A METHOD FOR DETERMINING AND REDUCING TRANSPORT DELAYS IN THE FLIGHT SIMULATION ENVIRONMENT

by

R. Marshall Smith

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
Electrical Engineering

APPROVED:

T. Pratt, Chairman

H. Van Landingham

J. Armstrong

December, 1991
Blacksburg, Virginia

A METHOD FOR DETERMINING AND REDUCING TRANSPORT DELAYS IN THE FLIGHT SIMULATION ENVIRONMENT

by

R. Marshall Smith

Committee Chairman: T. Pratt
Electrical Engineering

(ABSTRACT)

This report will describe the process of transport delay measurement along several points in the signal path for piloted flight simulators. Measurements were made in both the frequency and time domain. A new method for collecting data from video transitions was designed and tested. The accuracy and ease of use of the new method was compared to the previous data collection method. The transport delays for individual pieces of equipment were determined as well as the delays associated with the computational flow of two vehicle math models. Hardware and software was then modified to significantly improve the overall transport delay characteristics of the simulation.

CONTENTS

Contents **Page**

TABLES

FIGURES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

Flight Simulation

Since its inception flight simulation World War II, flight simulation has evolved to meet the increasing demands placed upon it by the training and engineering communities.[1] The first flight simulators were small blue boxes, with wings and instruments mounted on pedestals . Hence the name, Blue Box. The Blue Box served as a training aid in flight instruction for most of the war. Through its use it became apparent that simulation, although quite crude at this stage, was extremely valuable in minimizing in-flight instruction and in reducing training costs. As flight simulators increased in accuracy and realism, engineering concepts for the design of aircraft began to be tested on the simulator.

In the 1960s the flight simulator made the transition from analog computers to digital computers. This enhancement meant that training and engineering tasks could be made more realistic due to the relative ease of reprogramming the computer. The training tasks could be more versatile and now the engineering tasks had a method by which mathematical models could be easily modified. This transition opened the door to extensive use of flight simulation in many areas and as a result flight simulation equipment (computers, Input/Output systems, visual systems, etc.) began to rapidly evolve to meet the demands exerted by the flight simulation community.

Today research is conducted in vehicle handling qualities, avionics, stability and control, controls, human factors, displays and safety in the flight simulator. Engineering simulators used in conjunction with wind tunnels and flight test data add another dimension in the cost effective development of new aeronautical, control and operational concepts for aerospace vehicles. Such a simulation is achieved by mathematically modeling a vehicle's aerodynamics, controls, propulsion system, structures,

avionics, and environmental characteristics and using computer controlled displays and a control feel system to give the pilot the illusion of actual flight.[2]

During the last twenty years the flight simulation equipment evolution has lead to the development of equipment that can be integrated into a simulation facility system. These facilities typically have several subsystems such as the simulation computers, data transmission networks, cockpits, visual systems for out-the-window scenery and graphics equipment for cockpit instrumentation.

Simulation Facilities

During the evolution of the flight simulator, NASA Langley Research Center's Flight Simulation Facility (FSF) has been on the forefront of simulation technology development. The simulation equipment at the FSF has been through several upgrades and modifications since the early 1960s leading to the simulation system that is currently in use today. The FSF, similar to other facilities, is composed mainly of electronic, hydraulic and pneumatic hardware controlled by software that is executed either on the subsystem or on the main simulation computer. The composition of the FSF system includes two main simulation computers; a data input/output system capable of being configured into a Local-Area-Network (LAN); four controller consoles, from which to operate the simulation; several different kinds of cockpits; two motion base platforms, for cockpit acceleration cues; two Computer Generated Imagery (CGI) systems, for out-the-window visuals; three Calligraphic/Raster Display Systems (CRDS), for cockpit instrumentation; and video and audio distribution networks. Figure 1 shows the functional layout of the FSF. The following is a description of Figure 1.

Hardware

All simulations ultimately utilize a central computer, or computers, on which to start, operate and terminate the vehicle

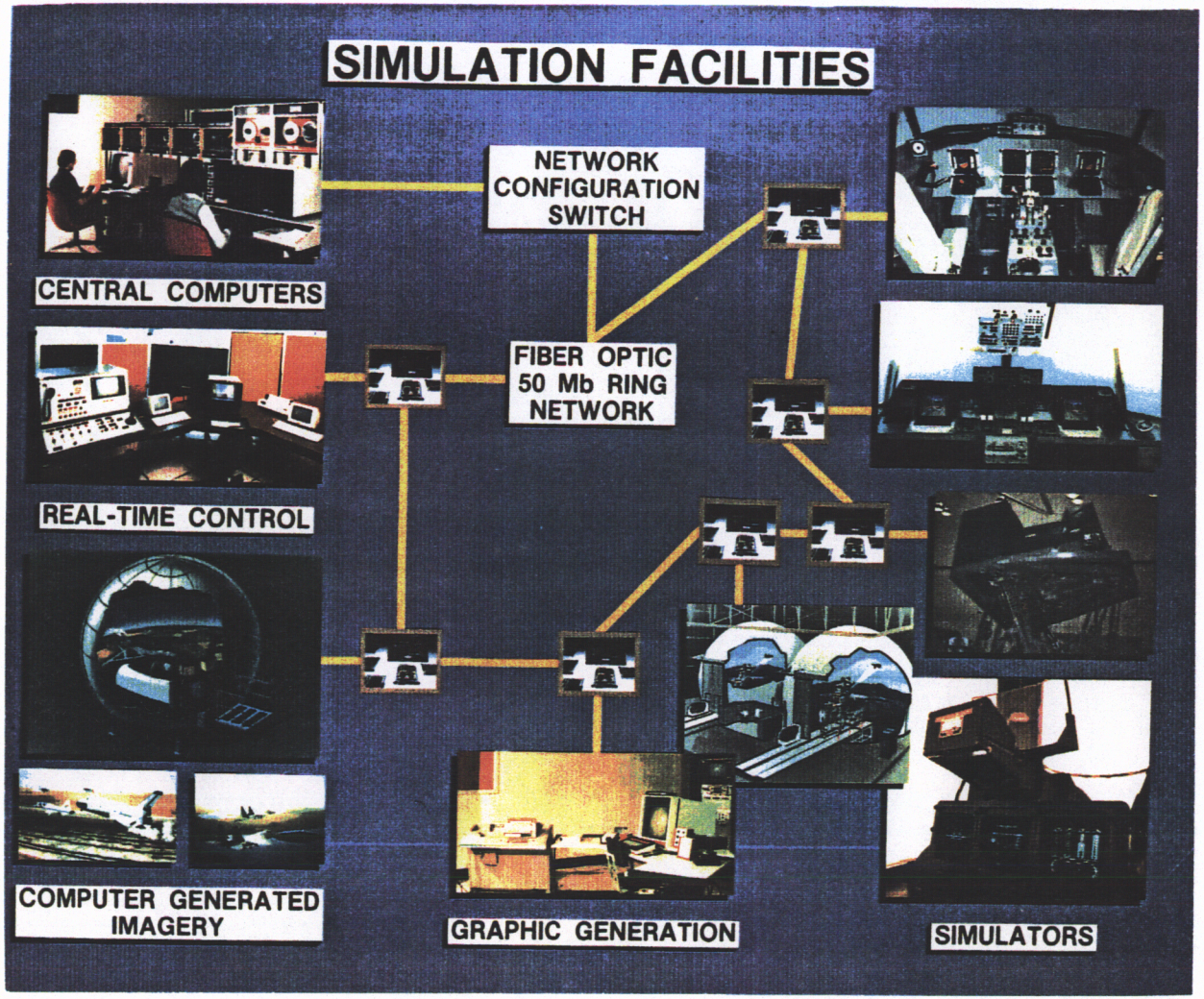


Figure 1 - The NASA Langley Flight Simulation Facility

simulation. In order to execute a real-time job, the central computer will configure the requested simulation equipment into a fiber optic LAN, through the network configuration switch, and control any assigned hardware through the LAN. Inter-computer communications, between the central computer and remote site computers, may take place via local protocols such as the DR-11W, VME or IEEE-488. The central computer completes all necessary functions within one simulation frame. During a simulation frame data is read into the computer from the remote sites. The data read into the central computer consists of analog data, emanating from the cockpit, which is converted to digital information via Analog-to-Digital-Converters (ADC), data returned from remote processing sites and the real-time control console. The computer then takes this information and completes all necessary calculations to determine the next state of the vehicle. New data is then passed out of the computer, via the LAN, to the remote sites for local processing and, if necessary, the data is converted to back to analog information via Digital-to-Analog-Converters (DAC). Completion of all of these functions is necessary for a simulation job to qualify for real-time status.

The central computer can schedule a single cockpit or several cockpits as the research needs require. Each cockpit contains control systems, communication systems and display systems necessary to emulate the environment of the actual flight hardware. To emulate cockpit control systems each cockpit will typically include a programmable control loader. The control loader is a hydraulic system that give the pilot the "feel" of the vehicle through the stick and rudder pedals. Each cockpit also contains the communications system that is present in the actual aircraft and it is interfaced with the simulation communications network. Each cockpit also has a display system for rendering the out-the-window scenery to the pilot. Typically these systems are Cathode-Ray-Tube (CRT) raster monitors that portray their images through collimating optics. The collimation is necessary to give the pilot the illusion that the image is farther

away than a few feet. In the training environment the cockpit typically is a exact replica of the vehicle being simulated. However, in the engineering environment cockpits typically only include what is functionally necessary.

Some cockpits are mounted on motion platforms, also called motion bases. These platforms give the pilot acceleration ques in order to aid in the realism of the simulation. Special washout algorithms, located on the simulation computer, monitor the travel of the motion base and bring the base back to center position whenever possible.[3]

In order to provide additional realism, an out-the-window CGI scene is generated from specific position coordinates in a database and displayed to the pilot in the cockpit on the CRT displays. A CGI systems cost increases dramatically as the quality of the image becomes more realistic. State-of-the-art systems may cost between $5 to $10 million dollars, depending on the configuration selected.

Finally as the trend toward CRT instrumentation in the cockpit continues the simulator must also duplicate, electronically, any cockpit instrumentation. Images may be presented to the pilot in either a raster or calligraphic video format. The information that is displayed usually contains information on the state of vehicle (airspeed, angle-of-attack, G-load, altitude, etc.). State-of-the-art systems capable of generating electronic displays cost between $0.5 to $1 million dollars.

Software

The main central computers execute the bulk of the software required for a real-time simulation job. The central computer software contains the math model, which is an exact representation of the vehicle being simulated. This includes the equations of motion, the models for all servos, actuators, surfaces, aero data from wind tunnels and atmospheric data.[4] In addition, the simulation software coordinates the scheduling of any assigned equipment,

formats the data for transmission and sends/receives the data to/from the remote sites.

The simulation computer is usually not the only system with operating software. Several digital subsystems typically have local operating systems to preform specific tasks. For example, the CGI system typically has an operating system that sends/receives data to/from the LAN, formats the data for the image generator and monitors image generation hardware as each scene is created. The graphics system used to generate cockpit instrumentation generally interfaces to the LAN for data exchange, and then user defined display programs are then executed to generate the instrumentation.

Transport Delay

Figure 2 shows the scene inside a typical dome fighter simulator and the following is a description of a typical simulation. The pilot flies the vehicle by control inputs through the stick, throttle, rudders and switches. These inputs are then passed through the LAN to the central computer. The central computer then executes the vehicles math model and generates the next state parameters. Once these parameters have been calculated then they are transmitted, via the LAN, to the appropriate supporting equipment. The supporting equipment then manipulates the data as required. Figure 2 shows the output of several pieces of equipment used in this simulation. The out-the-window scene, generated by the CGI, is projected on the interior of the dome. The Heads-Up-Display (HUD) and Heads-Down-Displays (HDD), generated by the graphics generation equipment, are displayed in the cockpit. The targets, generated by both laser and computer, are positioned and displayed on the inside of the dome. Finally, any analog dials and gauges in the cockpit are to set to their appropriate values.

In order for a simulation to truly be called real-time the above sequence of events must occur in the same amount of time as it would in the real world. The math model includes all the real world delays the vehicle would have (for example, actuator and servo lags). Therefore, since

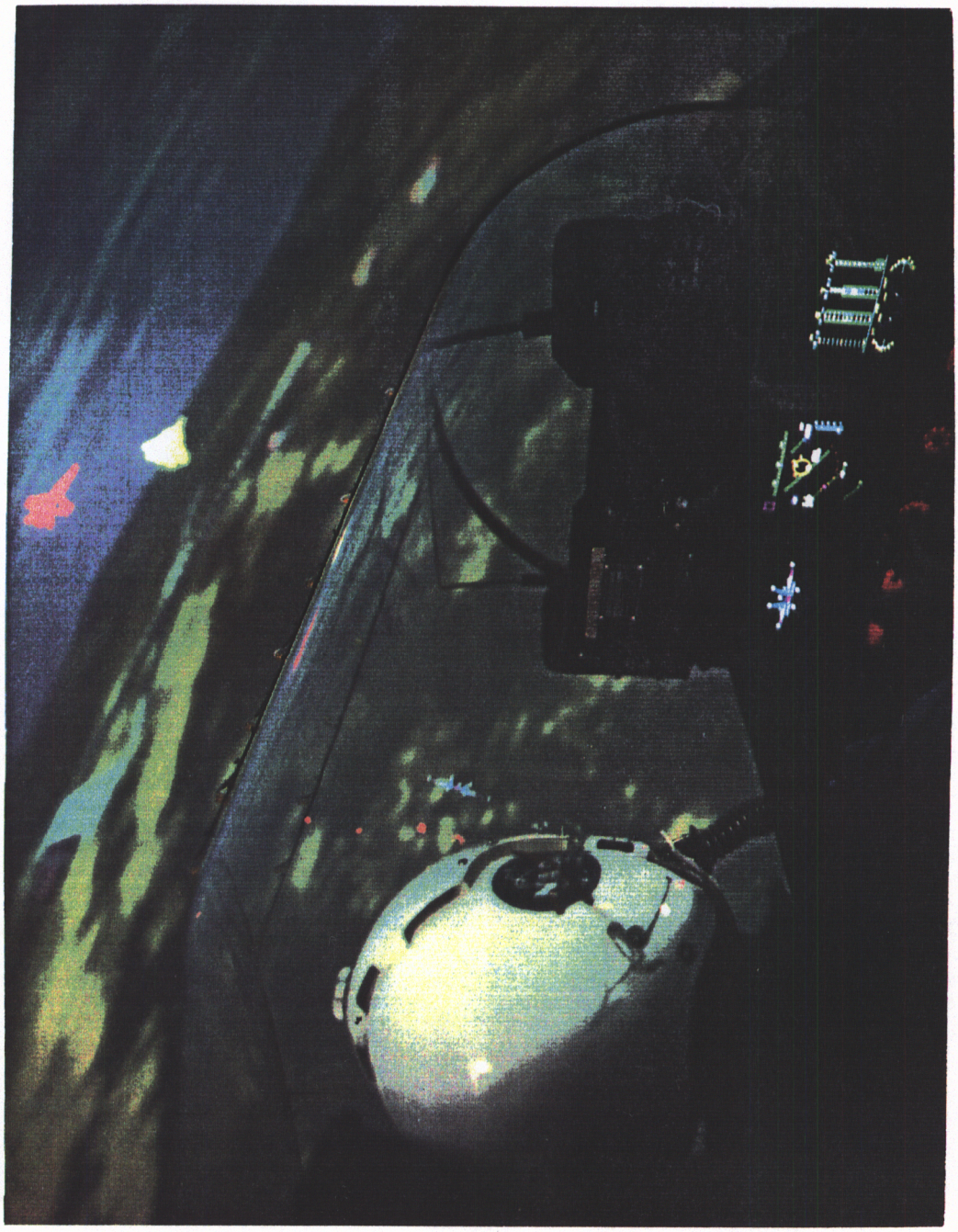


Figure 2 - A Typical Fighter Simulation Scene

the real world does not contain any extraneous simulation equipment (for example, CGIs, LANs, central computers, etc.) the amount of time, additional to the math model, that is required to give the pilot appropriate feedback about the state of the vehicle is considered a time lag or transport delay.

It is well known that the transport delay must be held to a minimum in order to avoid degraded pilot performance.[5] Transport delays, if excessive, can lead to pilot induced oscillations due to inappropriate queuing. The Federal Aviation Administration (FAA) has issued guidelines stating that average transport delays of less than 150 ms are required for certification of transport simulators and less than 100 ms for certification of military aircraft.[6] Recent research seems to indicate that these are appropriate guidelines.[7,8] When the transport delay is extremely long the pilot may suffer headaches, dizziness and nausea.[9] Also, if the transport delays of individual pieces of equipment are not approximately equal then the research or training task can become an invalid. The following example will illustrate this. Consider a target box, displayed on the HUD, which is supposed to overlay the laser target projected in a dome. Once a correct targeting solution is achieved the target box should overlay the laser target. If the delays were excessively mismatched, the target box may not overlay the target at the appropriate time, therefore indicating an incorrect solution. This would invalidate the data and make the research data useless.

The primary function of the FSF is to conduct basic aerospace research. This requires daily changes to be made to many different real-time programs and math models. Also, each of these math models may require different pieces of simulation equipment. Therefore as the math model software changes, the transport delay through that software could change as a result of different programming methods or undetected programming errors.

The optimum way to track transport delays in this environment would be to individually quantize the delays that are associated with each piece of equipment and with each math model. Once these delays are known then the programmer can calculate the transport delay associated

with the simulation based on timing of the particular math model and the known transport delays of the equipment utilized. To do this each programmer can internally, on the central computer, test the math model transport delay time, which can then be added to the known hardware delay time. This yields a simple method for determining total transport delay in a dynamic, multi-user environment. Further hardware testing would only be required for equipment that undergoes any changes.

It is evident, from the above discussions, that the transport delay characteristics, for both the hardware and software, of a simulation facility not only be known but that they not be excessive and that they be synchronized to within an acceptable level.

Research Objective

The objective of this study is to document the transport delays associated with the equipment and software which is used by the Transport Systems Research Vehicle (TSRV) 737 simulation and the Visual Motion Simulator (VMS) National Aerospace Plane (NASP) simulation. In order to accomplish this task multiple methods of measurement will be examined. Transport delay data will be measured in both the time and frequency domain and compared for accuracy and ease of use. In addition, data collection techniques for measuring transport delays will be examined, or developed, and compared. Once the domain of measurement and method of measurement have been determined the transport delays for the TSRV and VMS control loaders, the communication network/simulation computer, the CGI system, the graphics system and cockpit display monitors will be determined. The transport delays associated with the vehicle math models of the 737 and NASP simulation will also be determined. If these delays are excessive or mismatched correction methods will be investigated and implemented where possible.

CHAPTER 2

TRANSPORT DELAY

Definitions

Transport delay is defined as the time elapsed from a pilot input until an appropriate response is made to the pilot by the associated hardware and software that is in excess of the actual delay of the vehicle. This time includes all equipment delays (for example: the CGI or communication network delays) and the time to execute the vehicle math model (for example: the 737 or NASP). The definition of transport delay can be further divided into a software transport delay time and a hardware transport delay time.

For the purposes of this paper the software transport delay time is defined as the amount of time delay associated solely with the execution of the vehicle model that is being implemented. In order to perfectly simulate real-world systems each vehicle model already contains any real-world delays, such as actuator or on-board computer response time, so that the model is a perfect representation of the vehicle. Since there are no simulation computers on actual vehicles any additional amount of time required to execute the computer instructions associated with the vehicle model is extraneous time and therefore represents a software transport delay.

The hardware transport delay is defined as any amount of time delay resulting from a piece of simulation equipment that is not implemented in the same manner as on the vehicle being simulated. For example, a pilot flying a real aircraft flies his vehicle in the real-world and therefore does not need a CGI to give him out-the-window cues. Therefore the amount of time required to generate a CGI image is considered extraneous and is classified as hardware delay. On the other hand, the control loaders used in the FSF cockpits are the actual fight equipment and therefore have the same delays as in the real vehicles. The delay associated with the control loader is not

added in as part of the total hardware delay. The total hardware delay is the amount of time required for a signal to pass through all non-real-world simulation equipment.

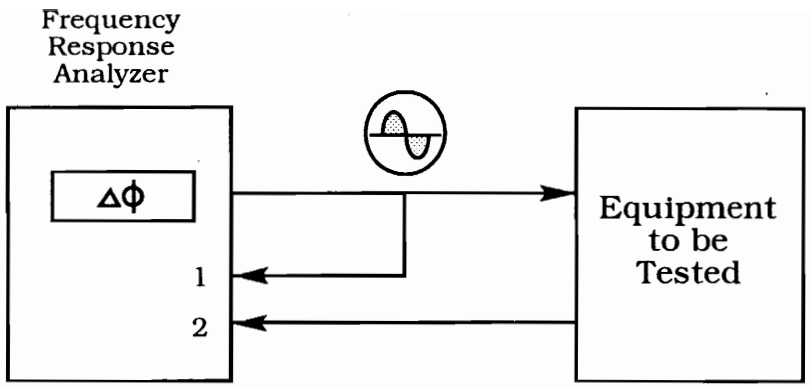
There may also be transport delays within individual pieces of simulation equipment that are generated by internal software. For example, a CGI system requires an operating system and database, or a LAN will require scheduling and protocol software. Since this software is not changed on a regular basis their transport delays remain constant from one simulation to another therefore this delay is quantified with the hardware transport delay for that piece of equipment.

Measurement Domains

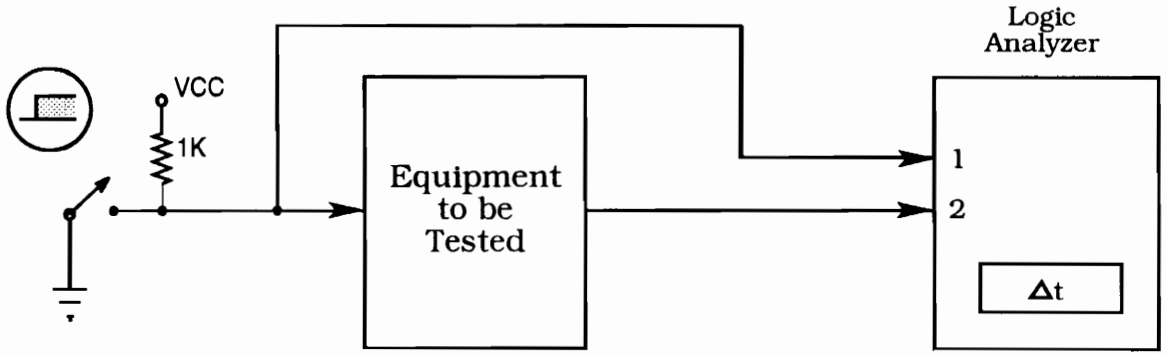
Transport delays may be measured in either the frequency domain or the time domain. Proponents of the frequency domain method contend that the measurement is more accurate since it removes the effects of asynchronous delays along the signal path. If the math model is left in the real-time program during testing the frequency domain method will give the response of the system with the steady state delays of the math model included. Since the transport delay is defined to be only the additional time lag due to superfluous hardware and software the steady state response of the math model must be known and removed from the measurement.[10] The time domain method only measures the step response of the system. Effects due to asynchronous inputs are eliminated by taking multiple readings and averaging them to determine the average transport delay. During the course of this study transport delays were measured in both the time and frequency domains and the results were compared for discrepancies.

Frequency Domain

Figure 3 shows the general setup for the frequency and time domain tests. In order to make measurements in the frequency domain a continuous sinusoid is input into a piece of equipment.



Frequency Domain Technique



Time Domain Technique

Figure 3 - Frequency and Time Domain Technique

The output of that piece of equipment is monitored with a Bafco 916 Frequency Response Analyzer (FRA) to determine the phase shift, $\Delta\phi$, in the signal. The FRA outputs a sinusoid which is fed into the equipment to be tested and constantly compares the continuous output generated by the tested equipment to the input sinusoid. The amount of phase shift in the output signal to that of the input signal can be converted to time by the following formula.[11]

$$t = \frac{\Delta\phi^\circ}{360^\circ \times f}$$

where:

t = time in seconds

$\Delta\phi$ = phase difference in degrees

f = test frequency in Hz

Time Domain

Time domain measurements are made by simply subjecting the input of a piece of equipment to a step signal and monitoring the time to which an output begins to occur as a result of that input. The time based measurements were recorded utilizing a Tektronix 1241 logic analyzer and/or a strip chart recorder.

Measurement Techniques

Several methods for data collection were also tested and compared. A Photo-Electric Diode (PED) and a Video Level Detector (VLD) were compared for accuracy and ease of use. A logic analyzer was also utilized to monitor events along digital paths and to measure transport delay in the time domain.

Photo-Electric Diode

The first method that was tested utilized a PED that was attached to a monitor. The operational theory behind the PED is that it monitors changes in the ambient light emanating from a monitor and outputs a voltage once a change is detected. A raster monitor draws an image starting in the upper left corner and proceeding across the top line. The beam is then retraced to the left side of the screen and the next line is drawn. In order to use a PED effectively it must be placed as close to the upper left corner of the monitor as possible to detect when new pixel data is written (see Figure 4). Once a change in the pixel intensity occurs then the diode will output, through additional circuitry, a Transistor-Transistor-Logic (TTL) signal to indicate that an event has occurred at the monitor. Figure 5 shows the schematic of the PED and its associated logic.

The PED method has several problems due to its design and operational characteristics. First the diode should be quantized to know its own delay characteristics. Ambient light in the cockpit can interfere with measurements. The ambient cockpit light may also cause the PED to signal an event has occurred when one has not. The PED also does not take into account the interlace effect on the monitor which is prevalent in most CGI systems. This can be a problem if the light level of the first field is not bright enough to trigger the diode then the addition of the second field causes the PED to trigger. In this case the measurement would be off a minimum of one video field. Finally, data collection with a PED is usually limited to the cockpit site.

Video Level Detector

A second method detects video levels on the Red, Green and Blue (RGB) inputs to the monitor. The VLD utilizes the incoming RGB signals to detect a video level change. Figure 6 shows the schematic for the VLD circuit. The detector monitors the RGB signals with

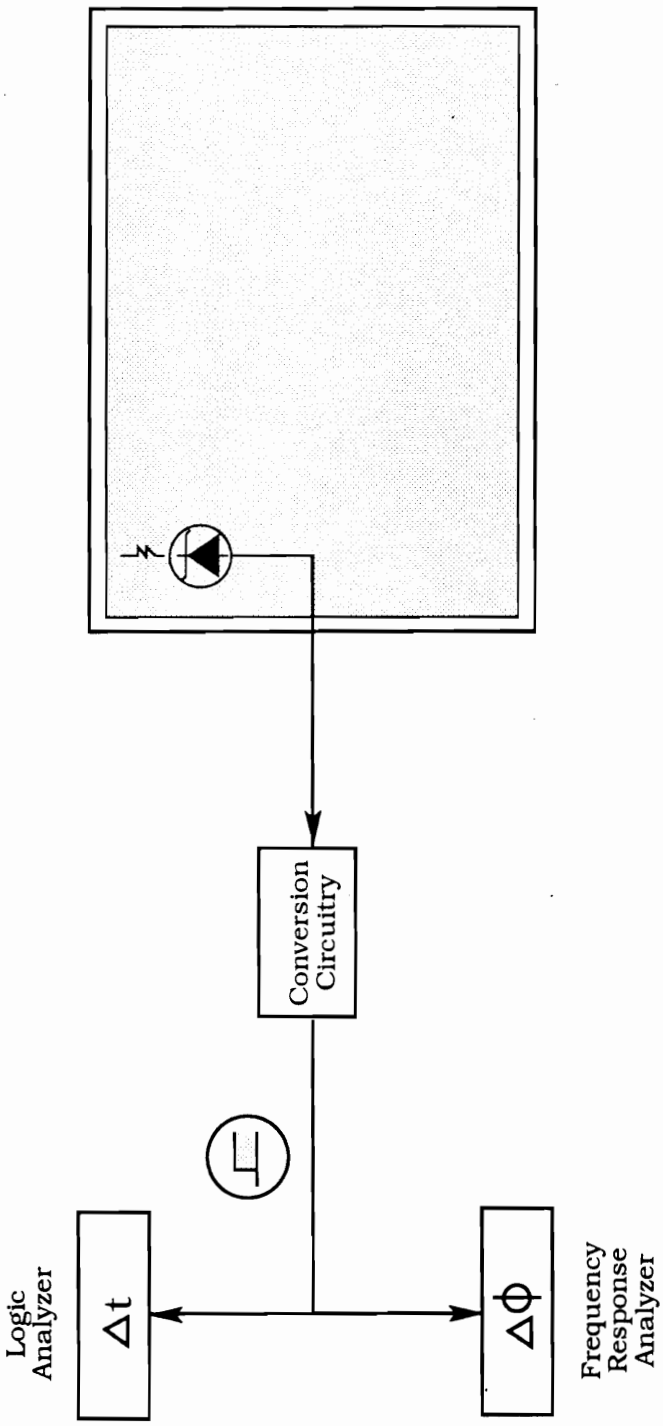


Figure 4 - PED Setup

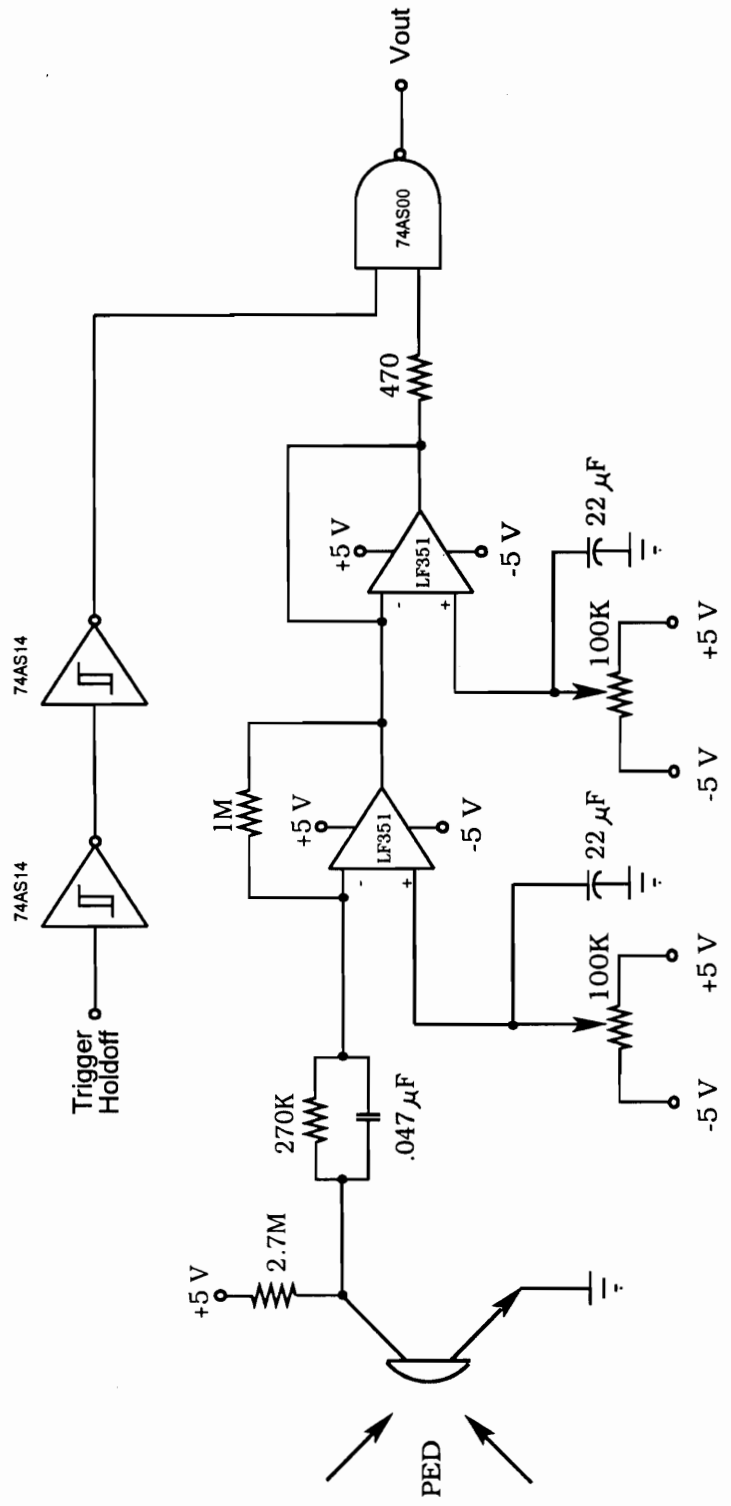


Figure 5 - PED Schematic

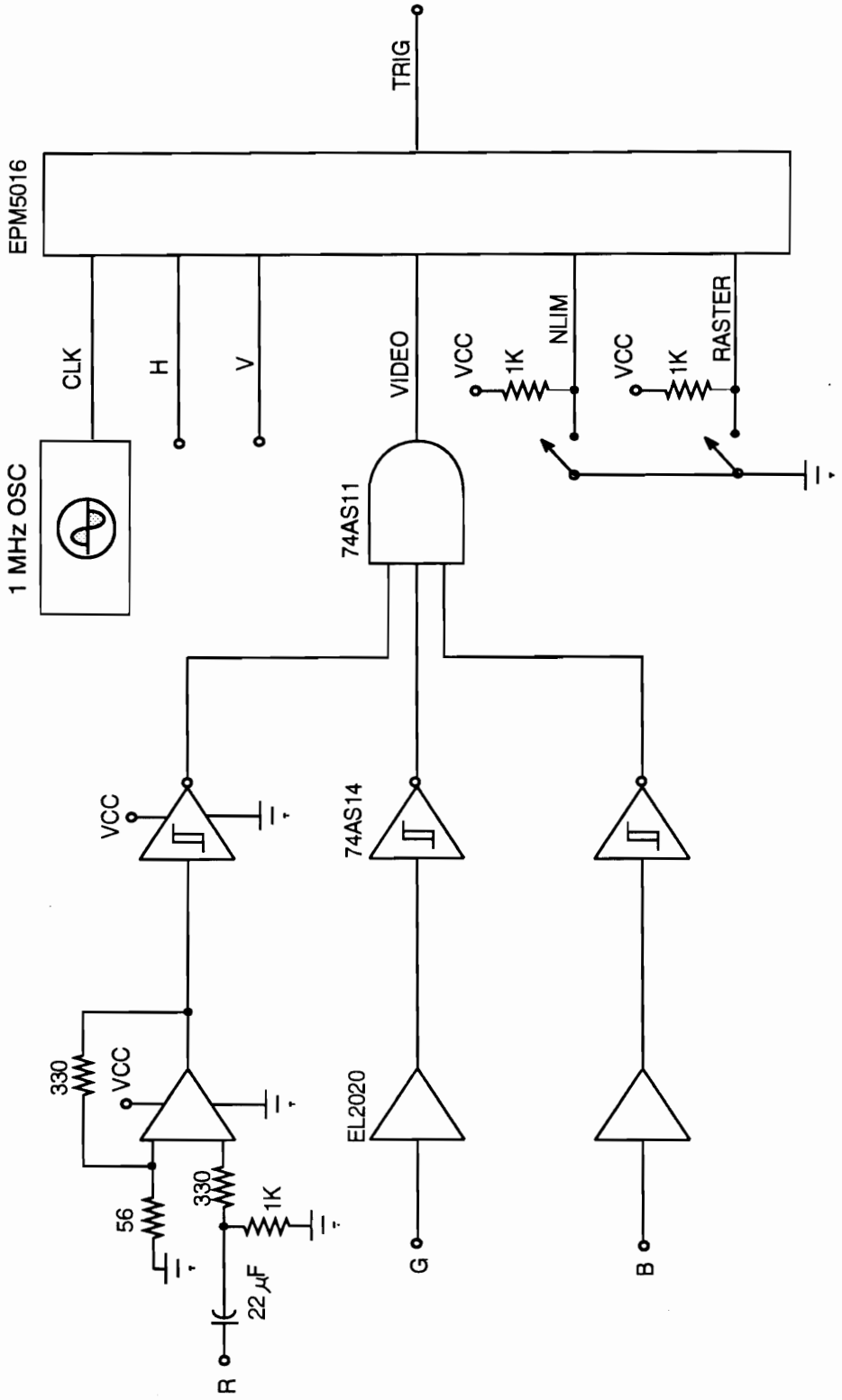


Figure 6 - VLD Schematic

three comparators. Once the video level reaches a preset threshold, the comparators will output a Transistor-to-Transistor Logic (TTL) level to Schmitt-triggers which will then square the signal for output to an AND gate. The output of the AND gate will only be high when RGB are high, meaning a video level corresponding to white is detected. This signal is then fed to an Erasable Programmable Logic Device (EPLD). An EPLD is used to give a trigger output signal based on what kind of input video signal is being used. The CGI system at the FSF is capable of generating several different video formats. A 771 line video format is used for transport simulator displays and an 841 line video format is used for fighter simulators. For each video format the RGB information is modulated with each pixel being displayed. Since the FRA requires a continuous output to calculate phase shift in the frequency domain the input signal that is fed into the FRA must have a solid TTL high during the white video field or a solid TTL low during the black video field. Standard RGB signals for a white field are not normally a solid high. The signal will have periods of lows during the horizonal retrace time and also for a period of time before, during and after the vertical retrace time (see Figure 7). The amount of time the signal remains low depends on the line rate and number of active lines for which the video signal is configured. Because of these intermittent lows simply monitoring the RGB output is not sufficient to conduct frequency domain analysis. The optimum VLD would trap and hold, for a complete video field, the state of the first pixel, high or low, of the first active line of every field. For a 771 line rate, at a 50 Hz update rate, the first pixel of the first line transitions 2.15 ms into the 20.0 ms field time. This would allow the output of the detector to reflect the state of each new video field. This circuit is not necessary if one is interested in only detecting transitions from a black image to a white image in the time domain but it is necessary to detect transitions from a white to a black image in the time domain or to collect any data in the frequency domain. Figure 6 shows two switches: NLIM, and Raster. The NLIM switch allows different timing mechanisms in the EPLD to

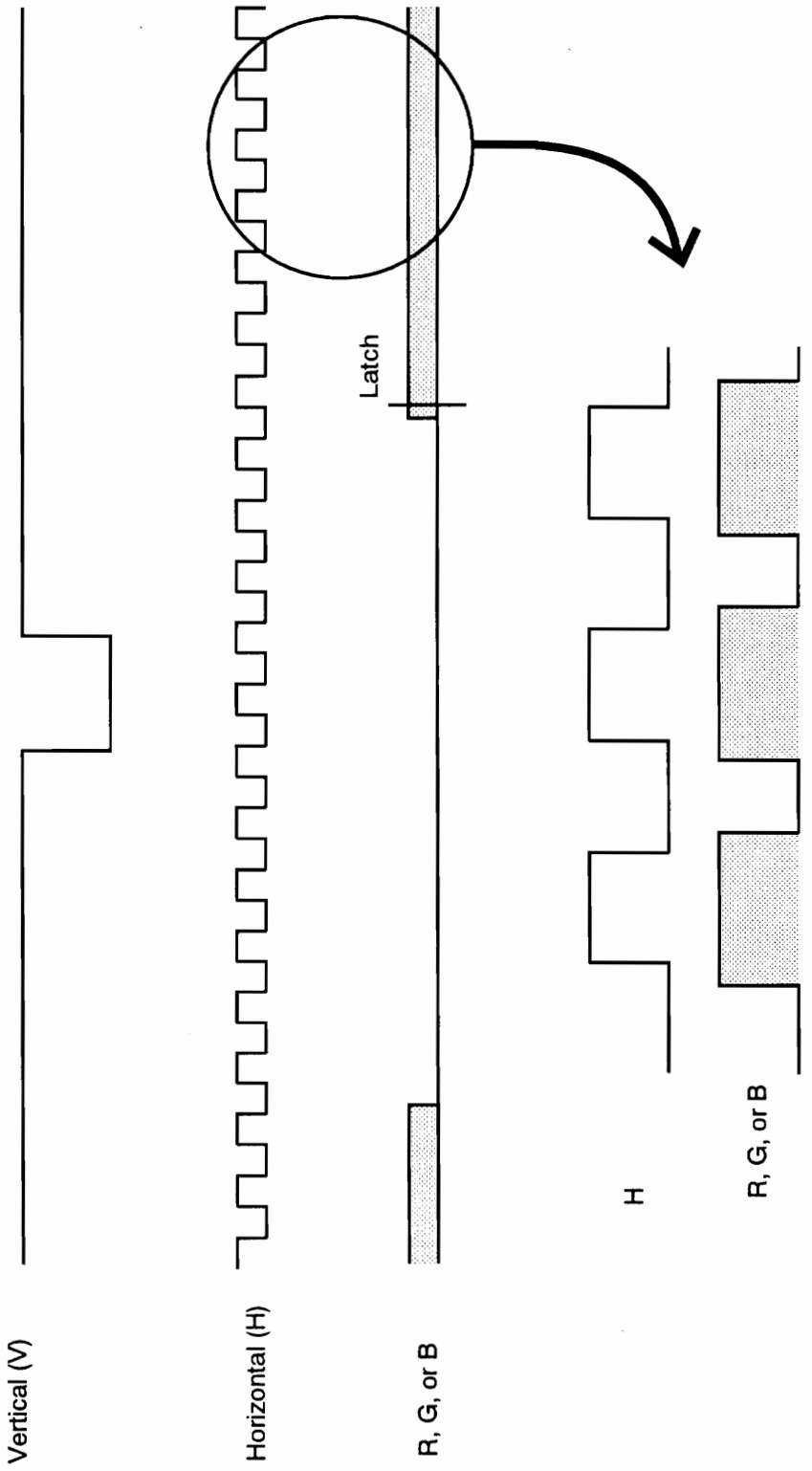


Figure 7 - Video Signal Characteristics

count different numbers of active and passive lines, depending on the video format being tested. The raster switch allows the circuit to be used for raster or calligraphic video. When calligraphic video is being tested the video input to the EPLD is routed directly to the trigger output. This will only give a TTL output when the video changes from black to white and is only useful for conducting time domain measurements. The code for programming the EPLD was generated using the Altera Max+plus II system and is shown in Appendix A.

The VLD and PED only measure the time until an image begins to be rendered. This does not include the time required to complete the drawing of the image. The timing of a raster image is almost always at a fixed frame rate. The CGI system at the FSF operates at 50 Hz update rate, or 20 ms for every video field. Therefore the time to the end of the first video field, once a change has been detected in video, is the time measured by the VLD or the PED plus 17.85 ms (20.0 ms - 2.15 ms). For calligraphic systems the only useful information that can be gathered is when the graphics image has started to be drawn. The time to image completion is dependent on the time required to complete all necessary graphics defined in the user's program.

Logic Analyzer

Finally, a Tektronix 1241 logic analyzer collected data along the digital path by interfacing with the digital portion of the simulation LAN. The logic analyzer was also used as a timing device between events to yield time domain measurements. This was accomplished by detecting the start event, either a step input at the cockpit or digital data on the ARTS highway, and the stop event, a signal from either the PED or the VLD.

CHAPTER 3

SIMULATION HARDWARE

Test Hardware

Chapter 1 discussed, in general, the equipment necessary to conduct a real-time simulation. The equipment that is described in this section is the specific simulation hardware for which individual and composite transport delays were determined. The cockpits that were tested were the Transport Systems Research Vehicle (TSRV) and the Visual Motion Simulator (VMS). Each cockpit has a McFadden sidearm control loader, four XKD monitors for out-the-window visuals and XYTRON monitors for graphical instrumentation. The cockpit interfaces to the central computer through the simulation LAN, which is called the Advanced Real-Time Simulation system (ARTS).[12] The central computers, used for real-time simulation are two CYBER 175s. The CGI system is an Evans and Sutherland (E&S) CT6 and the cockpit graphics system is driven by one of three Terabit Eagle 1000 systems. The Terabit system is called the Calligraphic/Raster Display System (CRDS) since it can generate both calligraphic and raster displays.

Transport Systems Research Vehicle

The TSRV cockpit simulates the flight deck of an advanced 737. Figure 8 show the exterior of the cockpit which is composed of a 737 cab and Figure 9 shows the monitors and optics mounted on the front of the cab. Figure 10 shows the interior of the simulator configured with communications systems, sidearm controller, throttle and rudder controls, Cockpit Display Units (CDU), CGI monitors and CRDS monitors. The TSRV cockpit is an "all-glass" flight deck which is representative of technology becoming available in future commercial aircraft.[13] The TSRV is used mostly for Air-Traffic-

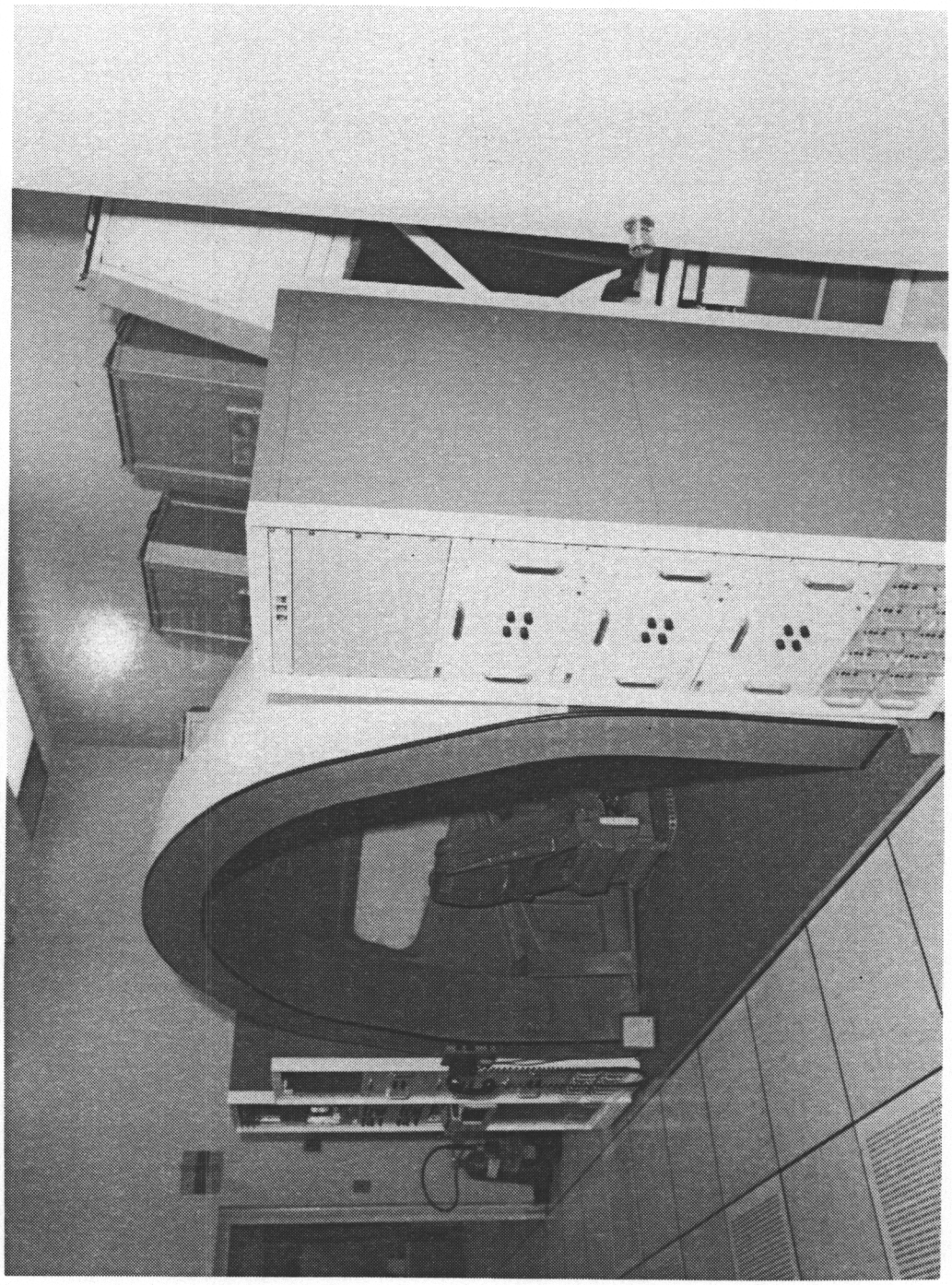


Figure 8 - TSRV Cockpit Exteroir (Rear View)

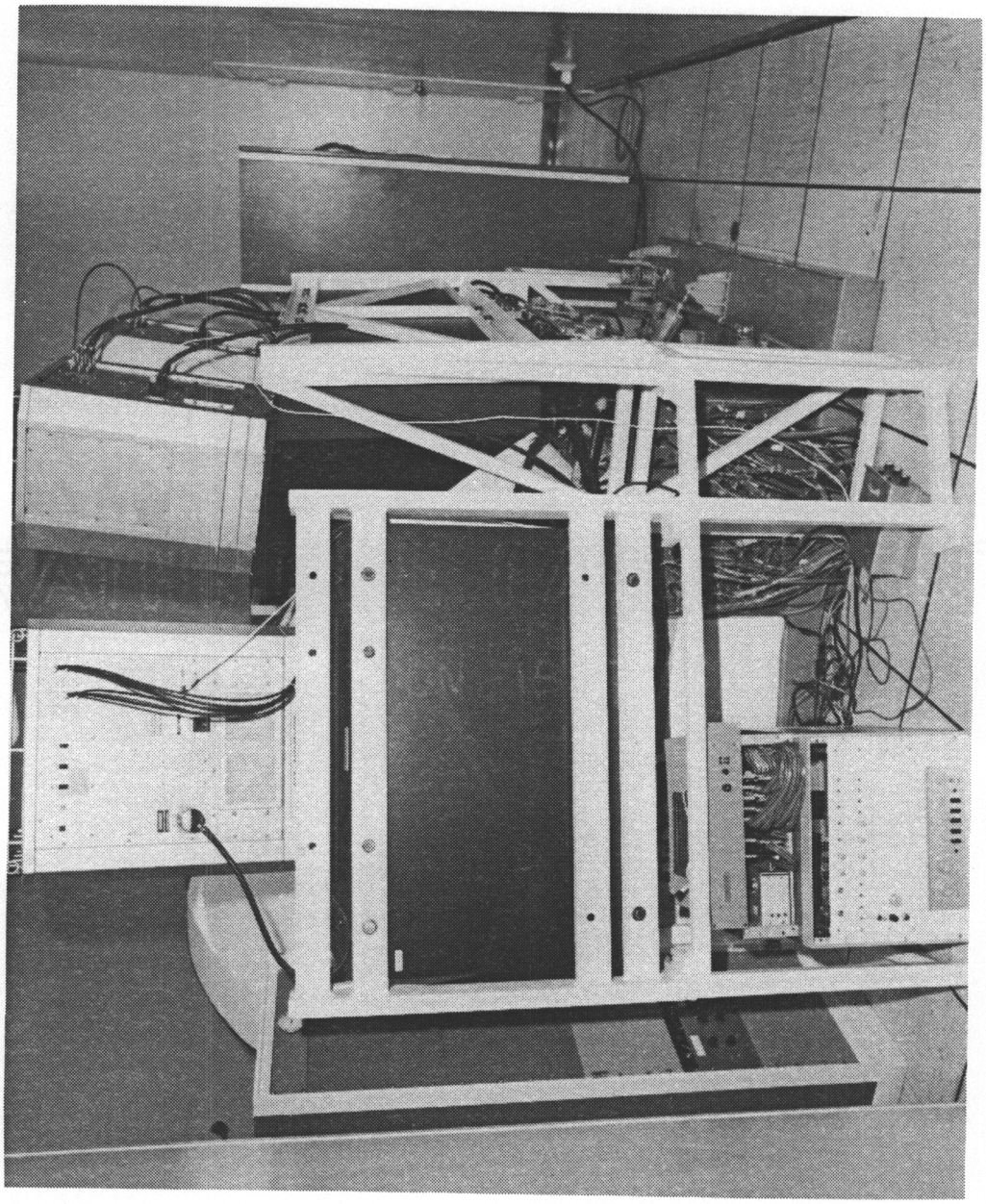


Figure 9 - TSRV Cockpit Monitors and Optics

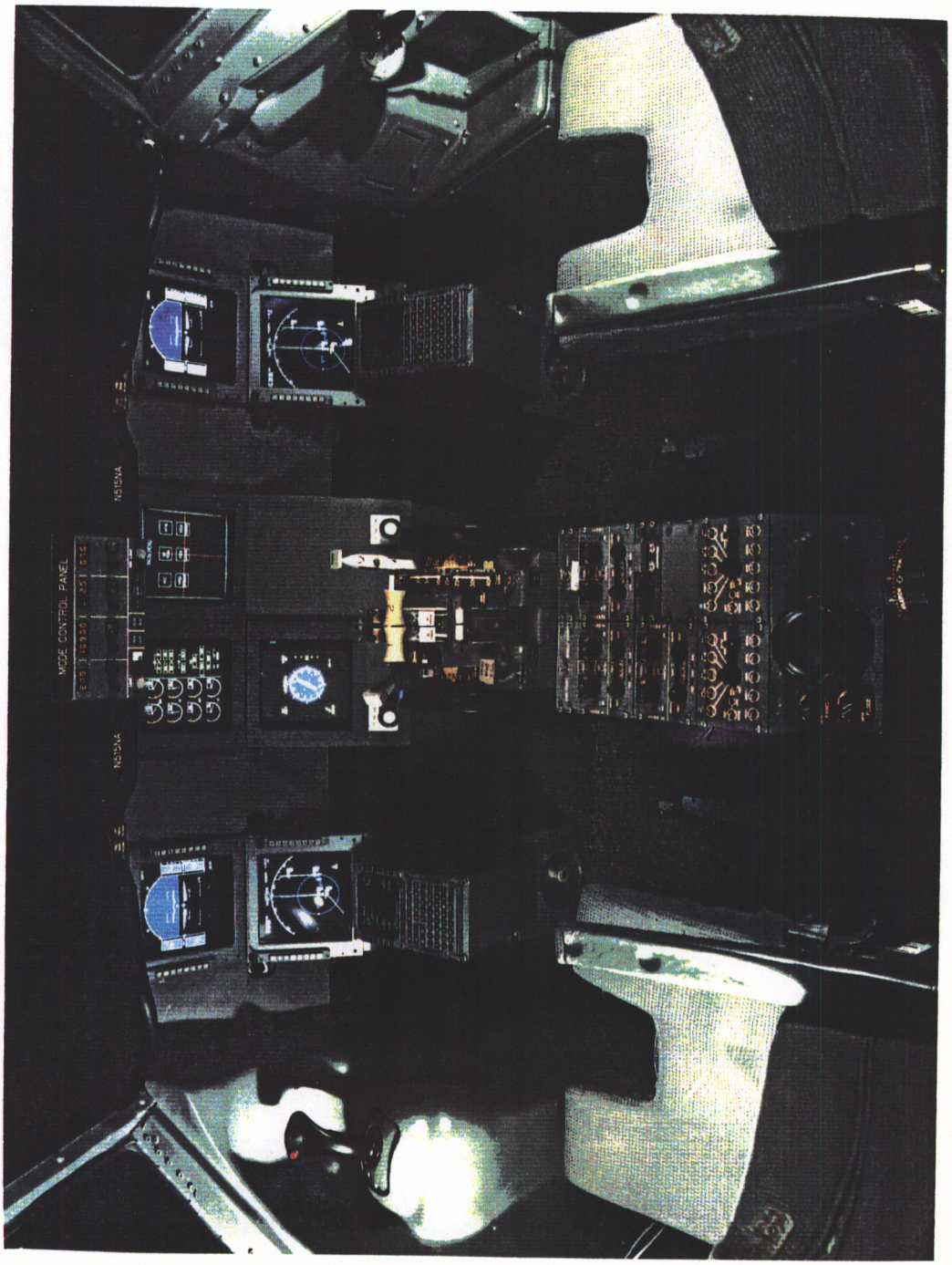


Figure 10 - TSRV Cockpit Interior

Control (ATC) and advanced display research.

Visual Motion Simulator

The VMS is a general purpose simulator which consists of a two person cockpit mounted on a six-degree-of-freedom synergistic motion base.[14] The exterior of the simulator is shown in Figure 11 and the interior is shown in Figure 12. The left side of the cockpit is configured to simulated advanced hypersonic aircraft such as the NASP. The right side of the cockpit can be configured to simulate helicopters. Like the TSRV the simulator is configured with a communications system, side-arm controller, throttle and rudder controls, CGI monitors and CRDS monitors. The VMS is used mostly for research into control laws for hypersonic aircraft.

Control Loader

Every cockpit at the FSF has at least one control loader. The VMS and the TSRV each have a McFadden sidearm control loader. Figure 13 show the sidearm unit mounted in the TSRV cab which is controlled by an analog computer mounted in the far left rack in Figure 9. The control loader converts stick and rudder motions, via the analog computer, into control-surface deflections and provides force feedback, though a controlled hydraulic system, cues to the pilot which assists him in evaluating the aircraft's dynamic state.[15] On the analog computer a second order differential equation approximates the forces and delays exhibited in the real aircraft. The position of the stick is feed to the simulation computer through the ARTS network.

Video Distribution System

The CGI and CRDS are centrally located so that they may provide graphics to any simulator. They are connected to the

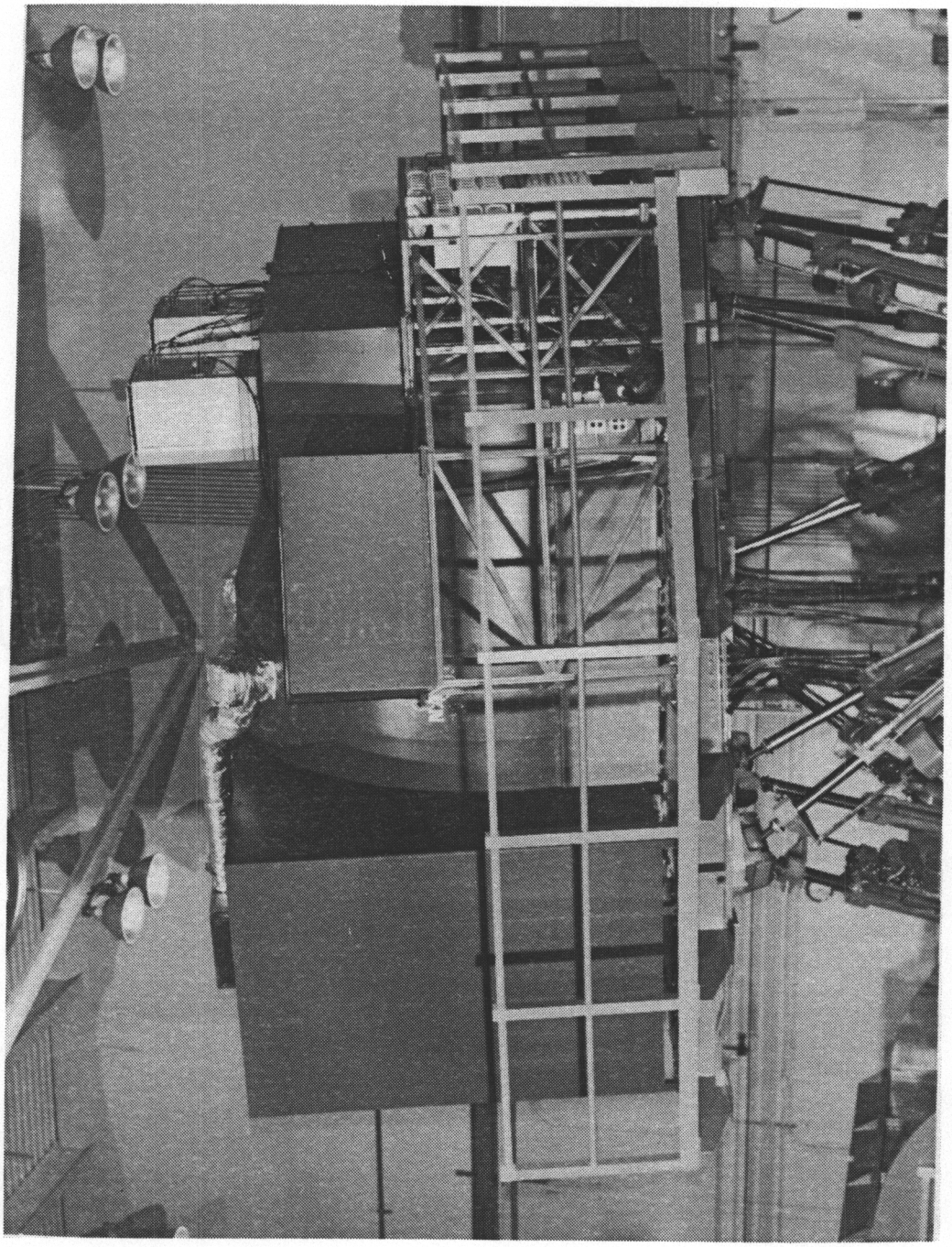


Figure 11 - VMS Platform and Cockpit

Figure 12 - VMS Cockpit Interior

Figure 13 - McFadden Control Loader

simulators through a video distribution system. The system, shown in Figure 14, can distribute images to monitors, located at each cockpit, for utilization. To provide out-the-window visuals the images from the CGI are displayed on an XKD monitor, through collimating optics (see Figure 15). Both the TSRV and the VMS each have four monitors located around the front of the cockpit. The CRDS graphics are displayed on the 8 inch XYTRON monitors in each cockpit (see Figures 10, 12 and 15). The VMS cockpit has six XYTRON monitors and the TSRV has eight XYTRON monitors.

Advanced Real-Time Simulation System

The ARTS system, which makes up the simulation LAN, allows the simulation equipment to communicate with each other as required.[16] Each piece of equipment interfaces to the ARTS fiber optic highway through Computer Automated Measurement And Control (CAMAC) crates (see Figure 16).[17,18] At each remote equipment site the CAMAC crate contains all of the DACs, ADCs, and digital interfaces required by that site. The controlling software, called the supervisor, resides on the central computer and handles the communication synchronization. Data is read in synchronously and can be shipped out to the crates either synchronously or asynchronously. Synchronous data is always input at the beginning of the simulation frame and output at the end of the simulation frame, synchronously with a frame clock. Asynchronously data can be shipped anytime during the frame except during the transmission of synchronous data. Since data is only read in synchronously the data may have actually changed anytime since the last input was read, the average transport delay due this can be represented by the Zero-Order-Hold (ZOH) effect and corresponds to one-half the sampling rate.[19] This would yield an expected transport delay due to sampling of 15 ms for a 30 ms sampling rate (one-half of 30 ms). Since the simulation computer is prohibited from executing the simulation during the shipment of synchronous data, the larger the

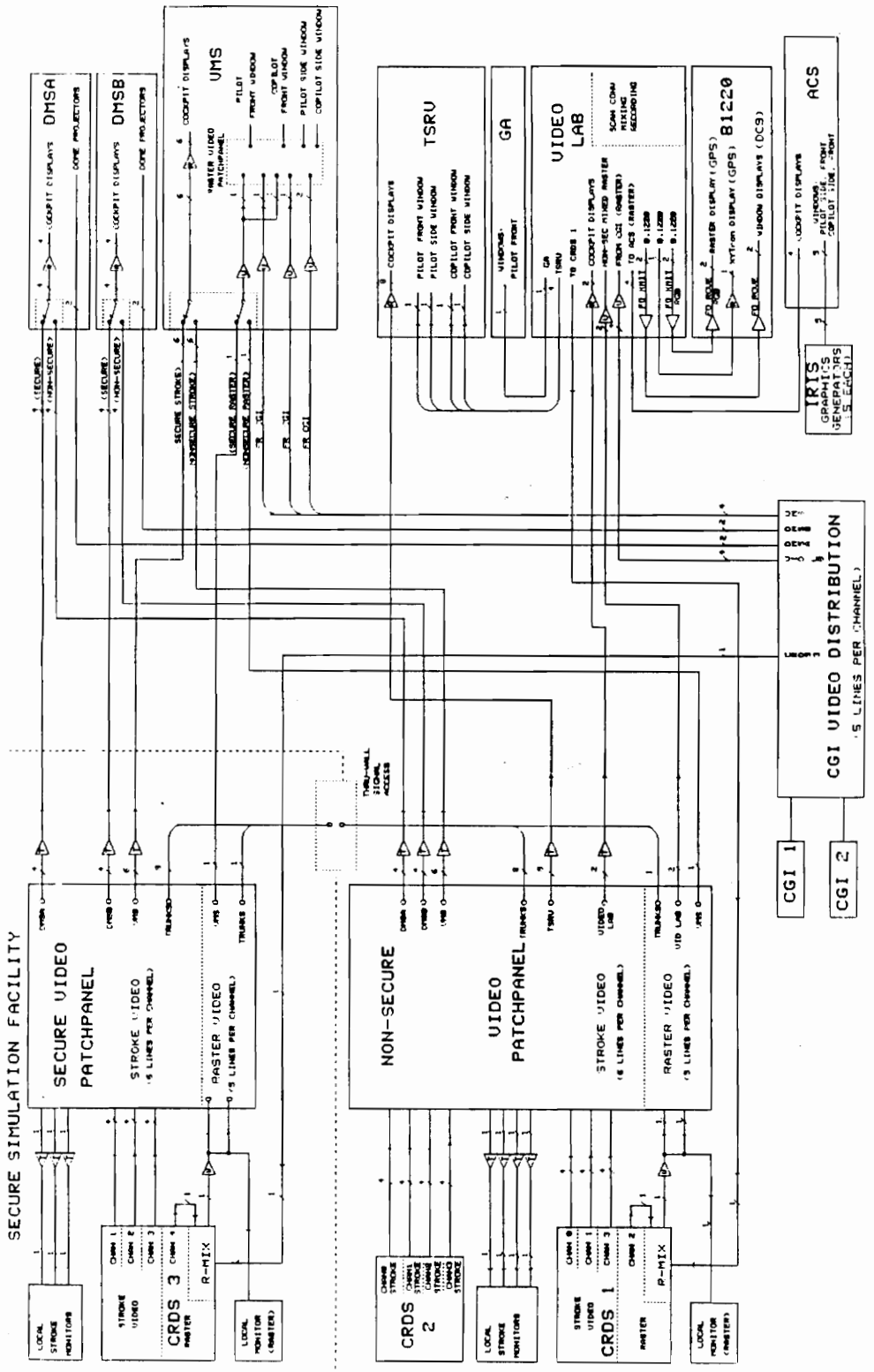


Figure 14 - Video Distribution System

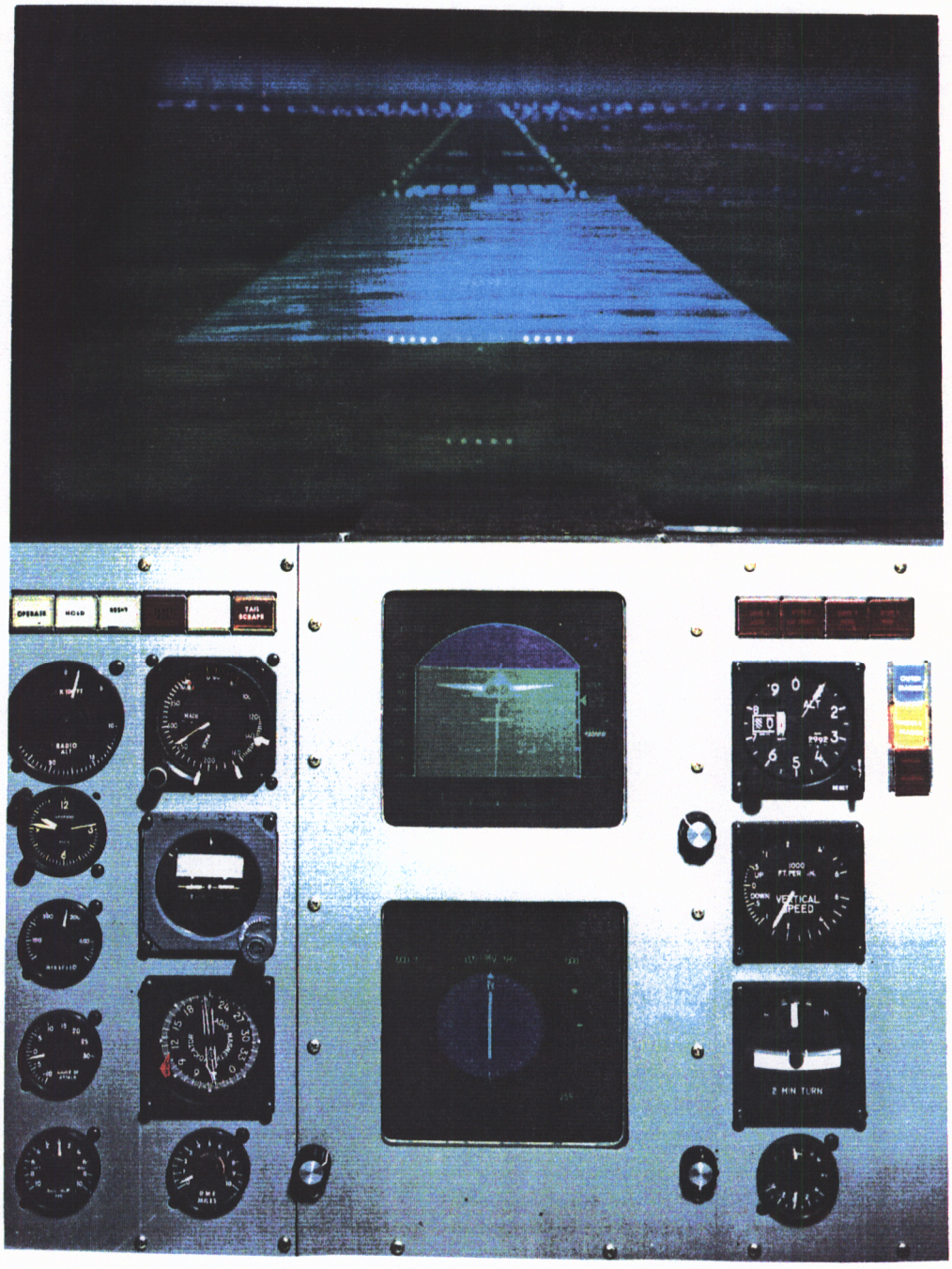


Figure 15 - XKD Monitor

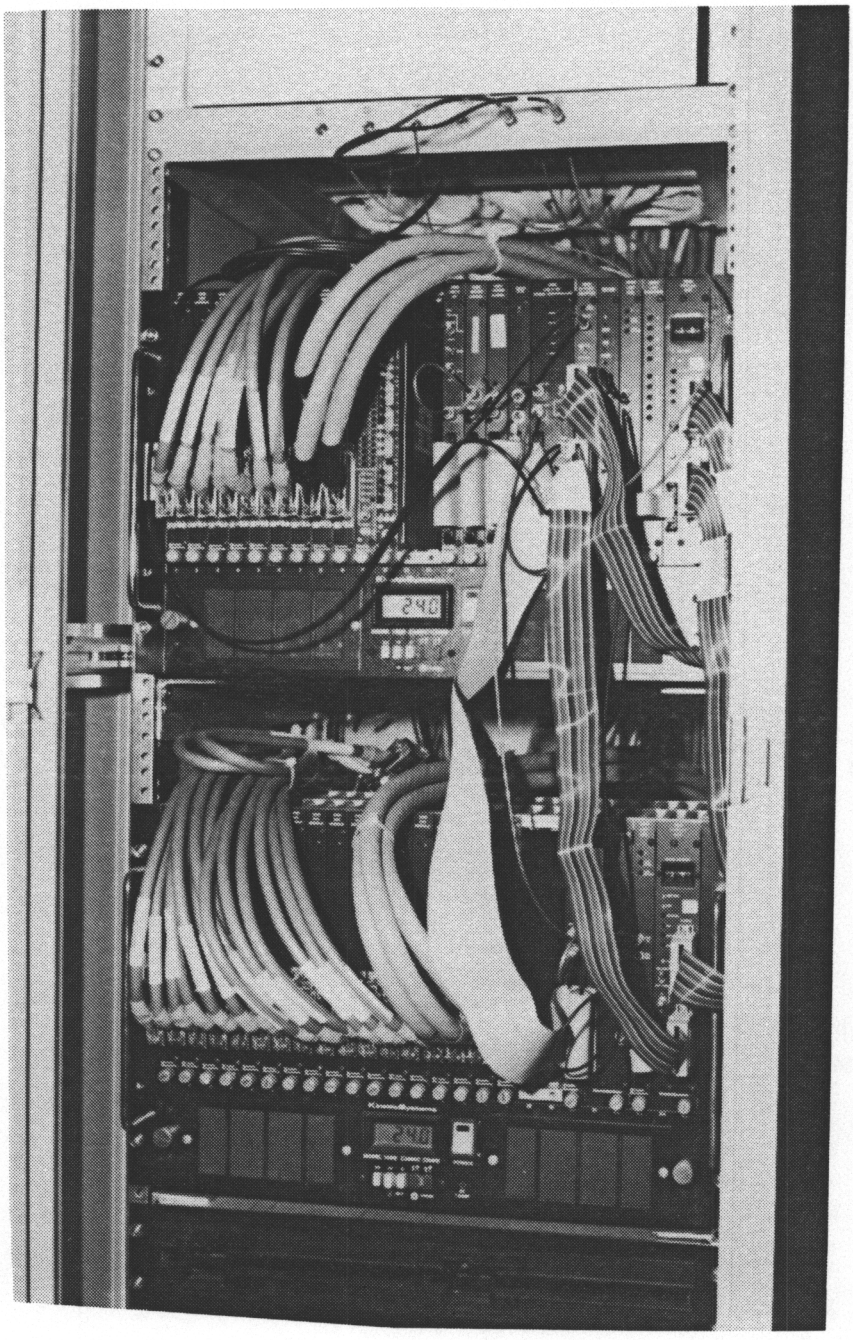


Figure 16 - CAMAC Crate

amount of data to be shipped, the less time the central computer has to execute the math model. Therefore, due to the large amount of data to be shipped, and in order to maximize the amount of execution time, the data shipped to the CGI and CRDS is shipped asynchronously. The remaining data to the simulation equipment is shipped synchronously.

Simulation Computers

The central computers are two Control Data Corporation (CDC) CYBER 175s (see Figure 17). Several other CYBERs are available for program development and checkout in a "non-real-time mode". The processing power for each CYBER is roughly equivalent to ten VAX 11/780s. The CYBER is capable of operating multiple simulations at the same time, as long as there is enough Computer Processing Unit (CPU) power and memory available. The amount of CPU power available is highly dependent upon the update rate, or frame rate, of the simulation. The update rate of a simulation typically runs at 33 Hz (or 30.0 ms). In other words, all data input, calculations and data output necessary to operate the simulation are executed and transmitted through the ARTS system 33 times every second. Faster iteration rates will reduce the asynchronous delays associated with sampling the ADCs and loading the DACs on the ARTS system due to the reduced ZOH effect. In general, with simulations operating at 33 Hz, each CYBER will operate two simultaneous simulations. Since the FSF has two CYBERS devoted to real-time the FSF will typically operate four, simultaneous, real-time simulations. These simulations are usually be changed every three hours, to support research into many different vehicles, and the facility is usually operated 16 hours per day.

Computer Generated Imagery System

The E&S CT6 consists of 4 output channels with each channel

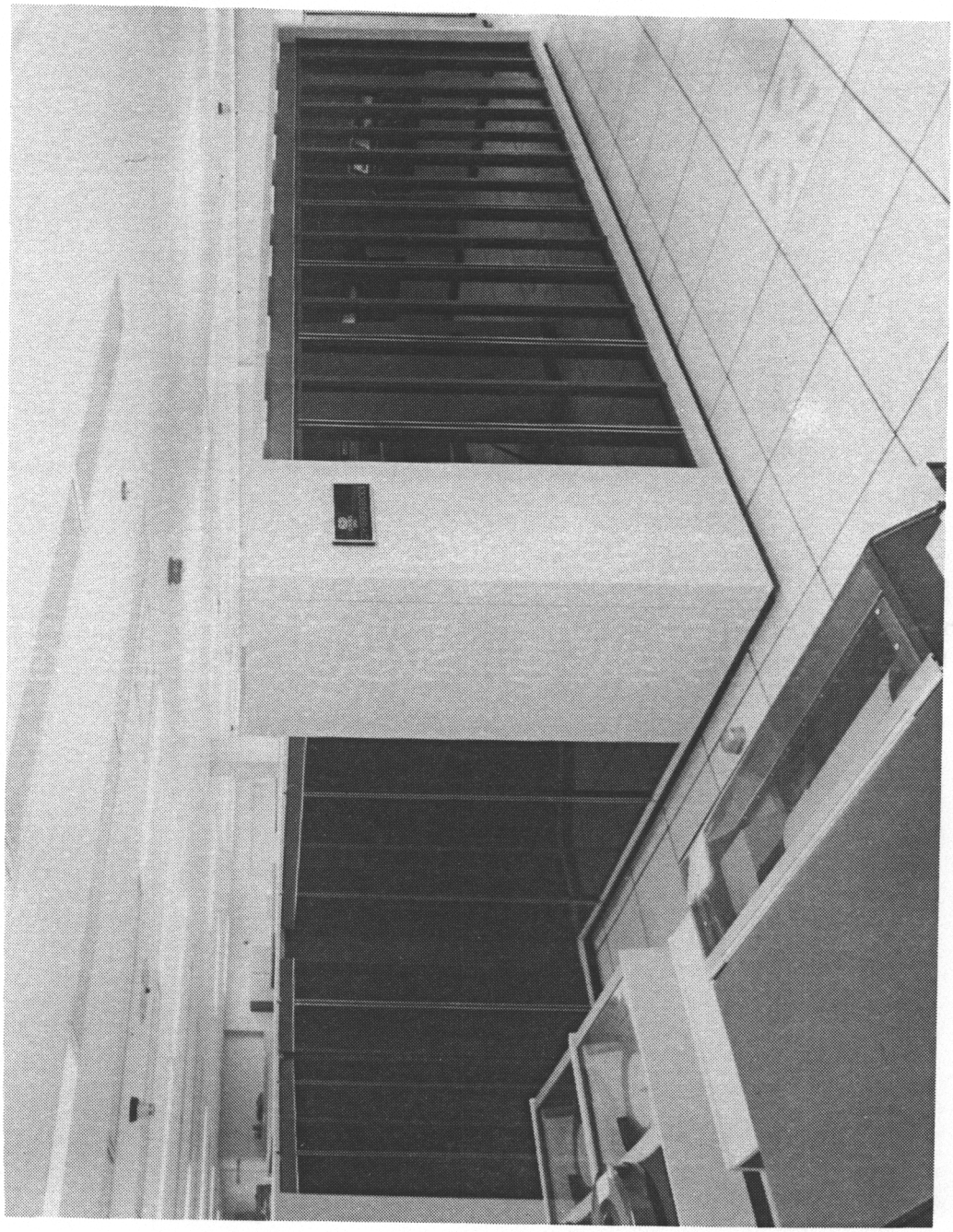


Figure 17 - CYBER Computer

capable of displaying 200,000 pixels (see Figure 18). The CGI is capable of supporting two independent, simultaneous simulations through two independent eyepoints. Two databases are available for simulations to utilize. The first database consists of generic terrain and is used in the fighter simulations (see Figure 2) and the second is a detailed representation of Denver Stapleton Airport, and 200 nautical miles of surrounding terrain (see Figure 19). The CGI system processes real-time commands, sent to it via the ARTS highway, on a Gould 67/80 computer. The Gould controls the two eyepoints of the CGI and formats data to be passed to it. The image rendered on the monitors is dependent upon the positional information passed to the Gould from the CYBER controlling the simulation. The CGI operates asynchronously from the central computers and has an image update rate of 50 Hz. In other words a new image is drawn every 20.0 ms. If positional data is not received within 20.0 ms of the last data packet, the CGI will predict the next position and render it while waiting for new positional data. The CGI has a pipelined architecture and consists of four stages each requiring 20 ms to complete. Therefore the CGI's specified transport delay is 80 ms. In addition, since the CGI is operating asynchronously with the CYBER the ZOH transport delay due to sampling the crate is 10.0 ms (0.5 * 20.0 ms). Therefore, the total expected average transport delay in the CGI is 90.0 ms (80.0 ms + 10.0 ms).

Calligraphic/Raster Display System

The CRDS consists of three Terabit Eagle 1000s (see Figure 20). Two of the Eagle 1000s each can drive eight calligraphic monitors and one raster monitor. The third Eagle drives eight calligraphic monitors with increased graphical computational power. Typical cockpit displays are shown in Figures 10, 12, 15 and 21. Figure 22 shows how the raster channel can be mixed with an incoming CGI picture to simulate a HUD on out-the-window monitors. The

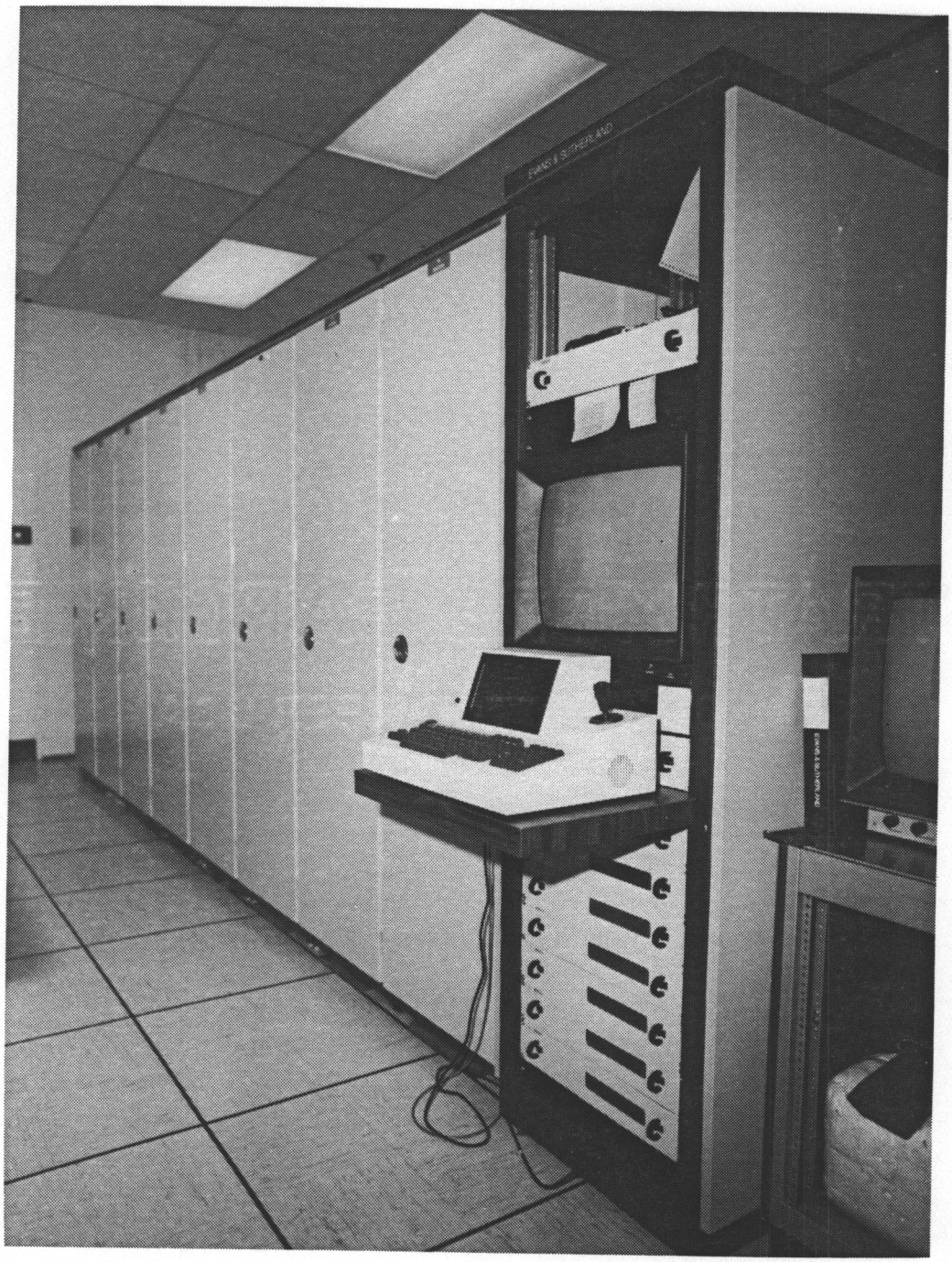


Figure 18 - CGI Hardware

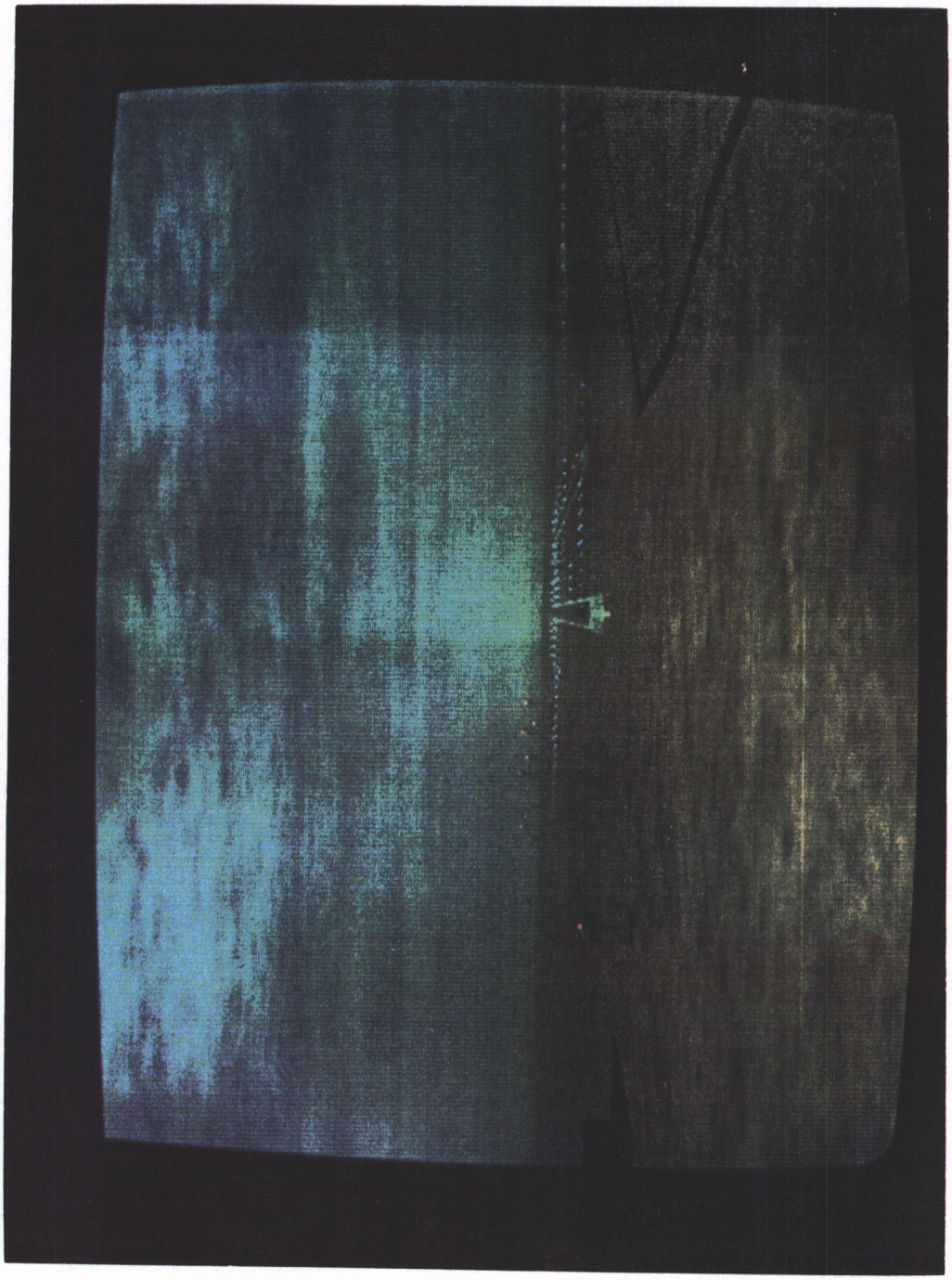


Figure 19 - Denver Stapleton Airport

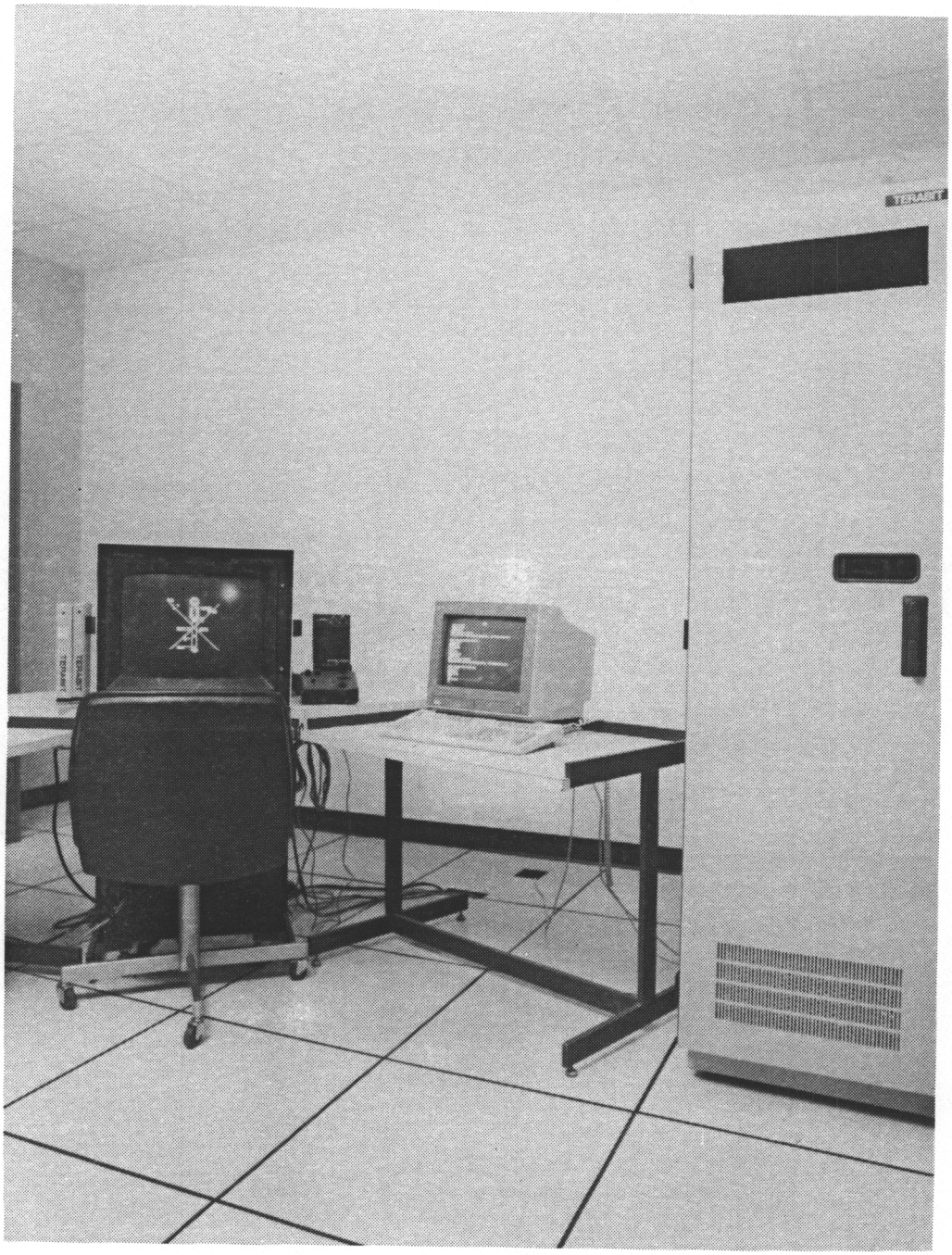


Figure 20 - CRDS Hardware

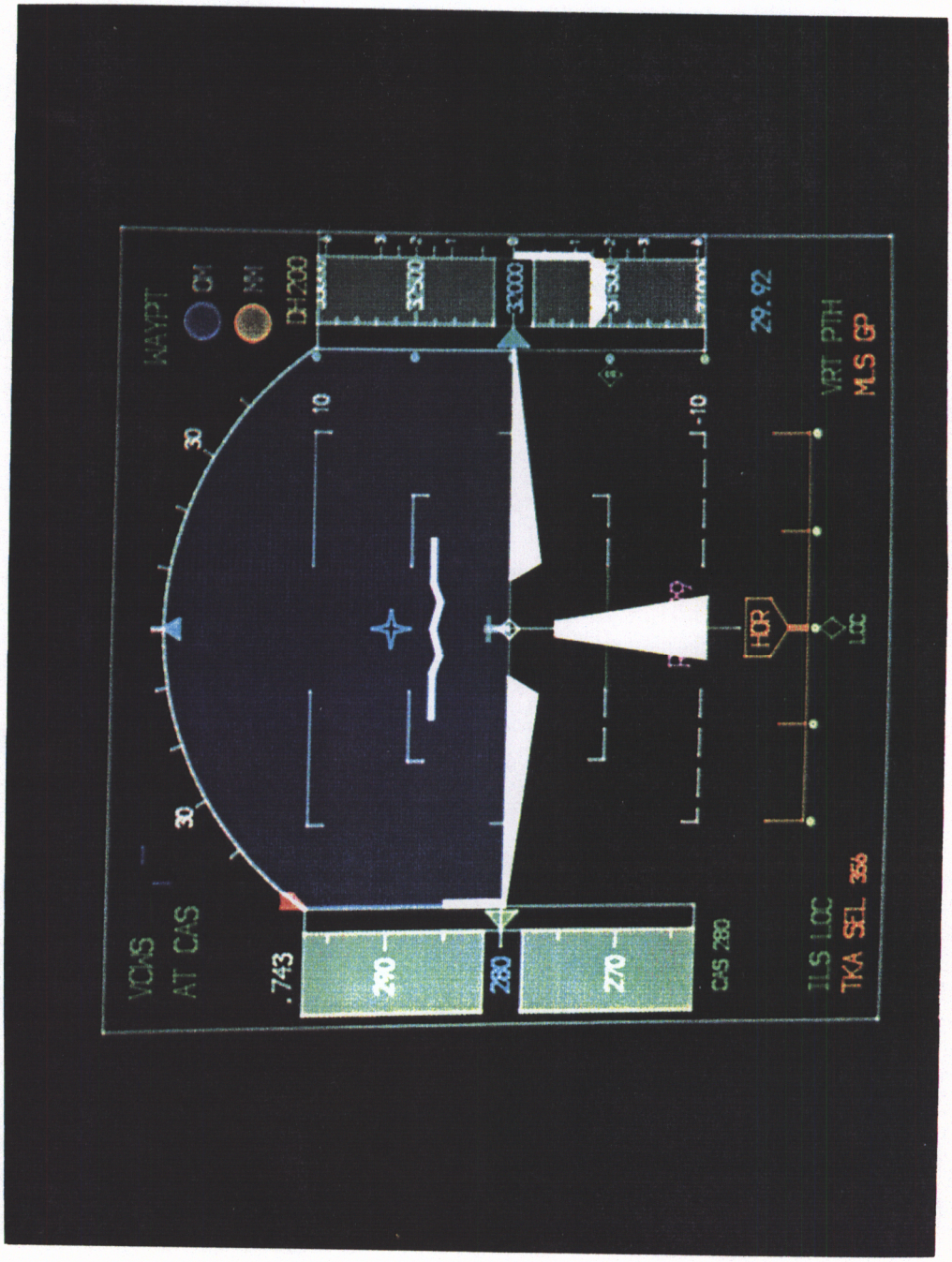


Figure 21 - CRDS Calligraphic Display

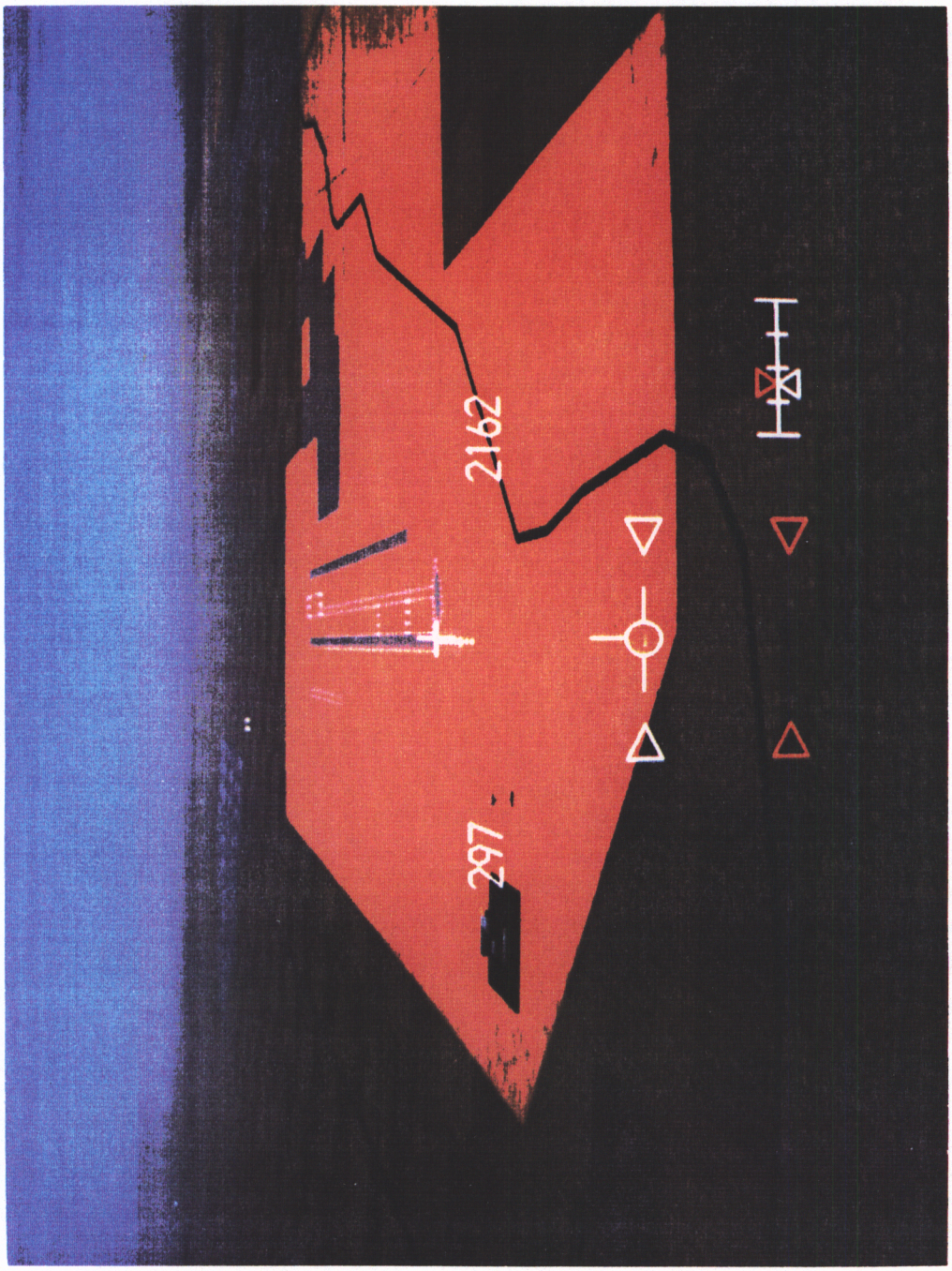


Figure 22 - CRDS Mixed Display

transport delay for each Eagle is dependent upon the update rate of the display. The more complex the display, the longer it will take the CRDS output device to render the image once a display list has been generated. The Eagle's output update rate will determine the frame rate for the entire graphics pipeline. The transport delay is specified to be four times the update rate plus the ZOH delay due to the asynchronous operation of the CRDS. Typical displays have update rates of either 60 Hz or 40 Hz. At a 60 Hz update rate the transport delay of the CRDS is expected to be 66.7 ms (4 * 16.7 ms). In addition, since the CRDS is operating asynchronously with the CYBER the transport delay due to the ZOH effect of sampling the crate is 8.3 ms (one-half of 16.7 ms). Therefore, the total expected average transport delay in the CRDS is 75.0 ms (66.7 ms + 8.3 ms) at 60 Hz and 112.5 ms (4 * 25.0 ms + 0.5 * 25.0 ms) at 40 Hz.

CHAPTER 4

TEST SETUP

General Setup

The simulation hardware required for the TSRV simulation is shown in Figure 23. The equipment required for the VMS simulation is similar except that a motion base is required. The TSRV cockpit has eight XYTRON monitors on which one of the three Eagle 1000s in the CRDS can display electronic instrumentation and four XKD monitors for out-the-window visuals. The VMS cockpit has six XYTRON and 4 XKD monitors. Only one of the CGI eyepoints is required by the TSRV or VMS simulation. The hardware is tied to the CYBER 175 through the ARTS system through a serial highway driver and a peripheral processing unit. The signal flow shown in Figure 24 depicts the entire path an input signal at the control loader must take to cause a change in the CGI or CRDS. The major components of the simulation (the control loader, the ARTS/CYBER system, the CGI and the CRDS) each have their own transport delays. The test setup for measuring the delay in each piece of equipment is described below. The test setup for measurement technique and domain determination is discussed below. Also discussed are any changes that must be made to hardware and software to run these tests.

Measurement Technique Setup

Chapter 2 discussed several methods for measuring transport delay. The first method discussed utilized the method that has become a standard in industry, the PED. The PED will be compared to the VLD in a side by side comparison. The PED is located as close to the upper-left hand corner of the visual window as possible (the TSRV collimation optics prevent direct contact to the monitor. The VLD is placed in line with the video signals feeding that monitor. The logic analyzer monitors the time elapsed from a

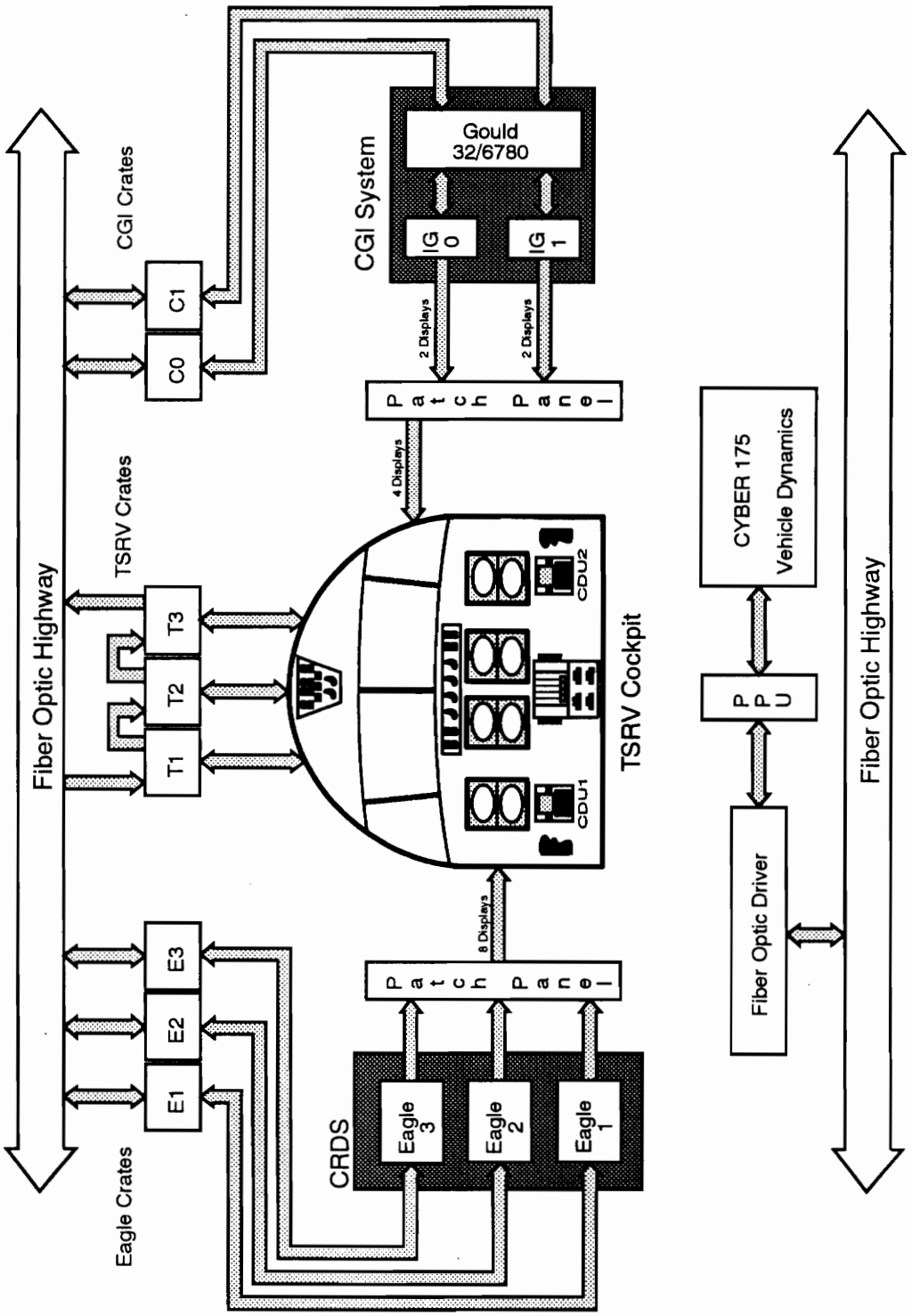


Figure 23 - TSRV Simulation Test Hardware

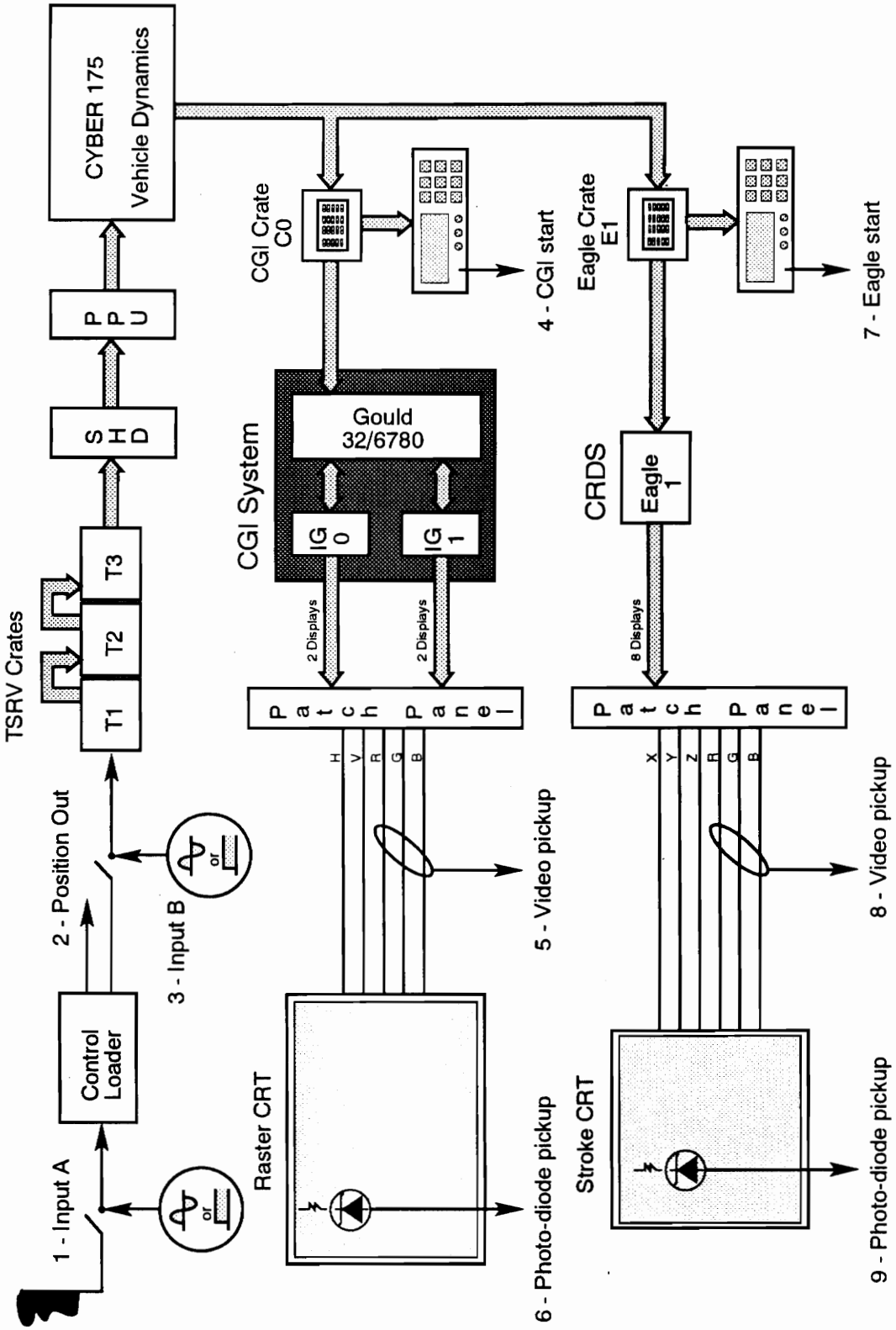


Figure 24 - Signal Flow Diagram

step input until a response from the PED and the VLD. The logic analyzer is also directly monitoring when the video actually makes a transition. The output of the CGI will be toggled from a black to a white image by the method described in CGI Setup. The position of the eyepoint will be translated via local CGI control therefore real-time support is not required for this portion of the testing.

Frequency Domain Setup

In order to record measurements in the frequency domain, refer to chapter 2 for requirements, the FRA is connected to the equipment to be tested in the following manner. The Bafco 916 is a dual channel FRA that also contains a precision low frequency generator. Since it is impossible for a human pilot to excite the stick with a near perfect sinusoidal an artificial signal will be substituted for the pilot's input. The generator is set to output a sinusoid, ± 2.0 Volts peak-to-peak (V_{pp}), and it is fed into channel 1 of the FRA and the simulation equipment being tested. The sinusoid is then input to the control loader or the TSRV pitch axis ADC, located in the TSRV crate. If the control loader is being tested the position output of the control loader is then input into channel 2 of the FRA (for more details see the section on Control Loader Setup). If the output of the transport delay of the simulation is being tested then the output of the PED or the VLD, depending on the measurement technique selected, will be input to channel 2 of the FRA. During the simulation testing the simulation program, without the math model in-the-loop, will monitor the pitch axis ADC from the control loader and output a specific position to the CGI, that corresponds to a white or a black screen, based upon the input voltage (for more information see CYBER 175 Setup, Control Loader and CGI Setup). The FRA will then continually monitor and display the phase difference between channel 1 and 2. The frequency of the sinusoid will be varied to observe the effect on the test equipment.

Time Domain Setup

Since it is impossible for a human pilot to excite the stick with a near perfect step an artificial signal will be substituted for the pilot's input. The setup for the time domain consists of injecting a step TTL (0.15 V to 4.15 V) signal from a debounced switch at either the control loader or the ADC (see Control Loader Setup for details on injection points). Figure 25 shows a shcematic of the step generator. Again, depending on the test being conducted, the position output of the control loader, the PED or the VLD is then monitored with a logic analyzer or a strip chart recorder to record times to response. If the output of the simulation is being tested then the input ADC from the pitch axis is monitored by the simulation program, without the math model in-the-loop, for a transition from a TTL low to a TTL high (for more information see CYBER 175 Setup, Control Loader and CGI Setup). Multiple steps will be fed to the system in order to obtain an average trasport delay.

Control Loader Setup

The control loader's transport delay is measured from the input of a sinusoid or step input at the pitch axis, Δ, force transducer (node 1, Figure 24), to the output of the stick position to the host computer (node 2, Figure 24). To do this the normal inputs from the force transducer in the stick are disconnected from the signal conditioning card. Figure 26 show the schematic for the control loader signal conditioner card. The ΔP inputs on pins 2 and 3 are disconnected from the stick to open the control system loop and eliminate feedback effects. The sinusoid or step is injected into the TP2 node to simulate the signal that would normally be generated by the pilot. The output of the control loader that is normally sent to the simulation porgram is a voltage that corresponds to the stick position. This voltage can be monitored at TP7 to determine the transport delay of the control loader.

Since the control loader is actual flight hardware and as such it's transport delay is not included in the total hardware transport delay, it is

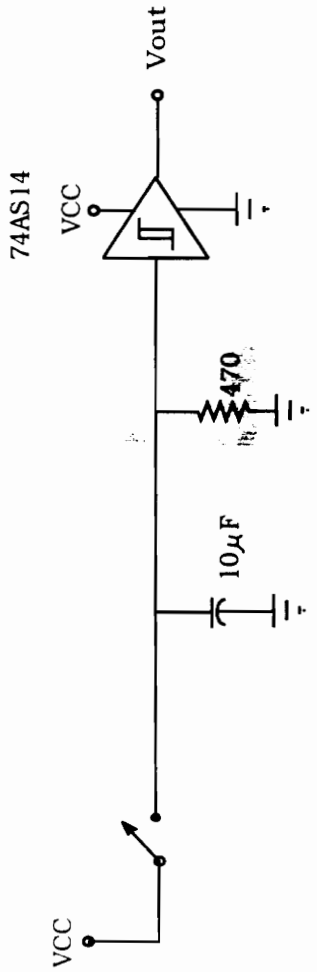


Figure 25 - Step Generator

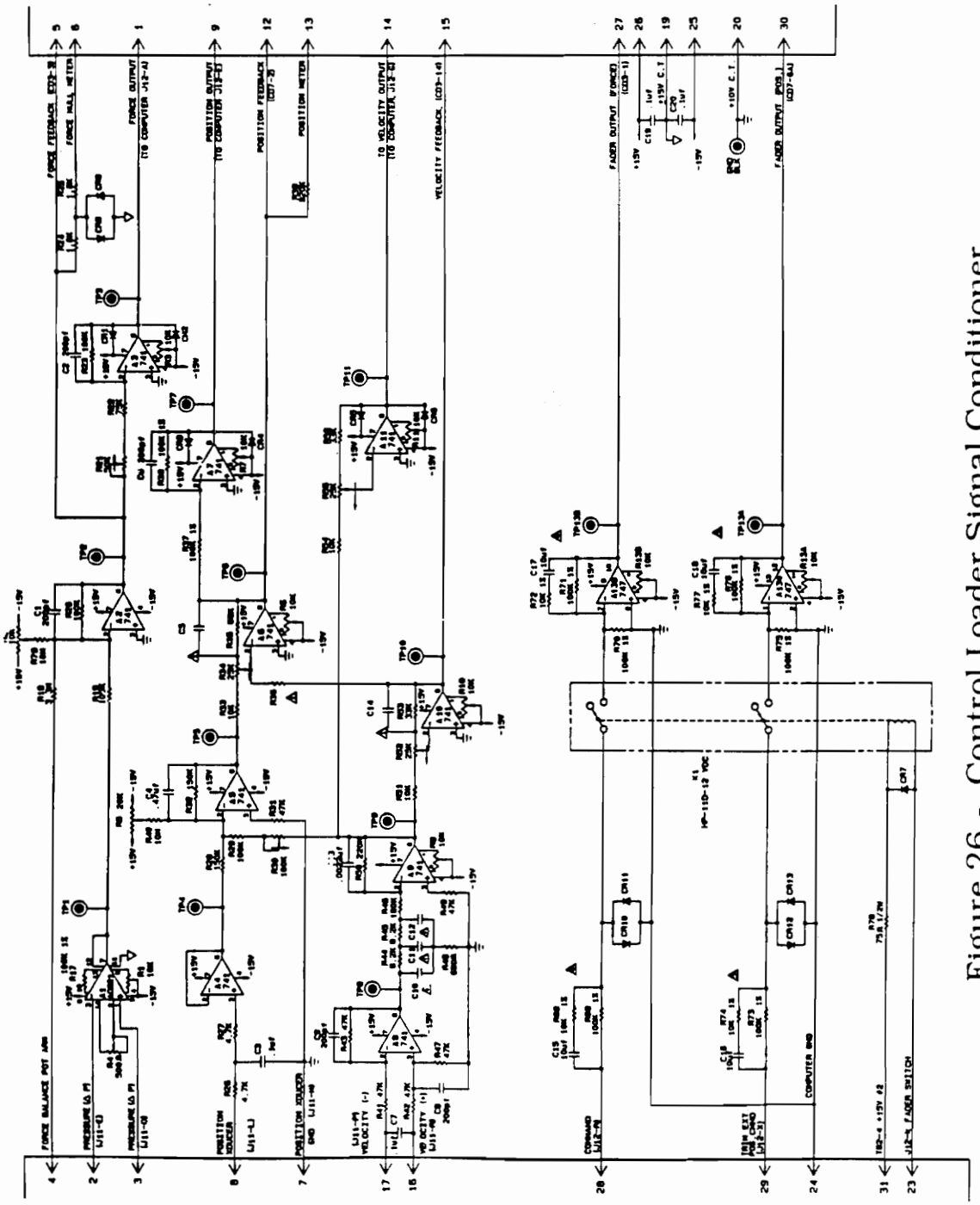


Figure 26 - Control Loader Signal Conditioner

bypassed for the remainder of the tests. To do this the position output of the control loader is disconnected from the crate and the sinusoid or step is then presented to the ADC. The simulated position is converted to digital information, by an ADC in the TSRV crate, and read by the simulation program at the beginning of each frame.

CYBER 175 Setup

Referring to Figure 24, the ARTS/CYBER delay is measured from the input of a step or a sinusoid at the pitch ADC in the cockpit, node 3, to the output of the system at the CGI crate or the Eagle crate, nodes 4 or 7. The ADC data is read into the CYBER 175 at the beginning of each real-time frame, once every 30.0 ms. In order to detect that a change has occurred at the cockpit the real-time program is modified to monitor that ADC and make decisions based on the voltage at the control loader. Appendix B contains the modifications required to the real-time program. In normal operation, when there is no deflection of the stick, the voltage sent to the CYBER 175 is approximately 0.0 V. If the pilot makes an input (moves the stick) the output voltage of the control loader can vary between ± 10.0 V.

For time domain testing the real-time code is modified such that when a step function is input to the ADC, the program will detect when the voltage it reads has exceeded a preset threshold, indicating a transition has taken place. If such a transition does occur then the CGI and CRDS are sent the required information that corresponds to a white or black screen (for more information see CGI Setup and CRDS Setup). Since the step function varies between 0.15 V and 4.15 V an arbitrary threshold was selected of 0.5 V. In other words, when the input voltage is $\geq$ 0.5 Volts (V) the CGI and CRDS will be commanded to display a white screen. When the input voltage is $<$ 0.5 V then the CGI and CRDS will be commanded to display a black screen.

For frequency domain testing the real-time code is modified such that when an sinusoid is input to the ADC, the program will detect when the voltage it reads has exceeded a preset threshold, indicating a transition has taken place. If such a transition does occur then the CGI and CRDS are

sent the required information that corresponds to a white or black screen (for more information see CGI Setup and CRDS Setup). Since the sinusoid varies between ± 2.0 V an threshold of 0.0 V was selected. In other words, when the input voltage is ≥ 0.0 Volts (V) the CGI and CRDS will be commanded to display a white screen. When the input voltage is < 0.0 V then the CGI and CRDS will be commanded to display a black screen. The program variable that controls the decision breakpoint, for both the time and frequency domain cases, is called COLBP.

A software switch was developed to determine whether or not the math model will be executed. Appendix B shows that the variable NOPTION controlled this decision. When NOPTION is set to 0 no modified code will be implemented. When NOPTION is set to 4 the real-time program will execute the math model while the output decision criteria is being executed. When NOPTION is set to 5 the math model is bypassed while the decision criteria is still executed.

For testing with the math model in-the-loop, NOPTION = 4, the aircraft's pitch axis was monitored for any deviation from a trimmed condition as a result of a response from the pitch ADC. To do this the aircraft is placed into a trimmed condition to stabilize the pitch variable, THETADEG (θ_{deg}). Once θ_{deg} has changed by a preset threshold amount the appropriate output is made to the CGI and CRDS indicating that the input has traveled through the math model. This is not to say that the output of the math model is correct, that requires more detailed analysis, but only that the onset of a response to a pitch input is occurring.

For testing the transport delay of the ARTS/CYBER section of the hardware and software without a math model in-the-loop, NOPTION = 5, several statements were inserted into the real-time program that will cause the program to skip all model dependant executable statements and then output to the CGI and CRDS a signal that indicates the pitch ADC has detected a change at the cockpit (see above decision criteria).

CGI Setup

Referring to Figure 24, the CGI system was timed using the logic

analyzer, which detected an input at node 4, to start the timing and the VLD or PED, at nodes 5 or 6, to stop the timing. In order to detect the output of the CGI at the logic analyzer, the CGI must be capable of rendering a black and white image to the monitor to give the maximum video level change (see Chapter 2 for a discussion of the PED or VLD). This is normally accomplished by developing a small black and white database on the CGI and translating the eyepoint within that database to yield the necessary changes. This can be a misleading results since the normal operational database is not being executed. Therefore, it is necessary to test the transport delay with the operational database. In order to get black and white transitions from the operational database a 737 aircraft was positioned in front of the eyepoint at a 90° roll angle and so close to the aircraft landing gear that the entire screen was filled with a black image. By translating the eyepoint a few feet in altitude, z, the screen will be filled with white from the wing of the aircraft (see Figure 27). By changing the screen from black to white the PED and VLD both have a input upon which to detect changes in data. The outputs of the PED and VLD are monitored by the logic analyzer. This method also allows the detection of transition from one state to another by monitoring the incoming data for a particular pattern that corresponds to the eyepoint position. When a transition from one state to another is required the real-time program will send down to the CGI a predetermined set of database coordinates (x_1, y_1, z_1, $heading_1$, $pitch_1$, $roll_1$) to display a black screen and another set of data base coordinates (x_1, y_1, z_2, h_1, p_1, r_1) that differs only by altitude, z_2, to display a white screen. The output of the CGI can then be toggled between black and white by toggling between z_1 and z_2. The logic analyzer can then time the CGI transport delay from the receipt of a new altitude command until the appropriate image is drawn on the screen.

CRDS Setup

The CRDS transport delay time is measured in a setup similar to the CGI's in that a logic analyzer was used to detect an input at node 7, to start the timing, and the PED or the VLD were used at nodes 8 and 9 to end the

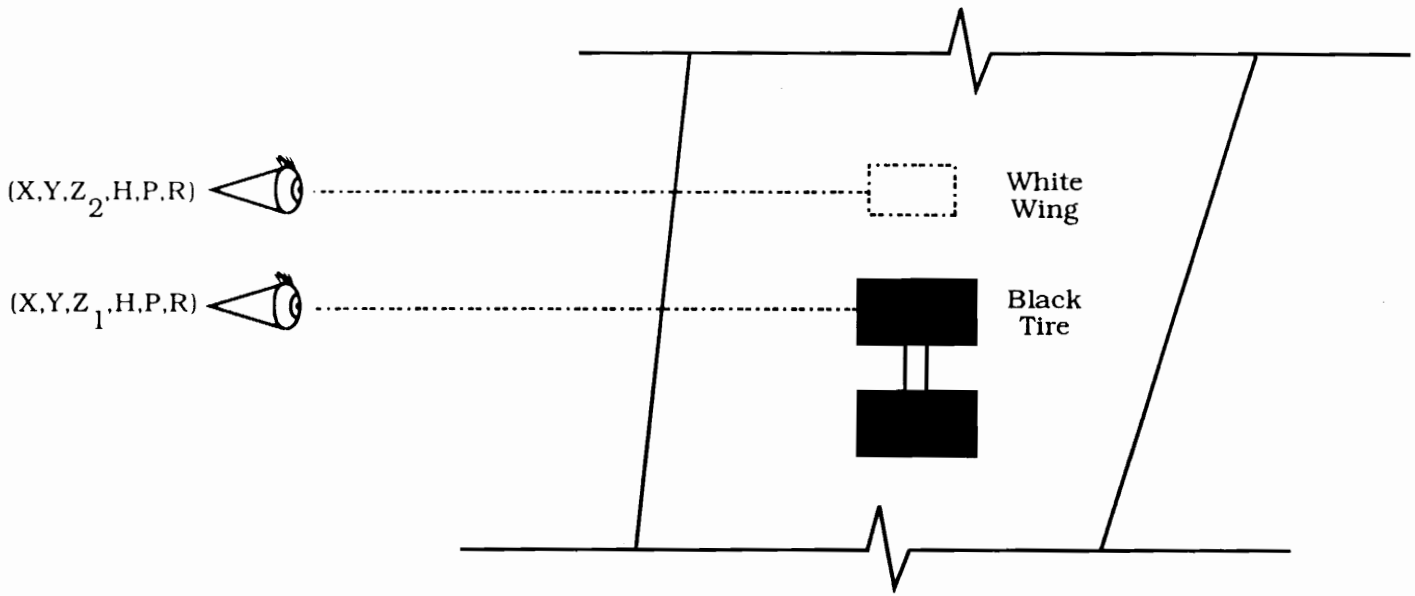


Figure 27 - CGI Visual Test Setup

timing. Again the PED and VLD require a display to cause a change in video level therefore it is necessary to develop a simple program capable of displaying a small white square (code for this display is located in Appendix C). The square can then be turned on and off by the use of a flag transmitted to the CRDS from the CYBER 175, which indicates that an event has occurred. The specific position sent to the CGI can also be sent to the CRDS to give the logic analyzer a start event for the timing is the arrival of a data packet that contains the changed flag. The stop event will be the detection of the beginning of the image being drawn by either the PED or the VLD.

Motion Base Setup

The transport delay of the motion base can be measured from the response of accelerometers located at the platform's center of gravity. The output of the accelerometer is an analog ± 10.0 V range which is not suitable for direct interface to the logic analyzer (requires inputs between ± 6.35 V). The circuit shown in Figure 28 converts the output of the accelerometer to TTL signals. The circuit is a comparator circuit that can be adjusted to output a TTL low when the motion base is in a trimmed condition. The gain of the circuit is high to exaggerate any small changes in the accelerometer signal to detect the onset of a change in the signal. The circuit is also nonlinear but that is not a concern since it is the onset of motion we are trying to detect and not any of the signal characteristics. The timing of the motion base will be conducted with a logic analyzer or an FRA using the input of a step or sinusoid, at the VMS crate, and the output of the accelerometer.

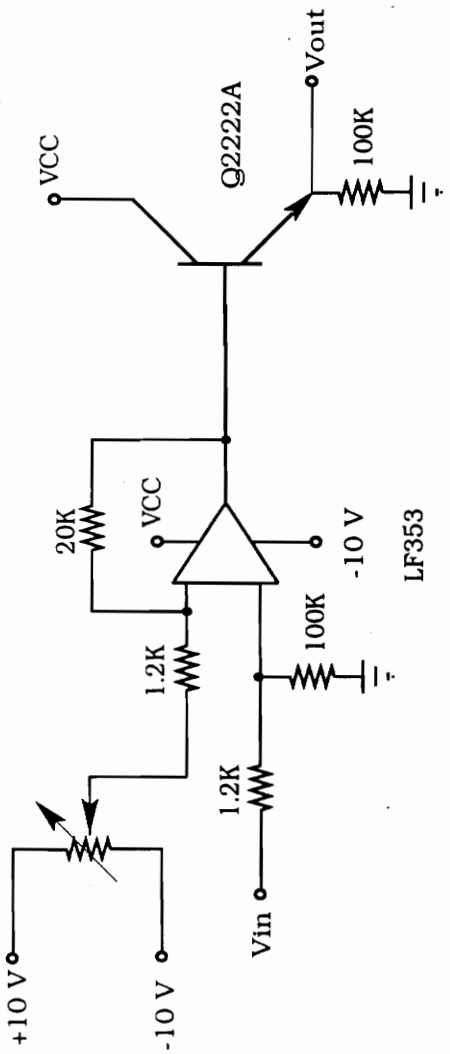


Figure 28 - Accelerometer Signal Conditioner

CHAPTER 5

RESULTS

Measurement Technique Comparison

The output from the PED, the VLD and the actual RGB video signal were monitored by the logic analyzer. Since the RGB video signal ultimately feeds both the VLD and PED it was considered the criterion against which the other signals will be measured. In Appendix D, Table D-1 twenty measurements are recorded using a step input to drive the output image from black to white. The average time until the VLD transition was recorded was 114.85 ms. The average time until the PED transition was recorded was 122.0 ms. The actual RGB video signal transition was observed to precede the VLD by a consistent 0.01 ms. Therefore, the PED lagged behind the actual video transition by 7.16 ms and the VLD lagged behind the video transition by 0.01 ms. Based upon the above data, the VLD was determined the most accurate measurement method and was utilized for the remainder of the transport delay measurements.

Domain Comparison

A step and a sinusoid were input directly into the pitch ADC of the TSRV crate. Again no math model calculations were executed. The output of the CGI was monitored using the VLD. For the frequency domain test, the output of the VLD was routed to the FRA while the pitch ADC was oscillated at various frequencies. Ideally, the FRA should yield a steady $\Delta\phi°$ reading at each frequency but due to the large amount of asynchronous delays in the system the readings fluctuated. Therefore twenty $\Delta\phi°$ readings were taken at each frequency (see Table E-1 in Appendix E for data). The equation discussed in chapter 2 is applied to the average $\Delta\phi°$ for each frequency and the results of the test are shown in Table 1. The measurements from the frequency domain were close to those obtained in

TABLE 1 - Frequency Domain Test Results

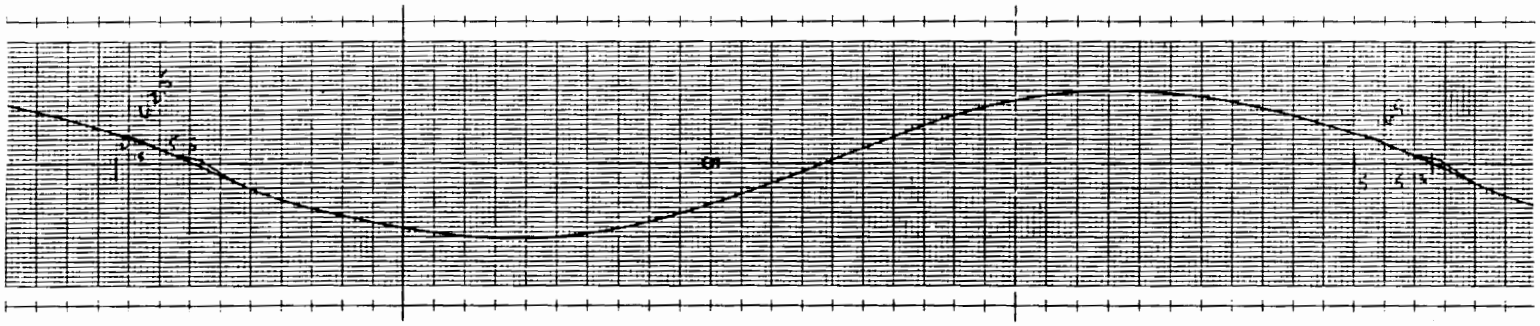
Frequency (Hz)	Delay (ms)
0.20	114.17
0.40	116.32
0.80	118.13
1.60	117.61
3.00	117.57

the time domain. The results were very close when the transport delays were measured using low frequencies (114.2 verses 114.9). In general, a low frequency input would be more indicative of the kind of input one would expect from a pilot during a simulation, particularly a transport simulation. Even at high frequencies the deviation is less than 4% of the total magnitude of the delay. Therefore, since the frequency and time domain measurements agreed closely with one another and much less data will be required for the time domain measurements it was decided to make all remaining measurements in the time domain.

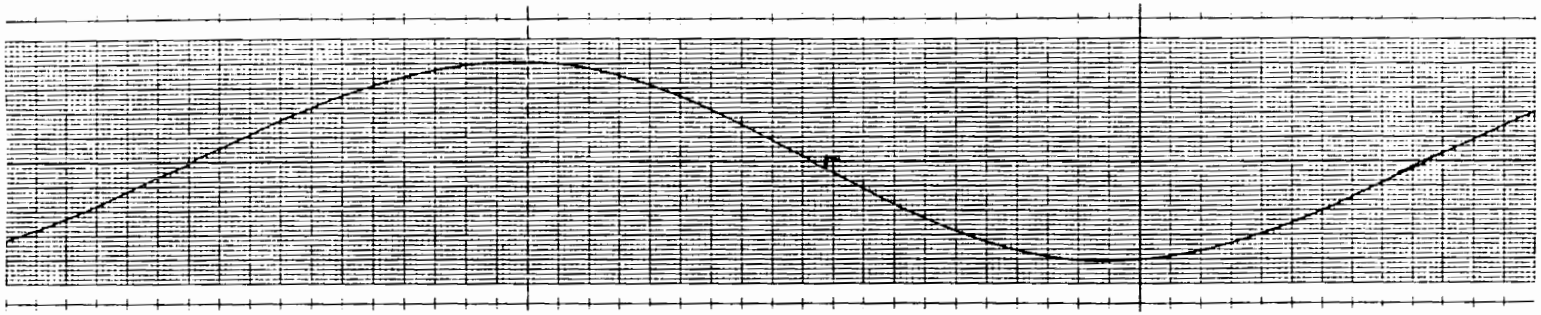
Hardware Transport Delays

Control Loader

The control loader was tested in both the frequency domain and the time domain. Since the control loader is an analog device, it begins responding almost instantaneously to a step input. Therefore, the transport delay tests utilized are those that evaluate steady state delays due to a sinusoid input. A sinusoid was input to the control loader and the output was monitored with both the strip chart recorder and the FRA. The FRA revealed an average transport delay of 59.7 ms (see Table F-1 in Appendix F for data). The strip chart recorder revealed an average delay of 68.4 ms. However, the accuracy of the strip chart recorder is very poor. The fastest chart speed is 200 mm/sec which corresponds to 5 ms/mm. Since the delay measurements are very small the recorder has a large error associated with it. Figure 29 shows a sample of the strip chart recording. Since the device is an analog system the $\Delta\phi°$ indicated by the FRA did not fluctuate during the testing. Therefore it was not necessary to collect multiple data points and average them at each frequency. The predicted transport delay of the control loader is on the order of 50.0 ms to 75.0 ms, depending on the settings of the damping and the velocity feedback functions implemented in the analog computer. Since the measured delay falls within this range



Position Output Signal



Input Signal (1.00 Hz)

Scale : 200mm/sec
0.1 V/div

Figure 29 - Sample Strip Chart of Control Loader

the test is considered valid.

ARTS/CYBER

To determine the transport delay associated with the ARTS/CYBER hardware and software the step presented to the pitch ADC, the math model was bypassed and the CGI was commanded to make the appropriate response to the input data (see CYBER Setup and CGI Setup). During the TSRV test a logic analyzer monitored the step input to start timing and stopped the timing when data arrived at the CGI crate. A second logic analyzer started timing from the data arrival at the CGI crate and monitored the output of the VLD to stop the timing. Forty measurements were made and averaged to yield the delay time (see CGIIN in Appendix G, TABLE G-1, TSRV - no math model). The time required for a step input in pitch to cause a change in altitude data, CGIIN, arriving at the CGI crate was 19.8 ms for the TSRV simulation. This is in agreement with the predicted value of 15.0 ms (average) to sample the ADC plus 4.8 ms to ship the data to the CYBER, execute the real-time program shell and to ship the required data to the CGI.

During the VMS test a logic analyzer monitored the step input to start timing, recorded when the data arrived at the CGI crate and monitored the output of the VLD to stop the timing. Forty measurements were made and averaged to yield the delay time (see CGIIN in Appendix G, TABLE G-2, VMS - no math model). The time required for a step input in pitch to cause a change in altitude data, CGIIN, arriving at the CGI crate was 18.5 ms for the VMS simulation. This is in agreement with the predicted value of 15 ms (average) to sample the ADC plus 3.5 ms to ship the data to the CYBER, execute the real-time program shell and to ship the required data to the CGI.

CGI

The CGI's transport delay was measured from the arrival of data

at the CGI crate to an output from the VLD. Forty measurements were recorded each time ARTS/CYBER data was collected in the previous two tests and the two tests that have the math model in-the-loop. This yields a total of four tests and one hundred and sixty data points (see CGIIN and CGIOUT in Appendix G, TABLE G-1 and G-2). The TSRV measurement for CGIOUT indicates the amount of time was required to generate an output at the VLD from an input at the crate. The average time in the CGI during the TSRV tests was 92.78 ms. The VMS measurement for CGIOUT indicates the average time measured to a VLD response from the step input. In order to determine the time in the CGI, for the VMS test, the average of CGIIN must be subtracted from CGIOUT. The average time in the CGI during the VMS tests was 93.37 ms. Averaging the VMS time with that of the TSRV time the average transport delay for the CGI was measured at 93.08 ms, using 160 data points. Next we must add the 17.85 ms to determine the transport delay to the end of the first field. This yields a total CGI transport delay of 110.9 ms. The predicted average transport delay was 90.0 ms. This leaves approximately 21.0 ms unaccounted for. After discussions with the CGI vendor, E&S, it was revealed that the Image Generator portion of the CT6 required 80.0 ms but the Gould front-end computer required an additional 20.0 ms to preprocess the data for the Image Generator. Also, the trigger signal to the logic analyzer, that starts the timing of the CGI, was found to precede the actual signal that interrupts the CGI by 0.85 ms. If one adds 20 ms, for the Gould, and subtracts 0.85 ms from the measured data for the signal adjustment, we have a measured transport delay of 110.05 and a predicted transport delay of 110.0 ms. These measurements are in good agreement.

CRDS

The CRDS's transport delay was measured, similar to that of the CGI, from the arrival of data at the CRDS crate to a change in the VLD. Forty data points were taken for the calligraphic channel and

twenty data points were taken for the raster channel (see Appendix H). The average transport delay, for the 40 Hz update rate on the calligraphic channel, was 111.7 ms which was approximately the predicted value of 112.5 ms. The average transport delay, for the, 60 Hz update rate on the raster channel, was 74.9 ms which was approximately the predicted value of 75.0 ms.

Motion Base

The measurement of the transport delay of the motion base is analogous to that of the control loader. Except in this case the motion base must be driven by the central computers. The response of the motion base can be measured in the time domain by subjecting it to a step and measuring when the platform begins to move via accelerometers. Ideally, the response to a step should be instantaneous. However, since the analog motion base control computers and the compression of the hydraulic fluid takes a finite amount of time the motion base does not respond instantaneously. Although it was never formally documented the previous transport delay due to a step was approximately 20.0 ms at the time the system was installed. Since installation in 1973 the motion base has been upgraded which added approximately 3000 pounds to the platform. It is expected that this will have an effect on performance. The actual VMS motion base response can be determined by deduction from the measured data. The motion algorithm, which calculates motion commands based upon pilot inputs, depends upon accurate, calculated accelerations from a pilot input. Therefore tests on the motion platform must be conducted with the math model in the loop. Appendix G, TABLE G-2 contains the forty measurements of ACCEL when the NASP math model is in-the-loop. The measurement ACCEL is the time required, from a step input until a response is detected in the accelerometer of the motion base. Figure 30 shows the CYBER and motion base timing diagram. One can see the average time to the motion base response in ACCEL is 86.1 ms.

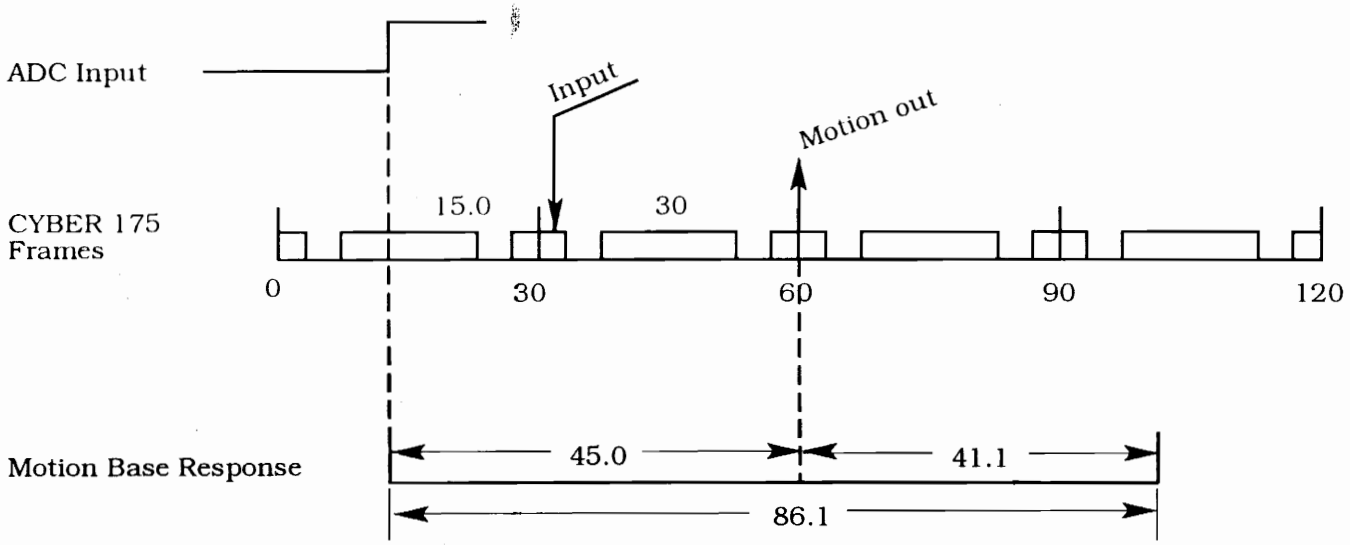


Figure 30 - Motion Base Timing Diagram

The average time required to read an input to the computer is 15.0 ms. Since the motion base output data is synchronous and the data is known (see Software Transport Delays below) to remain in the CYBER for one entire frame, 30.0 ms, the average time until data is available to the motion base in 45.0 ms. The motion base response can then be determined to be 41.1 ms (86.1 ms - 45.0 ms).

A frequency domain test may also be conducted similar to the test conducted on the control loader. The motion base will be driven by a sinusoid generated by the computer[20]. The sinusoid will drive the center of gravity in the X, Y and Z planes at several frequencies. The position of the motion base legs will be returned to the computer and the actual position of the center of gravity is determined via a Newton-Raphson's method.[21] TABLEs I-1 and I-2, in Appendix I, show the frequency domain data for the motion base with the compensation algorithms in and out-of-the-loop. Figures 31 and 32 show the graphs of the motion base response as compared to graphs recorded in 1973.[22] From the data it would appear that the motion base is operating better today than in years past. This is deceiving because there are adjustments that can be made to the incoming motion base command before it is actually sent to the motion base computer. These adjustments can improve the response but they can also increase undesirable effects. There is no way of knowing where these adjustments were set in previous tests therefore it is almost impossible to determine if the performance of the motion base has degraded over time. The time domain response is a good indicator of degradation since the motion base seems to have slowed down by some 20 ms. The frequency domain data does tend to reveal this at the higher frequencies where phase error increases more rapidly than in 1973. Several factors contributed to this phenomenon. The age of the system, the hydraulic pressure was reduced by 400 psi and the platform had 3000 pounds added off center of the center of gravity.

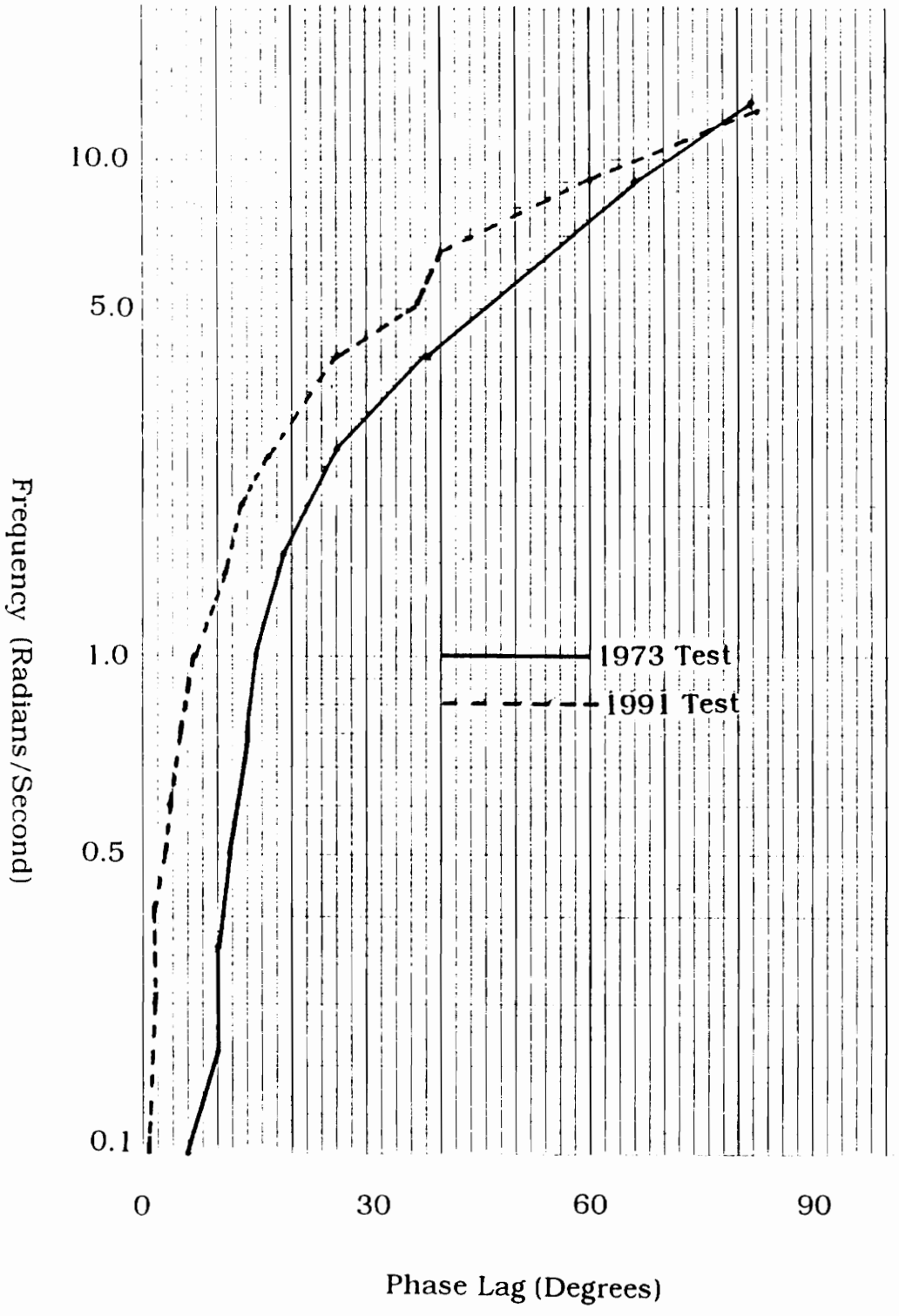


Figure 31 - Frequency Response (uncompensated)

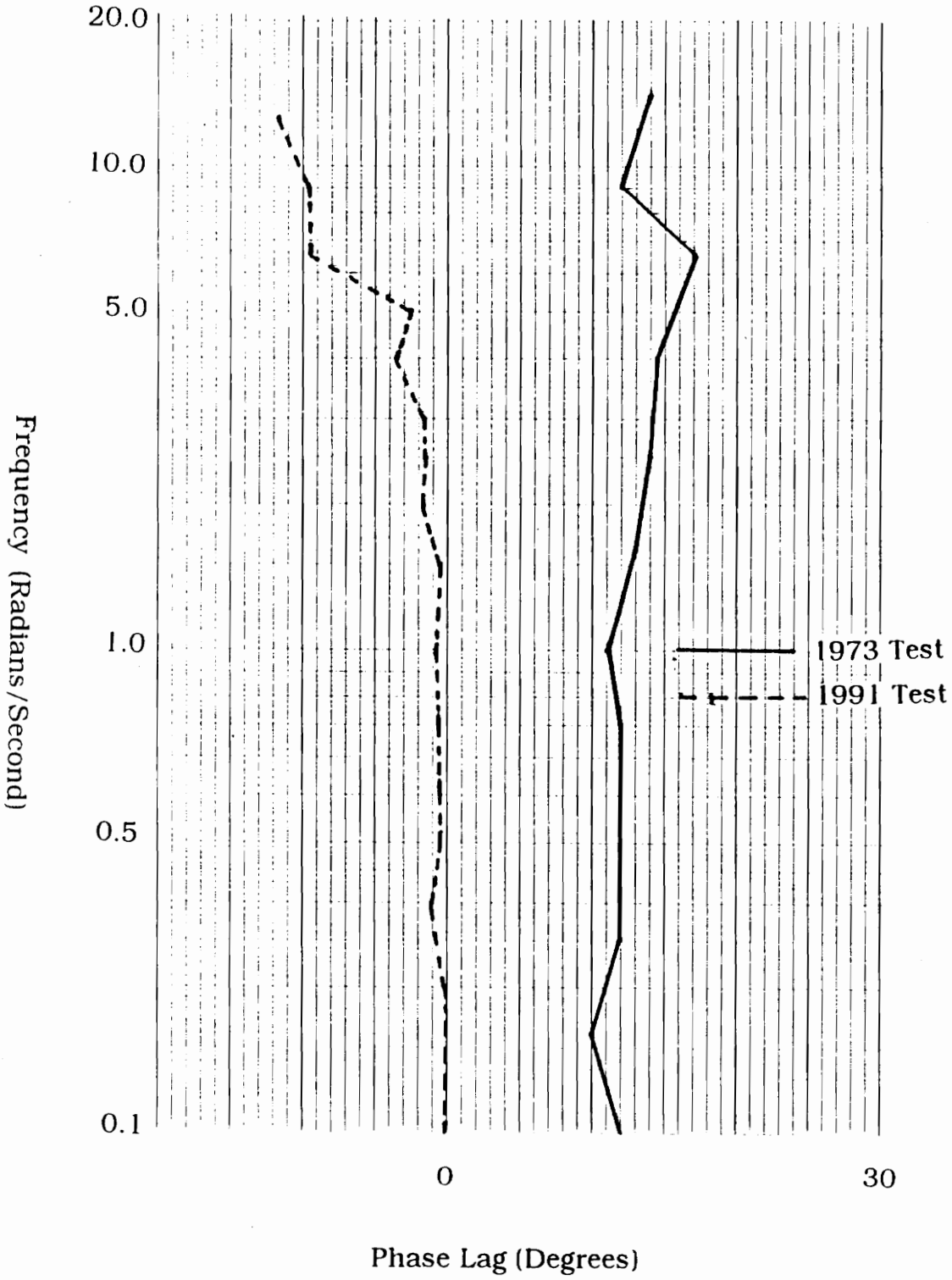


Figure 32 - Frequency Response (compensated)

Video Display Systems

The transport delay associated with the monitors are specified by the manufacturer as 12 ns plus the phosphor response time. In addition, the video distribution system adds several microseconds depending on the individual cable patching and number of analog buffer amplifiers used. Therefore, since the measurement apparatus cannot effectively measure delays in the microsecond range, and that the monitor delay is a negligible part of the overall transport delay, the video display systems transport delay will be ignored.

Software Transport Delay

The second part of the test consisted of measuring the software transport delay while the math model is being executed. Appendix G, TABLEs G-1 and G-2 also contain the data corresponding to the execution of the 737 and NASP models on the CYBER (forty measurements of CGIIN for each case). Since the only thing added to the simulation loop is the additional computation time due to the math model, the transport delay through the math model is the average measured time from an input until the data arrives at the CGI crate with a math model in-the-loop minus the average time to data arrival without the model in-the-loop. Therefore, the average software transport delay for the 737 model is 62.8 ms (82.6 ms - 19.8 ms) and the average delay for the NASP model is 39.5 ms (58.0 ms - 18.5 ms). The average transport delay for the motion base data is 30.0 ms since the leg positions are calculated via a separate subroutine and sent out at the end of the frame in which data is received.

Total Transport Delay

The total hardware transport delay for the simulation can be calculated by adding the transport delays of individual pieces of equipment. TABLE 2 contains a summary of the total average TSRV transport delays. The TSRV simulation requires all of the above pieces of equipment, except

TABLE 2 - Total TSRV Transport Delays

Device	No Math Model (ms)	With Math Model (ms)
ADC Sample	15.0	15.0
ARTS/CYBER	4.8	4.8
Math Model	0.0	62.8
CGI Sample	10.0	10.0
CGI	100.1	100.1
Total	129.9	192.7
ADC Sample	15.0	15.0
ARTS/CYBER	4.8	4.8
Math Model	0.0	62.8
CRDS Sample (40) (60)	12.5 8.3	12.5 8.3
CRDS (40) (60)	99.2 66.6	99.2 66.6
Total (40) (60)	131.5 94.7	194.3 157.5

the motion base, and therefore has an average hardware transport delay of 129.9 ms to the end of the first CGI field, 131.5 ms to the beginning of a 40 Hz CRDS display and 94.7 ms to the beginning of a 60 Hz CRDS display. The delay to completion of the CGI is composed of 15.0 ms to sample the pitch ADC in the crate, 4.8 ms to ship the data around the fiber optic loop, 0.0 ms in the math model, 10.0 ms for the CGI to sample the data on ARTS and 100.1 ms to the end of the CGI field. The delay to the beginning of the CRDS is composed of 15.0 ms to sample the pitch ADC in the crate, 4.8 ms to ship the data around the fiber optic loop, 0.0 ms in the math model, 12.5 ms for the CRDS (operating at 40 Hz) to sample the data on ARTS or 8.3 ms for the CRDS (operating at 60 Hz) to sample the data on ARTS and 99.2 ms to begin to render a 40 Hz display or 66.6 ms to begin to render a 60 Hz display. A timeline of these delays can be found in Figure 33. The 737 math model software transport delay of 62.8 ms was added to the hardware delay for a total transport delay of 192.7 ms to the end of the first CGI field and 194.3 ms to the beginning of a 40 Hz display or 157.5 ms to the beginning of a 40 Hz display. A timeline of these delays can be found in Figure 34.

TABLE 3 contains a summary of the total average VMS transport delays. The VMS simulation requires the same equipment as the TSRV with the addition of a motion platform. The average hardware transport delay for the VMS is 128.6 ms to the end of the first CGI field, 130.2 ms to the beginning of a 40 Hz CRDS display and 93.4 ms to the beginning of a 60 Hz CRDS display. The delay to completion of the CGI is composed of 15.0 ms to sample the pitch ADC in the crate, 3.5 ms to ship the data around the fiber optic loop, 0.0 ms in the math model, 10.0 ms for the CGI to sample the data on ARTS and 100.1 ms to the end of the CGI field. The delay to the beginning of the CRDS is composed of 15.0 ms to sample the pitch ADC in the crate, 3.5 ms to ship the data around the fiber optic loop, 0.0 ms in the math model, 12.5 ms for the CRDS (operating at 40 Hz) to sample the data on ARTS or 8.3 ms for the CRDS (operating at 60 Hz) to sample the data on ARTS and 99.2 ms to begin to render a 40 Hz display or 66.6 ms to begin to render a 60 Hz display. A timeline of these delays can be found in Figure 35. The NASP math model software transport delay

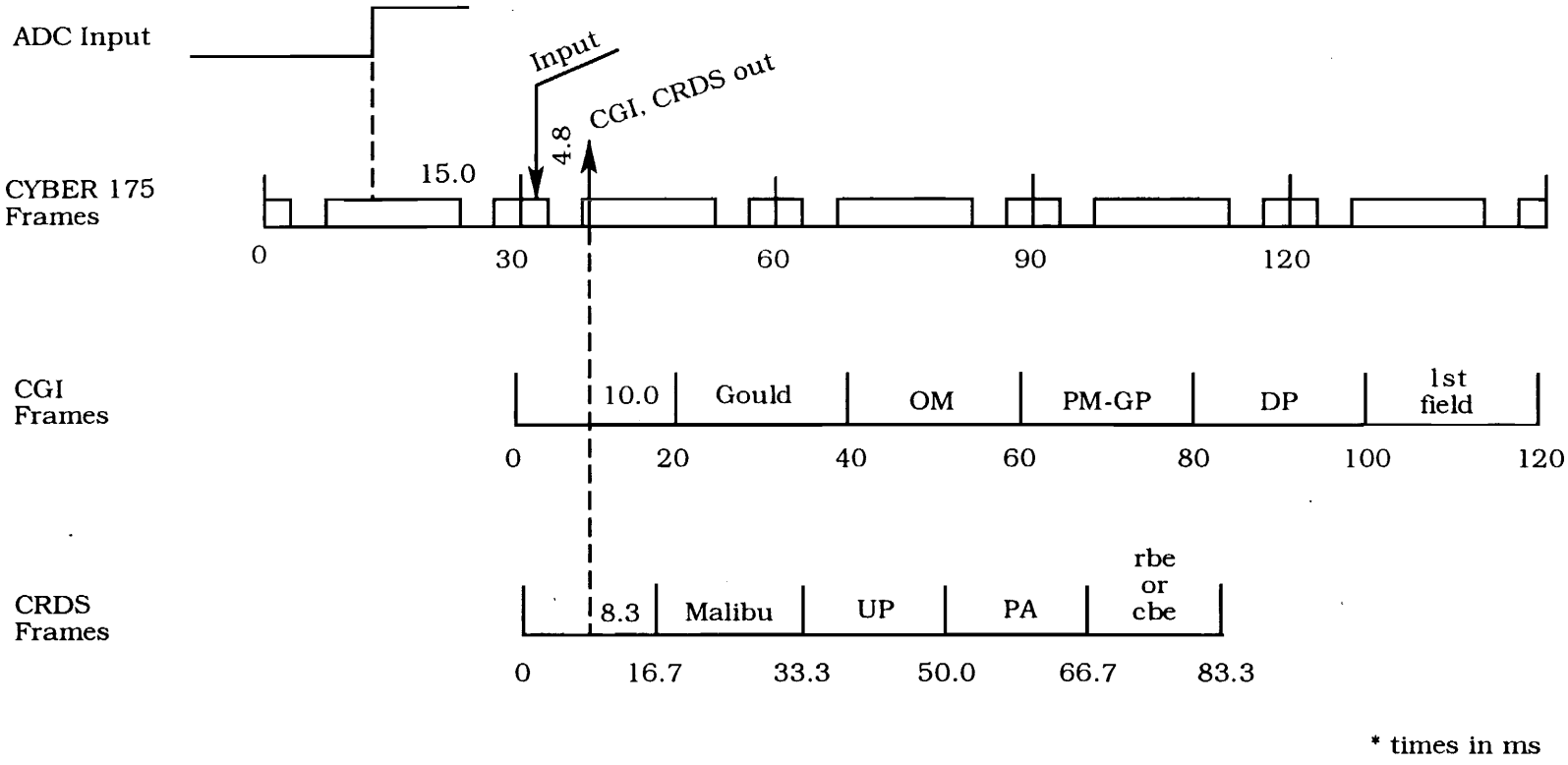


Figure 33 - TSRV Transport Delays (no math model)

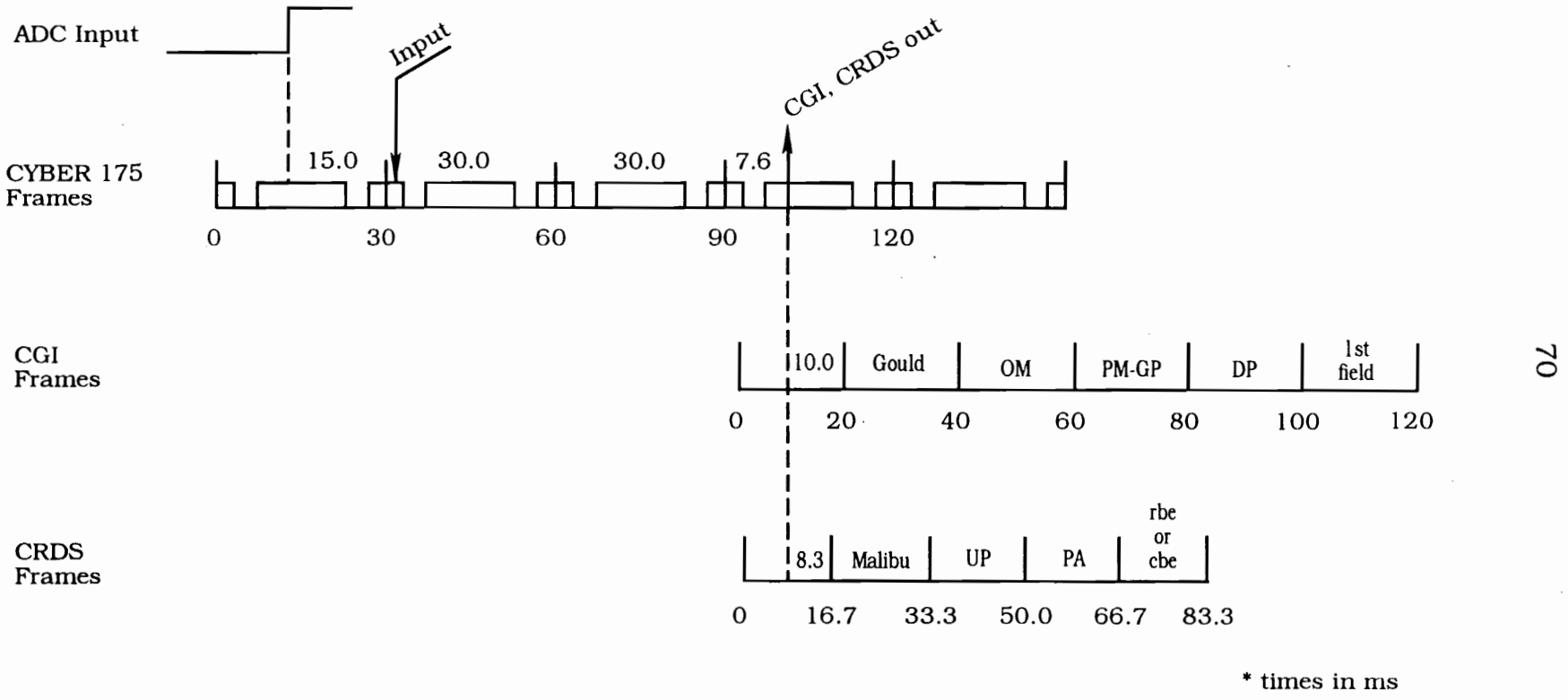


Figure 34 - TSRV Transport Delays (737 math model)

TABLE 3 - Total VMS Transport Delays

Device	No Math Model (ms)	With Math Model (ms)
ADC Sample	15.0	15.0
ARTS/CYBER	3.5	3.5
Math Model	0.0	39.5
CGI Sample	10.0	10.0
CGI	100.1	100.1
Total	128.6	168.1
ADC Sample	15.0	15.0
ARTS/CYBER	3.5	3.5
Math Model	0.0	39.5
CRDS Sample (40) (60)	12.5 8.3	12.5 8.3
CRDS (40) (60)	99.2 66.6	99.2 66.6
Total (40) (60)	130.2 93.4	169.7 132.9
ADC Sample	N/A	15.0
ARTS/CYBER/Model	N/A	30.0
Motion Base	N/A	41.1
Total	N/A	86.1

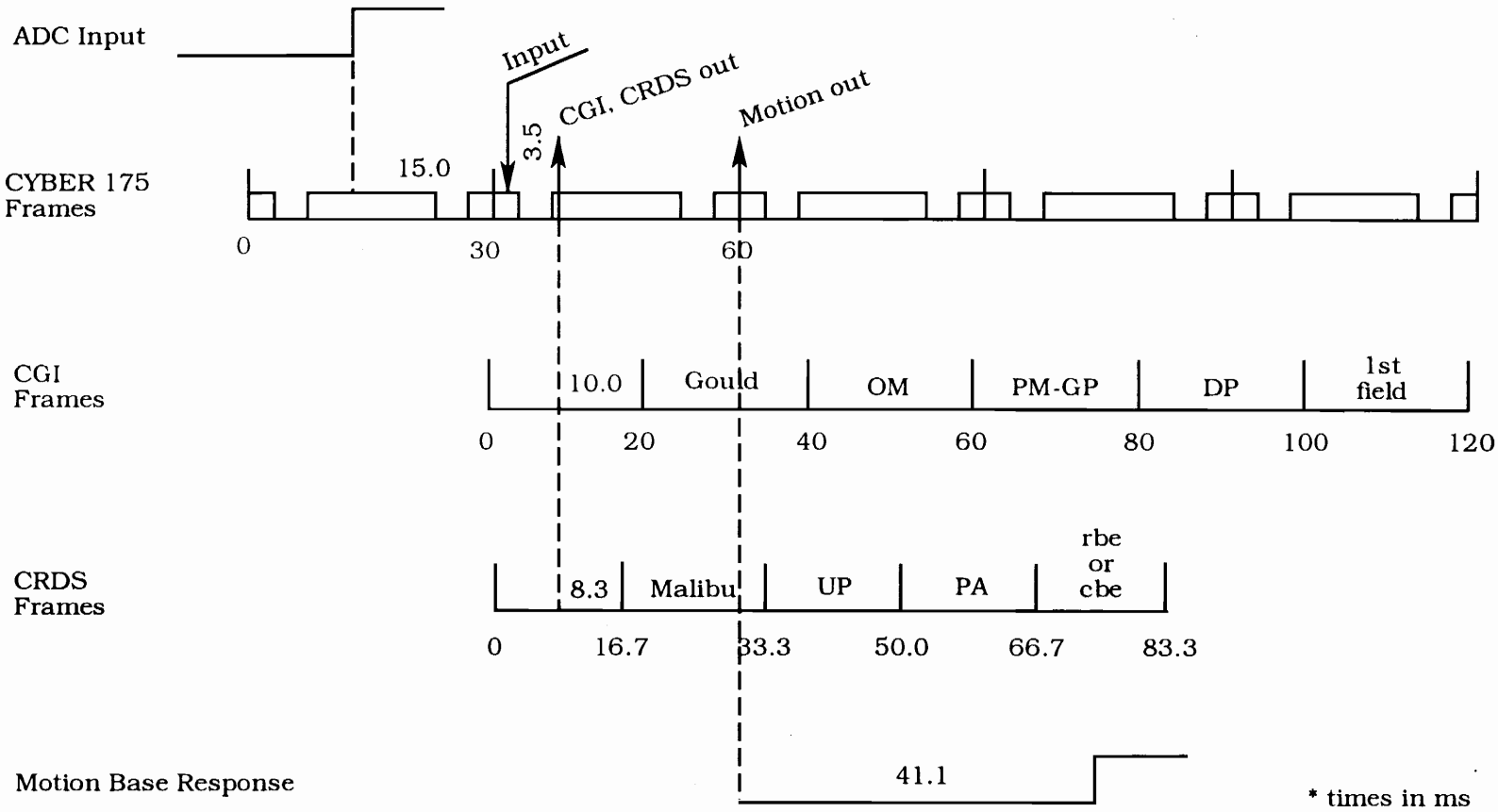


Figure 35 - VMS Transport Delays (no math model)

of 39.5 ms was added to the hardware delay for a total transport delay of 168.1 ms to the end of the first CGI field, 169.7 ms to the beginning of a 40 Hz display and 132.9 ms to the beginning of a 60 Hz display. The motion base was also tested with the model in-the-loop and found to have an 86.1 ms transport delay. A timeline of these delays can be found in Figure 36.

The overall total transport delays of both the TSRV and the VMS, with the exception of the 60 Hz display on the VMS, are excessive when compared to the FAA guidelines of 150 ms for the average transport delay.[23] Therefore the next task is to see if there are any measures that can be taken to reduce these delays to an acceptable level.

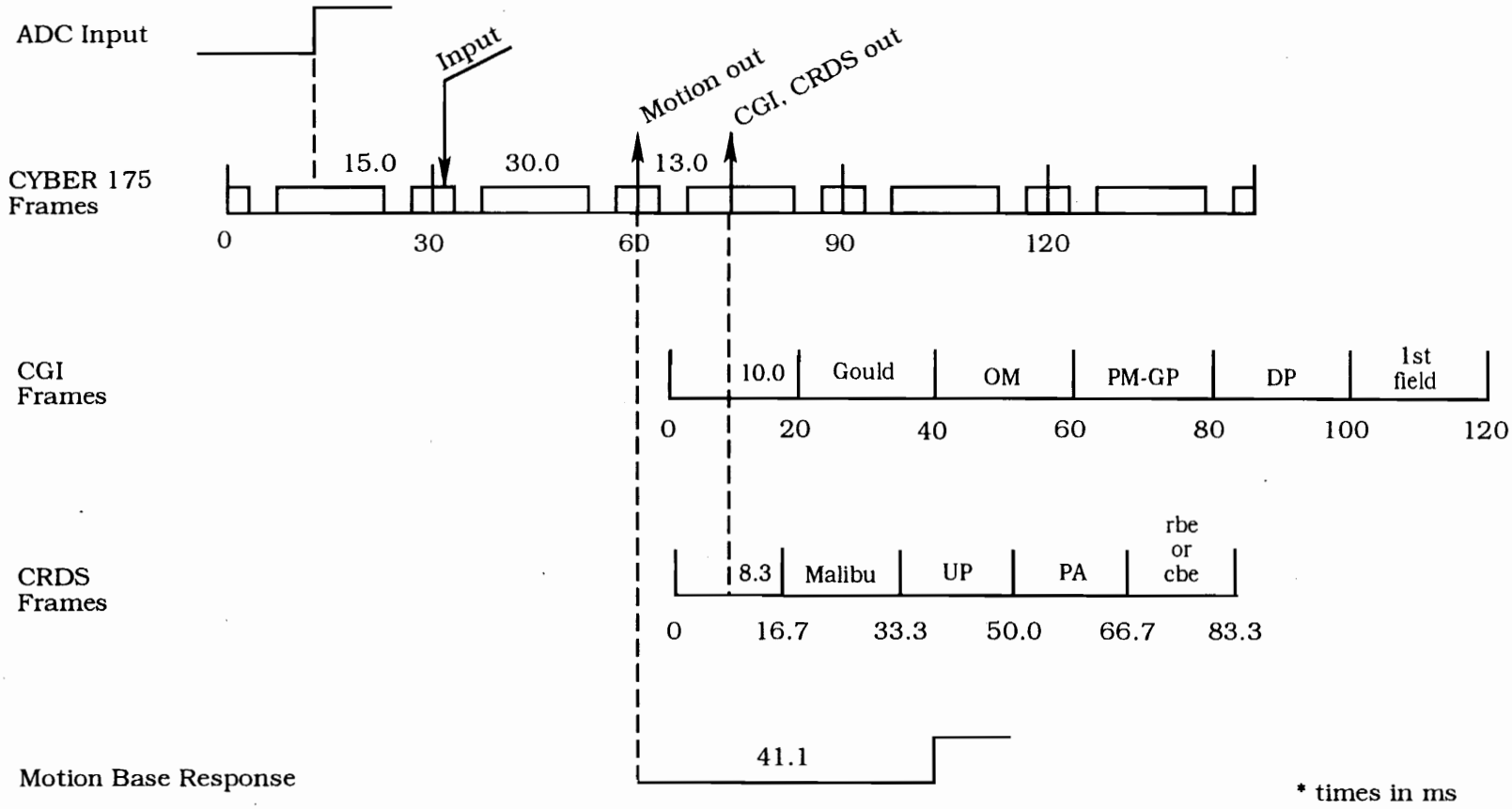


Figure 36 - VMS Transport Delays (NASP math model)

CHAPTER 6

CORRECTIVE MEASURES

Upgrade CGI System

Once the extra 20 ms transport delay for the Gould was discovered, the CGI vendor, E&S, was contacted to obtain verification of this delay and to outline any options for correcting this problem. It was learned that the system may take anywhere from 17 ms to 37 ms to get the data from the crate, parse it for the image generator and pass it to the image generator. However, E&S has recently determined a way to guarantee the data to be delivered to the image generator at an average of 17.0 ms sooner than presently possible. This upgrade would require extensive modification of the CGI's proprietary operating system software. Since this is a significant delay savings a contract with E&S to modify the NASA system is being pursued. The correction of the CGI transport delay will remove approximately 17 ms from the transport delays of the TSRV and the VMS. The transport delays will then be 175.7 ms and 151.1 ms, respectively. Since these delays are still in excess of the FAA guideline of 150.0 ms further corrective measures are required.[24]

Modify Simulation Program Flow

The next logical choice for reducing the transport delay is the simulation program. The simulation program should be outputting any data that requires further computing as soon as possible. In order to determine when data was leaving the real-time program a print statement was inserted into the TSRV and VMS code to determine what values were being output during which CYBER frame. The first print out shown in Figure 37 shows the data from the TSRV simulation, ALTCGI, sent to the CGI and CRDCLR, sent to the CRDS, being changed in the third frame from when the input was received. Figure 38 shows the same data from the

04/12/91. 09.47.26. OPT= 4 COLBP= .5000 LIMCOL= .0100 TSTHETA= .0020

T	W	ADC	THEDEG	THCGI	COUNT	INC	ALTCGI	THETA	QB	QBDOT
1.95	F	.15	5.54934	.00000	16	0	5692.25	.09686	.00008	-.00004
1.98	F	.15	5.54950	.00000	16	0	5692.25	.09686	.00008	-.00004
2.01	F	.15	5.54966	.00000	16	0	5692.25	.09686	.00008	-.00004
2.04	F	.15	5.54982	.00000	16	0	5692.25	.09687	.00008	-.00004
2.07	F	.15	5.54998	.00000	16	0	5692.25	.09687	.00007	-.00004
2.10	F	.15	5.55014	.00000	16	0	5692.25	.09687	.00007	-.00004
2.13	F	.15	5.55029	.00000	16	0	5692.25	.09687	.00007	-.00003
2.16	F	.15	5.55044	.00000	16	0	5692.25	.09688	.00007	-.00003
2.19	F	.15	5.55059	.00000	16	0	5692.25	.09688	.00007	-.00003
2.22	F	.15	5.55074	.00000	16	0	5692.25	.09688	.00007	-.00003
2.25	F	.15	5.55089	.00000	16	0	5692.25	.09688	.00007	-.00003
2.28	F	.15	5.55104	.00000	16	0	5692.25	.09689	.00007	-.00003
2.31	F	.15	5.55118	.00000	16	0	5692.25	.09689	.00007	-.00003
2.34	F	.15	5.55132	.00000	16	0	5692.25	.09689	.00007	-.00003
2.37	F	.15	5.55146	.00000	16	0	5692.25	.09689	.00006	-.00003
2.40	F	.15	5.55160	.00000	16	0	5692.25	.09690	.00006	-.00002
2.43	F	.15	5.55174	.00000	16	0	5692.25	.09690	.00006	-.00002
2.46	F	4.39	5.55188	.00000	17	1	5692.25	.09690	.00089	.01836
2.49	F	4.39	5.55201	.00000	18	1	5692.25	.09694	.00221	.03541
2.52	T	4.39	5.55428	.00000	19	0	5695.25	.09703	.00398	.05117
2.55	T	4.39	5.55924	.00000	19	0	5695.25	.09717	.00617	.06575
2.58	T	4.39	5.56763	.00000	19	0	5695.25	.09739	.00875	.07915
2.61	T	4.39	5.58014	.00000	19	0	5695.25	.09769	.01167	.09140
2.64	T	4.39	5.59742	.00000	19	0	5695.25	.09809	.01491	.10254
2.67	T	4.39	5.62002	.00000	19	0	5695.25	.09858	.01344	.11257
2.70	T	4.39	5.64847	.00000	19	0	5695.25	.09919	.02231	.12337
2.73	T	4.39	5.68323	.00000	19	0	5695.25	.09992	.02628	.12944
2.76	T	4.39	5.72492	.00000	19	0	5695.25	.10077	.03048	.13639
2.79	T	4.39	5.77354	.00000	19	0	5695.25	.10174	.03484	.14238
2.82	T	4.39	5.82956	.00000	19	0	5695.25	.10286	.03934	.14744
2.85	T	4.39	5.89321	.00000	19	0	5695.25	.10410	.04395	.15161

Figure 37 - TSRV Program Output

07/17/91. 13.33.06. OPT= 4 COLSF= .5000 LINCOL= .1000 TSTHETA= .0002

T	CADCLR	ADC	THETAO	THECGI	COUNT	INC	ALTCGI	THETA	Q	QCOT
.38	.00	.14	4.89606	.00000	0	0	5692.25	.08545	.00001	.00002
.42	.00	.14	4.89608	.00000	0	0	5692.25	.08545	.00001	.00002
.45	.00	.14	4.89610	.00000	0	0	5692.25	.08545	.00001	.00002
.48	.00	.14	4.89612	.00000	0	0	5692.25	.08545	.00001	.00002
.51	.00	.14	4.89614	.00000	0	0	5692.25	.08545	.00001	.00002
.54	.00	.14	4.89617	.00000	0	0	5692.25	.08545	.00001	.00002
.58	.00	.14	4.89619	.00000	0	0	5692.25	.08545	.00001	.00002
.61	.00	.14	4.89622	.00000	0	0	5692.25	.08546	.00001	.00002
.64	.00	.14	4.89624	.00000	0	0	5692.25	.08546	.00002	.00002
.67	.00	.14	4.89627	.00000	0	0	5692.25	.08546	.00002	.00002
.70	.00	.14	4.89630	.00000	0	0	5692.25	.08546	.00002	.00002
.74	.00	.14	4.89634	.00000	0	0	5692.25	.08546	.00002	.00002
.77	.00	.14	4.89637	.00000	0	0	5692.25	.08546	.00002	.00002
.80	.00	.14	4.89641	.00000	0	0	5692.25	.08546	.00002	.00002
.83	.00	.14	4.89644	.00000	0	0	5692.25	.08546	.00002	.00002
.86	.00	.14	4.89648	.00000	0	0	5692.25	.08546	.00002	.00002
.90	.00	.14	4.89652	.00000	0	0	5692.25	.08546	.00002	.00002
.92	.00	.14	4.89656	.00000	0	0	5692.25	.08546	.00002	.00002
.96	.00	.14	4.89660	.00000	0	0	5692.25	.08546	.00002	.00002
.99	.00	.14	4.89664	.00000	0	0	5692.25	.08546	.00002	.00002
1.02	.00	.14	4.89669	.00000	0	0	5692.25	.08546	.00002	.00002
1.05	.00	.14	4.89673	.00000	0	0	5692.25	.08546	.00003	.00002
1.09	.00	.14	4.89678	.00000	0	0	5692.25	.08546	.00003	.00002
1.12	.00	.14	4.89683	.00000	0	0	5692.25	.08547	.00003	.00002
1.15	.00	.14	4.89688	.00000	0	0	5692.25	.08547	.00003	.00002
1.18	.00	.14	4.89693	.00000	0	0	5692.25	.08547	.00003	.00002
1.22	.00	3.91	4.89698	.00000	1	1	5692.25	.08547	.00003	.06834
1.25	1.00	3.91	4.89875	.00000	2	0	5695.25	.08550	.00283	.11335
1.28	1.00	3.91	4.90726	.00000	2	0	5695.25	.08563	.00734	.15995
1.31	1.00	3.92	4.92561	.00000	2	0	5695.25	.08587	.01354	.21934
1.34	1.00	3.92	4.95686	.00000	2	0	5695.25	.08651	.02139	.27083
1.38	1.00	3.92	5.00403	.00000	2	0	5695.25	.08734	.03091	.32171
1.41	1.00	3.92	5.07009	.00000	2	0	5695.25	.08843	.04193	.37329
1.44	1.00	3.92	5.15800	.00000	2	0	5695.25	.09002	.05471	.42275

Figure 38 - VMS Program Output

VMS program changing in the second frames from when the input received. Ideally, the output data should change in the same frame the as input was received. Therefore, since the data should be ready to be sent at the end of the first frame a study of the program flow was implemented.

TSRV Program Structure

For the purposes of this paper, a simplified view of the program structure of the TSRV is presented in Figure 39. The major subroutine calls that calculate the math model and hardware data are shown in the order they are called. First the program reads in new data, through TCVCAB, at the beginning of the frame. Then MISCEL and RUNWAY perform final calculations on information that was either received or generated in the previous frame. ACCEL then calculates accelerations based on data received in this frame. SWITCH, MSPLOG and INPUT continue processing data read in at the beginning of this frame. They do not use any of the variables calculated in ACCEL. The acceleration data and new inputs are then used to determine forces and moments on the engines, landing gear and aero model in ENGFM, LNGFM and AEROFM. These forces and moments are summed in SUMFM then the model's new state variable derivatives are calculated in DERIV. Data that was calculated in previous frames is set to the CGI, the CRDS and the cockpit through CGIINTF, CDCAGT and DACS. Finally the state variables are integrated in IGRATE6 and the program is recycled at the beginning of the next frame.

The path that an input in pitch would take is described in Figure 40. The input arrives at the beginning of the frame and is first processed in ACCEL to generate QBDOT, which is the pitch acceleration in the body axis. QBDOT is then integrated at the end of the frame, in IGRATE6, to yield QB. The integration method used is the Adams and Bashforth (AB-2) method and the program is recycled at the end of the frame.[25] THETADOT, the velocity in the earth axis, is then generated in DERIV and then integrated to THETA

TCVCAB	-	read analog inputs cockpit
MISCEL	-	determine the ψ, θ, ϕ, for the last frame (old), solve all miscellaneous equations
RUNWAY	-	converts the lat and long to CGI frame of reference
ACCEL	-	compute new body accelerations, UDOT, VDOT, WDOT
SWITCH	-	stores guidance and navigation data into buffers
MSPLOG	-	determines mode selected from cockpit switches
INPUT	-	calls autopilot, SAS, engine model and yaw damper
ENGFM	-	compute new engine forces and moments
LNGFM	-	compute new landing gear forces and moments
AEROFM	-	compute new aerodynamic forces and moments
SUMFM	-	summation of forces and moments
DERIV	-	calculate new derivatives and PDOT, QDOT, RDOT
CGIINTF	-	send old x, y, z, ψ, θ, ϕ to the CGI
CDCAGT	-	packs and sends old data to the CRDS
DACS	-	packs and sends old data to analog controls in the cockpit
IGRATE6	-	computes new state variables and U, V, W, P, Q using AB-2

Figure 39 - TSRV Program Flow

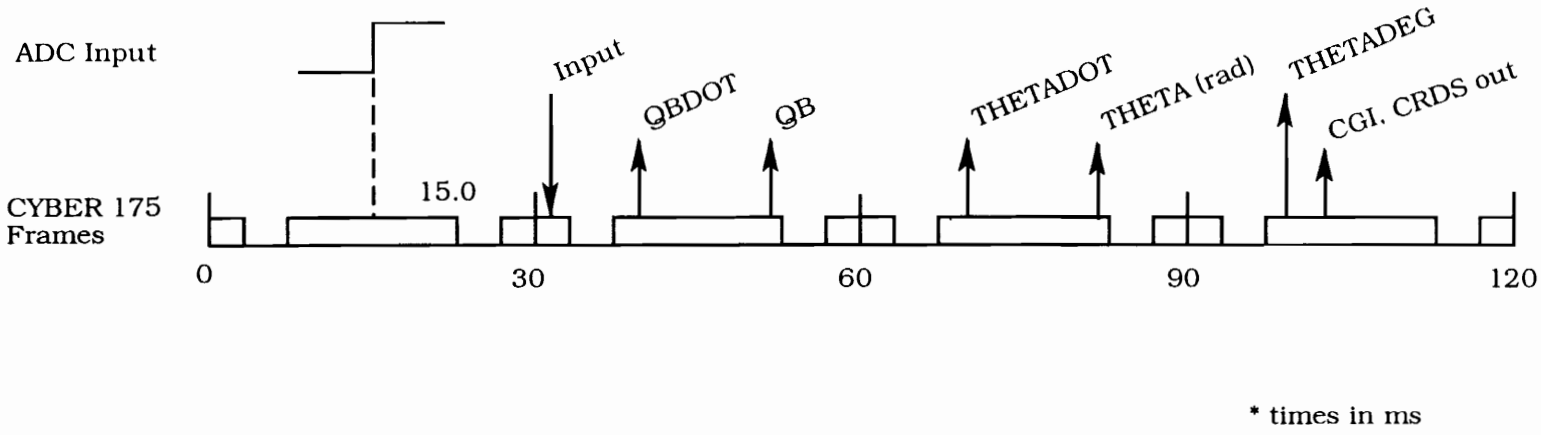


Figure 40 - TSRV Program Signal Path

at the end of the frame. THETA is then converted from radians to THETADEG in degrees in MISCEL. Since the cockpit is synchronous the data goes out at the end of the third frame. The CGI and CRDS are asynchronous and therefore goes out as soon as the calls to the CGIINTF and CDCAGT are made (assuming the CYBER's Peripheral Processing Unit is not busy) sometime in the middle of the third frame. This accounts for a two and a half frame software delay.

The integrations are located at the end of the frame because the AB-2 method is a prediction method and it will determine the value of the state variable at the frame time, t, plus the step size of the integration, h. Therefore, since the output of IGRATE6 is the state information for the next frame it should be output in the next frame. This program structure has been in use since the early seventies and, at the time of development, the only external simulation equipment was synchronous and analog. The analog systems typically had a very low transport delay therefore there was not much concern over the total system transport delay. However with the addition of asynchronous, digital systems to the network, such as the CGI and CRDS, the transport delay for the system has increased dramatically. One way to compensate for the increased delay is to output the calculated parameters as soon as possible. In other words, the data will be given a two frames of lead by being output earlier than it should be released. It should be noted that this does not change the numerical value of the data but only affects when it is calculated and sent out.

Proposed TSRV Program Structure

From the description above it is clear that the simulation program can be reordered to calculate and output time critical data as soon as possible. Figure 41 shows the reordered subroutines. Since the object is to calculate new data as soon as possible new control inputs are read in by TCVCAB, SWITCH, MSPLOG and INPUT at the beginning of the frame. The data is then used in ENGFM,

TCVCAB	-	read analog inputs cockpit
SWITCH	-	stores guidance and navigation data into buffers
MSPLOG	-	determines mode selected from cockpit switches
INPUT	-	calls autopilot, SAS, engine model and yaw damper
ENGFM	-	compute new engine forces and moments
LNGFM	-	compute new landing gear forces and moments
AEROFM	-	compute new aerodynamic forces and moments
SUMFM	-	summation of forces and moments
ACCEL	-	compute new body accelerations, UDOT, VDOT, WDOT
DERIV	-	calculate new derivatives and PDOT, QDOT, RDOT
QUAT	-	calculate new quaternion coefficients and ψ, θ, ϕ
IGRATE6	-	computes new state variables and U, V, W, P, Q using AB-2
RUNWAY	-	converts the lat and long to CGI frame of reference x, y, z
CGIINTF	-	send new x, y, z, ψ, θ, ϕ to the CGI
MISCEL	-	solve all miscellaneous equations
CDCAGT	-	packs and sends new data to the CRDS
DACS	-	packs and sends new data to analog controls in the cockpit

Figure 41 - Reordered TSRV Program Flow

LNGFM, AEROFM and SUMFM to calculate new forces and moments on the model. ACCEL and DERIV calculate the state variables and derivatives. Next a new subroutine is added called QUAT. QUAT performs a transformation using the quaternion (or Euler parameter) rate equations on the accelerations from this frame and the velocities from the last frame. It also integrates these equations using the AB-2 method to yield positional information in the earth axis. Previously the transformation was done separately in the conversion from QDOT to THETADOT. Since the output of QUAT only contains positional data such as THETA, IGRATE6 is then used to generate the velocities for this frame from the accelerations. The conversion from radians to degrees is executed in the QUAT subroutine. RUNWAY converts positional coordinates from latitude and longitude to CGI database coordinates. The data is then output to the CGI. Next any miscellaneous equations are calculated in MISCEL and the data is sent to the CRDS and the cockpit through CDCAGT and DACS.

These changes should accelerate the availability of data to the CGI and CRDS by almost two frames (60 ms). However since the call to the CGI and CRDS is now later in the frame the time to execute the code will be slightly longer. The time to the output to the cockpit will be accelerated by exactly two frames. This should yield a predicted a total TSRV transport delay of approximately 137.7 ms to the end of the first CGI field , 139.3 ms to the beginning of a 40 Hz CRDS display and 102.5 ms to the beginning of a 60 Hz display. These figures include 5.0 ms for extra computational time. With the proposed CGI upgrade the predicted CGI transport delay would be approximately 120.7 ms to the end of the first field.

VMS Program Structure

A simplified view of the VMS program structure of the VMS is presented in Figure 42. The major subroutine calls dealing with the math model and hardware are shown, in the order they occur in one frame. The program reads in new data at the beginning of the frame

ADINPUT	-	read inputs from all remote sites
AUXEQ	-	compute direction cosines and determine the ψ, θ, ϕ, for the last frame (old), solve all auxiliary equations
FDIRECT	-	models the flight director
TRIM	-	computes the trim of the aircraft in reset
ACEQM	-	compute new body forces and moments from the inputs
ACCEL	-	compute new body accelerations
SUMSQ	-	obtain any root-mean squares required for data analysis
HDWOUT	-	packs and sends old data to analog controls in the cockpit
HUD	-	packs and sends old data to the CRDS
CGIINTF	-	send old x, y, z, ψ, θ, ϕ to the CGI
MBINTF	-	send new accelerations to the motion base washout algorithm
QUAT	-	calculate new quaternion coefficients
LVELOC	-	compute new x, y, z
DERIV	-	swaps data to tables to prepare for integrations
IGRATE6	-	integrate new UDOT, VDOT, WDOT, PDOT, QDOT, RDOT, using AB-2 to get new U, V, W, P, Q, R
INTEGRAL	-	swaps data back from tables after integrations have been performed

Figure 42 - VMS Program Flow

through ADINPUT. Then AUXEQ, FDIRECT and TRIM perform final calculations on information that was either received or generated in the previous frame. ACEQM, ACCEL and SUMSQ generate forces and moments and accelerations on new data. The output section of the program sends data to the cockpit, CRDS, CGI and the motion base through HDWOUT, HUD, CGIINTF and MBINTF. It should be noted that the computations on the data going to the cockpit, CRDS and the CGI were completed at the end of the last frame. Although ψ, θ and ϕ, are generated in AUXEQ, they are still considered data from the last frame since no inputs read in by ADINPUT affect their calculated result. The motion base is sent the new acceleration data in order to accelerate the response from the hardware (see Chapter 5, Motion Base). Since the cockpit and motion base are synchronous the data goes out at the end of the frame. The CGI and CRDS are asynchronous therefore the data is shipped as soon as the calls to the CGIINTF and HUD are made. QUAT then integrates the quaternion rate equations and LVELOC calculates the x, y and z that will be output to the CGI in the middle of the next frame. In order to obtain the velocities the body accelerations UDOT, VDOT, WDOT, PDOT, QDOT and RDOT are integrated using the AB-2 method and the program is recycled at the end of the frame.[26] The integrations are located at the end of the frame for the same reasons as described in the TSRV Program Structure above. If the new parameters can be calculated in the first frame the data will be given a frame of lead by being output before the data technically should be released.

Figure 43 describes the path that an input in pitch would take through the program. The input arrives at the beginning of the frame and is first processed in ACCEL to generate the pitch acceleration in the body axis QBDOT. The new QBDOT and the QB from the last frame are transformed and integrated into THETA and THETADEG in the subroutine QUAT. At the end of the frame IGRATE6 generates the new velocity QB. The program is recycled and in the next frame the data is output to all devices. This accounts for a one and a half frame software delay.

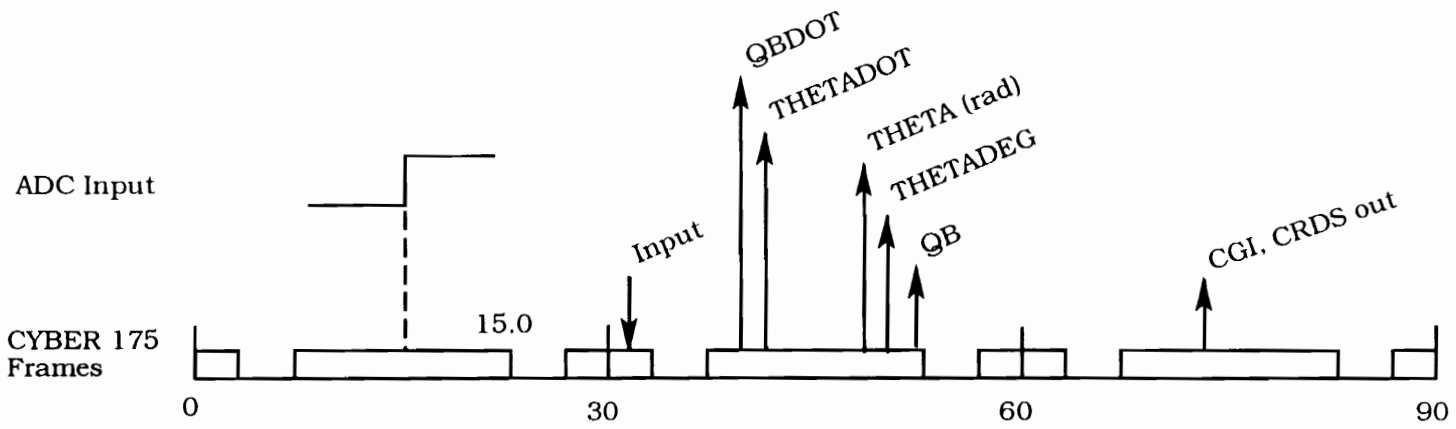


Figure 43 - VMS Program Signal Path

Proposed VMS Program Structure

The VMS is similar to the TSRV in that it's program structure can also be optimized so that time critical data can be calculated and output as soon as possible. Figure 44 shows the reordered routines. Since the object is to calculate new data as soon as possible ADINPUT gets the new control inputs at the beginning of the frame. The data is then used in ACEQM, ACCEL, SUMSQ and QUAT to calculate new forces, accelerations and quaternion coefficients. LVELOC then calculates the new x, y and z position of the aircraft. The subroutine DIRCOS, which originally was called in AUXEQ, was the first subroutine executed in the previous program flow and does not utilize any data that may be affected by DERIV, IGRATE6 or INTEGRAL. Therefore it can be removed from AUXEQ and placed after LVELOC. Once the new direction cosines have been determined then the new ψ, θ, ϕ can be calculated for the CGI. The information the CGI requires is now ready and can be transmitted by the subroutine CGIINTF. The motion base can also be sent the new accelerations however, since the transmission is synchronous the data will not actually go out until the end of the frame. Next we integrate the body accelerations into velocities and send the latest data to the CRDS or the cockpit instruments via the HUD and HDWOUT subroutines. Finally, the remaining equations in AUXEQ, FDIRECT and TRIM are calculated to prepare for the next frame.

These changes should accelerate the availability of data to the CGI and CRDS by slightly less than one frame (30 ms). Again, as in the TSRV case, the CGI and CRDS calls have been moved to later in the frame and their outputs will be delayed by the amount required to execute the additional code, estimated at approximately 2.0 ms. The time to output to the cockpit will be advanced by one frame. This should give a predicted a total VMS transport delay of approximately 140.1 ms to the end of the first CGI field, 141.7 ms to the beginning of a 40 Hz CRDS display, 104.9 ms to the beginning of a 60 Hz display and 86.1 ms to the motion base response. With the

ADINPUT	-	read inputs from all remote sites
ACEQM	-	compute new body forces and moments from the inputs
ACCEL	-	compute new body accelerations
SUMSQ	-	obtain any root-mean squares for data analysis
QUAT	-	calculate new quaternion coefficients
LVELOC	-	compute new x, y, z
DIRCOS	-	compute new direction cosines and determine the new ψ, θ, ϕ
CGIINTF	-	send new x, y, z, ψ, θ, ϕ to the CGI
MBINTF	-	send old accelerations to the motion base washout algorithm
DERIV	-	swaps data to tables to prepare for integrations
IGRATE6	-	integrate new UDOT, VDOT, WDOT, PDOT, QDOT, RDOT, using AB-2 to get new U, V, W, P, Q, R
INTEGRAL	-	swaps data back from tables after integrations have been performed
HUD	-	packs and sends new data to the CRDS
HDWOUT	-	packs and sends new data to analog controls in the cockpit
AUXEQ	-	solves all auxiliary equations (remove call to DIRCOS subroutine)
FDIRECT	-	models the flight director
TRIM	-	computes the trim of the aircraft in reset

Figure 44 - VMS Reordered Program Flow

proposed CGI upgrade the predicted CGI transport delay would be approximately 123.1 ms to the end of the first field.

Matching Subsystem Delays

Using the predicted transport delays above for the VMS, the CGI upgrade and the program flow modifications will reduce the phase difference between the CGI and the CRDS to +18.2 ms (123.1 ms - 104.9 ms) for a 60 Hz display, -18.6 ms (123.1 ms - 141.7 ms) for a 40 Hz display and +37.0 ms (123.1 ms - 86.1 ms) to the motion base response. Since only slightly more than half of the displays at NASA update at 40 Hz we must make the transport delays match at as close to both rates as possible. Since the data can only be slowed down by frame increments (30 ms) it would not be desirable to add 30 ms to the CRDS delay since that would synchronize the 60 Hz displays to within -11.8 ms (123.1 ms - 134.9 ms) but the 40 Hz displays would be -38.6 ms (123.1 ms - 161.7 ms) out of phase. Since the present delays for each display rate are centered about the CGI delay it would not be desirable to delay the CRDS an additional frame.

The FAA guidelines state that the motion onset cue shall occur within 150.0 ms of the visual cue. The visual cue must not transition before the motion onset cue.[27] While this may be a good specification for transport aircraft on study has indicated that delays that are asynchronous by more than 80.0 ms or more can substantially degrade pilot performance in vehicles that require high response rates.[28] Although no studies have been conducted to determine the maximum asynchronous delay allowable, with no pilot degradation, it is NASA's goal to keep the differences in the delay limited to within ±30.0 ms of each other. Presently the motion base response is significantly faster than the CGI response. However, as in the case of the CRDS, the motion base response can be delayed in 30.0 ms increments. If the motion base is delayed 30.0 ms the response will be approximately 116.1 ms (86.1 ms + 30.0 ms). Since the CGI and CRDS are going to be accelerated together as discussed above, the motion base would only be out of phase by -7.0 ms (116.1 ms - 123.1 ms) with the CGI, +11.2

ms (116.1 ms - 104.9 ms) out of phase with a 60 Hz CRDS display and -25.6 ms (116.1 ms - 141.7 ms) with a 40 Hz CRDS display. Delaying the motion base more than one frame would have the motion base lagging the CGI and the CRDS by a maximum of +41.2 ms (146.1 ms - 104.9 ms). Since this would exceed the synchronization goal of ±30.0 ms only one frame of motion base delay is required. There are several methods by which the motion command may be delayed by one frame. One is to move the call to MBINTF (see Figure 42) after ADINPUT. This will ensure that the values sent to the motion base at the end of that frame were the accelerations calculated in the previous frame. A second method would be to ladder the data. This is accomplished by saving the new accelerations, generated in the present frame, to a variable to be used in the next frame. Since either method is equally acceptable the second method was implemented in order to allow the equipment output calls to remain grouped together. The code for the ladder delay is located in Appendix J.

CHAPTER 7

FINAL RESULTS

Hardware Transport Delays

The physical hardware transport delays remain unchanged, with the exception of the CGI upgrade, by any of the software modifications described in the previous chapter. Each subsystems hardware transport delay is shown in TABLEs 4 and 5. The present CGI transport delay was also reverified and is shown in the TABLEs 4 and 5. The anticipated CGI transport delay is 17.0 ms less than measured CGI data.

Software Transport Delays

Once the changes described in the previous chapter were incorporated into the TSRV and VMS real-time programs, the output variables to the CGI and CRDS were printed again (see Figures 45 and 46) and the variables were observed to change within one frame in both programs. The total TSRV transport delay was retested according the procedure described in chapters 4 and 5. The data for the reordered program flow was collected (see Appendix G TABLE G-3) and an analysis of these transport delays is shown in TABLE 4. The transport delay of the math model was 5.4 ms (25.2 ms - 19.4 ms). The total transport delay to the end of the first CGI field was measured at 135.0 ms. Subtracting 17.0 ms for the CGI upgrade yields a transport delay of 118.0 ms. Since the transport delays for the CRDS were moved with the CGI and the output parameters were verified to be changing during the first frame the transport delays for the CRDS were not remeasured. The transport delay to the beginning of a 40 Hz display is 136.9 ms and 100.1 ms to the beginning of a 60 Hz display. A timeline of these delays can be found in Figure 47.

The total VMS transport delay was retested according the procedure described in chapters 4 and 5. The data for the reordered program flow

TABLE 4 - Reordered TSRV Transport Delays

Device	With Math Model (ms)
ADC Sample	15.0
ARTS/CYBER	4.8
Math Model	5.4
CGI Sample	10.0
CGI (present)(upgraded)	99.8 82.8
Total (present)(upgraded)	135.0 118.0
ADC Sample	15.0
ARTS/CYBER	4.8
Math Model	5.4
CRDS Sample (40) (60)	12.5 8.3
CRDS (40) (60)	99.2 66.6
Total (40) (60)	136.9 100.1

TABLE 5 - Reordered VMS Transport Delays

Device	With Math Model (ms)
ADC Sample	15.0
ARTS/CYBER	3.5
Math Model	7.0
CGI Sample	10.0
CGI (present)(upgraded)	104.0 87.0
Total (present)(upgraded)	139.5 122.5
ADC Sample	15.0
ARTS/CYBER	3.5
Math Model	7.0
CRDS Sample (40) (60)	12.5 8.3
CRDS (40) (60)	99.2 66.6
Total (40) (60)	137.2 100.4
ADC Sample	15.0
ARTS/CYBER/Model	60.0
Motion Base	37.2
Total	112.2

11/01/91. 12.01.37. OPT= 4 COLBP= .5000 LIMCOL= .1000 TSTHETA= .000300

T	W	ADC	THEDEG	THCGI	COUNT	INC	ALTCGI	THETA	QB	QBDOT
2.01	F	.14	5.54038	.00000	180	0	5692.25	.0966978	.0000238	-.0000026
2.04	F	.14	5.54042	.00000	180	0	5692.25	.0966985	.0000237	-.0000032
2.07	F	.14	5.54046	.00000	180	0	5692.25	.0966992	.0000235	-.0000053
2.10	F	.14	5.54050	.00000	180	0	5692.25	.0966999	.0000232	-.0000070
2.13	F	.14	5.54054	.00000	180	0	5692.25	.0967006	.0000230	-.0000071
2.16	F	.14	5.54058	.00000	180	0	5692.25	.0967013	.0000228	-.0000073
2.19	F	.14	5.54062	.00000	180	0	5692.25	.0967020	.0000225	-.0000090
2.22	F	.14	5.54065	.00000	180	0	5692.25	.0967027	.0000222	-.0000104
2.25	F	.14	5.54069	.00000	180	0	5692.25	.0967033	.0000218	-.0000116
2.28	F	.14	5.54073	.00000	180	0	5692.25	.0967040	.0000214	-.0000126
2.31	F	.14	5.54077	.00000	180	0	5692.25	.0967046	.0000210	-.0000134
2.34	T	4.30	5.54128	.00000	181	0	5695.25	.0967135	.0008476	.0183639
2.37	T	4.30	5.54364	.00000	181	0	5695.25	.0967548	.0021630	.0353532
2.40	T	4.30	5.54868	.00000	181	0	5695.25	.0968427	.0039328	.0511128
2.43	T	4.30	5.55713	.00000	181	0	5695.25	.0969903	.0061216	.0656778
2.46	T	4.30	5.56969	.00000	181	0	5695.25	.0972095	.0086947	.0790733
2.49	T	4.30	5.58699	.00000	181	0	5695.25	.0975114	.0116183	.0913267
2.52	T	4.30	5.60961	.00000	181	0	5695.25	.0979061	.0148590	.1024577
2.55	T	4.30	5.63805	.00000	181	0	5695.25	.0984025	.0183847	.1124998
2.58	T	4.30	5.67283	.00000	181	0	5695.25	.0990095	.0222455	.1232955
2.61	T	4.30	5.71440	.00000	181	0	5695.25	.0997351	.0262174	.1293641
2.64	T	4.30	5.76298	.00000	181	0	5695.25	.1005830	.0304119	.1363317
2.67	T	4.30	5.81892	.00000	181	0	5695.25	.1015594	.0347716	.1423269
2.70	T	4.30	5.88249	.00000	181	0	5695.25	.1026688	.0392695	.1473945
2.73	T	4.30	5.95390	.00000	181	0	5695.25	.1039151	.0438793	.1515730
2.76	T	4.29	6.03332	.00000	181	0	5695.25	.1053012	.0485761	.1548981
2.79	T	4.30	6.12087	.00000	181	0	5695.25	.1068294	.0533406	.1575088
2.82	T	4.30	6.21667	.00000	181	0	5695.25	.1085013	.0581511	.1594042
2.85	T	4.30	6.32078	.00000	181	0	5695.25	.1103181	.0629858	.1605716
2.88	T	4.30	6.43318	.00000	181	0	5695.25	.1122802	.0678243	.1610463

Figure 45 - TSRV Reordered Program Output

08/02/91. 11.57.38. OPT= 4 COLBF= .5000 LIMCOL= .1000 TSTHETA= .0002 RUN= 3 TAPE42

T	CRDCLR	ADC	SX	SY	ALT	PSID	THETAD	PHID	ALTCGI	THETA	Q	QDOT
.03	.00	.14	-24972.06	2500.00	2500.00	.00000	4.89595	.00000	5692.25	.08545	.00000	-.00001
.06	.00	.14	-24958.09	2500.00	2500.00	.00000	4.89595	.00000	5692.25	.08545	.00000	-.00001
.10	.00	.14	-24944.11	2500.00	2500.00	.00000	4.89595	.00000	5692.25	.08545	.00000	-.00001
.13	.00	.14	-24930.14	2500.00	2500.00	.00000	4.89595	.00000	5692.25	.08545	.00000	-.00002
.16	.00	.14	-24916.17	2500.00	2500.00	.00000	4.89595	.00000	5692.25	.08545	.00000	-.00002
.19	.00	.14	-24902.20	2500.00	2500.00	.00000	4.89595	.00000	5692.25	.08545	.00000	-.00002
.22	.00	.14	-24888.22	2500.00	2500.00	.00000	4.89594	.00000	5692.25	.08545	.00000	-.00002
.26	.00	.14	-24874.25	2500.00	2500.00	.00000	4.89593	.00000	5692.25	.08545	.00000	-.00002
.29	.00	.15	-24860.28	2500.00	2500.00	.00000	4.89593	.00000	5692.25	.08545	.00000	-.00002
.32	.00	.15	-24846.30	2500.00	2500.00	.00000	4.89592	.00000	5692.25	.08545	.00000	-.00001
.35	.00	.14	-24832.33	2500.00	2500.00	.00000	4.89591	.00000	5692.25	.08545	.00000	-.00001
.38	.00	.15	-24818.35	2500.00	2500.00	.00000	4.89590	.00000	5692.25	.08545	-.00001	-.00001
.42	.00	.15	-24804.38	2500.00	2500.00	.00000	4.89589	.00000	5692.25	.08545	-.00001	-.00001
.45	.00	.14	-24790.41	2500.00	2500.00	.00000	4.89588	.00000	5692.25	.08545	-.00001	-.00001
.48	.00	.14	-24776.43	2500.00	2500.00	.00000	4.89586	.00000	5692.25	.08545	-.00001	-.00001
.51	.00	.14	-24762.46	2500.00	2500.00	.00000	4.89585	.00000	5692.25	.08545	-.00001	-.00001
.54	.00	.14	-24748.48	2500.00	2500.00	.00000	4.89584	.00000	5692.25	.08545	-.00001	-.00001
.58	.00	.14	-24734.50	2500.00	2500.00	.00000	4.89582	.00000	5692.25	.08545	-.00001	-.00001
.61	.00	.14	-24720.53	2500.00	2500.00	.00000	4.89581	.00000	5692.25	.08545	-.00001	-.00001
.64	.00	.14	-24706.55	2500.00	2500.01	.00000	4.89579	.00000	5692.25	.08545	-.00001	-.00001
.67	.00	.14	-24692.58	2500.00	2500.01	.00000	4.89577	.00000	5692.25	.08545	-.00001	-.00001
.70	.00	.14	-24678.60	2500.00	2500.01	.00000	4.89576	.00000	5692.25	.08545	-.00001	-.00001
.74	.00	.14	-24664.62	2500.00	2500.01	.00000	4.89574	.00000	5692.25	.08545	-.00001	-.00001
.77	.00	.15	-24650.65	2500.00	2500.01	.00000	4.89572	.00000	5692.25	.08545	-.00001	-.00001
.80	.00	.15	-24636.67	2500.00	2500.01	.00000	4.89570	.00000	5692.25	.08545	-.00001	-.00001
.83	.00	.15	-24622.69	2500.00	2500.01	.00000	4.89568	.00000	5692.25	.08545	-.00001	-.00001
.86	.00	.14	-24608.72	2500.00	2500.01	.00000	4.89566	.00000	5692.25	.08545	-.00001	-.00001
.90	.00	.14	-24594.74	2500.00	2500.01	.00000	4.89563	.00000	5692.25	.08544	-.00001	-.00001
.93	.00	.14	-24580.75	2500.00	2500.01	.00000	4.89561	.00000	5692.25	.08544	-.00001	-.00001
.96	.00	.14	-24566.78	2500.00	2500.01	.00000	4.89559	.00000	5692.25	.08544	-.00001	-.00001
.99	.00	.14	-24552.80	2500.00	2500.01	.00000	4.89556	.00000	5692.25	.08544	-.00001	-.00001
1.02	.00	.14	-24538.82	2500.00	2500.01	.00000	4.89554	.00000	5692.25	.08544	-.00001	-.00001
1.06	.00	.14	-24524.84	2500.00	2500.01	.00000	4.89551	.00000	5692.25	.08544	-.00001	-.00001
1.09	.00	.14	-24510.87	2500.00	2500.01	.00000	4.89548	.00000	5692.25	.08544	-.00001	-.00001
1.12	.00	.14	-24496.89	2500.00	2500.02	.00000	4.89546	.00000	5692.25	.08544	-.00001	-.00001
1.15	.00	.15	-24482.91	2500.00	2500.02	.00000	4.89543	.00000	5692.25	.08544	-.00002	-.00001
1.18	.00	.15	-24468.93	2500.00	2500.02	.00000	4.89540	.00000	5692.25	.08544	-.00002	-.00001
1.22	1.00	4.28	-24454.95	2500.00	2500.02	.00000	4.89709	.00000	5695.25	.08547	-.00002	.05849
1.25	1.00	4.28	-24440.97	2500.00	2500.01	.00000	4.90554	.00000	5695.25	.08562	.00279	.11368
1.28	1.00	4.27	-24426.98	2500.00	2500.00	.00000	4.92386	.00000	5695.25	.08594	.00731	.16746
1.31	1.00	4.27	-24413.00	2500.00	2499.99	.00000	4.95513	.00000	5695.25	.08648	.01353	.22003
1.34	1.00	4.27	-24399.01	2500.00	2499.96	.00000	5.00236	.00000	5695.25	.08731	.02141	.27171
1.38	1.00	4.27	-24385.03	2500.00	2499.92	.00000	5.06855	.00000	5695.25	.08846	.03094	.32278
1.41	1.00	4.27	-24371.03	2500.00	2499.87	.00000	5.15666	.00000	5695.25	.09000	.04208	.37353

Figure 46 - VMS Reordered Program Output

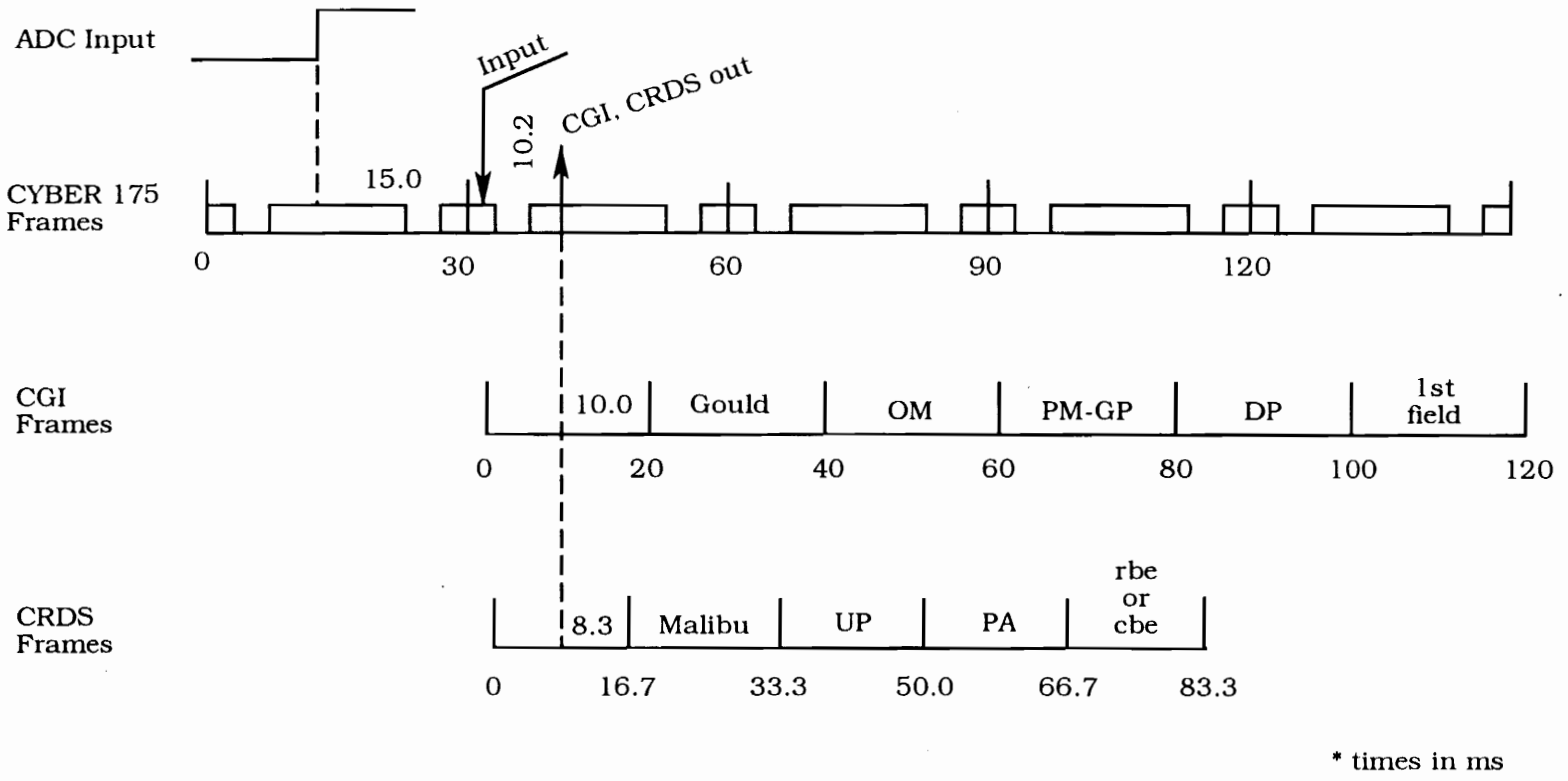


Figure 47 - TSRV Transport Delay (Reordered Program Flow)

was collected (see Appendix G TABLE G-4) and an analysis of these transport delays is shown in TABLE 5. The transport delay of the math model was 7.0 ms (25.5 ms - 18.5 ms). The total transport delay to the end of the first CGI field was measured at 139.5 ms. Subtracting 17.0 ms for the CGI upgrade yields a transport delay of 122.5 ms. Again, since the transport delays for the CRDS were moved with the CGI and the output parameters were verified to be changing during the first frame the transport delays for the CRDS were not remeasured. The transport delay to the beginning of a 40 Hz display is 137.2 ms and 100.4 ms to the beginning of a 60 Hz display. The motion base response was measured at 112.2 ms, 26.1 ms longer than in the previous test. The expected delay was 116.1 ms therefore a discrepancy of 3.9 ms exists. One reason for the difference can be in the data collection/averaging process. This is unlikely since in all other tests this has proved to yield a fairly consistent, accurate measurement. A second, and more probable, reason is the signal conditioning circuit shown in Figure 28. The voltage characteristics of the motion base's trim condition change on a daily basis thus requiring the trim potentiometer on the signal conditioner to be adjusted from test to test. Since the voltage differences detected and amplified by the conditioner are very small an even smaller error on the potentiometer can cause a slightly earlier or later detection of transition. However, since the motion base transition occurred approximately 1 frame later the delay was considered a success. The phase difference between the upgraded CGI and the motion base is only +10.3 ms (122.5 ms - 112.2 ms). Between the CRDS and the motion base the phase difference is +25.0 ms (137.2 ms - 112.2 ms) for 40 Hz displays and -11.8 ms (100.4 ms - 112.2 ms) for 60 Hz displays. All video phase differences are within ±30 ms of the motion cue. A timeline of these delays can be found in Figure 48.

Verifying Numerical Accuracy

In order to verify the numerical accuracy of the software changes the modifications discussed in chapter 6 were implemented with duplicate variables so that the previous program flows (Figure 39 and Figure 42)

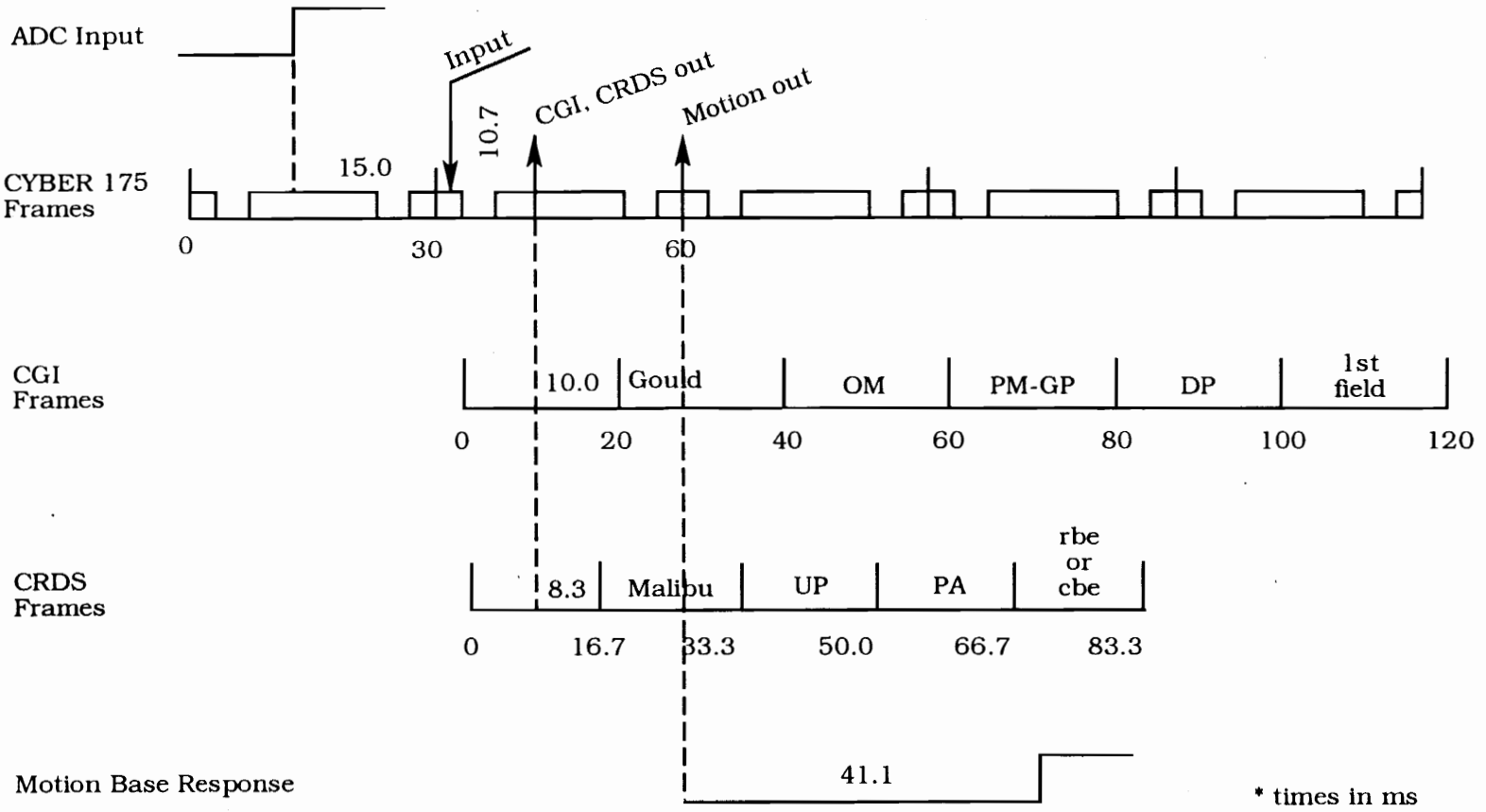


Figure 48 - VMS Transport Delays (Reordered Program Flow)

could still be executed and compared to those of the new program flows (Figure 41 and Figure 44). For both the TSRV and the VMS the new program flow was only computed when two preset switches at the real-time control console were active. The first switch controlled which program flow was to be implemented. This allowed accurate timing of the model delay under different program structures. The second switch allowed both program flows to be executed in order to compare the variables calculated. This method allows the direct comparison of data in any variable to determine if the program flows are numerically the same. A detailed analysis is still being conducted however at this time no discrepancies in variable data has been discovered. It is not expected that any discrepancies in the data will be discovered because the equations are still being executed in the same order as in the previous program flow, they are just shifted with respect to when they are executed.

CHAPTER 8

RECOMMENDATIONS AND CONCLUSIONS

The intent of this paper was originally to develop a method of detecting and documenting transport delays within the FSF at NASA Langley Research Center. However, as the project grew the data revealed several problem areas in the FSF that required immediate attention. Research was conducted into real-time program improvements and hardware improvements.

A VLD was developed which provided a more accurate and reliable method to detect video transitions in the output of an image or display generator.

Testing was conducted in both the frequency and time domains. The data that resulted from the time domain revealed negligible differences from that of the frequency domain, especially in the lower frequency range. Although the deviation between data was larger at higher frequencies this is not considered significant since pilot inputs are typically low frequency inputs. This is particularly true in the case of transport aircraft.

The hardware transport delays were measured for several pieces of equipment and used to calculate the overall transport delay of the simulation. The math model software was then added to the simulation and the additional transport delay was attributed to the math model. Through the use of this technique unexpected transport delays were discovered in the CGI and the software implementation of the math model. The additional CGI transport delay is currently being corrected by the manufacturer. A detailed study of the real-time program implementation was conducted.

The results of the study showed that the integration method utilized in the TSRV required two frames to complete and the data was output in the third frame. The VMS integration methods only required one frame but the output did not occur until the second frame. The TSRV required a reordering of the code and the addition of quaternion transformations and

integrations to the program in order to correct the problem. This improved the TSRV transport delay by 57.7 ms. The VMS code only required reordering to accelerated data availability to peripheral hardware. The reordering of the code improved the VMS transport delay by 28.6 ms. Since these savings do not impact the computational accuracy of the math model it is recommended that all of the real-time programs be modified to reflect this program order.

Once the unexpected delays are eliminated from the CGI and the software delays are corrected the TSRV should be accelerated by 74.7 ms and the VMS 45.6 ms. The overall transport delay for the TSRV and VMS, including the math model, to the end of the first CGI field will be reduced to 118.0 ms and 122.5 ms, respectively. The TSRV transport delays for the CRDS are accelerated to 136.9 ms for a 40 Hz display and 100.1 ms for a 60 Hz display. The VMS transport delays for the CRDS are accelerated to 137.2 ms for a 40 Hz display and 100.4 ms for a 60 Hz display. The motion base transport delay was delayed by one frame to respond at 112.2 ms. Since these transport delays include the math model the FSF at NASA Langley the transport delays now fully meet the FAA guidelines of 150.0 ms. In addition the synchronization between motion and visuals is within ± 30.0 ms to reduce possibility of miscuing.[29]

REFERENCES

[1] Copeland, J.L., Research through Simulation, NASA Langley Research Center, NF-125, p. 1.

[2] Ibid., p. 1.

[3] Martin, D.J., Jr., A Digital Program For Motion Washout On Langley's Six-Degree-Of-Freedom Simulator, NASA Contractor Report 145219, July 1977, p. 2.

[4] Roskam, J., Flight Dynamics of Rigid and Elastic Airplanes - Part One, Roskam Aviation and Engineering Corporation, 1972, pp. 333-333.

[5] Hettinger, L.J., Lane, N.E. and Kennedy, R.S., Diagnostic Measurement Approaches to the Problem of Simulator Sickness in Flight Simulation, Flight Simulation Technologies Conference, AIAA paper 88-4624-CP, September 1988, pp. 288-290.

[6] Federal Aviation Administration, Airplane Simulator and Visual System Evaluation, Advisory Circular AC 120-40A, July 31 1986, Appendix 5 p. 7.

[7] Middendorf, M.S., Fiorita, A.L. and McMillian, G.R., The Effects of Simulator Transport Delay on Performance, Workload, and Control Activity During Low-Level Flight, Flight Simulation Technologies Conference, AIAA paper 91-2965-CP, August 1991, p. 425.

[8] Bailey, R.E., Knotts, L.H., Horowitz, S.J., and Malone, H.L., Effect of Time Delay on Manual Flight Control and Flying Qualities During In-Flight and Ground-Based Simulation, Flight Simulation Technologies Conference, AIAA paper 87-2370, August 1987, pp. 37-38.

[9] Cook, A., McCauley, M. and Voorhees, J., Recent Proceeding of the

NASA Steering Committee on Simulator Induced Sickness, Flight Simulation Technologies Conference, AIAA paper 91-2973, August 1991, pp. 478-488.

[10] Johnson, W.V. and Middendorf, M.S., Simulator Transport Delay Measurement Using Steady States Techniques, Flight Simulation Technologies Conference, AIAA paper 88-4619-CP, September 1988, p. 250.

[11] Ibid., p. 252.

[12] Crawford, D., Cleveland, J. and Staib, R., The Langley Advanced Real-Time Simulation (ARTS) System, Flight Simulation Technologies Conference, AIAA paper 88-4595, August 1988, pp. 109-121.

[13] Langley Aerospace Test Highlights, NASA Langley Research Center, NASA Technical Memorandum 104090, 1990, pp. 92-94.

[14] Ibid., p. 106.

[15] Parish, R.V. and Ashworth, B.R., The Effect of Digital Computing on the Performance of a Closed-Loop Control-Loading System, NASA TN D-8371, December 1976, p. 5.

[16] Crawford, D., Cleveland, J. and Staib, R., pp. 110-111.

[17] Crawford, D., Cleveland, J. and Staib, R., pp. 111-112.

[18] CAMAC Instrumentation and Interface Standards, Institute of Electrical and Electronic Engineers (IEEE) Standard 583-1975, IEEE Standard 595-1976, IEEE Standard 596-1975, IEEE Standard 683-1976, 1976, pp. 7-12.

[19] Johnson, W.V. and Middendorf, M.S., p. 252.

[20] Parish, R.V., Dieudonne, J.E., Martin, D.J., Jr. and Copeland, J.L., Compensation Based On Linearized Analysis For A Six-Degree-Of-Freedom Motion Simulator, NASA TN D-7349, November 1973, pp. 1-8.

[21] Dieudonne, J.E., Parish, R.V. and Bardusch, R.E., An Actuator Extension Transformation for a Motion Simulator and an Inverse Transformation Applying Newton-Raphson's Method, NASA TN D-7349, November 1973, pp. 4-12.

[22] Parish, R.V., Dieudonne, J.E., Martin, D.J., Jr. and Copeland, J.L., pp. 17-18 and 33-34.

[23] Federal Aviation Administration, p. 7.

[24] Ibid., p. 7.

[25] Battin, R.H., An Introduction To Mathematics And Methods Of Astrodynamics, AIAA Education Series, 1987, pp. 93-95.

[26] Barker, L.E., Jr., Bowles, R.L., Williams, L.H., Development And Application Of A Local Linearization Algorithm for the integration of Quaternion Rate Equations In Real-Time Flight Simulation Problems, NASA TN D-7347, December 1973, pp. 5-8.

[27] Federal Aviation Administration, p. 7.

[28] Cardullo, F.M., The Need for Non-Visual Motion Cuing in Flight Simulators Employed for Engineering or Research, Cardullo, Brown and Associates, December 30, 1990, p. 13.

[29] Federal Aviation Administration, p. 7.

APPENDIX A

VLD Program Code

```
DESIGN IS "VID_3"

BEGIN
        DEVICE IS "EPM5016"
        BEGIN
            clk@20,h@19,v@12,video@11 : INPUT;
            raster@7,nraster@8,nlim@9,nnlim@10 :INPUT;
            trig@3 : OUTPUT;
        END;
END;

SUBDESIGN VID_3
(
        clk             : INPUT;
        h               : INPUT;
        v               : INPUT;
        video           : INPUT;
        raster          : INPUT;
        nraster         : INPUT;
        nlim            : INPUT;
        nnlim           : INPUT;
        trig            : OUTPUT;

)
VARIABLE

        delay[5..0]     : DFF;
        count[6..0]     : DFF;
        latch_vid       : DFF;
        raster0         : EXP;
        raster1         : EXP;
        nlim0           : EXP;
        nlim1           : EXP;

BEGIN

        % DEBOUNCE SWITCHES %

        raster0 = raster & raster1;
```

```
raster1 = nraster & raster0;
nlim0 = nlim & nlim1;
nlim1 = nnlim & nlim0;

delay[].clk = clk;
count[].clk = clk;
latch_vid.clk = clk;

% IF RASTER VIDEO AND NON-NLIM PICTURE %

if (raster0==1 & nlim0!=1) then

    % DELAY H BY 6 MICRO-SECOND INTERVAL %

    if (h==1 & delay[]!=7) then
      delay[] = delay[] + 1;
    elsif (h==1 & delay[] ==7) then
      delay[] = 7;
    elsif (h==0) then
      delay[] = 0;
    end if;

    % count 45 pulses for non-NLIM video %

    if (v==1 & count[]!=45 & delay[]==6) then
      count[] = count[] + 1;
    elsif (v==1 & count[]==45) then
      count[] = 45;
    elsif (v==0) then
      count[] = 0;
    else
      count[] = count[];
    end if;

    % latch video state to trigger %

    latch_vid.clrn = video & count[]==45;
    latch_vid.d = (video & count[]==45 # latch_vid.q);
    trig = latch_vid.q;
```

```
% IF RASTER VIDEO AND NLIM PICTURE %

elsif (raster0==1 & nlim0==1) then

    % DELAY H BY 23 MICRO-SECOND INTERVAL %

    if (h==1 & delay[]!=24) then
      delay[] = delay[] + 1;
    elsif (h==1 & delay[] ==24) then
      delay[] = 24;
    elsif (h==0) then
      delay[] = 0;
    end if;

    % count 12 pulses for non-NLIM video %

    if (v==1 & count[]!=12 & delay[]==23) then
      count[] = count[] + 1;
    elsif (v==1 & count[]==12) then
      count[] = 12;
    elsif (v==0) then
      count[] = 0;
    else
      count[] = count[];
    end if;

    % latch video state to trigger %

    latch_vid.clrn = video & count[]==12;
    latch_vid.d = (video & count[]==12) # latch_vid.q;
    trig = latch_vid.q;

% IF CALLIGRAPHIC VIDEO %

elsif (raster0!=1) then
    trig = video;
end if;

END;
```

APPENDIX B

Modified CYBER 175 Code

TSRV Main Program (FOUR2) - The entire compiled listing of the main routine is included. In order to conserve space the variable declarations and initializations have been deleted.

```
      OVERLAY(TCVRSFS,4,2)
      PROGRAM FOUR2
*
**    THIS PROGRAM IS THE REAL TIME EXECUTIVE
*

*** Common blocks and variable inlitizations have been deleted
    except where changes have been made ***

C     LOGICAL VARIABLE TO USE QUATERNION METHOD
      LOGICAL QUATOP
      REAL IK,JK,KK
C     LIMIT FUNCTION
      XLIM(P,Q,R) = AMIN1(R,AMAX1(P,Q))
C
C     SAVE CURRENT SECONDARY OVERLAY NUMBER

      ISLP = ISL
*
 9041 CONTINUE
*
*
**    INITIALIZE MFD
*
      CALL MFDINT
*
**    UNPACK IDIS INTO TCVI'S AND MODE LOGICS
*
      CALL UNDISCR
*
**    ESTABLISH RECOVERY ADDRESS FOR POPUP OF PROGRAM
*
      CALL RESADD(1,*9000)
      DATA IRSADIC / 0 /
      IF (IRSADIC .NE. 0) CALL RSADDIC
*
**    ENTER SYNCHRONIZED REAL-TIME
*
*
```

```
**    CALL RTART 1ST IN CASE ALREADY IN SRT
*
 9000 CALL RTART
      CALL RTSRT
**
*** POST REQUEST TO FILL ASYNCHRONOUS INPUT DATA BUFFER IN CASE A LAM
*** (INTERRUPT) HAS BEEN SET AND THE CYBER HAS NOT READ THE DATA
*** (THIS RESULTS IN A DEVICE HANG )
**
      IF ( ISENSW(1) .EQ. 1 ) CALL RTNETIN ( STATMI, IDUM )
*
**    IF IN THE MIDDLE OF A RUN WANT TO BE ABLE TO RECOVER IN
**    OPERATE  (E.G. IF WANT TO GET DUMPS IN A SERIES TO DEBUG
**    A PROBLEM) WITHOUT RESETTING EVERYTHING (E.G. LATIC)
*
      IF ( T .GT. 60. ) GOTO 9006
*
      CALL SETUPIC
      CALL PACKC
      CALL SETUP0
      CALL UNPACKC
      IF (ISENSW(1) .EQ. 1) CALL MIMIC
*
**                    **********
**                    ENTRY POINT FOR RESET
*                     **********
 9003 CONTINUE
*
**    UNPACK DISCRETES
*
*
      CALL UNPACKC
      CALL UNDISCR
*
**    RESET TOUCHDOWN PRINT AND AUTO HOLD FLAGS
*
      IPRTHCG=1
*
      LITE(45) = .FALSE.
```

```
*
**   RNWAYIC GETS EADI RUNWAY COORDINATES - LOGIC(6) IS SET
**   WHENEVER THERE IS NEW RUNWAY INFORMATION (NCDK3)
*
     IF (LOGIC(6)) CALL RNWAYIC
*
**   IF NON-HARDWARE OVERLAY, WANT TO FORCE THROTTLE
**   POSITION (THROTLC) TO THE TRIM VALUE
*
     IF (ISL .EQ. 4) THROTLC = THRTRIM
*
**   RESET LOOP FOR THE AUTO PILOT
*
     CALL APRESET
     IF (T .NE. 0.) THEN
*
**     THE 1ST PASS BACK FROM OPERATE, T WILL BE NON-ZERO (UNTIL
**     SRESET IS CALLED) SO PUT HERE ANY TYPE OF RE-INITIAIZATION
**     FOR 1ST PASS IN RESET
*
       ICMIM   = .TRUE.
       TMFD    = 0.
       TSTREAM = 9999999.
     ENDIF
*
**   RESET STATE VARIABLES TO IC'S
*
     CALL SRESET
*
**   TO TRIM OR TO CHANGE SECONDARY OVERLAYS DEPRESS SWITCH
*
     IF (.NOT. FSS(1)) GOTO 2
**
*** DISABLE VOTAN PORT
**
     MIMOUT(1) = O" 12000 00000 00000 00000 "
**
     IF (ISENSW(1) .EQ. 1) CALL RTNETOT ( STATMO , IDUM )
*
```

```
**    NEED TO GO TO HOLD1 TO CHANGE OVERLAYS
*
      CALL RTART
*
**    IF NOT TRIMMING, THEN MUST HAVE SET INTEG(2) TO THE
**    DESIRED OVERLAY - RETURN SO CAN LOAD IT
*
      IF (ISL .NE. ISLP) RETURN
*
**    TRIMMING, TURN OFF SUCCESSFUL TRIM LITE AND SET
**    SECONDARY OVERLAY NUMBER TO 1 (4,1)
*
      ISL = 1
      LITE(48) = .FALSE.
      CALL PACKC
      CALL SETUP0
*
      CALL RTCYCLE
      RETURN
*
    2 CONTINUE
**                    **********************
**                    ENTRY POINT FOR RECYCLE
**                    **********************
 9006 CONTINUE
*
**    UNLESS FSS(23) IS ON, RE-INITIALIZE THE MIM
*
      IF ((ISENSW(1) .EQ. 1 .OR. VBATCH .EQ. 0) .AND. ICMIM) CALL MIMIC
*
*
**    DISCRETES ARE UNPACKED ONCE EACH ITTERATION
*
*
      IF (.NOT. MCP(3)) CALL UNPACKC
*
      IF (.NOT. MCP(3)) CALL UNDISCR
*
**    IF HAVE A BAD ADC CAN CHANGE IBADADC TO THE ADC THAT
```

```
**    IS BAD AND CHANGE INEWADC TO A FREE ADC AND THEN
**    ASK THE HARDWARE PERSONNEL TO MAKE THE NEEDED SWAP
*
      ADC(IBADADC) = ADC(INEWADC)
*
*
 C TRANSPORT DELAY STUDY
 C OPTION TO BYPASS MOST OF THE MATH MODEL AND CHANGE CRDCLR
ACCORDING TO
 C   CHANGE IN ADC(IADCCC) = COLUMC
      IF (NOPTION .EQ. 5 ) THEN
        CRDCLR = ADC(IADCCS) .GT.  COLBP
        CALL CGIINTF
        IF (FSS(7)) CALL CDCAGT
        GOTO 54
      ENDIF

 C NOPTION = 6 SINE/SQUARE WAVE = ADC(IADCCS) = COLUMC AND CRD & CGI
 C            POSITIONS AND SKIP MOST OF THE MATH MODEL

      IF  (NOPTION .EQ. 6 )   THEN
        TW = AMOD(T, SPERIOD )
        CRDCLR = .FALSE.
        IF  (TW .GE. 0.5* SPERIOD )    CRDCLR = .TRUE.
        CALL CGIINTF
        IF (FSS(7)) CALL CDCAGT
        GOTO 54
       ENDIF

*
**    CONTROL FROM COCKPIT - PROCESS INPUTS FROM COCKPIT
*
      CALL TCVCAB
*
*
      CRDSKED = (( ISENSW(2) .EQ. 1 ) .OR. ( ISENSW(6) .EQ. 1 ))
     *      .AND. (.NOT. FSS(16))
**    IF IN AUTOHOLD STATE DO NOT CALCULATE DERIVATIVES AND MOMENTS
*
```

```
      IF (LITE(45)) GOTO 8888
*-* WAS CALLING MISCEL, RUNWAY, ACCEL HERE
      IF (RESET) THEN
        CALL MISCEL
        CALL RUNWAY
        CALL ACCEL
      ENDIF
C
C
C*    INITIALIZE FOR INTEGRATION OF QUATERNIONS
C*
      IF (QUATOP.AND.RESET) THEN
      CPSI2    = COS(.5*PSI)
      SPSI2    = SIN(.5*PSI)
      CTHETA2  = COS(.5*THETA)
      STHETA2  = SIN(.5*THETA)
      CPHI2    = COS(.5*PHI)
      SPHI2    = SIN(.5*PHI)
C*
      AB1      = CPSI2*CTHETA2*CPHI2 + SPSI2*STHETA2*SPHI2
      AB2      =-CPSI2*STHETA2*SPHI2 + SPSI2*CTHETA2*CPHI2
      AB3      = CPSI2*STHETA2*CPHI2 + SPSI2*CTHETA2*SPHI2
      AB4      = CPSI2*CTHETA2*SPHI2 - SPSI2*STHETA2*CPHI2
      ENDIF
C*
**    MICRO-WAVE LANDING SYSTEM
*
      IF (LOGIC(25)) CALL MLS
*
**    DETERMINE WHETHER MLS OR INS VARIABLES WILL FEED TO DISPLAY AUTO
**       PILOT, AND/OR GUIDANCE
*
      CALL SWITCH
*
**    MODE SELECT PANEL LOGIC
*
      CALL MSPLOG
*
**    COMPUTE CONTROL SYSTEM OUTPUTS
```

```
*
      CALL INPUT
*
**    COMPUTE ENGINE FORCES AND MOMENTS
*
      CALL ENGFM
*
**    COMPUTE LANDING GEAR FORCES AND MOMENTS
*
      CALL LNGFM
*
**    COMPUTE AERODYNAMIC FORCES AND MOMENTS
*
      CALL AEROFM
*
**    SUM ALL FORCES AND RESOLVE INTO LOCAL FRAME
*
      CALL SUMFM
*+*   CALCULATE ACCELERATIONS
      CALL ACCEL
**                    **********
**                    ENTRY POINT FOR HOLD
**                    **********
 9002 CONTINUE
*
**    IF HOLD, WANT TO UNPACK DISCRETES
*
      IF (MCP(2)) CALL UNPACKC
      IF (MCP(2)) CALL UNDISCR
*
**    COMPUTE DERIVATIVES OF STATE VARIABLES
*
      CALL DERIV
*
 8888 CONTINUE
C*
C** INTERFACE WITH THE CGI ?
C
      IF (FSS(18)) THEN
```

```
        CGIINT = .TRUE.
        FSS(18) = .FALSE.
      ENDIF
C     THEDEG = 57.2958 * THETA
      IF (RESET .AND. NOPTION .NE. 5 .AND. NOPTION .NE. 6) THEN
       IF ( ISENSW(4) .EQ. 1  .OR.  ISENSW(5) .EQ. 1 ) CALL CGIINTF
       IF (FSS(7)) CALL CDCAGT
      ENDIF
*
*** TO BE ABLE TO RE-INITIALIZE THE CRD IN ALL PROGRAM STATES
*
      IF (FSS(5)) THEN
      IF ( CRDSKED ) CALL CRDINIT ( 0 )
        FSS(5) = .FALSE.
        NUPAGE = -1
      ENDIF
*
*
**    IF THE USER WISHED TO USE VOICE, THEY COULD PUT THE CALL
**    HERE AND COULD LOOK AT VOICE ON DLWLIB (WEI) AS A SAMPLE
*
C     IF (AMOD(TMFD+.015, .5) .LT. .028) CALL VOICE
**    INTERFACE WITH THE CRD (GRAPHICS COMPUTER)
*
*
**    PROCESS NCDU KEYBOARD INPUTS
*
      IF (FSS(11)) CALL IONCDU
*
**    NCDU FAST LOOP AND GUIDANCE EXECUTIVE
*
    4 CALL ARTEXEC
*
*** TEST FOR DATA INPUT VIA MIM (MINICOMPUTER INTERFACE MODULE)
*
      IF (ISENSW(1) .EQ. 1) CALL MIMTALK
*
**    MIMTALK WILL HAVE CALLED UPAKMIM WHICH WILL HAVE UNPACKED THE
**    TOUCHES - THEY ARE NOW AVAILABLE
```

```
*
**    PROCESS NCDU OUTPUTS (NCDU PAGE)
*
*
**    COMMUNICATE WITH NCDU MICROPROCESSOR
*
      IF (NUPAGE .NE. 0 ) CALL BYTES
*
      IF (FSS(11) .AND. ISENSW(1) .EQ. 1) CALL IO8080(IBYTES)
*
**    CLEAR 'NEED A NEW PAGE' FLAG
*
      NUPAGE = 0
**
*** MIM DEBUG TERMINAL PRINT OF INPUT/OUTPUT AND DEMAND ARRAYS, ALSO
*** OUTPUT MIM TEST MESSAGE IF TESTMES=.TRUE. (CONTINUOUSLY IF
CONTMES)
**
      IF (ISENSW(1) .EQ. 1) CALL MIMDBUG
*
   50 CONTINUE
*
**    SET OPERATE,HOLD,AND RESET LIGHTS FOR THE CAB
*
      TCVO2(20) = LOPER
      TCVO2(21) = HOLD
      TCVO2(22) = RESET
*
**    SETTING LOGIC(10) GIVES A PRINT OUT OF THE GUIDANCE BUFFERS
*
      IF (.NOT. LOGIC(10)) GOTO 79
*
**    MUST BE IN HOLD1 TO ACCESS THE PRINTER
*
      CALL RTART
      GBFLAG = AGBF .EQ. LOCF(GBUF1)
*
      IF(.NOT.GBFLAG) WRITE(MF,779)
      IF(     GBFLAG) WRITE(MF,780)
```

```
**
 779 FORMAT(1H1//,6X,'**    ACTIVE    **',5X,'** PROVISIONAL  **')
 780 FORMAT(1H1//,6X,'** PROVISIONAL  **',5X,'**    ACTIVE    **')
*
**   K WILL BE THE WAYPOINT NUMBER, J THE BUFFER ELEMENT
*
     K = 1
     DO 78 J=1,30
*
     WRITE(MF, 781) J, GBUF2(K), GBUF1(K), WPN(GBUF2(K)), PN(GBUF1(K))
 781 FORMAT(//2X, 'WAYPOINT NUMBER',I3 /6X, O20, 5X, O20, (6X,A5))
*
     DO 77 I=2,19
     KJ = K + (I-1)
*
     WRITE(MF,782) I, GBUF2(KJ), GBUF1(KJ)
 782 FORMAT(2X, I2, 2X, 1PE16.7, 9X, 1PE16.7)
  77 CONTINUE
*
     K = K + 19
*
  78 CONTINUE
*
**   TURN OFF LOGIC(10) AND DISPOSE MF TO THE PRINTER
*
     LOGIC(10) = .FALSE.
*
*
**   WANT TO ROUTE GBUF PRINT IN R.T. BUT NOT IN BATCH (LOGIC(10)
**   GETS THE GBUF PRINT)
*
     IF (VBATCH .NE. 0) THEN
       REWIND MF
       CALL RTOUT("MF", "RB")
     ENDIF
     CALL REWRTF(1)
*
**   RETURN TO SYNCHRONIZED REAL TIME
*
```

```
      CALL RTSRT
*
   79 CONTINUE
*
**    DETERMINE VARIABLE SET TO BE DISPLAYED ON CONSOLE READOUT
*
*
      IF ( LOPER ) GOTO 5
*
**    OUTPUT DACS AND UPDATE CONSOLE DISPLAY IN RESET AND HOLD LOOPS
*
**    EVEN IF THERE IS A SUCCESSFUL TRIM SOLUTION (LITE(48)), IT TAKES
**    A WHILE FOR THE COCKPIT THROTLE AND STAB TO DRIVE TO THE
**    TRIM VALUES - THEY ARE THERE WHEN LITE(47) GOES ON
*
*
**    IF ACNORM IS NOT TINY, NEED TO TRIM ON 6 VARIABLES RATHER
**    THAN JUST 3 (INTEG(6) = 6)
*
      LITE(47) = ABS(VTDOT) .LT. .05 .AND. ABS(ACNORM) .LT.
*                                          .05
**    SET THE ANALOGS FOR THE COCKPIT
*
      CALL DACS
*
**    STRIP CHART RECORDERS
*
      CALL BRUSH
*
**    PACK THE TCVO'S AND THE RED AND WHITE LIGHTS INTO ODIS
*
      CALL PKDISCR
*
      CALL PACKC
*
**    IF FSS(19) IS ON, SET TURRET LABELS
*
      IF (FSS(19)) CALL SETUPL
*
```

```
**    IF FSS(20) IS ON, SET TURRET VALUES
*
      IF (FSS(20)) CALL TURVAL
*
      CALL SETUP0
*
    5 CONTINUE
*
**    READ IN AN NCDU KEY EVERY 5TH PASS
*
      DATA NKNT / 0 /
*
**    LOGIC(42) IS TO BYPASS NCDU KEYBOARD
*
      IF (.NOT. LOGIC(42)) GOTO 54
*
      IF (MOD(NKNT,5) .EQ. 0) INTEG(14) = INTEG14(I14)
**    THE INPUT FILE WILL CONTAIN A "80" WHEN DONE
      IF (INTEG(14) .EQ. 80) CALL FINI
*
      IF (MOD(NKNT,5) .EQ. 0) I14 = I14 + 1
*
      NKNT = NKNT + 1
*
   54 CONTINUE
*
**    IF HAVE A BAD DAC CAN CHANGE IBADDAC TO THE DAC THAT
**    IS BAD AND CHANGE INEWDAC TO A FREE DAC AND THEN
**    ASK THE HARDWARE PERSONNEL TO MAKE THE NEEDED SWAP
*
      DAC(INEWDAC) = DAC(IBADDAC)
*
**    RTMODE DETERMINES WHICH MODE THE PROGRAM IS IN AND
**    BRANCHES ACCORDINGLY
*
*
      IF (ISL .EQ. 4 .AND. .NOT. RESET) THEN
*
        IF (.NOT. FSS(15) .AND. VBATCH .EQ. 0) THEN
```

```
*
        CALL BATCHU
*
       ELSE
*
        CALL BATCH
*
       ENDIF
*
      ENDIF
      IF (.NOT.RESET)CALL RTMODE(*9001,*9002,*9003,*9004,*9014,*9015)
*
**    IF RESET, BEGIN THE BACKGROUND CALCULATIONS
*
      CALL BEGBACK(TIMEU,TIMER,*9003)
*
**    BKEXEC IS THE BACKGROUND EXECUTIVE
*
      IF (FSS(7)) CALL BKEXEC
*
**    WHEN BKEXEC HAS COMPLETED, END THE BACKGROUND
*
      CALL ENDBACK
*
**    MOD TO DOUBLE THE OUTPUT RATE TO THE NCDU
*
*** SECOND CALL TO OUTSBC TO DOUBLE NCDU OUTPUT FREQUENCY N
LONGER
*** NEEDED
*
**    IN RESET, DETERMINE WHERE TO BRANCH
*
      IF (ISL .EQ. 4) THEN
*
       IF (.NOT. FSS(15) .AND. VBATCH .EQ. 0) THEN
*
        CALL BATCHU
*
       ELSE
```

```
*
       CALL BATCH
*
      ENDIF
*
    ENDIF
*
    CALL RTMODE(*9001,*9002,*9003,*9004,*9014,*9015)
*
**                  **********************
**                  ENTRY POINT FOR OPERATE
**                  **********************
 9001 CONTINUE
*
**   TURN OFF RED LIGHT 8 (TRIMMED SUCCESSFULLY)
*
    LITE(48) = .FALSE.
*
**   INITIALLY TURN ON THE AUTO-HOLD LIGHT AND THEN IF ALL THE
**   TRAPS ARE PASSED, TURN IT OFF
*
    LITE(45) = .TRUE.
*
**   PROJECT THE ALTITUDE FOR THE NEXT ITTERATION AND IF IT
**   IS LESS THAN 5 FEET, GO INTO AN AUTOHOLD MODE
*
    DATA HCGMIN / 5. /
    DATA HDPAST / 0. /
*
    ALTPROJ = HCG + .5 * H * (3. * HDOT - HDPAST)
    HDPAST  = HDOT
*
    IF (ALTPROJ .LE. HCGMIN) GOTO 40
*
**   CAN SET A VALUE IN T(80) TO FORCE AUTO HOLD AT SOME HCG
*
    IF (HCG .LE. TABLE(80)) GOTO 40
*
**   ALSO WANT TO GOTO AUTO HOLD IF THE ACCELERATIONS ARE LARGE
```

```
*
      DATA HLDTHR / 96. /
      IF (ABS(AX + AY + AZ) .GT. HLDTHR) GOTO 40
*
**    SINCE PASSED ALL THE TRAPS, TURN OFF AUTO-HOLD LIGHT
*
      LITE(45) = .FALSE.
*
**    FIRST PASS,  RESET TIME OUT OF DETENT COUNTERS
*
      IF (T .EQ. 0) TABLE(68) = TABLE(69) = 0.
*
**    COMPUTE TIME OUT OF DETENT FOR COLUMN T(68)
**       AND WHEEL T(69)
*
      IF (PCWOD) TABLE(68) = TABLE(68) + H
      IF (RCWOD) TABLE(69) = TABLE(69) + H
*
**    REAL-TIME RECORDING
*
      IF (.NOT. LOGIC(11)) GOTO 14
      NT = INTEG(4) = 1

*
      XNT = NT * H
*
**    BASELINE PRINTS EVERY NT ITERATIONS BUT ALSO GIVES A TOUCHDOWN
**       PRINT AND ENDS THE PRINTS AT TOUCHDOWN
*
*
**    WHEN THIS IS 0, NO MORE PRINTING
*
      IF (HCG .LE. TABLE(79)) IPRTHCG = 0
*
   14 CONTINUE
*
*-*      WAS CALLING IGRATE6 HERE
*+*
*
```

```
C ******* QUATERNION OPTION **********************
      DATA QUATOP / .TRUE. /
      IF (QUATOP) THEN
C
C     BARKER'S METHOD  DESCRIBED IN NASA TN D-7347
C*
C*     QUATERNION UPDATE
C*
      OMEGAK2  = PB*PB + QB*QB + RB*RB
      OMEGAK2 = XLIM (OMEGAK2,1.E-6,1.E10)
      OMEGAK   = SQRT(OMEGAK2)
      RHOK     = H*OMEGAK*.5
      SRHOK    = SIN(RHOK)
      CRHOK    = COS(RHOK)
      C1       = CRHOK
      C2P      = SRHOK/OMEGAK
      C3P      = 2.*(1.-CRHOK) / OMEGAK2
      C4       = 4.*(H-2.*C2P) / OMEGAK2
      AK       = -.25*(PB*PBDOT + QB*QBDOT + RB*RBDOT)
      BK       =  .25*(PB*QBDOT - QB*PBDOT)
      CK       =  .25*(RB*PBDOT - PB*RBDOT)
      DK       =  .25*(QB*RBDOT - RB*QBDOT)
      HK       = C1 + C4*AK
      IK       = -C2P*RB - C3P*RBDOT + C4*BK
      JK       =  C2P*QB + C3P*QBDOT - C4*CK
      KK       =  C2P*PB + C3P*PBDOT - C4*DK
      AT1      =  HK*AB1 + IK*AB2 - JK*AB3 - KK*AB4
      AT2      = -IK*AB1 + HK*AB2 - KK*AB3 + JK*AB4
      AT3      =  JK*AB1 + KK*AB2 + HK*AB3 + IK*AB4
      AT4      =  KK*AB1 - JK*AB2 - IK*AB3 + HK*AB4
C*
C*   NORMALIZE
C*
      CORRECT  = 1./ SQRT(AT1*AT1+AT2*AT2+AT3*AT3+AT4*AT4)
      AB1      = CORRECT*AT1
      AB2      = CORRECT*AT2
      AB3      = CORRECT*AT3
      AB4      = CORRECT*AT4
```

```
C*
      C11    = AB1*AB1 - AB2*AB2 - AB3*AB3 + AB4*AB4
      C21    = (AB3*AB4 - AB1*AB2)*2.
      C31    = (AB1*AB3 + AB2*AB4)*2.
      C12    = (AB3*AB4 + AB1*AB2)*2.
      C22    = AB1*AB1 - AB2*AB2 + AB3*AB3 - AB4*AB4
      C32    = (AB2*AB3 - AB1*AB4)*2.
      C13    = (AB2*AB4 - AB1*AB3)*2.
      C23    = (AB2*AB3 + AB4*AB1)*2.
      C33    = AB1*AB1 + AB2*AB2 - AB3*AB3 - AB4*AB4
C*
      PSI    = ATAN2(C12,C11)
      PSIDEG   = PSI*57.2958
      PSIDEG   = PSI*57.2958
C*
      PHI    = ATAN2(C23,C33)
      PHIDEG   = PHI*57.2958
C*
      STHLIM =  XLIM (C13,-1.,1.)
      QTHETA   = -ASIN(STHLIM)
      QTHEDEG  =  QTHETA*57.2958
C*
C     BOTTOM OF QUATERNION OPTION
      IF (LOGIC(50)) THEN
      IF(LOGIC(50)) PRINT*,'AB1,2,3,4=',AB1,AB2,AB3,AB4
      PRINT*,'QUAT PSI,THETA,PHIDEG=',PSIDEG,THEDEG,PHIDEG
      ENDIF
      ENDIF
*
*+*     CALL IGRATE AFTER QUATERNION AND WRITE OVER THETA
*
      CALL IGRATE6
*
*     REPLACE THETA AND THEDEG WITH QUATERNION VALUES
      THETA = QTHETA
      THEDEG = QTHEDEG
*
*
** COMPUTE PILOT AND CG LOCATION IN RUNWAY FRAME & ILS ERRRORS
```

```
*
      CALL RUNWAY
*
      IF (NOPTION .NE. 5 .AND. NOPTION .NE. 6) THEN
       IF ( ISENSW(4) .EQ. 1  .OR.  ISENSW(5) .EQ. 1 ) CALL CGIINTF
      ENDIF
*
**
*
      CALL MISCEL
*
*
      IF (NOPTION .NE. 5 .AND. NOPTION .NE. 6) THEN
      IF (FSS(7)) CALL CDCAGT
      ENDIF
*
*
 8889 CONTINUE
*
**    OUTPUT DACS IN OPERATE
*
**    LITE(45) INDICATES AUTO-HOLD
*
   40 CONTINUE
*
**    SET THE ANALOGS FOR THE COCKPIT
*
      CALL DACS
*
**    STRIP CHART RECORDERS
*
      CALL BRUSH
*
**    PACK THE TCVO'S AND THE RED AND WHITE LIGHTS INTO ODIS
*
      CALL PKDISCR
*
      CALL PACKC
*
```

```
**    IF (FSS(20) IS ON, SET TURRET VALUES
*
      IF (FSS(20)) CALL TURVAL
*
      CALL SETUP0
*
**    BACKGROUND OPERATION DURING OPERATE
      IF (INTEG(89).EQ.3.AND.LOPER.AND.FSS(2)) CALL RTWRITE(1)
*
 9999 CALL BEGBACK(TIMEU,TIMER,*9006)
      CALL BKEXEC
      CALL ENDBACK
*
      GOTO 9006
*
**                  ************************
**                  ENTRY POINT FOR TERMINATE
**                  ************************
**    GO TO HOLD1
*
 9004 CALL RTART
**
*** DISABLE VOTAN PORT
**
      MIMOUT(1) = O" 12000 00000 00000 00000 "
**
      IF (ISENSW(1) .EQ. 1) CALL RTNETOT ( STATMO , IDUM )
*
**    ATERM WILL CALL FINI - DONE
*
      CALL ATERM(*9001,*9002,*9003,*9004,*9014,*9015)
*
**                  ****************************
**                  ENTRY POINT FOR PRINT OR READ
**                  ****************************
*
**    SINCE EITHER PRINT OR READ BUTTON HAS BEEN DEPRESSED,
**    RETURN SO CAN LOAD IN OTHER OVERLAY
*
```

```
 9014 CONTINUE
 9015 CONTINUE
**
*** DISABLE VOTAN PORT
**
      MIMOUT(1) = O" 12000 00000 00000 00000 "
**
      IF (ISENSW(1) .EQ. 1) CALL RTNETOT ( STATMO , IDUM )
**
      RETURN
      END
```

VMS Main Program (MAIN) - The entire compiled listing of the main routine is included. In order to conserve space the variable declarations and initializations have been deleted.

```
      OVERLAY (ABSFILE,4,0)
      PROGRAM MAIN
C*
C*    THIS IS THE REAL-TIME PORTION OF THE MULTI-BODY SIMULATION
C*
C ************************************
```

Variable definitions and declarations have been deleted to conserve space

The program modifications made to CGIINTF are the same as in the TSRV code

```
C ************************************
C*
C*    SET UP RTGO ADDRESS
C*
      CALL RESADD(1,*50)
      GO TO 60
   50 CONTINUE
   60 CALL RECADD(1,*100)
      GO TO 101
  100 CONTINUE
      IOLDMOD  = MODE
      MODE     = 1
  101 CONTINUE
C*
      VARCHNG  = .FALSE.
C*
C*    RESTORE NO-TABLES CHANGES
C*
      CALL NOTABLR
C*
C*    SET UP RETURN FOR LOST TIME SYNCH
C*
      CALL LOSTIME(1,*8000)
C*
      IF ( ISSWTCH(2) .EQ. 0 ) NOHUD = .TRUE.
      IF ( .NOT.NOHUD ) AGTINIT = .TRUE.
C*
```

```
C*   BEGIN REAL-TIME
C*
     CALL RTSRT ("MAIN")
C
C*   INITIALIZE THE HARDWARE INTERFACE
C
     CALL SETUPIC
     CALL PACKC
     CALL SETUP0
     CALL UNPACKC
C
C*
C ***************************************
C*          S T A R T   R E S E T   L O O P
C ***************************************
 1000 CONTINUE
C*
C*   CHECK FOR FIRST CYCLE IN RESET LOOP
C*
     IF ( MODE.EQ.1 ) GO TO 1500
C*
C*   THIS IS FIRST CYCLE IN RESET LOOP
C*
     IOLDMOD  = MODE
     MODE     = 1
C*
C*   RTWRITE(1) LAST TIME ON RUN
C*
     IF ( IOLDMOD.EQ.2.OR.IOLDMOD.EQ.3 ) GO TO 1100
     GO TO 1200
 1100 CALL RTWRITE(1)
     IRECCNT  = IRECCNT + 1
     TRECLST  = T
 1200 CONTINUE
C*
C*   SET RUN NUMBER INCREMENT FLAG
C*
     RUNINC   = .TRUE.
C*
```

```
C*    STORE INITIAL CONDITIONS INTO PROGRAM VARIABLES
C*
 1500 CONTINUE
C*
C*      CHECK FOR NEW PRINT LIST
C*
      IF ( IWRITE.EQ.IWRITEL ) GO TO 1300
      IWRITE   = MAX0(MIN0(IWRITE,4),1)
      CALL RTMRT ("MAIN")
      GO TO 5000
 1300 CONTINUE
C*
C*    DEFINE AIRCRAFT
C*
      IF ( .NOT.REINIT ) GO TO 1830
      CALL RTART("MAIN")
      REINIT   = .FALSE.
      CALL GEARIC
      CALL RTSRT ("MAIN")
 1830 CONTINUE
C*
      CPHOLD   = .FALSE.
      AUTOHLD  = .FALSE.
      CALL IC
C*
      GO TO 2500
C*
C ****************************************
C*          S T A R T   H O L D   L O O P
C ****************************************
C*
 2000 CONTINUE
C*
C*    CHECK FOR FIRST CYCLE OF HOLD LOOP
C*
      CALL UNPACKC
      CALL UNDISCR
      IF ( MCP(2) ) CPHOLD = AUTOHLD = .FALSE.
      IF ( MODE.EQ.2 ) GO TO 3500
```

```
C*
C*   THIS IS FIRST CYCLE OF HOLD LOOP
C*
     IOLDMOD  = MODE
     MODE     = 2
C*
C*   IF COMING INTO HOLD LOOP FROM RESET LOOP THEN ERASE REAL TIME
C*   RECORDING FILE
C*
     IF ( IOLDMOD.NE.1 ) GO TO 3500
C*
     CALL RTART("MAIN")
     CALL REWRTF(1)
     CALL RTSRT ("MAIN")
C*
C*   RESET THE AUTOHOLD INDICATOR
C*
     ITEST     = 0
     GO TO 3500
C*
C*   INITIALIZE INTEGRATION
C*
 2500 CALL INTIC
C*
C ****************************************
C*        S T A R T   O P E R A T E   L O O P
C ****************************************
 3000 CONTINUE
     CALL UNPACKC
     CALL UNDISCR
C*
C*   PICK UP AIRCRAFT CONTROL INPUTS
C*
C  OPTION 5 SKIPS THE MATHEMATICAL MODEL
     IF( NOPTION .EQ. 5 )  GOTO  3999

     CALL ADINPUT
     IF (MODE.EQ.1) THEN
       CALL DIRCOS
```

```
      CALL AUXEQ
      CALL FDIRECT
      CALL TRIM
    ENDIF
C*
C*   COMPUTE BODY FORCES AND MOMENTS
C*
    CALL ACEQM
C*
C*   COMPUTE BODY ACCELERATIONS
C*
    CALL ACCEL
C*
C*   COCKPIT DRIVE SIGNALS AND RECORDERS
C*
C*
C*   ROOT MEAN SQUARES
C*
    CALL SUMSQ
C*
 3500 CONTINUE
    GOPER   = MODE.EQ.3
    GHOLD   = MODE.EQ.2
    GREST   = MODE.EQ.1
    IF ( .NOT.FSS(1) ) CALL LNDGEAR(GREST,GHOLD,GOPER)
    IF ( MODE.NE.3 .AND. VARCHNG ) CALL TYPEVAR
C*
C*   DISPOSE VARIABLE CHANGES TO PRINTER WHEN FSS(15) DEPRESSED
C*
    IF ( .NOT.FSS(15) ) FSS15L = .TRUE.
    IF ( MODE.NE.1 .OR. .NOT.FSS(15) .OR. .NOT.FSS15L ) GO TO 3601
    FSS15L  = .FALSE.
    ENDFILE 33
    REWIND 33
    CALL RTMRT ("MAIN")
    CALL RTOUT (L"TAPE33","RB",*3600)
 3600 REWIND 33
    CALL RTSRT ("MAIN")
 3601 CONTINUE
```

```
C*
C*   PRINT THE "POST-RUN" DATA IF FSS(12) IS ON
C*
     IF ( FSS(12) ) CALL SUMPRNT
C*
C*
C*   HARDWARE DRIVE (CRD, MOTION BASE, CGI)
C*
     IF ( MODE.GT.1 ) FSS(5) = .FALSE.
C*
C
C     START WRITING TO DISK AT INTERVALS OF WRINTV SEC
C     AND THEN WRITE FOR IWRITER ITERATIONS
C
C     DEFAULT VALUES ARE SET TO WRITE EVERY FRAME
     DATA WRINTV,IWRITER,KWRITER / 0., 1, 0 /
C
C      DO NOT WRITE IF IN RESET
     IF (MODE.EQ.1) WRFLAG = .FALSE.
C
C     MINIMUM WRITE INTERVAL IS ONE ITERATION
     IF (WRINTV.LT.H) WRINTV = H
C
C     START WRITING WHEN T MOD WRINTV < TIMESTEP  OR 1ST ITERATION
     IF (MODE.EQ.3 .AND. (AMOD(T+.005,WRINTV).LT.H .OR. T.EQ.0.)) THEN
       WRFLAG = .TRUE.
       KWRITER = 0
     ENDIF
C
C     IF WRITING INCREMENT COUNTER & STOP IF FINISHED WRITING
     IF (WRFLAG) THEN
       KWRITER = KWRITER + 1
       IF (KWRITER.GT.IWRITER) WRFLAG = .FALSE.
     ENDIF
C
C NOPTION = 4  CGI & CRD STATES ARE CHANGED ACCORDING TO DELTA THETA
C            AND EXECUTE THE MATH MODEL
C INITIALIZE PREVTHE IN RESET
     IF (MODE .EQ. 1)    PREVTHE = THETAD
```

```
C DO NOT CHANGE CRDCLR UNTIL THETA(THETAD) HAS CHANGED MORE THAN
C TEST THETA, TSTHETA

C     IF( NOPTION .EQ. 4) THEN
C       DLTHETA = THETAD - PREVTHE
C       IF (DLTHETA .GE. TSTHETA)  CRDCLR = 1.0
C       IF (DLTHETA .LT. -TSTHETA)  CRDCLR = 0.0
C       PREVTHE = THETAD
C     ENDIF

C     PRINT*,'THETA: DEL PREV DEG TEST',DLTHETA,PREVTHE,THETAD,TSTHETA

C INCREMENT COUNTER IF THE CHANGE IN CADC(6), DCPILOT, HAS NOT
C CHANGED MORE THAN THE THRESHOLD LIMIT, LIMCOL
C     DCOLCNT - COUNTER
C     COLFLAG - DELTA: FLAG THAT INDICATES IF DCPILOT HAS CHANGED
C                      MORE THAN THE THRESHOLD

      DCOLCNT = DCOLCNT + COLFLAG
      IF ( CRDCLR .NE. PREVCLR)   COLFLAG = 0
      PREVCLR = CRDCLR

C     PRINT*,'CNT  FLAG  CRDCLR PREVCLR ',DCOLCNT,COLFLAG,CRDCLR,PREVCLR

      IF (MODE.NE.3) THEN
       IF (MODE.GT.1) FSS(5) = .FALSE.
       CRDCLR = 0.
       CALL HDWOUT
       IF (FSS(16) .AND. VBATCH.NE.0) CALL HUD
       IF (IROUT.EQ.1) THEN
        CALL CGIINTF
        CALL MBINTF
       ENDIF
      ENDIF
      IROUT   = 1
C*
C**************************************
C*     E N D   O F   R E S E T   A N D   H O L D   L O O P S
C**************************************
```

```
C*
      CALL PKDISCR
      CALL PACKC
C*
C*    IF LOGIC( 8) IS TRUE, SEND THE TURRET LABELS TO RAM (LEAVE THIS
C*    LOGIC ON FOR ABOUT 1 SECOND TO GIVE THE LIBRARY TIME TO SEND
C*    ALL THE LABELS)
C*
      IF ( LABELS ) THEN
       LABLCNT    = LABLCNT + 1
       CALL SETUPL
       IF ( LABLCNT .GT. 32 ) THEN
        LABELS     = .FALSE.
        LABLCNT    = 0
       ENDIF
      ENDIF
C*
      CALL SETUP
      CALL SETUP0
C*
C      IF BATCH   SET IPILOT = 99  = BATMAN OPTION
      IF (VBATCH.EQ.0) IPILOT = INTEG(1) = 99
      IF (IPILOT.EQ.99) CALL BATMAN
C      WAS -- IF (VBATCH.EQ.0) CALL RTBATCH
C*
      IF (MODE .NE. 1 ) GO TO 3999
      IF (.NOT. MCP(4)) GO TO 3999
      CALL RTCYCLE
 3999 CONTINUE
C*
      IF ( AUTOHLD.OR.CPHOLD ) GO TO 4100
      CALL RTMODE (*4000,*2000,*1000,*7000,*6000,*5000)
C*
C*    OPERATE LOOP ONLY
C*
C*
C*    SET AUTOMATIC HOLD CONDITIONS
C*
 4000 CONTINUE
```

```
      IF ( NOPTION .EQ. 5 ) GOTO 4300

C*
C*    WANT TO BE ABLE TO SKIP ALL AUTO HOLD TESTS EXCEPT ALTITUDE
C*    AND PILOT OVERRIDE
C*
      IF ( .NOT.SKPAHLD ) THEN
C*
C*    WANT THE LIMITS TO BE SMALLER CLOSER TO THE GROUND
C*
        ALTSCL   = 1. + (ALTLG / 3333.)
        IF ( ALTSCL .GT. 2.5 ) ALTSCL = 2.5
C*
        ANGDLMP  = ANGDLIM * ALTSCL
        ACCELMP  = ACCELIM * ALTSCL
C*
        IF ( ABS(PDOT) .GT. ANGDLMP ) ITEST = ITEST + 1
        IF ( ABS(QDOT) .GT. ANGDLMP ) ITEST = ITEST + 2
        IF ( ABS(RDOT) .GT. ANGDLMP ) ITEST = ITEST + 4
        IF ( ABS(UDOT) .GT. ACCELMP ) ITEST = ITEST + 10
        IF ( ABS(VDOT) .GT. ACCELMP ) ITEST = ITEST + 20
C       IF ( ABS(WDOT) .GT. ACCELMP ) ITEST = ITEST + 40
C*
        IF ( (ABS(ALPDEG) .GE. ALPHOLD) .AND. (ALT .GT. 50.) )
     *       ITEST = ITEST + 600
C*
        IF ( ALTLG .LE. ALTLIM ) THEN
C*
          IF ( ABS(PHI) .GT. PHILIMT) ITEST = ITEST + 100
C*
          PHIPROJ  = (PHI + H*(PHI - PHIDP))*DEGRAD
          IF ( ABS(PHIPROJ) .GT. PHIPLIM ) ITEST = ITEST + 5000
          PHIDP    = PHI
C*
          PPRJ     = (P + .5*H*(3.*PDOT - PDOTP))*DEGRAD
          IF ( ABS(PPRJ) .GT. PPLIM ) ITEST = ITEST + 7000
          PDOTP    = PDOT
C*
          HDPROJ   = HDOT + .5*H*(3.*WDOT - WDOTP)
```

```
        IF ( HDPROJ .LT. HDOTLIM ) ITEST = ITEST + 200
        WDOTP    = WDOT
C*
      ENDIF
C*
    ENDIF
C*
    ALTPROJ    = ALTLG + .5*H*(3.*HDOT - HDPAST)
    IF ( ALTPROJ .LT. ALTMIN ) ITEST = ITEST + 1000
    HDPAST     = HDOT
C*
C*   MAXIMUM TIME
C*
    IF ( T .GT. TIMEMAX ) ITEST = 3000
C*
    IF ( ITEST .NE. 0 )  AUTOHLD = .TRUE.
    CPHOLD   = LDISI(1)

    IF ( .NOT.(AUTOHLD.OR.CPHOLD) ) GO TO 4200
 4100 MCP   (1) = LMCP   (1) = .FALSE.
    MCP   (2) = LMCP   (2) = .TRUE.
    CALL RTCYCLE
    GO TO 2000
C*
C*   INCREMENT RUN NUMBER AND INITIALIZE RECORDING FREQUENCY AND
OUTPUT
C*   COUNTER
C*
 4200 IF ( .NOT.RUNINC ) GO TO 4300
    IRUN     = IRUN + 1
    NTCNT    = NT - 1
    IRECCNT  = 0
    RECFLAG  = .FALSE.
    RUNINC   = .FALSE.
C*
C*   IF THIS IS FIRST CYCLE IN OPERATE - UPDATE MODE
C*
 4300 IF ( MODE.EQ.3 ) GO TO 4400
    IOLDMOD  = MODE
```

```
      MODE     = 3
      YDEVMAX  = 0.
      TDDSPLY    = .FALSE.
C*
C*    IF ENTERING OPERATE FROM RESET, THEN ERASE REAL-TIME FILE
C*
      IF ( IOLDMOD.NE.1 ) GO TO 4400
      CALL RTART ("MAIN")
      CALL REWRTF(1)
      CALL RTSRT ("MAIN")
 4400 CONTINUE
      IF ( NOPTION .EQ. 5 ) GOTO 9000

C*
C*    RECORD ON REAL-TIME FILE
C*
      RECFLAG  = T.GE.PRNTALL(1) .AND. T.LE.PRNTALL(2)
      IF ( RECFLAG ) GO TO 4500
      GO TO 4600
 4500 NTCNT    = NTCNT + 1
      IF ( NTCNT.NE.NT ) GO TO 4600
      NTCNT    = 0
      IRECCNT  = IRECCNT + 1
      IF ( IWRITE .NE. 5 ) CALL RTWRITE(1)


      RECFLAG  = .FALSE.
 4600 CONTINUE
C*
C*    STORE MAXIMUM Y-DEVIATION
C*
      IF ( ABS(SY).GT.ABS(YDEVMAX) ) YDEVMAX = SY
C*
C*    UPDATE THE DISPLAY OF VARIOUS VARIABLES UNTIL TOUCHDOWN
C*
      IF ( .NOT.TDDSPLY .AND. (GRND .EQ. 1.) ) TDDSPLY = .TRUE.
C*
      IF ( .NOT.TDDSPLY ) THEN
        TIMETD     = T
```

```
      SXTD      = SX
      SYTD      = SY
      HDOTTD    = HDOTLG
      PSITD     = PSID
      VIASTD    = IAS
    ENDIF
C*
    IF ( T .EQ. 0. ) THEN
      TAUTOFF   = 0.
      AUTOPST   = AUTO
    ENDIF
C*
    IF ( .NOT.AUTO .AND. AUTOPST ) TAUTOFF = T
    AUTOPST   = AUTO
C*
C*   OVERIDE CLOCK FROM RTMODE
C*
    ITRUVAL  = 0
    ICOUNT  = ICOUNT  + 1
    IF ( ICOUNT  .EQ. IFIX(1./H) ) THEN
      ITRUVAL = -1
      ICOUNT  =  0
    ENDIF
    CODIS(1) = (CODIS(1).AND.FMASK( 2)) .OR. (TMASK( 2).AND.ITRUVAL)
    CODIS(1) = (CODIS(1).AND.FMASK(12)) .OR. (TMASK(12).AND.ITRUVAL)
    CODIS(1) = (CODIS(1).AND.FMASK(22)) .OR. (TMASK(22).AND.ITRUVAL)
    CODIS(2) = (CODIS(2).AND.FMASK( 8)) .OR. (TMASK( 8).AND.ITRUVAL)
C*
C*   INTEGRATE
C*
    CALL QUAT
    CALL LVELOC
    CALL DIRCOS
C
      IF( NOPTION .EQ. 4) THEN
        DLTHETA = THETAD - PREVTHE
        IF (DLTHETA .GE. TSTHETA)  CRDCLR = 1.0
        IF (DLTHETA .LT. -TSTHETA)  CRDCLR = 0.0
        PREVTHE = THETAD
```

```
      ENDIF

    CALL CGIINTF
    CALL MBINTF

    IF (MODE.EQ.3 .AND. WRFLAG)   CALL RTWRITE(1)
C
    CALL DERIV
    CALL IGRATE6
    CALL INTGRAL
    IF (FSS(16) .AND. VBATCH.NE.0) CALL HUD
C
    CALL HDWOUT
C
    CALL AUXEQ
C
    CALL FDIRECT
C
    IF (MODE.GT.1) FSS(1) = .FALSE.
C
C*
C*   CONTINUE IN REAL TIME OPERATE
C ENTRY POINT FOR OPTION 5 TO SKIP MATHEMATICAL MODEL
C AND CHANGE CRDCLR ACCORDING TO THE CHANGE IN ADC(1) = DCPILOT

9000  IF (NOPTION .EQ. 5 ) THEN
      IF (MODE .EQ. 3 ) THEN
        IOLDMOD = MODE
        MODE = 3
        YDEVMAX  = 0.
        TDDSPLY    = .FALSE.
      ENDIF
      CALL ADINPUT
      CALL HDWOUT
      IF  (ADC(1) .GT. COLBP)   THEN
        CRDCLR = 1.0
      ELSE
        CRDCLR = 0.0
      ENDIF
```

```
      CALL CGIINTF
      IF ( .NOT. FSS(3))   CALL HUD
      CALL MBINTF
    IF (IWRITE.EQ.5 .AND. MODE.EQ.3 .AND. FSS(2) .AND. WRFLAG)
   .  CALL RTWRITE(1)

    ENDIF


C*
    CALL RTCYCLE
    GO TO 3000
C*
C*            R E A D
C*
 5000 CONTINUE
    IOLDMOD  = MODE
    MODE     = 5
    IPL      = 3
    RETURN
C*
C*            P R I N T
C*
 6000 CONTINUE
    IOLDMOD  = MODE
    MODE     = 6
    IPL      = 2
    RETURN
C*
C*         T E R M I N A T E
C*
 7000 CALL RTMRT ("MAIN")
    ENDFILE 33
    REWIND  33
    IF ( VBATCH .EQ. 0 ) GO TO 7100
    CALL RTOUT(L"TAPE33","RB",*7100)
 7100 CALL RTART ("MAIN")
    CALL ATERM
    IPL      = 99
```

```
      RETURN
C*
C*    L O S T   T I M E   S Y N C H
C*
 8000 CALL RTART ("MAIN")
      CALL TTPRINT (MESSAGE)
      CALL RTSRT ("MAIN")
 8100 CALL RTCYCLE
      CALL UNPACKC
      MOPER  = MCP   (1)
      IF ( MOPER ) GO TO 8100
      CALL RTMODE (*4000,*2000,*1000,*7000,*6000,*5000)
C*
      END
```

VMS CGI Driver (CGIINTF) - The subroutine CGIINTF drives the CGI and contains the decision logic which determines, depending upon the option, what position to send to the CGI. The code listed here is from the VMS however there is little difference between that and the VMS code. Again the variable definitions and declarations have been deleted.

```
      SUBROUTINE CGIINTF
C
C**** MAKE THE CALLS NECESSARY TO DRIVE THE CGI:
C
C       CALL CGIINIT(IRUNWAY,PSI_RNWAY,LAT_RNWAY,LON_RNWAY
C                    LCKSUM,DEBUG FLAG VALUES)
C       CALL CGICHAN(IGN,CHN,HFOV,VFOV,  H ,  P ,  R )
C       CALL CGIEYEO(CHGRP,XOFF,YOFF,ZOFF)
C       CALL CGILUM(CHGRP,SUN INTEN,FOG INTEN,POLY INTEN)
C       CALL CGIVIS(CHGRP,LITE MASK,POLY MASK,VISIB DIST)
C       CALL CGISUN(IGN,SUN PSI,SUN THE)
C       CALL CGIDCS(IGN,DCS,SPC,SEQ,MOD, X  , Y  , Z  , H  , P  , R  )
C       CALL CGIFOG(IGN,TYPE,FOG VIS,FOG BOT,FOG TOP)
C       CALL CGICLD(IGN,TYPE,CLD VIS,CLD BOT,CLD TOP)
C       CALL CGIFLSH(IGN,ON OFF)
C       CALL CGILOBE(CHGRP,LIT SEL,INTENSITY)
C       CALL CGISLGC(CHGRP,LIT GRP,ON OFF,INTEN)
C       CALL CGIMLGC(CHGRP,LIT NUM,LIT DATA)
C       CALL CGIMSEQ(IGN,SEQ,SEQ DATA)
C       CALL CGIMSTP(IG NUM, SEQ ID)
C       CALL CGITXMO(IGN,MOTION VECT,X VEL,Y VEL,X OFF,Y OFF)
C       CALL CGICOLD(IGN,ON OFF,SEQ NUM)
C       CALL CGIHAT(IGN,PCSB,ON OFF,DCS NUM)
C       CALL CGIRTN (CGI INPUT BUFFER, INPUT STATUS, DEMAND COUNTER)
C       CALL CGIHATR(IGN, DCS NUM, HEIGHT-ABOVE-TERRAIN)
C       CALL CGIXFR(CGI OUTPUT BUFFER,OUTPUT STATUS)
C       CALL CGIERRP(WRITE TO TAPE FLAG)

C
C*    DETERMINE WHICH IG IS GOING TO WHICH SIDE OF THE COCKPIT
C
      MAP    (1) = MAPCGI / 10
      MAP    (2) = MAPCGI -10*MAP(1)
C
C*    SET THE IG NUMBER BASED ON THE IG SCHEDULED:
C*      MAP(1) = 1 --> IG 0 SELECTED FOR LEFT SIDE
C*      MAP(1) = 2 --> IG 0 SELECTED FOR RIGHT SIDE
C*      MAP(2) = 1 --> IG 1 SELECTED FOR LEFT SIDE
C*      MAP(2) = 2 --> IG 2 SELECTED FOR RIGHT SIDE
```

```
C
      IGNUM      = -1
      IF ( MAP(1) .GT. 0 ) IGNUM = IGNUM + 1
      IF ( MAP(2) .GE. 1 ) IGNUM = IGNUM + 2
C
C*    CALL THE CGI ROUTINES
C
      IF ( CGIINT  ) THEN
        CALL CGIINIT(IRNWAY,PSIRWY,LATR,LONR,LCHKSUM,LDEBUG)
        CGIINT  = .FALSE.
        EYEPNT  = .TRUE.
        LUMINT  = .TRUE.
        VISINT  = .TRUE.
        SUNINIT = .TRUE.
      END IF
C
      IF ( (IGNUM .EQ. 0) .OR. (IGNUM .EQ. 1) ) THEN
C
        J         = IGNUM + 1
C
        IF ( EYEPNT ) THEN
          CALL CGIEYEO(CHNGRPN(J), XBAR   , YBAR   , ZBAR   )
          EYEPNT  = .FALSE.
        END IF
C
        IF ( LUMINT ) THEN
          CALL CGILUM (CHNGRPN(J), SUNINT , FOGINT , POLYINT)
          LUMINT  = .FALSE.
        END IF
C
        IF ( VISINT ) THEN
          CALL CGIVIS (CHNGRPN(J), LITEMSK, POLYMSK, VISDIST)
          VISINT  = .FALSE.
        END IF
C
      ELSE IF ( IGNUM .EQ. 2 ) THEN
C
        DO 10 J = 1, 2
C
```

```
      IF ( EYEPNT ) THEN
        CALL CGIEYEO(CHNGRPN(J), XBAR   , YBAR   , ZBAR   )
      END IF
C
      IF ( LUMINT ) THEN
        CALL CGILUM (CHNGRPN(J), SUNINT , FOGINT , POLYINT)
      END IF
C
      IF ( VISINT ) THEN
        CALL CGIVIS (CHNGRPN(J), LITEMSK, POLYMSK, VISDIST)
      END IF
C
      IF ( J .EQ. 2 ) THEN
        EYEPNT  = .FALSE.
        LUMINT  = .FALSE.
        VISINT  = .FALSE.
      ENDIF
C
   10 CONTINUE
C
    ENDIF
C
    IF ( SUNINIT ) THEN
      CALL CGISUN (IGNUM  , SUNPSI , SUNTHE )
      SUNINIT       = .FALSE.
    END IF
C
    IF ( FSS( 8) ) THEN
C
      SXCGI     =  SX*COSD(PSIR) - SY*SIND(PSIR) + SXRWY
      SYCGI     =  SX*SIND(PSIR) + SY*COSD(PSIR) + SYRWY
      ALTCGI    =  ALT + ALTRWY
      PSICGI    =  PSID + PSIR
      PSICGI    =  MOD(PSICGI+720., 360.)
C
C SEND SPECIFIED COORDINATES TO CGI ACCORDING TO THE VALUE OF
CRDCLR.
C USED DATA STATEMENTS FOR COORDINATES SO THAT THEY CAN BE CHANGED
C DURING REAL TIME.
```

```
      DATA  SXCGI1 , SYCGI1 , ALTCGI1  /47816.43, 78545.074, 5692.25 /
      DATA  SXCGI2 , SYCGI2 , ALTCGI2  /47816.43, 78545.074, 5695.25 /

      IF ( (NOPTION .EQ. 4) .OR. (NOPTION .EQ. 5) )  THEN
        IF ( CRDCLR .EQ. 0.0) THEN

C WHITE
          SXCGI = SXCGI1
          SYCGI = SYCGI1
          ALTCGI = ALTCGI1

        ELSE
C BLACK

          SXCGI = SXCGI2
          SYCGI = SYCGI2
          ALTCGI = ALTCGI2

        ENDIF

      ENDIF

      IF ( (NOPTION .EQ. 4) .OR. (NOPTION .EQ. 5))  THEN
        THECGI =  0.0
        PSICGI = -90.0
        PHID   =  0.0
      ELSE
        THECGI = THETAD
      ENDIF

       CALL CGIDCS (IGNUM  , 0     , 0     , 0     , 0     ,
     *             SXCGI  , SYCGI  , ALTCGI , PSICGI , THECGI ,
     *             PHID   )
C
       IF ( FOG ) THEN
         CALL CGIFOG (IGNUM  , FOGTYPE, FOGVIS , FOGBOT , FOGTOP )
         FOG          = .FALSE.
       END IF
```

```
C
      IF ( CLOUDS ) THEN
       CALL CGICLD (IGNUM  , CLDTYPE, CLDVIS , CLDBOT , CLDTOP )
       CLOUDS       = .FALSE.
      END IF
C
      IF ( LGTNGON ) THEN
       CALL CGIFLSH(IGNUM  , LGHTING)
       LGTNGON      = .FALSE.
      END IF
C
      IF ( (IGNUM .EQ. 0) .OR. (IGNUM .EQ. 1) ) THEN
C
       J        = IGNUM + 1
C
       IF ( LOBE ) THEN
        CALL CGILOBE(CHNGRPN(J), LLSEL  , LLINT  )
        LOBE   = .FALSE.
       END IF
C
       IF ( SLGON   ) THEN
        CALL CGISLGC(CHNGRPN(J), SLGNUM , SLGSEL , SLGINT )
        SLGON  = .FALSE.
       END IF
C
       IF ( MLGON   ) THEN
        CALL CGIMLGC(CHNGRPN(J), MLGNUM , MLGARRY)
        MLGON  = .FALSE.
       END IF
C
      ELSE IF ( IGNUM .EQ. 2 ) THEN
C
       DO 20 J = 1, 2
C
        IF ( LOBE ) THEN
         CALL CGILOBE(CHNGRPN(J), LLSEL  , LLINT  )
        END IF
C
        IF ( SLGON   ) THEN
```

```
          CALL CGISLGC(CHNGRPN(J), SLGNUM , SLGSEL , SLGINT )
        END IF
C
        IF ( MLGON  ) THEN
          CALL CGIMLGC(CHNGRPN(J), MLGNUM , MLGARRY)
        END IF
C
        IF ( J .EQ. 2 ) THEN
          LOBE   = .FALSE.
          SLGON  = .FALSE.
          MLGON  = .FALSE.
        ENDIF
C
  20    CONTINUE
C
      ENDIF
C
      IF ( MSEQON ) THEN
        CALL CGIMSEQ(IGNUM, SEQID(1), MSEQDAT)
        MSEQON  = .FALSE.
      END IF
C
      IF ( MSEQOFF ) THEN
        CALL CGIMSTP(IGNUM, SEQID(1))
        MSEQOFF = .FALSE.
      END IF
C
      IF ( TEXMOTN ) THEN
        CALL CGITXMO(IGNUM  , MVECNUM, XTEXVEL, YTEXVEL, XTEXOFF,
     *               YTEXOFF)
        TEXMOTN      = .FALSE.
      END IF
C
      IF ( COLDON  ) THEN
        CALL CGICOLD(IGNUM  , COLDECT, CDSEQNO)
        COLDON   = .FALSE.
      END IF
C
      IF ( HATON   ) THEN
```

```
        CALL CGIHAT(IGNUM  , PDCSON , HATSEL , HATDCSN)
        HATON  = .FALSE.
      END IF
C
      IF ( HAT ) THEN
        CALL CGIRTN (RTBUFC(CGIINDX), STCGII , IDEMAND)
        CALL CGIHATR(IGNUM  , HATDCSN, ZHAT   )
      END IF
C
    IF ( (ISSWTCH(3) .EQ. 1) .OR. (ISSWTCH(4) .EQ. 1) ) THEN
        CALL CGIXFR(RTBUFC(CGIONDX),STCGIO )
      END IF
C
    END IF
C
    IF ( ERRWFLG ) THEN
C
      CALL RTART
C
      CALL CGISTAT
C
      CALL CGIERRP(12,ERRWFLG)
C
      ERRWFLG = .FALSE.
C
      CALL RTSRT
C
    END IF
C
    RETURN
    END
```

APPENDIX C

CRDS Display Code

```
tds1::
 if white_flag_dyn
   pmode 8
   pcolor white
   fill_density 6.0
   float_add  x_tran    x_tran  10.0
   push
     tran     x_tran   y_tran   0.0
     polys      0.0    0.0
     poly       0.0    5.0
     poly       5.0    5.0
     polyf      5.0    0.0
   pop
 endif
end
```

APPENDIX D

Measurement Technique Data

TABLE D-1 - Measurement Technique Data

Run	Time to PED (ms)	Time to VLD (ms)
1	116.5	110.5
2	118.5	111.0
3	119.5	112.0
4	129.5	122.5
5	121.0	113.5
6	111.0	104.0
7	126.0	118.5
8	134.0	127.0
9	121.0	114.5
10	109.0	102.0
11	116.0	109.0
12	139.5	132.0
13	139.0	131.0
14	127.0	119.5
15	127.5	120.5
16	109.0	102.0
17	114.0	107.0
18	115.5	108.0
19	112.5	105.5
20	134.0	127.0

APPENDIX E

Frequency Domain Data

TABLE E-1 - Frequency Domain Data

Run	$\Delta\phi°$ for Selected Frequencies				
	0.2 Hz	0.4 Hz	0.8 Hz	1.6 Hz	3.0 Hz
1	8.1	16.9	34.5	69.8	124.3
2	7.5	14.3	33.8	69.8	130.8
3	7.3	15.9	34.8	68.8	127.3
4	8.6	15.8	35.6	68.6	121.6
5	8.3	15.0	33.7	67.9	132.7
6	7.6	16.3	33.2	67.3	124.9
7	8.1	16.2	32.6	66.1	129.5
8	8.6	17.0	34.7	64.7	127.2
9	8.2	16.3	35.7	64.5	126.1
10	7.7	16.6	34.7	66.2	125.9
11	8.9	17.1	34.4	67.3	130.3
12	8.5	16.6	32.2	67.2	125.7
13	8.3	18.1	33.4	67.0	126.8
14	7.8	17.0	34.1	67.4	134.2
15	8.8	17.2	35.2	69.7	119.7
16	8.2	16.3	35.9	70.6	126.4
17	8.4	17.8	33.7	69.2	129.7
18	8.8	17.6	33.1	68.0	121.8
19	8.5	18.6	31.4	67.5	127.4
20	8.2	18.4	33.7	67.3	127.2

APPENDIX F

Control Loader Data

TABLE F-1 - Control Loader Data

Frequency (Hz)	$\Delta\phi°$ to Position Output	Delay (ms)
0.250	3.60	40.00
0.400	7.60	52.78
0.600	12.65	58.56
0.800	17.25	59.90
1.000	23.15	64.31
1.200	28.00	64.81
1.500	34.75	64.35
1.755	40.70	64.42
2.005	46.15	63.94
2.500	56.40	62.67
3.000	65.50	60.65

APPENDIX G

ARTS/CYBER/CGI/Motion Base Data

TABLE G-1 - TSRV Data

Run	TSRV - no math model (ms)		TSRV - 737 math model (ms)	
	CGIIN	CGIOUT	CGIIN	CGIOUT
1	11.25	96.2	96.2	97.9
2	16.85	93.3	69.8	90.6
3	11.55	96.2	78.4	93.6
4	25.10	101.5	76.2	101.0
5	28.65	87.5	72.6	92.0
6	16.75	84.4	80.8	91.3
7	32.30	87.0	92.0	86.8
8	22.65	83.1	71.2	94.2
9	31.05	96.8	76.6	88.6
10	13.90	95.6	87.4	91.0
11	13.50	88.4	82.4	89.6
12	10.95	88.3	80.6	98.4
13	33.85	95.7	95.4	104.4
14	13.70	86.0	74.6	99.6
15	17.60	89.2	93.0	96.2
16	19.75	97.1	89.6	95.1
17	5.20	91.3	84.8	86.8
18	5.45	87.2	92.0	96.3
19	6.65	97.5	97.2	87.5
20	19.20	92.0	75.0	99.6

TABLE G-1 - TSRV Data (continued)

Run	TSRV - no math model (ms)		TSRV - 737 math model (ms)	
	CGIIN	CGIOUT	CGIIN	CGIOUT
21	5.25	85.0	87.2	94.4
22	32.00	96.4	75.0	86.1
23	23.10	96.9	97.8	101.5
24	15.80	104.1	78.8	96.5
25	14.35	89.4	72.2	89.8
26	20.80	99.0	88.8	90.1
27	21.15	90.1	85.0	86.6
28	27.20	90.9	71.0	85.2
29	29.85	92.4	79.4	88.1
30	22.75	90.1	72.8	93.5
31	12.95	101.1	78.2	83.7
32	26.65	101.1	87.4	98.8
33	6.90	86.3	97.4	89.3
34	22.00	97.8	95.0	93.8
35	21.70	97.3	80.6	89.0
36	27.80	103.7	86.6	88.8
37	8.85	94.6	73.6	87.7
38	34.00	87.6	85.6	93.9
39	34.55	89.5	71.2	96.9
40	26.90	97.4	76.2	83.2

TABLE G-2 - VMS Data

Run	VMS - no math model (ms)			VMS - NASP math model (ms)		
	ACCEL	CGIIN	CGIOUT	ACCEL	CGIIN	CGIOUT
1	56.8	8.60	95.2	97.6	65.4	161.6
2	52.2	3.70	106.8	84.2	56.0	143.6
3	65.2	16.45	118.4	97.2	50.5	134.6
4	60.2	13.95	106.2	90.4	61.0	158.4
5	62.0	12.65	104.0	82.0	55.9	157.6
6	78.2	29.25	122.0	81.2	53.3	138.2
7	76.8	29.20	125.2	67.6	43.4	143.0
8	72.0	22.60	116.0	66.4	41.3	145.4
9	67.6	18.20	107.4	71.2	50.1	139.4
10	58.0	10.55	111.2	87.0	60.5	148.6
11	60.6	11.50	97.8	83.4	60.9	147.0
12	69.4	20.50	108.8	70.8	47.4	133.2
13	79.2	30.95	123.0	94.8	66.8	153.6
14	67.2	21.10	111.6	118.6	66.8	156.6
15	71.0	23.35	113.6	88.6	62.5	149.6
16	52.8	4.30	100.6	95.2	69.8	161.6
17	82.2	33.75	125.6	94.8	67.1	156.4
18	61.8	14.10	106.4	58.4	43.2	129.6
19	67.2	18.15	116.6	67.0	45.8	149.0
20	64.4	16.10	101.4	89.4	61.7	150.2

TABLE G-2 - VMS Data (continued)

Run	VMS - no math model (ms)			VMS - NASP math model (ms)		
	ACCEL	CGIIN	CGIOUT	ACCEL	CGIIN	CGIOUT
21	52.4	17.90	110.6	103.0	73.3	160.6
22	79.8	32.40	120.6	113.8	61.8	160.6
23	69.0	20.15	120.8	85.4	54.8	148.2
24	78.4	29.15	116.6	93.2	67.4	170.8
25	79.2	34.25	136.8	93.4	59.8	151.4
26	65.0	14.90	109.2	93.0	69.9	160.8
27	61.0	13.65	113.6	71.0	42.8	130.0
28	65.8	16.20	115.6	89.0	66.3	154.2
29	63.0	12.75	112.0	85.8	62.8	149.4
30	68.4	18.65	105.6	95.4	71.9	168.2
31	55.2	4.70	92.2	94.2	69.0	165.8
32	65.2	16.55	107.2	90.8	64.6	152.2
33	64.6	12.80	115.8	89.4	58.6	153.6
34	68.8	15.95	118.2	67.6	41.5	135.6
35	69.0	17.75	120.0	75.4	48.3	140.6
36	70.4	18.60	110.8	95.6	65.2	152.6
37	65.2	18.20	116.4	69.0	43.1	135.7
38	69.4	19.10	122.4	90.0	61.6	157.4
39	66.6	16.45	116.4	71.6	45.8	148.2
40	79.0	30.00	126.4	90.8	62.4	151.4

TABLE G-3 - TSRV Reordered Program Data

Run	TSRV - Reordered Program Flow (ms)	
	CGIIN	CGIOUT
1	33.0	122.2
2	24.2	125.0
3	33.8	137.8
4	29.7	115.0
5	25.4	113.4
6	11.0	101.2
7	19.1	106.2
8	27.9	118.2
9	32.2	124.8
10	21.2	124.4
11	34.6	120.2
12	13.0	103.0
13	37.2	133.2
14	18.0	123.2
15	13.5	112.4
16	37.1	103.0
17	25.9	98.6
18	15.0	99.8
19	24.6	120.4
20	32.5	137.2

TABLE G-3 - TSRV Reordered Program Data (continued)

Run	TSRV - Reordered Program Flow (ms)	
	CGIIN	CGIOUT
21	25.5	122.6
22	27.9	111.6
23	29.1	131.4
24	28.2	124.2
25	17.8	122.6
26	26.2	112.4
27	20.3	121.0
28	13.1	107.2
29	22.1	108.6
30	17.6	120.6
31	36.7	103.4
32	29.1	116.0
33	24.5	117.2
34	33.2	134.2
35	22.7	124.4
36	28.3	116.2
37	25.5	128.2
38	17.2	103.2
39	34.4	106.6
40	20.2	116.0

TABLE G-4 - VMS Reordered Program Data

Run	VMS - Reordered Program Flow (ms)		
	ACCEL	CGIIN	CGIOUT
1	108.2	26.0	113.2
2	115.6	37.0	141.8
3	131.8	41.2	134.8
4	112.8	21.4	115.2
5	123.8	34.8	124.8
6	113.0	21.4	119.2
7	109.0	17.0	114.0
8	103.4	11.0	102.4
9	113.2	25.2	123.8
10	102.2	19.0	120.2
11	107.0	25.6	128.8
12	126.8	45.8	147.2
13	93.6	11.8	97.8
14	109.8	21.2	116.6
15	107.4	20.6	111.6
16	104.8	27.6	126.2
17	104.4	18.4	109.6
18	122.0	42.0	146.8
19	105.4	17.4	104.6
20	93.0	15.2	117.6

TABLE G-4 - VMS Reordered Program Data (continued)

Run	VMS - Reordered Program Flow (ms)		
	ACCEL	CGIIN	CGIOUT
21	110.0	27.2	131.8
22	110.0	26.0	125.8
23	112.8	24.6	128.2
24	107.8	25.0	115.6
25	104.2	20.8	106.8
26	111.0	30.6	120.8
27	127.2	46.2	151.2
28	124.4	35.6	130.4
29	107.2	21.4	113.0
30	102.2	15.0	116.6
31	103.8	23.2	124.6
32	101.6	10.6	99.2
33	103.2	18.8	116.0
34	101.8	16.2	116.6
35	127.4	39.4	138.6
36	120.4	30.6	128.0
37	115.2	26.4	123.0
38	105.6	18.2	103.8
39	110.2	24.0	116.6
40	122.6	38.8	140.0

APPENDIX H

CRDS Data

TABLE H-1 - CRDS Data

Run	Calligraphic (ms)	Raster (ms)
1	139.5	111.4
2	102.5	91.2
3	117.5	44.8
4	110.5	92.6
5	106.0	78.8
6	114.0	51.0
7	107.0	66.8
8	113.0	59.8
9	116.5	81.8
10	106.0	102.8
11	113.5	54.0
12	120.0	86.4
13	106.6	88.6
14	130.0	43.8
15	97.5	90.2
16	114.0	43.2
17	110.5	55.8
18	113.5	67.4
19	121.5	78.8
20	111.0	109.4

TABLE H-1 - CRDS Data (continued)

Run	Calligraphic (ms)
21	118.8
22	119.4
23	111.9
24	111.6
25	116.3
26	115.7
27	118.6
28	109.9
29	106.4
30	106.1
31	105.9
32	105.6
33	116.9
34	106.5
35	116.3
36	115.8
37	107.1
38	76.8
39	106.2
40	105.9

APPENDIX I

Motion Base Frequency Domain Data

TABLE I-1 - Motion Base Data (uncompensated)

Frequency (Rad)	Amplitude Ratio	Phase (Deg)
0.09817	1.00749	0.43702
0.19635	1.00920	1.50961
0.29452	1.00536	1.09340
0.39270	1.00587	2.31840
0.49087	1.02349	3.48737
0.68722	1.00320	5.13656
0.98175	1.00118	6.68907
1.47262	1.00566	11.08379
1.96350	1.00708	13.40763
2.45437	0.99128	17.56512
3.04342	0.98547	21.99145
4.02517	0.96590	27.43413
5.00691	0.93754	36.57910
6.47953	0.98983	40.80396
9.03208	0.86000	60.86443
12.56637	0.83100	81.80953

TABLE I-2 - Motion Base Data (compensated)

Frequency (Rad)	Amplitude Ratio	Phase (Deg)
0.09817	1.00867	-0.28885
0.19635	1.00761	-0.06500
0.29452	1.00593	-1.13693
0.39270	1.00652	-0.59818
0.49087	1.02316	-0.28245
0.68722	1.00482	-0.20792
0.98175	1.00174	-0.86391
1.47262	1.01141	-0.26550
1.96350	1.01509	-1.58288
2.45437	1.00452	-1.34504
3.04342	1.00658	-1.49032
4.02517	1.00641	-3.77018
5.00691	1.00086	-2.55173
6.47953	1.10287	-9.35072
9.03208	1.09851	-9.50640
12.56637	1.38868	-11.40889

APPENDIX J

Motion Base Ladder Code

VMS Motion Base Driver - The subroutine MBINTF drives the motion base and contains the ladder code that can delay the output of motion base data to the platform by one frame. The variable definitions and declarations have been deleted.

```
      SUBROUTINE MBINTF
C*
C*    SIX DEGREES OF FREEDOM MOTION BASE INTERFACE
C*
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C*

*** Variable declarations have been deleted here  ***

C CALL REAL TIME WRITE AFTER CGI IS CALLED IF FSS(4) IS ON
C IF FSS(6) IS ON THE DO NOT CALL CGI HERE CALL BELOW CALL TO LVELOC
C CGIFLAG IS SET TO INDICATE WHICH SET OR EULER ANGLES IS SENT TO THE
C CGI  CGIDCS FUNCTION
C*
C
C      INITIALIZE MOTION BASE DELAY BUFFERS IN RESET
C
      IF (MODE.EQ.1) THEN
        OLDP = P
        OLDQ = Q
        OLDR = R
        OLDPDOT = PDOT
        OLDQDOT = QDOT
        OLDRDOT = RDOT
        OLDNX = NX
        OLDNY = NY
        OLDNZ = NZ
        OLDGRND = GRND
      ENDIF
C
      TABW(1)  = XBAR*.3048
      TABW(2)  = (YBAR+2.25)*.3048
      TABW(3)  = (ZBAR+5.708333)*.3048

C OPTION 5 INPUT TO MOTION BASE

      IF (NOPTION .EQ. 5) THEN
        IF ( CRDCLR .EQ. 0.0 )  THEN
          DELTAQ = DELTAQ1
```

```
        ELSE
          DELTAQ = DELTAQ2
        ENDIF
      ELSE
        DELTAQ = OLDQ
      ENDIF

      TABW( 5) = DELTAQ
C
      TABW( 4) = OLDP
      TABW( 6) = OLDR
      TABW( 7) = OLDPDOT
      TABW( 8) = OLDQDOT
      TABW( 9) = OLDRDOT
      TABW(10) = OLDNX * GMTR
      TABW(11) = OLDNY * GMTR
      TABW(12) = -OLDNZ * GMTR
      TABW(180) = OLDGRND
C
      IF (LOGIC(50)) PRINT*,'NZ,OLDNZ=',NZ,OLDNZ
      TABW(180) = GRND
C*
      FHOLD     = ITEST .NE. 0
      HOLD      = (MODE .EQ. 2) .AND. .NOT.FHOLD
      OPER      =  MODE .EQ. 3
      RESET     = .NOT.(HOLD .OR. OPER)
      CALL WCNTRL(RESET,HOLD,OPER)
C*
      DAC(55)  = -(TABW(118) - BIAS) / SCAL
      DAC(56)  = -(TABW(119) - BIAS) / SCAL
      DAC(57)  = -(TABW(120) - BIAS) / SCAL
      DAC(58)  = -(TABW(121) - BIAS) / SCAL
      DAC(59)  = -(TABW(122) - BIAS) / SCAL
      DAC(60)  = -(TABW(123) - BIAS) / SCAL
C*
      AXCAB    = TABW( 97)/GMTR
      AYCAB    = TABW( 98)/GMTR
      AZCAB    =-TABW( 99)/GMTR
      PCAB     = TABW(100)
```

```
      QCAB    = TABW(101)
      RCAB    = TABW(102)
C
C     LADDER MOTION BASE DELAY BUFFER
C
       OLDP = P
       OLDQ = Q
       OLDR = R
       OLDPDOT = PDOT
       OLDQDOT = QDOT
       OLDRDOT = RDOT
       OLDNX = NX
       OLDNY = NY
       OLDNZ = NZ
       OLDGRND = GRND
C
C*
      RETURN
      END
```

VITA

R. Marshall Smith (12/27/63) is the engineer responsible for the procurement, integration and development of NASA Langley Research Center's flight simulation graphics systems. In addition, he is currently responsible for conducting research into flight simulation technologies in order to improve/validate flight simulator realism and performance. He has been with NASA LaRC since 1983 and in the Analysis and Simulation Branch since 1987. He holds a Bachelors Degree in Electrical Engineering from the University of Tennessee.

R. Marshall Smith