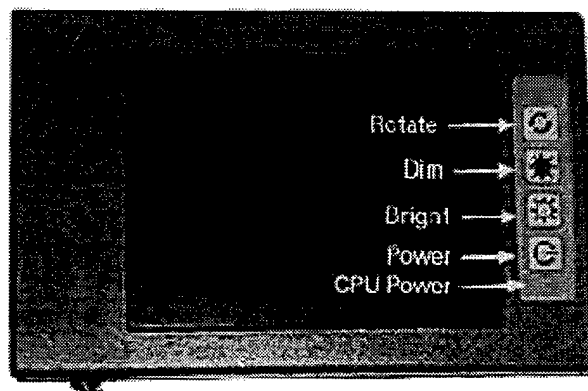


Figure 41. ViA Hand Held Display [Ref. 7]



Figure 42. FROG Voice Control System

4. Wearable Software

Jennifer Wightman of ViA has created a flight management program that is executed under Windows 95 on the wearable computer and is displayed on the hand-held screen (Figure 43). The software allows the user to control the FROG with voice commands, the touch screen, or a combination of both. All major commands have a button on the display to serve as a backup to the primary voice control method of inputting commands. The display presents current flight condition information as the latest voice commands sent to the aircraft. The display features a moving map that shows the position of the aircraft relative to the ground station. The scale of this map is adjustable by tapping on the map display or using the "SCALE MAP" voice command. Several voice commands allow the user to control the aircraft's flight path using yaw and climb rate commands to onboard autopilot. The flight control commands (Table 5) consist of two word phrases to aid the voice recognition.

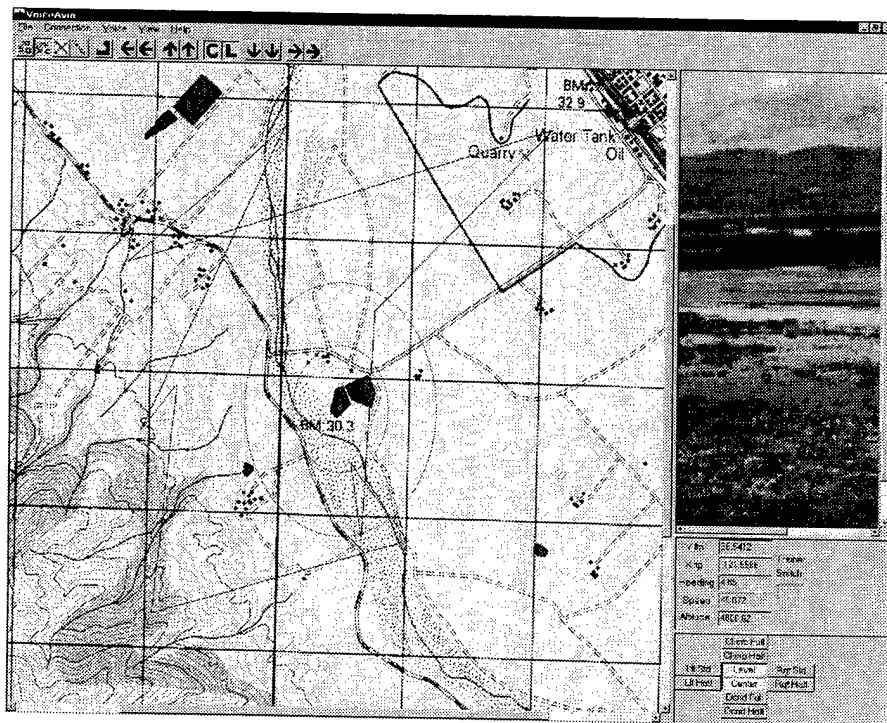


Figure 43. Touchscreen Display

VOICE COMMAND	ACTION COMMANDED
"RIGHT STANDARD"	Standard rate right turn
"RIGHT HALF"	Half rate right turn
"LEFT STANDARD"	Standard rate left turn
"LEFT HALF"	Half rate left turn
"CENTER"	Constant heading
"CLIMB FULL"	Standard rate climb
"CLIMB HALF"	Half rate climb
"DESCEND FULL"	Standard rate descent
"DESCEND HALF"	Half rate descent
"LEVEL"	Constant altitude

Table 5. Flight Control Voice Commands

5. Real-time Video Capture System

An additional feature of the VCS is the ability to view real-time video and capture images from the FROG's onboard camera (Figure 44). The camera, a commercially available Canon ES-970 8 mm Video Camcorder, is mounted in the nose of the aircraft behind a clear Plexiglas shield. Currently the camera is fixed and must be turned on before flight. A future project will allow control of all camera functions from the ground station. The video output of the camera is connected to a small transmitter/antenna assembly mounted on the rear of the FROG's fuselage, just under the tail boom.

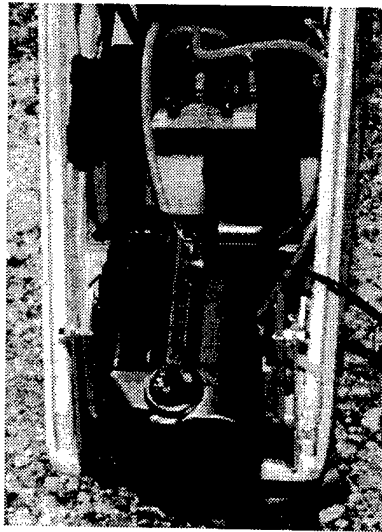


Figure 44. FROG Camera Installation

The television signal from the aircraft is received by the same laptop that is the host for the ViA Wearable. The VCS uses the Notebook TV system manufactured by Nogatech, Inc. of Cupertino, California. This system includes a type II , interactive real-time video and multimedia PC card and an external tuner module capable of receiving 155 UHF and VHF channels. The external tuner module and its available connections are illustrated in Figure 45.

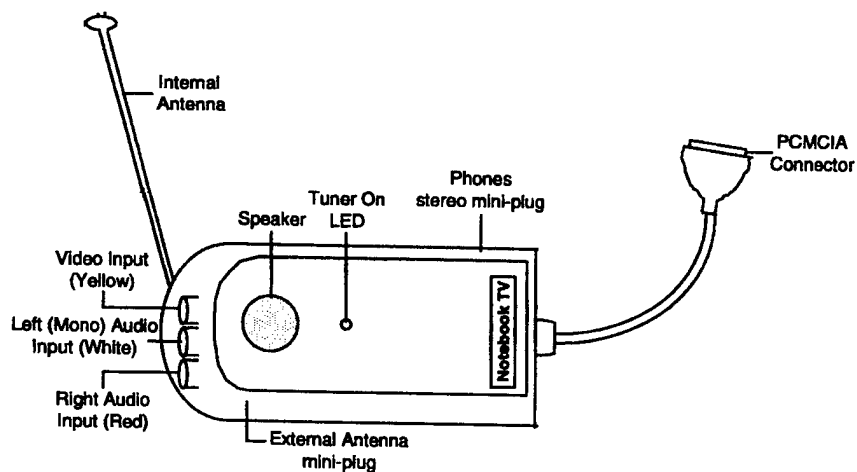


Figure 45. Notebook TV External Tuner [Ref. 8]

The Notebook TV system is capable of displaying full motion video and capturing 30 frames per second at a resolution of 320 x 240 pixels. It is also capable of hi-resolution, 24 bit still image capture at 640 x 480 pixels. The system comes with software; however, ViA has incorporated the Notebook TV drivers into the VCS software. The VCS software provides for viewing and capturing of stills only. Images are captured and displayed on the touchscreen, in the small box on the upper right side as depicted in Figure 43, by issuing the "GRAB IMAGE" command. Once a still is captured, it may be viewed in the larger map display box by issuing the command "VIEW IMAGE", tapping on the image box twice, or depressing the image button icon. The user may save the image by issuing the command "SAVE IMAGE" or by depressing the save icon on the touchscreen. Any number of images may be saved; each is given a sequential number starting at 1. The user may return to the default display by issuing the "VIEW MAP" command, by tapping twice on the displayed image, or deselecting the image pushbutton.

C. HARDWARE IN THE LOOP TESTING

All hardware in loop testing was conducted at the UAV laboratory facility at the NPS Golf Course. The FROG flight vehicle was set up with wings on to allow observation of the control surfaces. The RPS/VCS was assembled and thoroughly tested. Once up and running, all voice and touch screen commands were tested. The flight control commands were issued and the resulting flight control deflection observed. The variable aileron and elevator deflection gains were adjusted to obtain the desired surface deflections. These best guess numbers, relying on the experience of the UAV pilot, served as a starting point for the flight test. The video capture system was tested to ensure the wearable was able to grab, display and save still images. All VCS functions, with the exception of the moving map, were successfully tested in the laboratory setting.

The moving map did not operate as designed because LTP position information was used to attempt display the aircraft's position. The electronic map utilizes standard latitude and longitude measurements. A newer version of the VCS software has fixed this incompatibility.

D. FLIGHT TESTING

Flight testing was conducted the following day at the Chualar airfield. The RPS and VCS components were assembled and the same tests performed in the laboratory were repeated. The only problem was with the GPS tracking; the aircraft symbol did not show up on the map display. This was due to the use of GPS LTP information vice GPS latitude and longitude to fix the aircraft's position. This anomaly will be corrected in a future version of the software.

Weather conditions were good with the exception of a 15 knot wind out of the north. The ambient noise level was high due to the presence of several construction vehicles repairing the El Nino damaged bank of the Salinas River, located 50 yards from the airfield. After launch the aircraft was positioned to fly a north/south race track pattern. Once handed over to the computer, the aircraft was successfully flown for several minutes utilizing voice commands.

1. Flight Test Results

The ViA wearable and voice recognition software performed exceptionally well. Once the user determines the proper microphone placement and speaking volume level, the software recognized almost 100% of voice commands. This result was obtained despite the presence of major construction immediately adjacent to the test area.

The system exhibited a two second delay between the voice command and the command transmission. This is due to the time required by the voice recognition

software to decipher the command. This delay is affected by many factors and is slightly longer when the image capture capability is enabled. The delay required a little practice by the operator to become proficient at anticipating commands and leading turns in order to get the aircraft to fly the proper path.

2. Control Gains

During the initial voice runs, the variable gains for both aileron and elevator commands were adjusted to produce the desired turn and climb rates respectively. It was determined that the standard turn needed to be tighter (smaller radius) thus the gain was increased from 8 to 10, resulting in a 10°/sec turn rate. Figure 46 details the lateral commands and resulting aircraft reaction for one of the test runs. The top graph represents the input command from the VCS before it is augmented by the variable gain. A value of one represents a standard rate right turn; negative one a standard rate left turn. The middle graph is the resulting PWM transmitted. The final graph shows GPS heading and demonstrates that the aircraft does indeed turn in the direction commanded.

Figure 47 details the longitudinal results for the same run. Here, a value of negative one represents a full climb; -.5 represents a half rate climb. It was determined that the initial gain did not provide enough elevator authority in the negative (up) direction. The middle graph of Figure 47 shows a slight increase in PWM as the gain is increased. A final value of -1350 was determined to be adequate. The third graph demonstrates that the aircraft reacted properly to the climb command. It also shows the aircraft did not achieve the 1350 ft/min climb commanded. This result is due to the fact the aircraft was constantly in a turn to stay within the range limits of the Futaba controller and the visual limit of the computer operator. More testing is necessary to determine if a gain value of -1350 corresponds to a climb rate of 1350 ft/min in level flight.

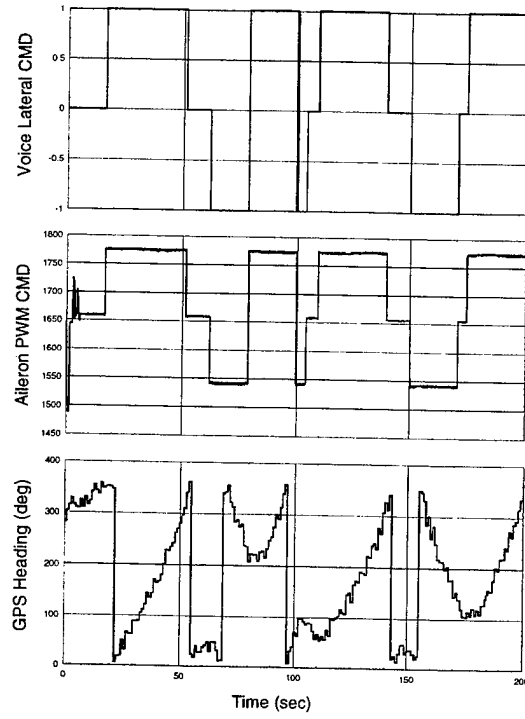


Figure 46. Lateral Voice Commands

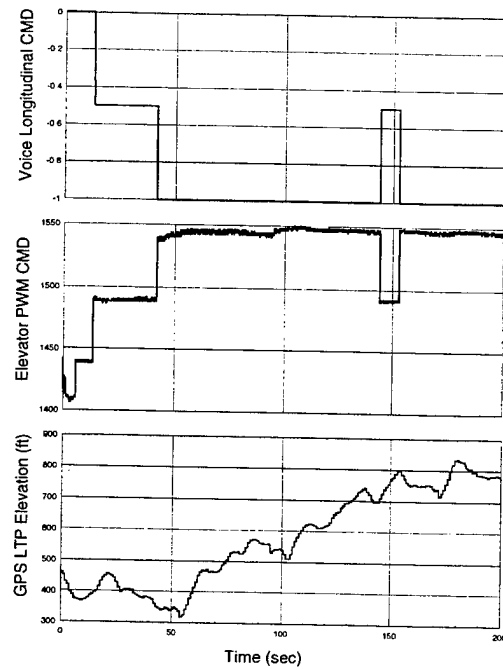


Figure 47. Longitudinal Voice Commands

3. Turn Performance

The FROG aircraft initially tended to descend during voice controlled turns. This was due to the lack of elevator commands to increase the g on the aircraft during the turn. The operator compensated for this by commanding a half rate climb immediately before the turn and level just after the turn.

The control scheme for turns was improved by automatically commanding elevator with the turn commands. This provides the necessary g to allow the aircraft to complete the turn without descending. It also relieves the operator from having to remember to command the elevator before and after each turn.

4. Image Capture

The image capture system worked as designed. The quality of the image was poor due to intermittent reception of the TV signal at the ground station. Further testing is necessary to determine whether this anomaly is due to the TV transmitter or the receive antenna.

5. Summary

Overall, the VCS was very successful in providing control commands to the aircraft. The voice recognition software proved very reliable. The ViA wearable was easy to setup and use. With a little fine tuning, the VCS will become the preferred method for controlling the FROG UAV.

V. CONCLUSIONS AND RECOMMENDATIONS

The projects described in this thesis have demonstrated the viability and versatility of the rapid prototyping concept. The time, resource and money saving potential of rapid prototyping is just beginning to be explored. The NPS RPS has allowed my participation in two extensive control projects from the initial design to the final flight testing stage over a period of about 7 months. While the products of these projects are strictly research oriented, it is easy to extend the method to actual production systems.

The work I have performed has lead me to formulate a few ideas on how to improve the RPS system in general, and the products of these two projects in particular. It is hoped that these recommendations will be helpful to the endeavors of future students who choose to pursue projects in rapid prototyping.

A. IMPROVEMENTS TO THE RPS

The RPS system developed by the NPS Aero Department is exceptionally helpful in the realization of control and navigation systems for the departments UAV fleet. However, there are a few areas where the ease of use and reliability of the system can be improved.

1. Sun Workstation/AC100

The equipment to do this is already on hand. The portability of the system will be greatly enhanced when the Sun/Luggable combination is replaced by a laptop and AC104. I recommend that the first priority of the next students working with the RPS be to get the new system up and running.

2. Comm Box/Modified Futaba Transmitter

These two components are the weak link in the system. The four ribbon cables that connect the Luggable I/O boards to the various components of the Comm Box are fragile and have had problems with bent pins. All connections in and out of the Comm Box need to be robust connectors permanently mounted on the sides of the box.

The Modified Futaba Transmitter has also been less than completely reliable, experiencing intermittent failures in its ability to transmit PWM commands. A duplicate transmitter that can serve as a backup is needed.

3. Airfield

The airfield at Chualar is too far away to be convenient. NPS Aero needs to reexamine the availability of the Marina Municipal Airport; or perhaps, locate a site on the former Fort Ord property. The 50 mile round trip to Chualar uses up too much time, too many resources, and makes a quick trip back to campus to retrieve a spare or forgotten part a major undertaking.

B. AIRSPEED CONTROLLER

In its current configuration the Airspeed Controller does an excellent job in maintaining a constant indicated aircraft speed. The following refinements would make the controller more versatile and user friendly.

1. Integrated Flight Management Page

The throttle controls need to be integrated into a new main flight IA interface. The current page is too complex and does not include all of the control inputs to efficiently direct the aircraft.

2. Control Groundspeed

The Airspeed Controller should be modified to give the user the choice of using the GPS velocity as the feedback signal. This would allow the aircraft to maintain a constant groundspeed, useful in trajectory control. The limited airspeed range of the FROG aircraft will have to be taken into account when trying to control groundspeed in a high wind environment.

C. VOICE CONTROL SYSTEM

The ViA wearable and voice recognition software proved highly successful under adverse conditions. The system proved reliable and easy to use. One issue with the hardware display must be addressed. Also, the interface with the RPS and commands provided by the VCS to the aircraft could be slightly refined to improve system performance.

1. ViA Touchscreen

The touchscreen is a great tool but is very hard to read outdoors; almost impossible in direct sunlight. This creates safety of flight issues if the operator does not receive feedback on the commands issued to the aircraft or its reaction to those commands. ViA is promising a sunlight readable screen with future versions of the wearable.

2. Airspeed Control

In order to allow complete control of the aircraft with the VCS, the Airspeed Controller must be integrated. The input should probably be via the touchscreen vice voice commands so as not to overload the voice recognition software.

D. CONCLUSION

The preparation of this thesis has been a challenging and rewarding experience. The Rapid Prototyping System has become a fairly mature system and in combination with the Voice Control System should offer some exciting opportunities for future projects.

APPENDIX A. BLOCKSCRIPT SOURCE CODE

This appendix contains the BlockScript code used in the Current Conditions Hold Block of the Speed Controller. It freezes the value of the selected input (U2) when the selected switch (U1) is engaged.

```
Inputs: u;
Outputs: y;
States: x;
Next_States: xnext;

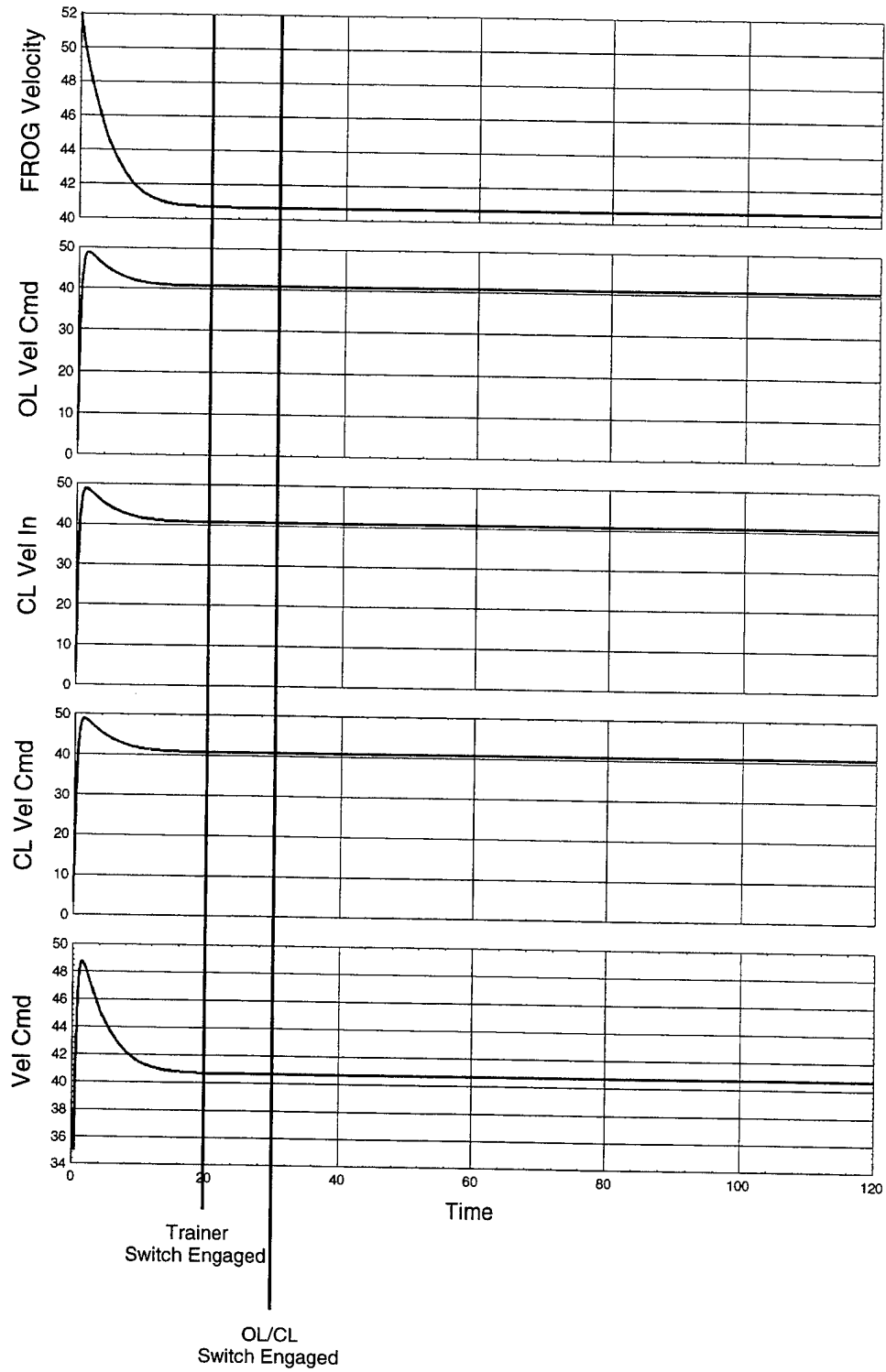
float u(:), y(:), x(:), xnext(:);

if u(1) < 1 then
for i=1:y.size do
y(i) = u(i+1);
xnext(i) = u(i+1);
endfor;
else
for i=1:y.size do
xnext(i) = x(i);
y(i) = x(i);
endfor;
endif;
```

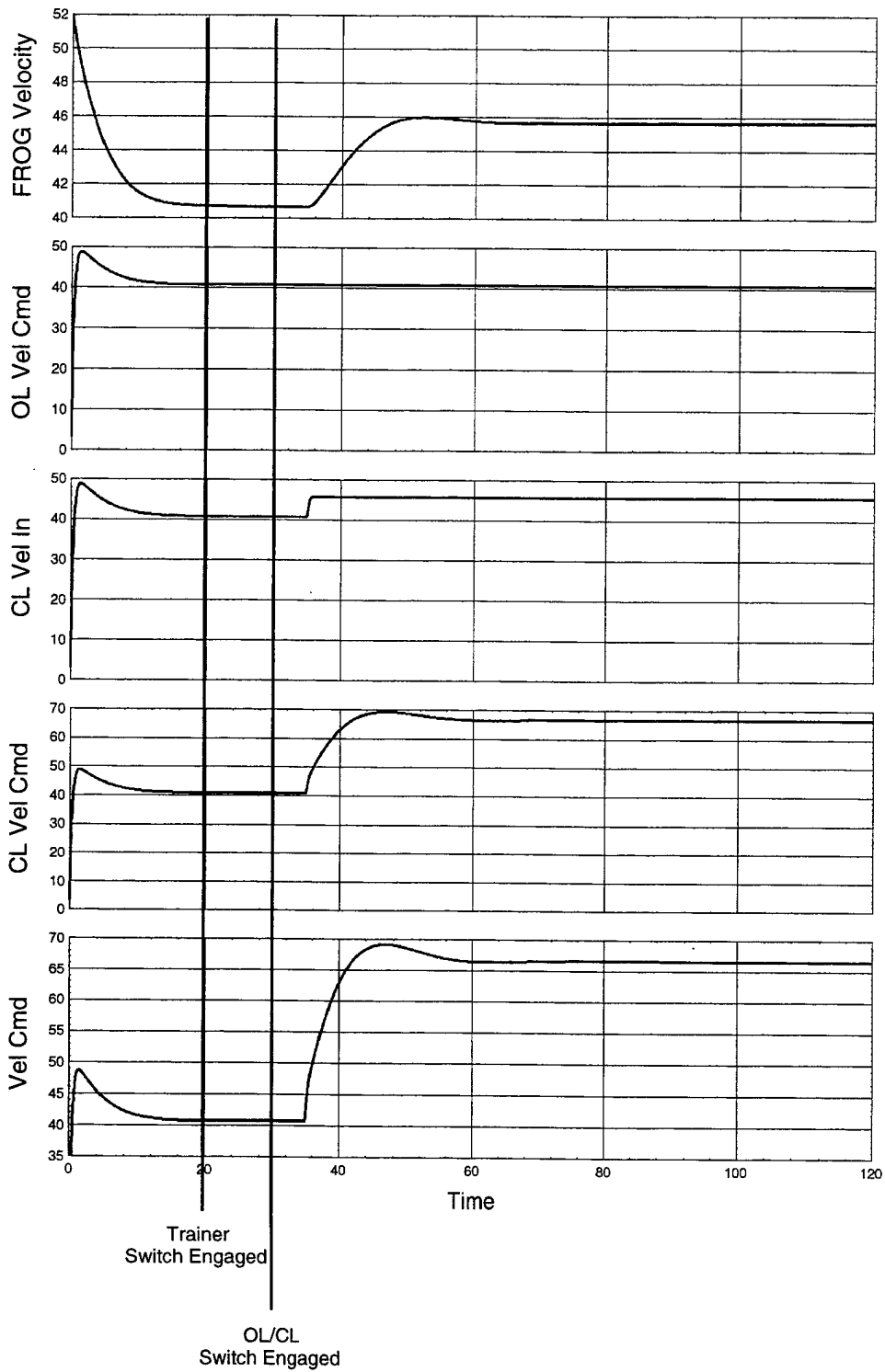

APPENDIX B. RESULTS OF AIRSPEED CONTROLLER SIMULATIONS IN XMATH/SYSTEMBUILD

The following pages detail the results of several simulation runs of the Airspeed Controller as described in Chapter III. The first four runs show the results of no wind situations. More interesting are remaining runs which show the controller in several head and tailwind scenarios. The long runs are provided to illustrate that the controller maintains a constant aircraft speed after a significant amount of time. The final run displays the differing reactions of the controller to the same inputs in no wind and headwind situations.

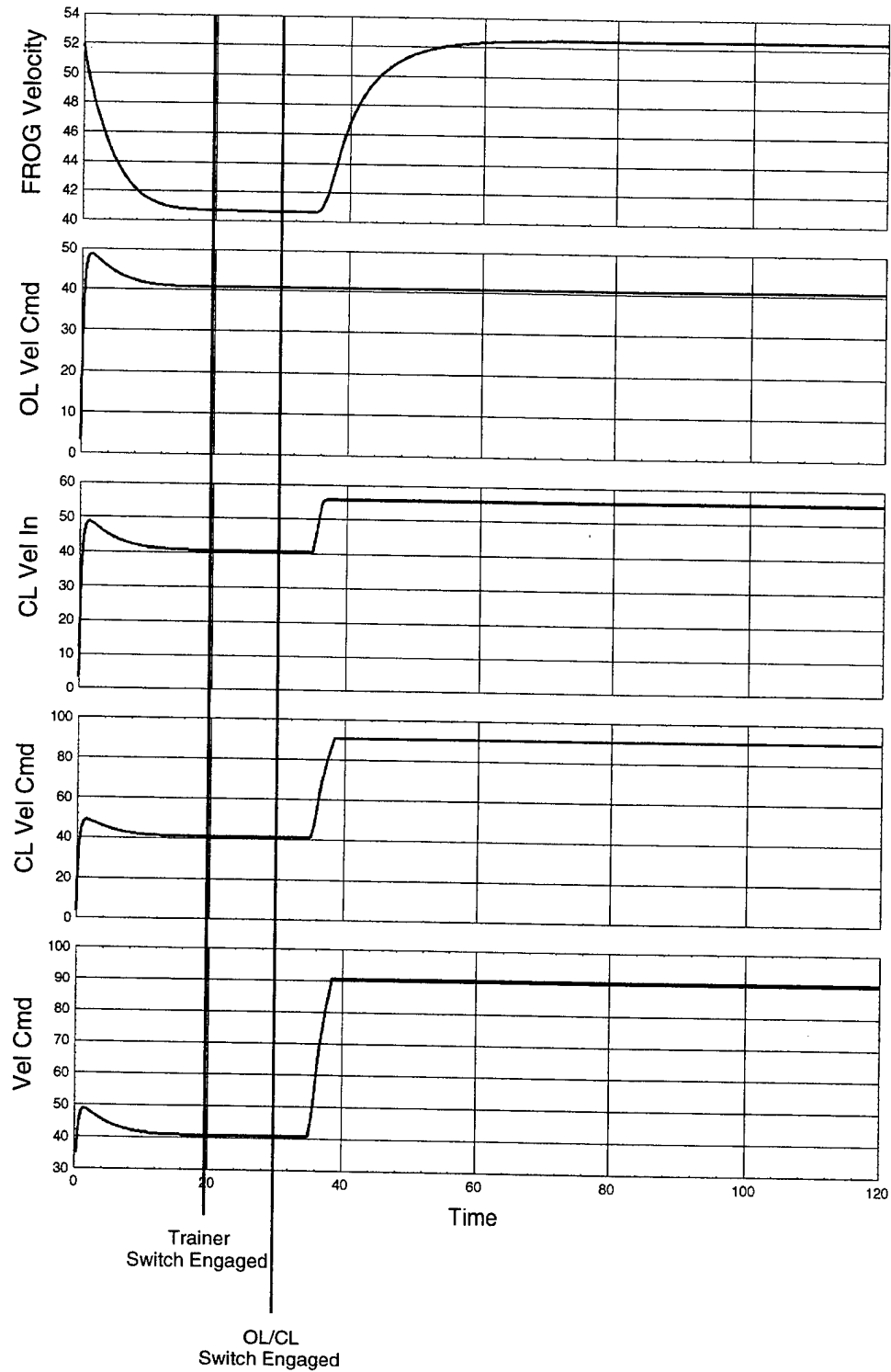
No Wind, No User Speed Inputs



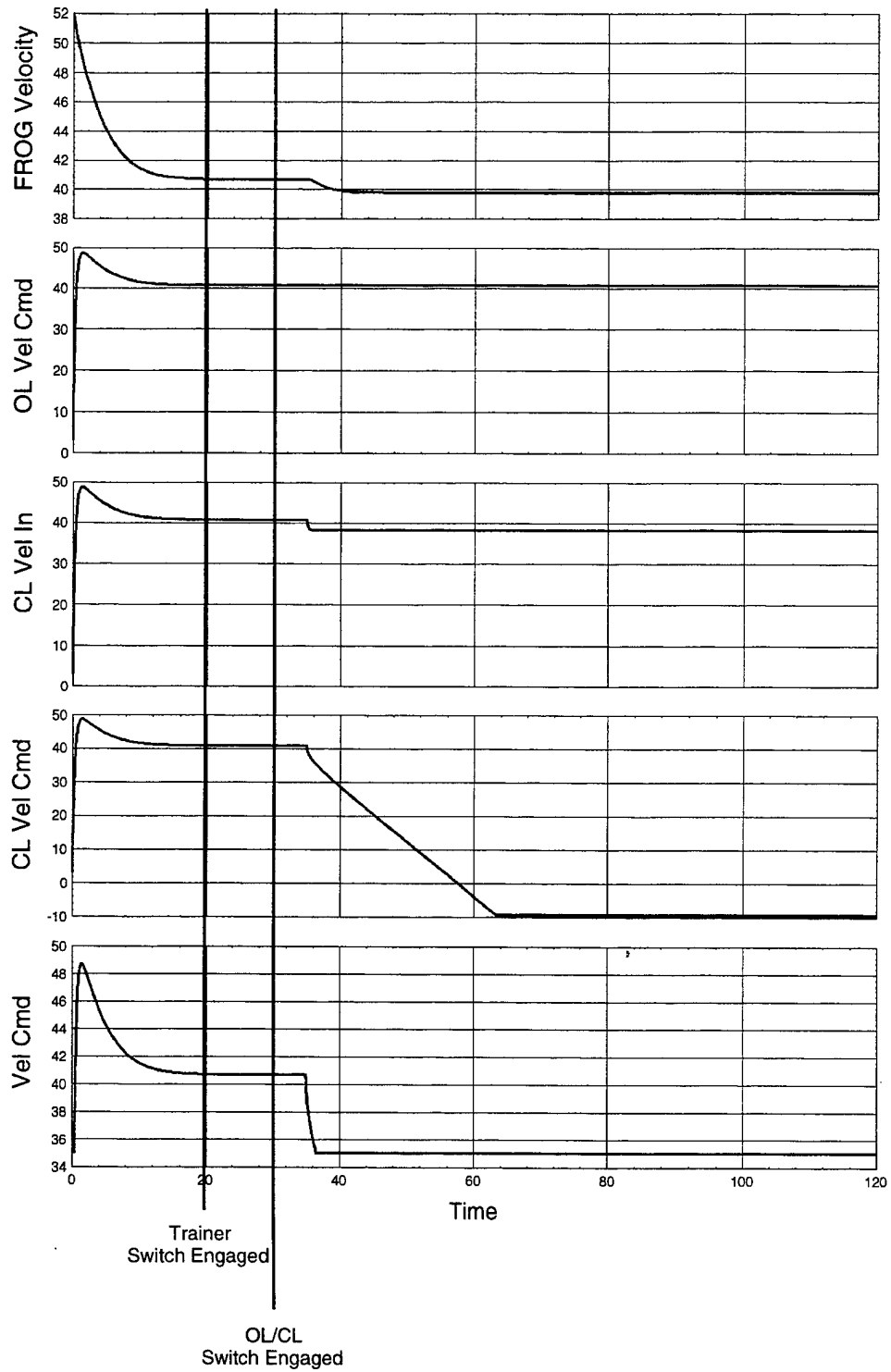
No Wind, +5 Knot Speed Change



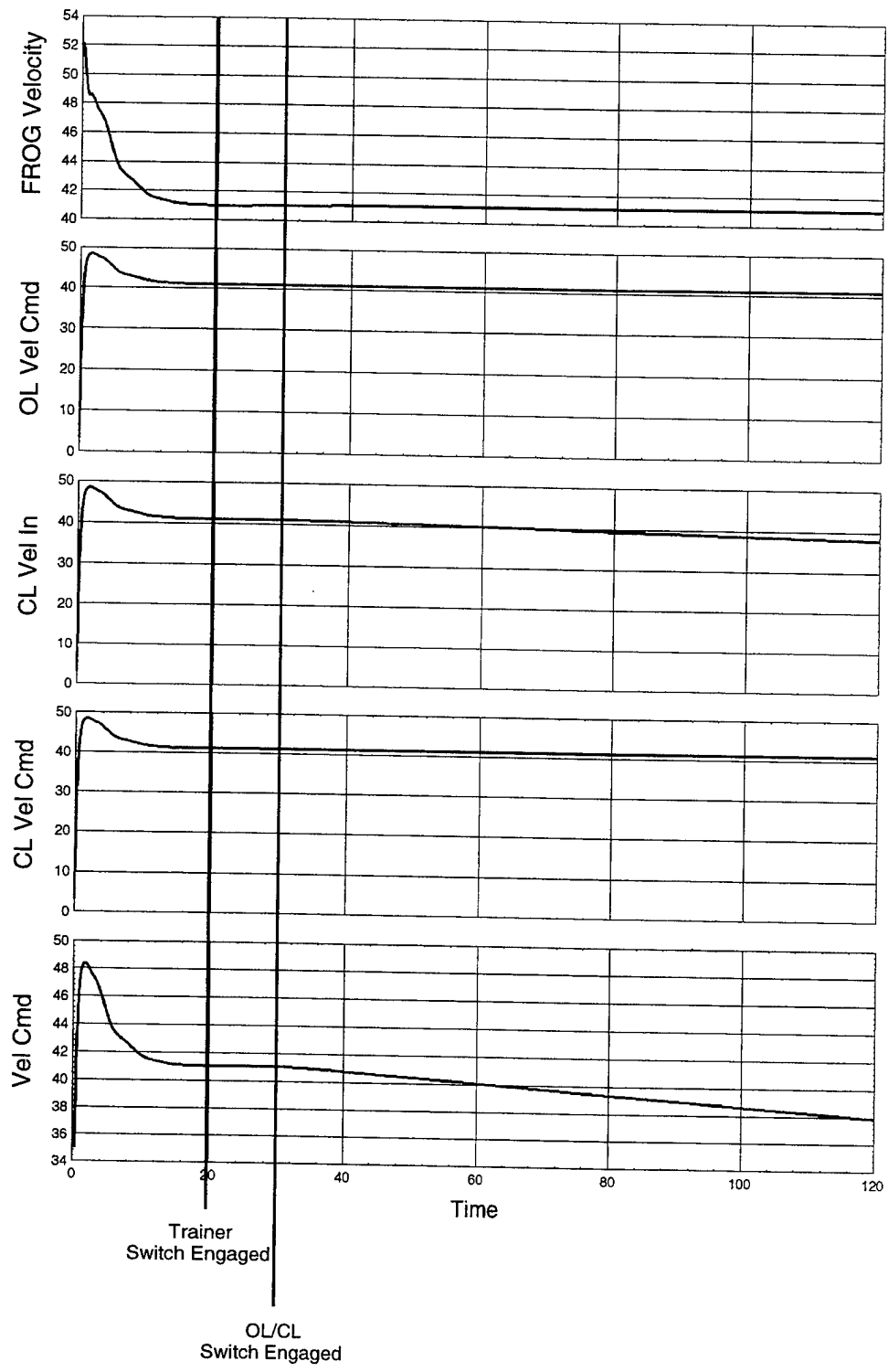
No Wind, + 15 Knots Speed Change



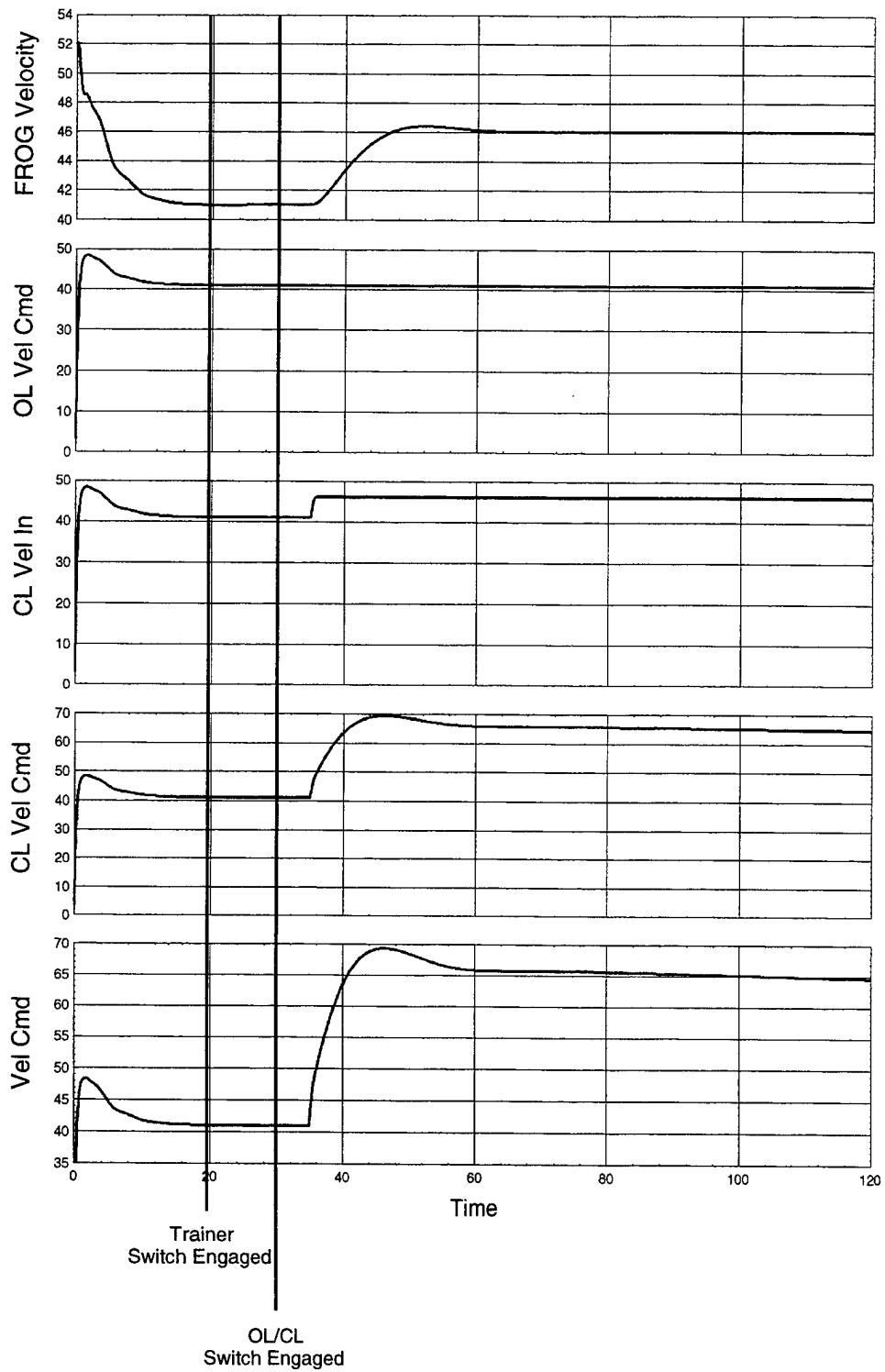
No Wind, - 2.5 Knot Speed Change



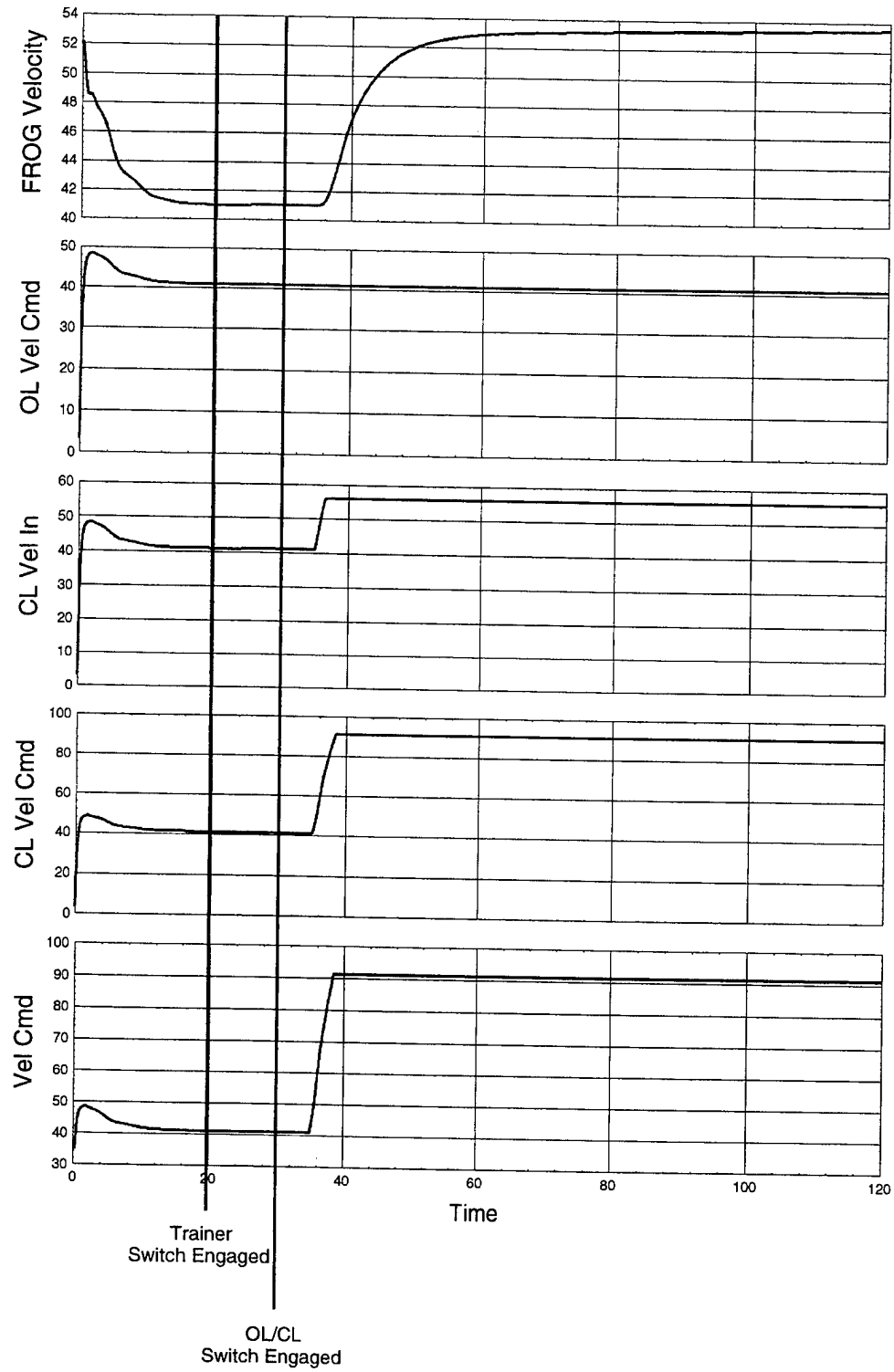
10 ft/s Headwind, - No User Speed Inputs



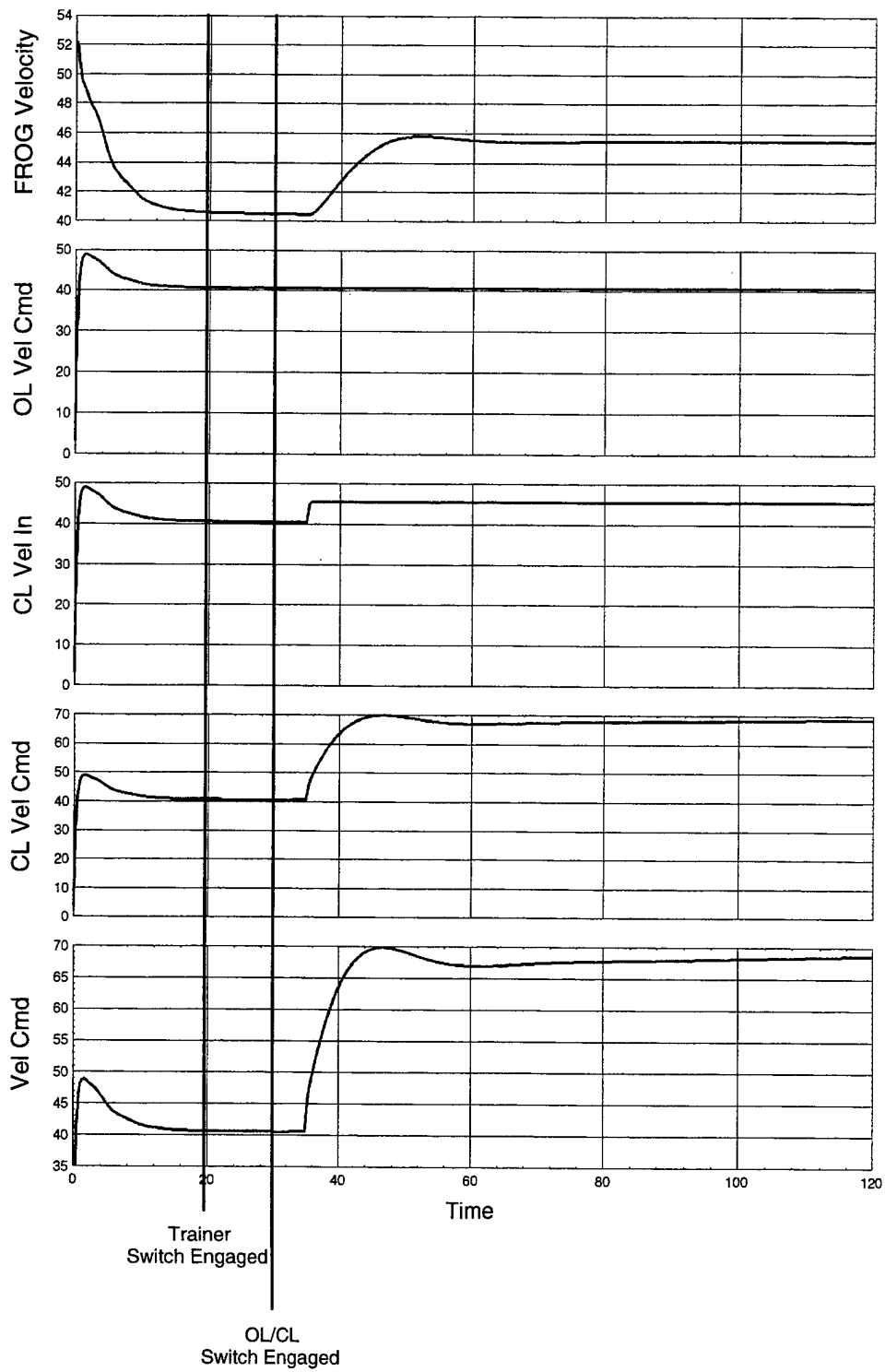
10 ft/s Headwind, +5 Knot Speed Change



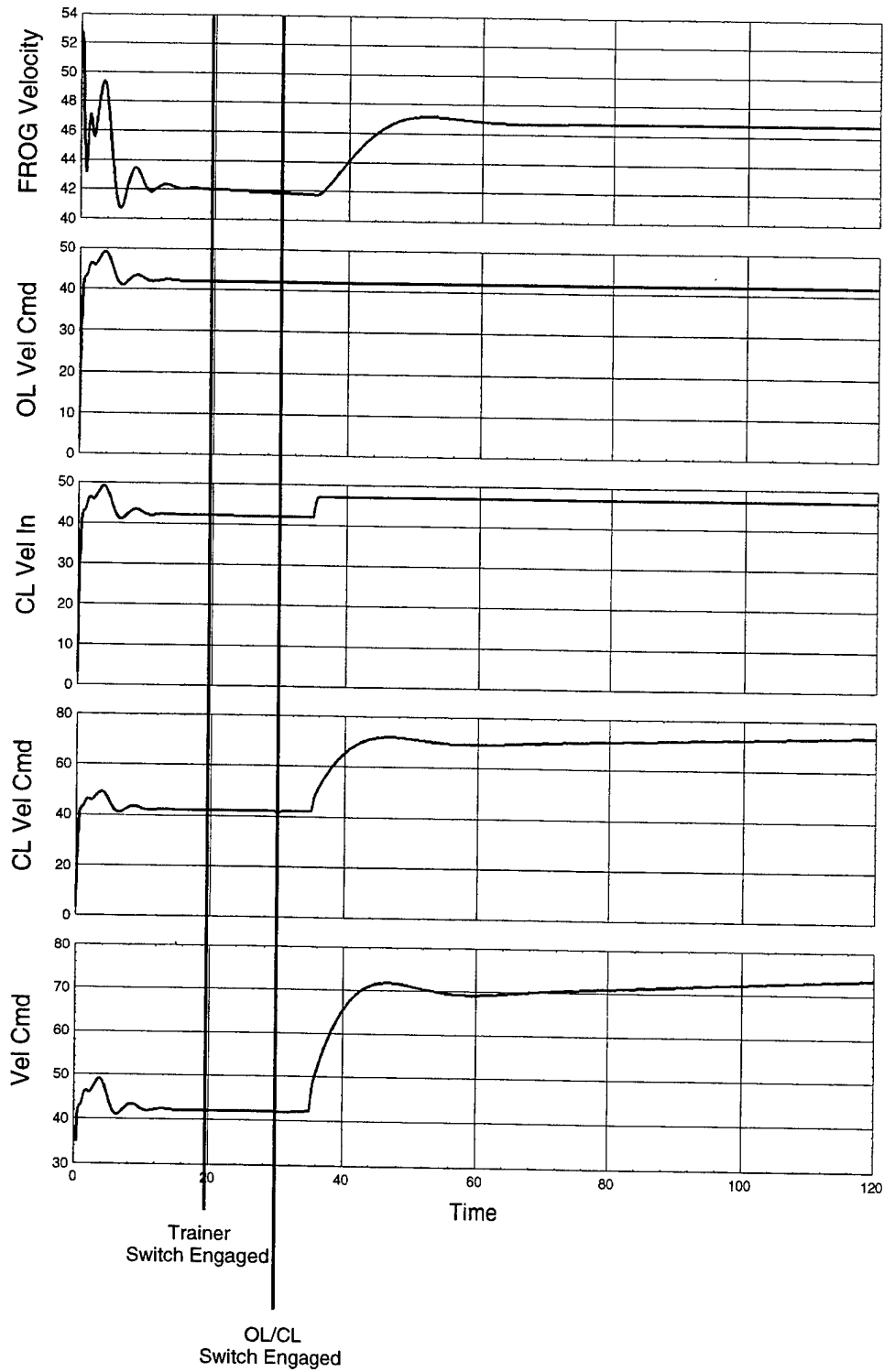
10 ft/s Headwind, +15 Knot Speed Change



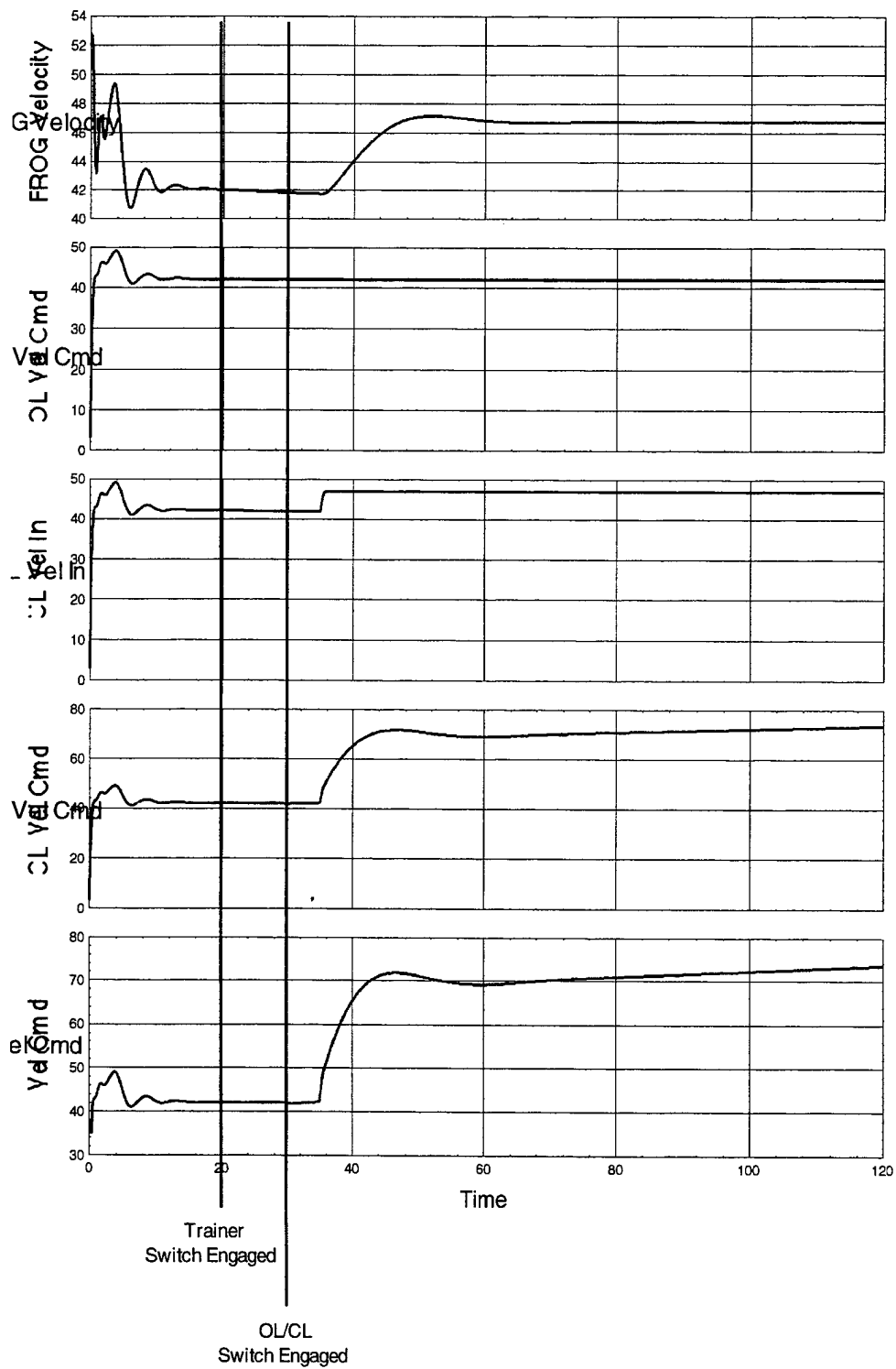
10 ft/s Tailwind, +5 Knot Speed Change



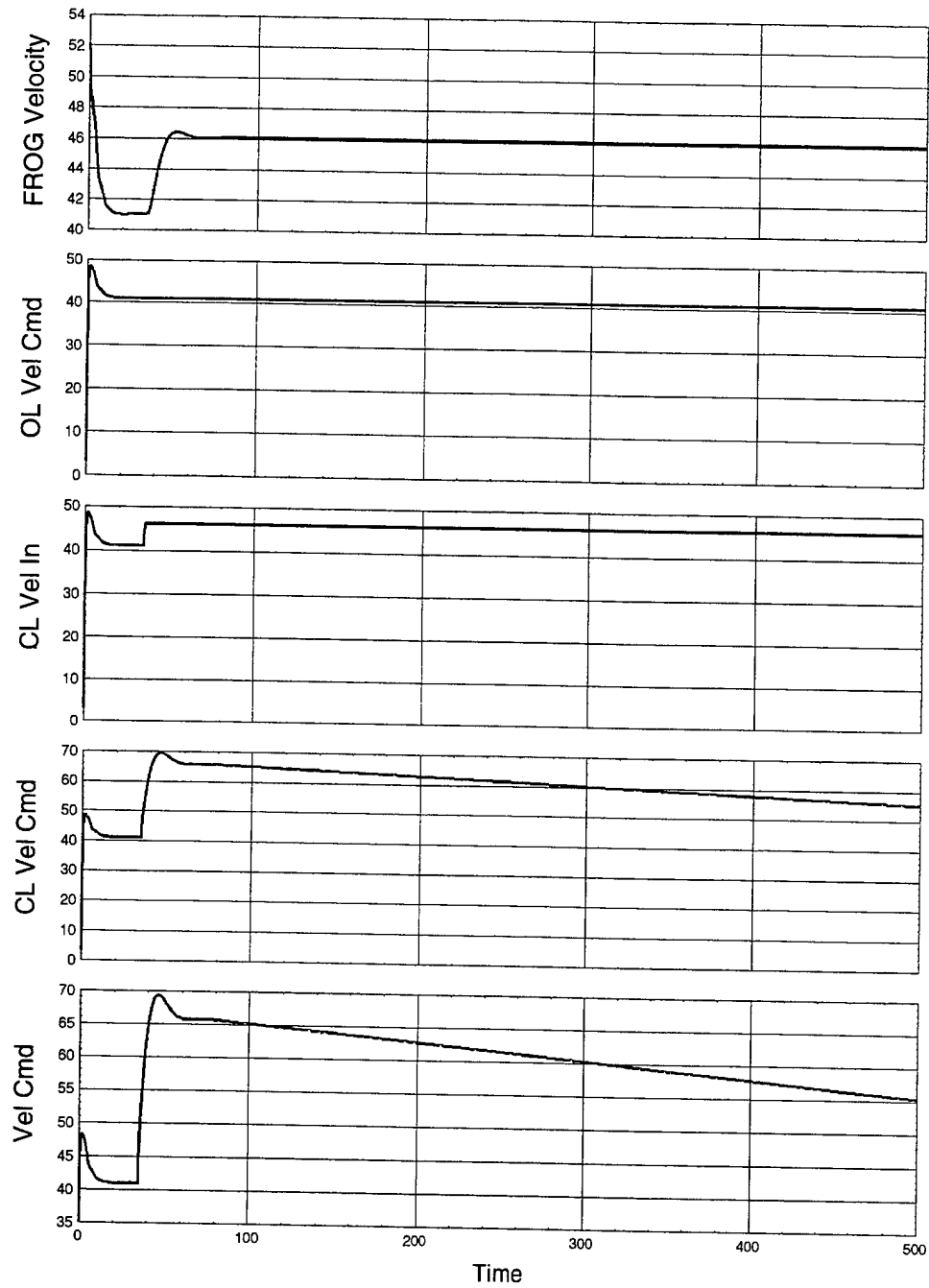
30 ft/s Headwind, +5 Knot Speed Change



30 ft/s Tailwind, +5 Knot Speed Change



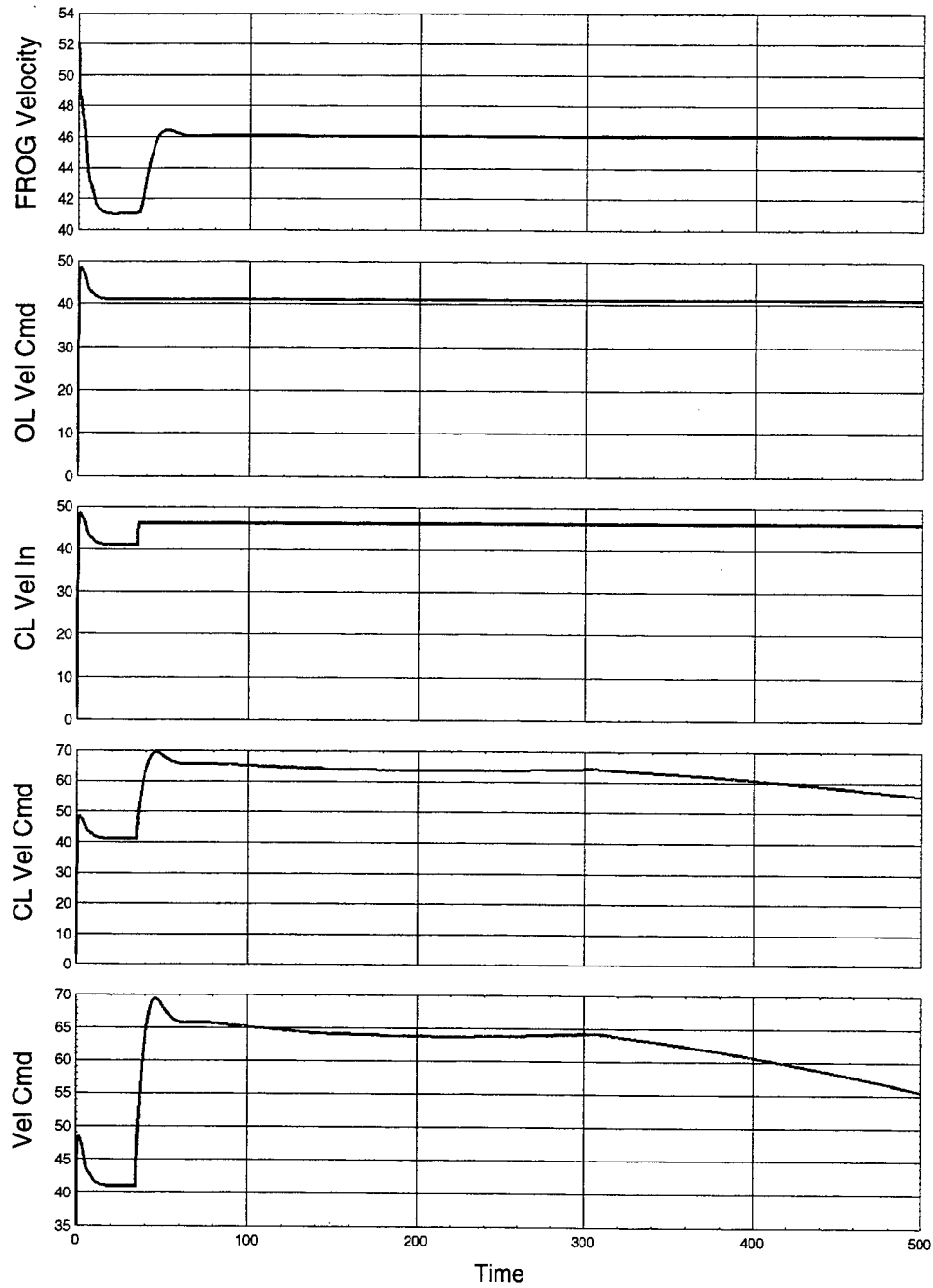
10 ft/s Headwind, +5 Knot Speed Change



Trainer Switch Engaged at 20 sec

OL/CL Switch Engaged at 30 sec

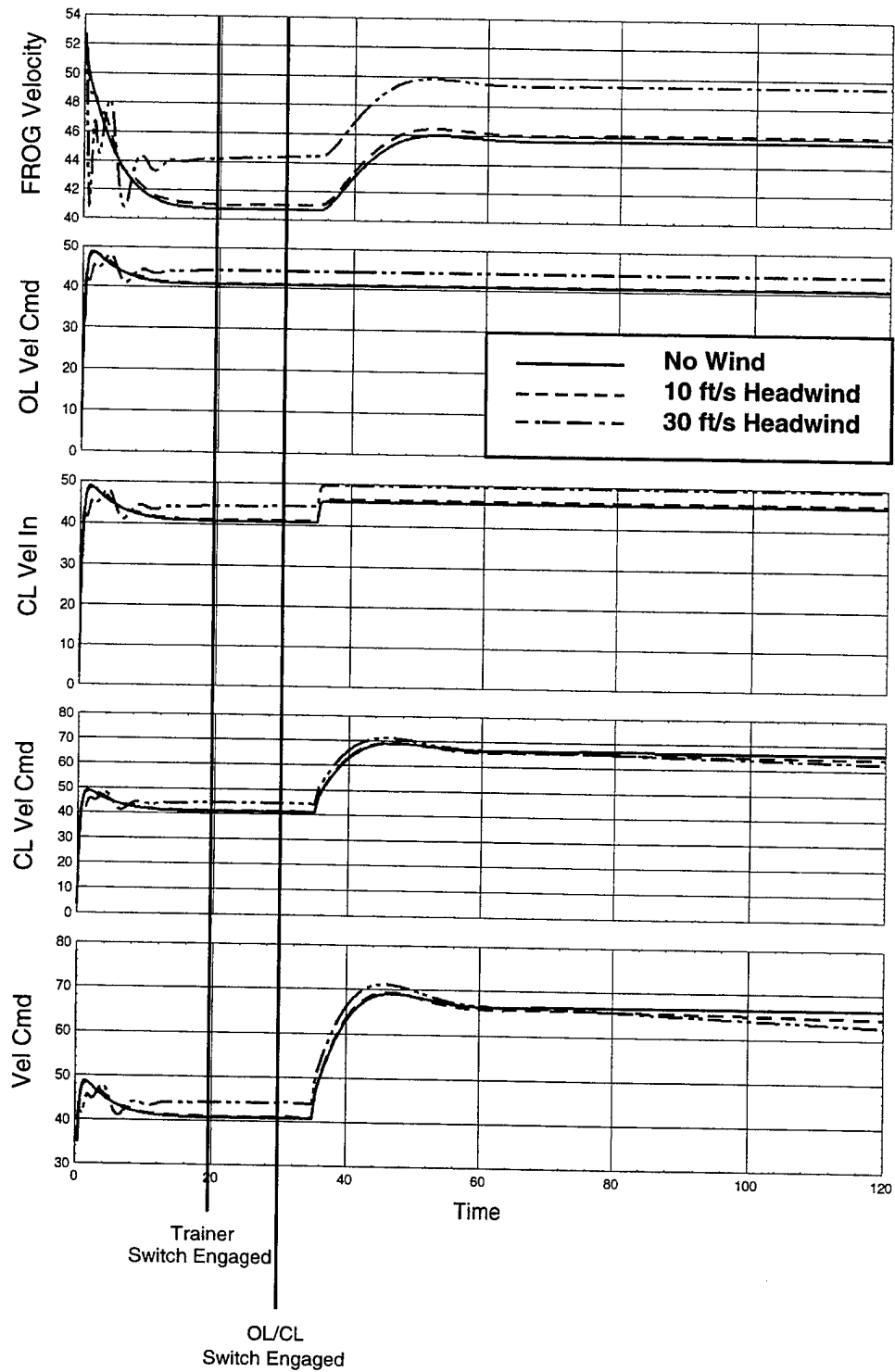
Variable Headwind, +5 Knot Speed Change



Trainer Switch Engaged at 20 sec

OL/CL Switch Engaged at 30 sec

Wind Comparison



APPENDIX C. FLIGHT TEST EQUIPMENT SETUP AND OPERATIONAL PROCEDURES

This appendix is included for the benefit of future students. Included is a list of the equipment that must be transported to the airfield for each flight test. Also included are instructions for setting up the equipment and preparing the aircraft for flight test.

RPS Component Checklist

Basic System

Students Responsibility

- ☐ SPARC2 CPU
- ☐ Monitor
- ☐ Monitor Cable
- ☐ Keyboard
- ☐ Mouse
- ☐ Mouse Pad
- ☐ Hard Disk
- ☐ Hard Disk Cable
- ☐ Ethernet Transceiver and Cable
- ☐ Luggable Computer
- ☐ Communications Box
- ☐ Futaba Battery for RX (2)
- ☐ Modified Futaba Transmitter
- ☐ Comm Box to Futaba Cable
- ☐ Trainer Cable (2)
- ☐ Antenna Cables for Freewaves
- ☐ GPS Antenna and Cable
- ☐ Antenna Array
- ☐ Antenna Stand
- ☐ 12 Volt Power Supply
- ☐ Generator
- ☐ Generator Power Cord
- ☐ Power Cords (3)
- ☐ Power Strip w/ 4 Power Cords
- ☐ Basket w/ extra parts
- ☐ Flight Test Log

UAV Pilots Responsibility

- ☐ FROG Aircraft
- ☐ Futaba Transmitter/Batteries
- ☐ Aircraft Fuel
- ☐ Starter
- ☐ Starter battery

Additional Items for Voice Control

- ☐ Laptop Computer
- ☐ ViA Wearable Computer
- ☐ Nogatech PC Card
- ☐ Nogatech TV RX
- ☐ RF LAN PC Card
- ☐ RF LAN RX

Additional Items for Video System

- ☐ Camera Battery(2)
- ☐ Battery Charger and AC Adapter
- ☐ Big Antenna
- ☐ Small Antenna
- ☐ Antenna Cables
- ☐ RF Down Converter
- ☐ DC Cables (2)
- ☐ 8 mm and VHS Tapes

Ground Station Setup Procedures

SPARC Setup

1. Hook up Monitor and Hard Disk
2. Keyboard Connected in 2nd port from right
3. Connect to Luggable w/ Ethernet Transceiver/cable

Luggable Setup

1. Connect Keyboard
2. CAREFULLY Connect 4 Ribbon Cables to Comm Box

Comm Box Setup

1. Connect Antenna Cables
 - a. Antenna directly to GPS
 - b. Two Freewave Cables on side of box
2. Ensure 3 switches Off, connect Power Supply
3. Connect Cable to modified Futaba TX
4. Connect Battery to Futaba RX
5. Connect Serial Cable to Laptop

Antenna Setup

1. Connect Cables to Freewave Helix Antennas
2. Place GPS Antenna on metal rings
3. Place Antenna Array onto of survey marker

Modified Futaba TX Setup

1. Connect Cable from Comm Box
2. Connect Trainer Cable

Power UP Procedures

1. Ensure all Boxes are OFF and connect power cables
2. Turn ON Power Supply and set to 12 volts
3. Turn ON switches on side of Comm Box
4. Turn ON Computers and Hard Disk

Preflight Checks

Control Checks

1. Check and ensure pilot's Futaba TX moves all control surfaces.

Trim Checks

1. Connect pilot's TX to Modified TX with Trainer Cable
2. Select Trainer Switch and observe reaction of control surfaces and actuators
3. Adjust trim until no movement is detectable with selection of TR Switch.

Calibrate DAC

1. Select Trainer Switch
2. Select CAL Mode on Calibrate DAC IA page.
3. Adjust input and record values for 2.4 and 2.7 volts for desired controls.
4. Enter values in slider control inputs
5. Deselect CAL Mode.

Other Checks

1. Look at GPS and IMU IA pages to ensure receiving data.
2. Turn on Video camera and transmitter and check TV reception.

Warm up checks

1. Start engine and warm-up for 5 minutes
2. Perform high speed taxi test

Takeoff

1. Start timer to track time airborne
2. Record data via IA page
3. Start VCR

LIST OF REFERENCES

1. Hallberg, E N., *On Integrated Plant, Control, and Guidance Design*, Ph.D. Dissertation, Naval Postgraduate School, Monterey, CA, September 1997.
2. Zanino, J. A., *Uniform System for the Rapid Prototyping and Testing of Controllers for Unmanned Aerial Vehicles*, Master's Thesis, Naval Postgraduate School, Monterey, CA, September 1996.
3. Allen, P. M., *Incorporation of a Differential Global Positioning System (DGPS) in the Control of an Unmanned Aerial Vehicle (UAV) for Precise Navigation in the Local Tangent Plane (LTP)*, Master's Thesis, Naval Postgraduate School, Monterey, CA, March 1997.
4. Integrated Systems Inc., *MATRIX_X Product Family System Manuals*, Version 5.0, 1996.
5. Papageorgiou, E. C., *Development of a Dynamic Model for a UAV*, Master's Thesis, Naval Postgraduate School, Monterey, CA, March 1997.
6. Sebastiao, L., *Technical Manual (Draft) for PC_S_C30*, Naval Postgraduate School, Monterey, CA, July 1997.
7. ViA, Inc., *ViA WearableTM User Manual*, September 1997.
8. Nogatech, Inc., *Notebook TVTM User's Manual*, June 1996.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center2
8725 John J. Kingman Rd., Ste 0944
Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library2
Naval Postgraduate School
411 Dyer Rd.
Monterey, CA 93943-5101

3. Mr. Steve Case3
ViA, Inc.
11 Bridge Square
Northfield, MN 55057

4. Dr. Isaac I. Kaminer Code AA/KA3
Department of Aeronautics and Astronautics
Naval Postgraduate School
Monterey, CA 93943-5121

5. Dr. Russ Duren Code AA/DR.....1
Department of Aeronautics and Astronautics
Naval Postgraduate School
Monterey, CA 93943-5121

6. Department of Aeronautics and Astronautics.....1
Code AA
Naval Postgraduate School
699 Dyer Rd. Rm. 137
Monterey, CA 93943-5121

7. Lieutenant Commander John A. Komlosy III3
10741 Portobelo Dr.
San Diego, CA 92124