Human-Machine Interface for Tactical Air Traffic Control Communications

Alan Mastin
University of North Florida

Suggested Citation
Mastin, Alan, "Human-Machine Interface for Tactical Air Traffic Control Communications" (1990). UNF Graduate Theses and Dissertations. 133.
https://digitalcommons.unf.edu/etd/133
HUMAN-MACHINE INTERFACE FOR
TACTICAL AIR TRAFFIC CONTROL COMMUNICATIONS

by

Alan Mastin

A thesis submitted to the
College of Computer and Information Sciences
in partial fulfillment of the requirements for the degree of

Master of Science in Computer and Information Sciences

UNIVERSITY OF NORTH FLORIDA
COLLEGE OF COMPUTER AND INFORMATION SCIENCES

August, 1990
The thesis "Human-Machine Interface for Tactical Air Traffic Control Communications" submitted by Alan Mastin in partial fulfillment of the requirements of Master of Science in Computer and Information Sciences has been approved by the thesis committee:

Approved by the thesis committee: Date

Signature Deleted

Thesis Adviser and Committee Chairman
Signature Deleted

Signature Deleted

Signature Deleted

Accepted for the College of Computer and Information Sciences:

Signature Deleted

Interim Dean

Accepted for the University:

Signature Deleted

Vice-President for Academic Affairs
ACKNOWLEDGMENT

I wish to thank the following people at the Jacksonville Center facility and at the Jacksonville TRACON facility for their help in my understanding of the ATC system and their contribution to the design specifications of this project. Jacksonville Center facility: Cliff Monteau, Facility Manager; Marvin Perkins, NAS Implementation Specialist; Dave Scott, Controller; Larry Haftel, Controller; L. J. Bennett, Automation Manager. Jacksonville TRACON facility: Bob Flemming, Controller; Ernie Reed, Data Systems Specialist.

I also wish to thank Dr. Leitner, my advisor, for his encouragement and guidance. Also Dr. Chua and Dr. Winton for being on my thesis committee and for providing additional insight and support.

I especially wish to thank my wife Marilyn and daughters Elan and Alisa for their support and encouragement and for putting up with the long hours spent in the "lab" and doing research at the ATC facilities.
TABLE OF CONTENTS

List of Figures........................................ v
Abstract.................................................. vi

Chapter 1: Introduction and Background......................... 1
1.1 ATC Communications and Facilities....................... 1
1.2 Digital Communications Background...................... 2
1.3 Mode S System........................................ 5
1.4 Airborne Component................................... 12
1.5 Controller's Job and Requirements....................... 12
1.6 Controller's Workstation............................ 14

Chapter 2: The Problem and the Solution......................... 17
2.1 Statement of the Problem.............................. 17
2.2 Proposed Solution and Design Criteria................... 19

Chapter 3: Device Specifications.............................. 26
3.1 Computer Hardware.................................... 26
3.2 Device Software....................................... 32
  3.2.1 Overview......................................... 32
  3.2.2 Building Messages................................. 37
  3.2.3 Device-Host Dialogue.............................. 44
3.3 Host Computer Software................................. 48

Chapter 4: Simulator........................................ 54
4.1 Description of the Simulator............................ 54
4.2 Results of the Simulation................................ 58

Conclusion................................................ 62
Recommendations........................................... 64
References.................................................. 65

Appendix A: Controller Phrases............................... 67
Appendix B: Menu Hierarchy.................................. 73
Appendix C: Device Pseudo Code.............................. 151
Appendix D: Host Pseudo Code............................... 188
Appendix E: Message Formats................................. 192
Appendix F: Simulator Source Code............................ 200
  Device Code.............................................. 201
  Host Computer Code..................................... 233

Vita.......................................................... 248
FIGURES

Figure 1: Mode S Data Formats.......................... 8
Figure 2: ATC Facility Computer Configuration............. 11
Figure 3: Mock-up of the Device.......................... 27
Figure 4: Device Face.................................. 30
Figure 5: Device Main Logic................................ 33
Figure 6: State Diagram................................ 34
Figure 7: Main Menu Items............................... 39
Figure 8: Altitude Menu Selections.......................... 43
Figure 9: Main Menu Hierarchy............................ 75
Figure 10: Vectoring Menu Hierarchy....................... 76
Figure 11: Altitude Menu Hierarchy......................... 77
Figure 12: Speed Menu Hierarchy........................... 78
Figure 13: Holding Menu Hierarchy......................... 79
Figure 14: Routing Menu Hierarchy......................... 80
Figure 15: Arrivals Menu Hierarchy......................... 81
Figure 16: Altimeter Menu Hierarchy....................... 82
Figure 17: Traffic Menu Hierarchy.......................... 83
Figure 18: Communication Menu Hierarchy................... 84
Figure 19: Alert Menu Hierarchy........................... 85
Figure 20: Radar Menu Hierarchy........................... 86
Figure 21: Weather Menu Hierarchy.......................... 87
ABSTRACT

This thesis proposes the design for a prototype device that would be used by Air Traffic Controllers in the radar environment to input tactical Air Traffic Control (ATC) instructions to be sent to aircraft via the Mode S digital data link network. The purpose of the device is to reduce the time required to issue instructions and to eliminate misunderstandings that occur when instructions are issued over voice transmission frequencies. The purpose of this thesis was to develop the device in the most ergonomically suited manner based on the air traffic controller's communications requirements. Digital communications systems include both airborne and ground based components. This project was concerned with the development of the ground-based aspect of the communications system.
Chapter 1
INTRODUCTION AND BACKGROUND

1.1 ATC Communications and Facilities

Air Traffic Control (ATC) facilities that control aircraft are divided into three types: control tower, terminal, and enroute. Terminal and enroute facilities control aircraft primarily by radar while control towers use visual means. Towers control aircraft on the ground, taking off and landing at an airport. Terminal facilities control aircraft departing and arriving from one or more airports in and around medium and large cities. Enroute facilities normally control aircraft between terminal facilities. Generally, terminal facilities handle the lower altitudes (up to 10 or 15 thousand feet) around metropolitan areas while enroute facilities handle the higher altitudes.

Communications within an ATC facility include information exchanges between controllers, between controllers and other agencies, and between controllers and pilots. The latter is known as tactical communications. All tactical ATC communications are currently being accomplished via
Voice communications on VHF, UHF and HF radio frequencies. Voice communication is adequate when traffic volume is normal and the radio equipment is functioning properly. During periods of air traffic congestion, however, the voice frequencies can become saturated with dialogue to the extent that some transmissions are interfered with by other transmissions. Voice communication is subject to equipment failure and degradations, interference and static. These potential problems can cause instructions to be lost, misunderstood, or acknowledged by the wrong pilot. However, digital communications would ensure that the intended aircraft receives a clear, easily understood message in a timely manner.

1.2 Digital Communications Background

The Federal Aviation Administration (FAA) has been working on the digital communications concept for over 10 years. The bulk of the research has been taking place at the Massachusetts Institute of Technology’s (MIT) Lincoln Labs [Baker89]. The initial concept was referred to as the Discrete Address Beacon System (DABS), which was developed in the late 1970’s. A beacon system is a system where the ground radar sends a signal that will interrogate the airborne transponders causing them to send a reply back
with the beacon code of the transponder and other data such as altitude information. This concept evolved into what is now known as the Mode Select (Mode S) Beacon System. This system is designed to eventually replace the existing Air Traffic Control Radar Beacon System (ATCRBS). The difference between the two systems is that Mode S provides discrete aircraft addressability and data link capabilities. The beacon code in the ATCRBS environment is a four digit code set by the pilot on his transponder set. In the Mode S environment, the beacon code is set uniquely by the manufacturer of the transponder. Mode S was designed to be completely compatible with ATCRBS allowing for a long transition period. Several engineering test sites have been set up around the country for evaluating the system. Scheduled deployment of the Mode S system is to begin in 1991 with full implementation by 1995 [FAA88].

One possible scenario for a digital communication system would include a ground component and an airborne component. The ground system allows the controller to rapidly input a message and send it to a specific aircraft. The airborne system would receive the message, display it on a device in the instrument panel and audibly inform the pilot that it was received. The pilot could acknowledge the message by pressing a button and the airborne system would then send an acknowledgment message
back to the ground system. The ground system could be implemented as an applications processor linked into the current ATC computer system utilizing the proposed Mode S transponder/data link system.

There are approximately 20 data link applications being considered by the FAA [Baker89]. One contract has already been awarded for the Aeronautical Data Link system scheduled to be implemented in 1994. This will establish a method for transmitting weather data to aircraft. The initial ATC services under consideration that are similar to the concepts proposed here are transfer of communications, altitude assignment, communications backup and menu text ATC instruction delivery. The FAA is in the process of site preparation for a new controller work station called the Sector Suite. The Sector Suite incorporates a standard typewriter keyboard and special function keys for computer input. The ATC services under consideration for the digital data link would require the human-machine interface to be limited to the design of the Sector Suite input devices. This interface interaction is similar to what a CRT operator would experience using a terminal while running a menu driven program. This would not be the best interaction for a controller whose attention needs to be focused on the radar display. A device specifically designed for tactical ATC communications is not being considered by the FAA.
1.3 Mode S System

The Mode S Beacon System is described by Vince Orlando of MIT Lincoln Laboratory [Orlando89] and by Baker, Orlando, Link, and Collins in paper to be published in an upcoming Institute of Electrical and Electronic Engineers (IEEE) special ATC issue [Baker89]. This system will provide a combined secondary surveillance radar (beacon) and an air-ground data link capability. The system is being designed to be compatible with the existing ATCRBS by allowing Mode S sensors to interrogate ATCRBS transponders, and ATCRBS sensors to interrogate Mode S transponders.

The Mode S system has certain features that are required for the surveillance functions that also enhance the data transmission functions. The Mode S signal format provides over 16 million addresses. Each Mode S transponder will have its address established at manufacture time. The Mode S coding structure provides for a high degree of error detection and provides for error correction on the downlink. Specifications call for less than one undetected error per billion messages. Adaptive reinterrogation provides for automatic reinterrogation of selected aircraft in the event of a garbled message. An all-call feature allows the sensor to interrogate and establish a link with all transponders within range that have not yet been identified.
All data communication occurs during the time the narrow radar beam is striking the target using a technique called monopulse signal processing. The monopulse technique is based on the use of an antenna with multiple patterns that effectively reduce the interrogation time [Baker89] [Orlando89]. This allows for a time period on each scan for data communications. The Mode S interrogation waveform consists of a 3.5 microsecond preamble with two pulses followed by a 16.25 microsecond or a 30.25 microsecond data block. The data is transmitted using differential phase shift keying (DPSK) modulation at a data rate of 4 mega bits per second. The reply waveform consists of an 8 microsecond preamble followed by a 56 or 112 microsecond data block. The data is transmitted in the reply using pulse position modulation (Ppm) at a data rate of 1 mega bits per second.

The Mode S waveform and selective addressing provide an ideal mechanism to transmit other than surveillance data. The communications control features of the data link have been designed to be compatible with the Open Systems Interconnection (OSI) reference model for digital communications. OSI is a standard developed by the International Standards Organization for data communications. The data link provides for messages to be sent both ground-to-air and air-to-ground.
There are three Mode S data formats (see figure 1), two of which can contain message data. The surveillance interrogation and reply format is used strictly for position information for radar tracking and does not include fields for data communications. The surveillance/communication interrogation and reply format is used for both radar positioning and data communications. The communication interrogation and reply format is used in conjunction with the surveillance/communication format to extend the size of the message to be transmitted.

The address and parity are combined into one 24 bit field to minimize channel overhead. The 24 bit parity field is generated from the entire message and overlaid onto the 24 bit address. All transponders will receive the message, decode the parity and receive the address. Only the addressed transponder will accept and process the message. If there is an error in the parity check, the transponder will not recognize the address and not acknowledge the message which will cause the sender to timeout and retransmit the message. The surveillance and communication control field contains control information on the uplink and Mode C (altitude information) or Mode A (transponder identity codes) on the downlink.
### Surveillance Interrogation and Reply Format

<table>
<thead>
<tr>
<th>FORMAT #</th>
<th>SURV/COMM CONTROL</th>
<th>ADDRESS/PARITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 bits</td>
<td>27 bits</td>
<td>24 bits</td>
</tr>
</tbody>
</table>

### Surveillance/Comm. Interrogation and Reply Format

<table>
<thead>
<tr>
<th>FORMAT #</th>
<th>SURV/COMM CONTROL</th>
<th>MSG FIELD</th>
<th>ADDR/PRTY</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 bits</td>
<td>27 bits</td>
<td>56 bits</td>
<td>24 bits</td>
</tr>
</tbody>
</table>

### Communication Interrogation and Reply Format

<table>
<thead>
<tr>
<th>FORMAT #</th>
<th>COMM CONTROL</th>
<th>MSG FIELD</th>
<th>ADDRESS/PARITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 bits</td>
<td>6 bits</td>
<td>80 bits</td>
<td>24 bits</td>
</tr>
</tbody>
</table>

Figure 1: Mode S Data Formats
The surveillance/communication format can contain from one to four data fields of 56 bits each. The communication formats are used in an extended length message (ELM) protocol. This protocol allows a window of up to 16 eighty bit messages for a single reply on the uplink. ELM provides for the linking up of 32 windows on the downlink for a message of up to 40,960 bits.

The tactical communications messages that would be used would have to be coded in a way to be at most 1280 bits to fit into the communication format for the uplink. Ideally, most messages should be coded to fit into the 224 bits of the surveillance/communication format with four message fields to minimize channel overhead. Each sweep of the radar will require a surveillance transaction. Adding data to that transaction would require much less overhead than if additional communication transactions were required. These messages would be forwarded to the transaction processing program of the Mode S sensor applications software for formatting and transmission on the data link.

The ATC computer system at a typical enroute facility consists of a pair of IBM 3083 computers. A software system called the National Airspace System (NAS) runs solely on one 3083. Terminal facilities typically have older Sperry machines that run a smaller version of NAS.
The terminal computers are linked as satellites to the enroute computer. The NAS software includes the operating system and handles all of the ATC functions (displays, I/O, radar data, communications, data bases, etc.). A skeleton version of NAS runs in low core of the other 3083 at the enroute facility with MVS (the IBM user operating system) running above it. Each NAS system checks the other every 30 seconds and the primary saves its state on disk. If the primary fails to respond to the secondary, the secondary kicks MVS out of core, loads the full version of NAS, loads the last saved state of the primary and takes over as the primary. The NAS software is highly controlled by a central authority at the FAA headquarters. A terminal or enroute facility wishing to modify their copy must first get permission.

The Mode S sensor application software runs in a computer called the Ground Data Link Processor (GDLP). This processor is linked to the ATC computer through a gateway. Other application processors are linked to the system through the gateway. The FAA is in the process of designing a weather services applications processor and a tower workstation applications processor.

The primary concern of the Mode S system is to provide for accurate identification of aircraft. The data link
Figure 2: ATC Facility Computer Configuration

capability is provided to handle anticipated data communications requirements. The discrete address concept of Mode S is provided by having each transponder manufactured with a discrete address code built into it. Thus, the transponder code becomes like a data network address and is transparent to the higher level users. Under the present ATCRBS system, transponder codes consists of only four digits and are manually dialed in by the pilot. This limits the number of discrete codes available, and also leaves room for a pilot to make a mistake setting a code that could be assigned to another aircraft.
1.4 Airborne Component

The airborne component of a digital communications system is not described here. An ergonomically designed unit could be the subject of further research. MIT Lincoln Labs has developed and tested an airborne unit called the Airborne Intelligent Display (AID) [Anderson80]. This system provides for receipt and acknowledgment of messages. The unit is a microprocessor based system that shares the video screen with a weather radar system. A keyboard is attached to the pilot’s control wheel and a printer is mounted on the center console. It also has a speech annunciator system to verbally alert the pilot with its 12 word vocabulary.

1.5 Controller’s Job and Requirements

There are many aspects of the controller’s job, including the radar room environment and the ATC computer system, that have an impact on the design of a digital communication system. One of the critical areas of concern is the proper acknowledgment of aircraft clearances. This area accounts for a major portion of air traffic control mistakes. In some instances, a pilot may acknowledge instructions that he may have misunderstood. The
controller would have no idea that the pilot did not understand the instructions until the aircraft is observed executing the wrong maneuver.

Another problem area is that an aircraft with a similar call sign may acknowledge a clearance intended for another aircraft. In this case, the wrong aircraft might begin to execute a maneuver that could be unsafe. The controller might or might not catch the mistake.

Busy sectors are manned by two controllers. One does nothing but watch the scope and talk to pilots while the other, called the data position, handles data input, updates the flight strips and talks to other ATC facilities.

Controllers currently have four methods for entering aircraft identification into the computer. One is by using a computer identification number (CID), another is by entering the aircraft's call sign, another is by using the transponder code and the fourth is by using the track ball to slew the cursor on the radar display over the aircraft's target. The transponder code will not be available when the Mode S system is implemented. Most controllers use either the CID or the track ball. The call sign is not used by many controllers. The CID is always displayed in
the data block on the radar display in front of the controller.

 Controllers normally have one hand busy at all times. They usually have the mike button in one hand and use the other hand to hold a pencil, roll a track ball that slews a cursor on the radar display or punch the existing computer keyboard buttons. Terminal facilities have a foot mike switch which frees both hands. Enroute facilities are gradually changing to this type of mike switch also. Enroute controllers are required to input clearance data into the ATC computer through a keyboard at their workstation whenever the clearance is given over the radio. The enroute facility computer keyboard is laid out like a standard keyboard, while terminal facility keyboards are not. Each facility has a set of special function keys (start track, drop track, hand-off, etc.). Controllers get used to the position of these keys and are able to find them by touch or a quick glance.

1.6 Controller’s Workstation

Each aircraft on the radar screen being tracked by the computer has a data block placed next to the aircraft’s target. The data block consists of various fields such as
call sign, track, CID, altitude and a "carat" byte that is used by the ATC computer or controller as a flag for various information. A "scratch pad" area is also on the radar screen that the controller can use to hold information. The controller can move the scratch pad to any position on the screen. The pad consists of 20 lines with the first being 32 characters long and the other lines 9 characters in length. The enroute facilities have a computer readout device called the "CRUD" that is a 25 line by 26 character screen to the right of the radar display. One problem controllers have with the CRUD is that it is out of the primary focus of their view while they are looking at the radar display. Important information displayed there can go unnoticed.

The ATC computer holds data about each aircraft including the call sign, altitude, track and groundspeed. The aircraft’s call sign is established when the flight plan is filed, and the transponder code is first detected up by the radar after takeoff. The altitude can be either verified or unverified. Verified altitudes are read from the aircraft’s encoding altimeter by the beacon system (Mode C) and verified by the pilot. Unverified altitudes are either unverified Mode C altitudes or altitudes given verbally by a pilot. The track and groundspeed are computed based on the target’s movement. Altitudes of 18,000 feet and above
are referred to as flight levels, while those below 18,000 feet are referred to by thousands of feet. The CID is assigned to an aircraft when the flight plan data enters a facilities computer database. The CID changes when a flight is handed off from one facility to another. The CID is a unique identifier that can be used to quickly access all data concerning an aircraft. A controller can select a target on the radar display to do a computer function by slewing the cursor over the target and pressing a computer function key.

Each target being tracked by the computer is assigned to a particular sector. A sector is an area of airspace that the controller is responsible for. The controller cannot update the data of an aircraft which is not in his sector. Sectors can be side-by-side or one on top of the other (high altitude/low altitude). Thus a target can be on the radar display of more than one sector. Each sector normally has one radio frequency assigned to it in each frequency band (UHF & VHF). Controllers cannot talk to aircraft in another sector since the aircraft is tuned to the frequency of the other sector. There are some instances that controllers may wish to broadcast a message to all aircraft on their frequency. Significant weather advisories, NAVAID (radio aids to navigation) status are some examples.
2.1 Statement of the Problem

Given the ever increasing volume of air traffic, voice communications are becoming saturated to a point where they are prone to errors and misunderstandings. Such saturation causes delays in critical clearances being executed on a timely basis. It causes controllers to have to repeat instructions, and it causes pilots to misunderstand instructions and perform the wrong operation or causes the wrong pilot to accept a clearance intended for another pilot. The problem of saturated voice communication adversely affects the safety of air traffic.

A method must be developed that will relieve controllers of their voice communication workload and ensure that tactical instructions are delivered expeditiously to the intended aircraft, free of errors.

Interviews were conducted with several controllers and data system specialists at the Jacksonville Center facility at
Hilliard, Florida and at the Jacksonville Terminal facility at the Jacksonville International Airport. The controllers were also observed working at their stations. The idea of using a digital communication system to transmit messages to pilots was presented. The controllers were asked if such a system would be helpful, what they thought it should look like and what functions it should include. They were asked what aspects of their communications workload caused the most problems with delayed transmissions and misunderstandings. Discussions followed covering what functions they thought should be included and what functions are not needed. The controllers were asked what would be the best method to display digital communication information and what would be the best method to input message data. The general consensus was that a digital communications system would be useful as long as it is easy to use and did in fact reduce their communications workload.

After examining the requirements of FAA controllers, it was found that the solution must have the following characteristics:

1) The method must behave in a predictable manner and easy to learn and use.
2) The controller must be free to use either the data link method or the voice method at his discretion for any particular message.

3) The method must be flexible to accommodate the dynamic air traffic control environment whereby controllers can tailor the method to whatever sector they are working. The content of instructions can vary from sector to sector. Some instructions are consistent in all sectors while the content of others are specific to a particular sector. Conditions within a sector are also dynamic due to changing traffic flow and weather. The content of an instruction to a pilot may need to be changed quickly.

4) The method used must be designed in a way that minimizes the controller's distraction from the radar display. As much information as possible about the status of the digital communication system should be displayed on the radar display.

2.2 Proposed Solution and Design Criteria

A solution to the problem of saturated voice communications is to give controllers a method to send tactical instructions to pilots over the Mode S data link thereby
reducing their verbal communications workload. A prototype device will be built that will be a hand held, microprocessor based device with a menu driven input methodology using as few input keys as necessary. The device will be used by the person in the scope position and communicate with the ATC NAS computer system and direct messages to the GDLP via the gateway. Minor modifications to NAS would be required to handle it.

In order to meet the requirement that the device be easy to learn and use, a hierarchical menu system will be used to build messages. There is a well defined set of phrases that constitute the bulk of communications between controllers and pilots. This set is easily incorporated into a menu driven input methodology.

The proposed device will have a menu display with menu selection keys next to the menu items. This provides for quick, positive selection by the controller. The most important aspect of the keyboard is its ease of use that requires the least amount of distraction from the radar display. The design will be such that, with practice, the controller can input messages from touch and memory with an occasional glance for verification. The controller will build a message by selecting menu items and send it by pressing a transmit key. The device will pass the message
to the host ATC computer which would send it to the aircraft via the Mode S system. The pilot will receive the message and reply by sending back an acknowledgment message. The host will receive the acknowledgment message from the Mode S system and pass it to the device.

To make it easier for a controller to use either the digital method or the voice method to give instructions to pilots, an auxiliary mike button will be attached to the device when held in the hand to allow controllers to key the mike when they wish to transmit a voice message. The device can also be placed on the work table in front of the radar display. With this arrangement, controllers can use either the voice or digital method, or both at the same time. Being a hand held device is important because it must be available regardless of what position the controller is seated in.

The device will both send data to and receive data from the host. In order for the device to operate in a predictable manner, the host computer should never send data to the device without being polled by the device. This will avoid interrupting the controller while he is inputting data.

Obtaining the identification of an aircraft for which a message is intended will be done by one of two methods.
One is by slewing the cursor on the radar display over the target using the track ball. The host will send an aircraft’s data to the device with each poll while the cursor is over a target. The second method is for the controller to enter the CID of the aircraft into the numeric keys of the device. The device will then ask the host for the aircraft data of that CID.

The radar display has areas for displaying data associated with digitally transmitted messages. The need for controllers to keep their eyes on the scope dictates that as much of the digital message data be displayed there as possible. The host computer will use the carat byte in the aircraft’s data block to place a status character of the most recent message. Information such as the last few messages will be placed in the scratch pad area of the radar display.

The host will use the data in transmitted messages to update appropriate information that would normally be keyed in by the controller. This will relieve some of the data entry that is currently required to keep an aircraft’s flight plan current.

Lighting inside the radar room must be taken into account for the design. Terminal facilities are generally very
dark and enroute facilities are fairly dim. Existing keyboards and communication panels have lighted keys. The device considered here will also have lighted keys. The design and layout of the keys will be done in the most ergonomical manner for ease of use.

Controllers indicated that the unit should be capable of being held in the hand, or placed on the table in front of the scope. Thus, it will have an easily held detachable pistol grip while held in the hand. While on the table, it will be attached to a mount with soft rubber skid pads so it will not slide. Since the controller is used to holding the mike button in one hand, the pistol grip should incorporate an auxiliary mike button. Discussions with controllers indicate that the controller population is comprised of a large portion of left-handed people. The grip will be designed to be held in either hand.

The display should either be a backlit color LCD or a plasma display. Plasma displays are fast, provide excellent visibility in low light conditions and can be driven by existing CRT drivers. The disadvantages of plasma displays are their cost, power requirements and weight. G. W. Dick, however, proposes a plasma panel design that reduces power requirements [Dick86]. LCD displays on the other hand are fairly inexpensive and have
very low power requirements. The disadvantages of LCD's have been speed and visibility in low light conditions. Advances in thin film transistor LCD's (TFT-LCD) used in avionics applications have provided high resolution, full color devices with response times under 30 mSEC [Firester87]. A well backlit TFT-LCD will be ideal for this application.

The format of phrases that controllers use to give instructions to pilots consists of aircraft ID, caller ID and message. For example, UNITED 323, JAX CENTER, TURN RIGHT HEADING 270. Controllers normally leave out the caller ID when talking to pilots. The set of phrases [ATC87] will be encoded to save transmission bandwidth. There are three basic phases of flight for which a subset of phrases are used: takeoff & landing, climbout & descent and cruise. These subsets are generally used by tower, terminal and enroute controllers respectively. Tower controllers control aircraft by visual contact. The device under consideration here would be used in the radar environment. The set of phrases would, therefore, be limited to those used by terminal and enroute controllers. Not all phrases would lend themselves to being incorporated into a digital data link system. Those that do lend themselves are listed in Appendix A. Phrases are listed by category based on usage. Some phrases have plain text data
such as routes of flight, weather data, SIDs (standard instrument departures) and STARs (standard arrivals).

A mechanism to load various menus into the device will allow it to be adaptable to the dynamic air traffic control environment. The content of some menus will vary from one sector to another and/or due to changing traffic conditions. The device will have a function whereby an updated menu can be loaded whenever the controller needs it.
Chapter 3

DEVICE SPECIFICATIONS

The device specifications are based on a study of air traffic controllers' requirements, ATC computer configurations and the radar room environment. It is designed to be easy to use and to be "user proof". A mock-up of the device was built to investigate the ergonomics of the keyboard and the grip (see figure 3).

3.1 Computer Hardware

The physical size of the device will be approximately 5.75 inches wide, 9.25 inches long and 1 inch thick. It will be made of lightweight, durable plastic with rounded edges. The unit will have an RS-232 communication jack with two of the unused pins used for a DC power supply. The connection is located in the bottom of the unit to be attached to one of two mounting options.

The first mounting option is a detachable pistol grip with a mike trigger. The unit will snap onto the grip making
Figure 3: Mock-up of the Device
the RS-232 connection. The data, power and mike circuits will exit the grip at the bottom in an integrated cord. It was found by experimentation with the mock-up that the grip will be able to be rotated up to 70 degrees clockwise from center for right-handed people holding the device in the left hand and up to 70 degrees counterclockwise from center, for left-handed people holding the device in the right hand.

The second mounting option will be a heavy, flat 5.5"x0.9"x0.25" slab with a full rubber anti-skid pad. This will be used by controllers who prefer to set the unit on the work table next to the track ball. The data and power circuits will exit the slab at the top in an integrated cord.

The unit will run on its own microprocessor and communicate with the host computer via the RS-232 interface. The processor will be fast to maximize throughput. An Intel 80386 or a Motorola 68030 will suffice. The unit will include a battery backup to provide uninterruptible power in the event the cord is disconnected or the device’s mount is changed. The use of a stand-alone processor will allow easy integration into the different host systems that exist in the terminal and enroute ATC facilities.
The memory space will include volatile RAM, nonvolatile RAM and EPROM. The program that runs the device will reside in the EPROM. The main program will begin at an address that will be the first instruction executed after the device is turned on. Thus, the device would be ready shortly after power is applied. The nonvolatile RAM will be used to store the menu data, current state of the device and the data structures used to hold the message grammar. The volatile RAM will hold temporary data used by the program in the course of execution. The EPROM chip will be programmed on a device suited for that purpose and inserted into the unit. The nonvolatile RAM will be loaded through the RS-232 port from the host computer with a program designed for that purpose.

The face of the device will include a 20 line by 31 character backlit, color LCD, on/off switch, battery and error LEDs, buzzer and the following keys: 18 function keys, CLEAR key, transmit key (XMIT), previous menu key (PREV MENU), left and right arrow keys, alpha keys and a numeric keypad. The alpha keys will include the letters A-Z and the space (see figure 4).

Rolf Ilg [Ilg87] did a study on keyboard design and found that the following provide for a good ergonomic design: Keys should be square with 14mm sides and 19mm apart. They
Figure 4: Device face
should be concave with a radius of at least 30mm. The
depth of depression should be 4mm. The force required to
press the key should be between 20 and 70 cN. The keyboard
should be inclined 8 degrees and have a hand-rest.

This design will follow Ilg's recommendations except that
the size of the keys will be changed for space
considerations. The table mounting option will be inclined
8 degrees. All keys except the CLEAR, XMIT and space keys
will be 10x12mm. The CLEAR, XMIT and space will be
10x20mm. All keys will be backlit and concave according to
Ilg's findings.

Controllers operating the radar position normally use one
hand to input data by the "hunt and peck" typing method.
The alpha keys will be arranged sequentially in 5 rows to
save space. The vowel keys will be a different color than
the other alpha keys to make them easy to find. The
numeric keypad will be a standard layout. There will be an
area under the numeric keypad to serve as a hand rest. The
CLEAR, XMIT, PREV MENU and arrow keys will be directly
under the display. The XMIT and CLEAR keys will be on
opposite sides of the device to avoid inadvertently
pressing the wrong key.
3.2 Device Software

3.2.1 Overview

The program that will run in the prototype will begin execution immediately after the device is turned on. The program will first initialize its data structures and then enter in a loop that will display the main menu, poll the host computer and check to determine if a key has been pressed (see figure 5). It will use program input/output (rather than interrupt input/output). If a key is pressed, the program will branch to a routine to accept the input. Poll requests will be processed before control is given back to the main loop.

The system can be in one of five states: checking if a key has been pressed, polling the host, accepting keyed input, receiving data from the host or building and sending a message to the host (see figure 6). The states are mutually exclusive and the tasks being performed in each state will be completed before leaving that state.

With these criteria, the device will poll the host only when the controller is not building a message. Once the controller begins building a message, the device will wait until the controller is finished or a certain time period
Figure 5: Device Main Logic
Figure 6: State Diagram

1 - NO KEY PRESSED
2 - HOST HAS NO DATA TO SEND
3 - HOST HAS DATA TO SEND
4 - DATA RECEIVED FROM HOST
5 - KEY PRESSED
6 - KEY OTHER THAN CLEAR PRESSED
7 - CLEAR PRESSED
8 - XMIT PRESSED AFTER VALID INPUT
9 - MSG SENT TO HOST
has elapsed. The device must not be allowed to stay in the input state indefinitely and cease polling the host (see state diagram in figure 6). The controller must be alerted to incoming pilot messages and acknowledgments on a regular basis. Thus, the device will set a timer and warn the controller if more than 10 seconds has passed since the first key was pressed to build the current message. It was found by experimentation with the simulator described in chapter 4 that most messages take less than five seconds to build and transmit.

The input sequence is well established based on the message being built. If an invalid key is pressed, a buzzer will sound and the controller given a chance to select the correct key. If the controller selects an incorrect menu item, he will be able to go back one menu level. The CLEAR key should be pressed if the controller wishes to abandon the message being built. Pressing the CLEAR key will always cause the program to branch back to the main loop. All of the keys on the device can be used to build messages. However, most input will be done using the menu selection keys. The alphanumeric keys will be used for the messages that require plain text.

The host computer is responsible for putting data into the communications port when the device asks for it. This
includes aircraft identification data, message acknowledgments, air-to-ground messages, memory load data, pilot weather report (PIREP) data, status data on radio aids to navigation (NAVAIDS) and altimeter data. Information being passed to and from the host will include a code to identify the type of data. A stop-and-wait protocol will be used with a timeout period of not more than half a second. A timeout will signal the controller to check the RS-232 connection by displaying a timeout message and sounding the buzzer. The controller must then press any key which will cause a branch back to the main loop.

A stop-and-wait protocol is one in which one computer sends some information to another computer and waits for the other computer to acknowledge that it was received error free. If an error in transmission did occur, a negative acknowledgment is returned to the sender and it tries again. A timeout occurs when the sender does not receive the acknowledgment from the other computer within a certain period of time. A timeout could occur if the communications link is broken, if the other computer fails or if the returning acknowledge message has errors whereby the sender does not recognize it.
The device's screen is laid out with the top 18 lines for the menu and the bottom 2 lines for a message area that shows the message being built. A line will be drawn between the 18th and 19th lines to separate these areas. Another line will be drawn down the center of the menu area to separate the left and right sides of the menu.

3.2.2 Building Messages

The program will go into the state of accepting keyed input if a numeric key or a function key is pressed. Any other key, except CLEAR, will cause an error condition (buzzer will sound and error LED will light for about 1 second) and the controller is given the opportunity to press a valid key.

The controller may press the CLEAR key at any time while building a message. This will cause the current message to be erased and the program will branch back to the main loop where the main menu is displayed.

If a number is pressed, the program will assume that a three digit CID is being input. It will accept the three digits and send a CID request to the host and wait for a response. The response will return the aircraft's call
sign which will be displayed in the message area. The program expects a function key or the CLEAR key to be pressed next.

If a function key is pressed, either initially or after the CID is entered, the program will branch to the appropriate routine to handle the menu item. If the menu item requires an aircraft id, the program checks if one is stored. If not, an error condition occurs, the message "NO AC IDENTIFIED" is displayed, and the program branches back to the main loop. An aircraft ID and its associated data can be stored in one of two ways. The first method is when the user inputs a valid CID. The second occurs with the following sequence: 1) The device polls the host on the next pass through the main loop. 2) The host checks if the cursor on the radar display is over a valid target in the controller's sector. 3) If so, the host sends the aircraft ID and data back to the device.

The aircraft ID data in the device will be cleared each time the host is polled or the CLEAR key is pressed. This means that the controller must positively identify an aircraft for each message. Once the controller begins inputting a CID, the host will not be polled until he has completed keying in a message or the CLEAR key is pressed. Thus the aircraft ID will stay current as long as the
controller is building a message. In addition, if the radar display cursor remains over a controller's target, the host will return the same aircraft ID with each poll. This will make it appear to the controller that the aircraft remains identified.

Once the controller has identified an aircraft to send a message to, he presses a function key next to the desired main menu item and the menu that was selected is displayed (see figure 7). The program traverses a hierarchy of menus and accepts alpha-numerics to build a message. The menu system is hierarchical in that, at each level, the controller is given several alternatives to pick from whereby he can travel down a branch of the hierarchy to build a specific message. See Appendix B for diagrams of the hierarchy and examples of the menus.
The message is displayed in the message area as it is being built with each new piece of information. If an invalid key is pressed, there will be an error indication and the user has the opportunity to press a correct key. If the CLEAR key is pressed at any time, the message area is cleared and the program returns to the main loop. If the controller presses the wrong menu key, he can press the PREV MENU key. This will cause the last part of the message to be cleared and the previous menu to be displayed. The user can use the arrow keys to move the cursor in order to rekey alphanumeric data.

The controller will press the XMIT key when the message is complete. If he takes longer than 10 seconds to build the message, the buzzer will sound every 10 seconds to remind him that he is still in the input state. The controller can always press the CLEAR key if he gets confused to get back to the main menu and start over. The following occurs once the XMIT key is pressed. A message number is assigned to the message and the message is coded into the transmission format and stored in a list of messages in the device memory in both text and coded form. The coded form is then sent to the host for transmission to the pilot via the Mode S system. Finally, the screen is cleared and program branches back to the main loop which displays the main menu.
The following is an example of how a message will be built. Figure 8 shows the branch of the altitude menu hierarchy that this sequence would follow. Assume that the controller wishes to send the following message:

UNITED 237, AT PILOT'S DISCRETION, DESCEND AND MAINTAIN FLIGHT LEVEL 240.

Step 1: The controller will identify the aircraft either by entering the CID or by slewing the radar display cursor over the target using the track ball.

Message Area: UNITED 237

Step 2: The controller will press the function key next to the "ALTITUDE" selection on the main menu. The altitude menu will be displayed next.

Message Area: UNITED 237

Step 3: The controller will press the function key next to the "AT PILOT'S DISCRETION" selection on the altitude menu.

Message Area: UNITED 237 AT PD
Step 4: The controller will press the function key next to the "CLIMB/DESCEND" selection on the altitude menu. The program will check if a verified altitude is stored with the aircraft’s ID. If not, the climb/descend options menu will be displayed and the controller will have to select the "CLIMB" or "DESCEND" option. If the aircraft’s altitude is known, the program will know whether it is a climb or descent. The altitude options menu (figure 8) will be displayed next.

Message Area: UNITED 237 AT PD DESCEND & MAINTAIN

Step 5: The controller will press the function key next to the "240" selection on the altitude options menu.

Message Area: UNITED 237 AT PD DESCEND & MAINTAIN FL240

Step 6: The controller will press the XMIT key. The message will then be stored in memory and sent to the host for transmission to the pilot via the Mode S data link.

The list of messages in memory will be used for the controllers to review what messages they have sent and to retransmit unacknowledged messages. One of the main menu
<table>
<thead>
<tr>
<th>VECTORED ALTIMETER</th>
<th>TRAFFIC COMMUNICATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTITUDE</td>
<td>ALERT</td>
</tr>
<tr>
<td>SPEED</td>
<td>EMERGENCY</td>
</tr>
<tr>
<td>HOLDING</td>
<td>RADAR</td>
</tr>
<tr>
<td>ROUTING</td>
<td>WEATHER</td>
</tr>
<tr>
<td>ARRIVALS</td>
<td>NAVIADS</td>
</tr>
<tr>
<td>ALTIMETER</td>
<td>ACKNOWLEDGE</td>
</tr>
<tr>
<td>ACKNOWLEDGE</td>
<td>RE-XMIT</td>
</tr>
<tr>
<td>AC LIST</td>
<td>OTHER FUNCTIONS</td>
</tr>
</tbody>
</table>

**UNITED 237**

- **VERIFY ALT**
- **VERIFY ASSIGN**
- **VERIFY AT ALT**
- **REQ ALT CHG**
- **EXPECT HI/LOW**
- **MAINTAIN BLOCK**
- **MAINTAIN ALT**
- **CLimb/DEC WHEN AT PD**

**UNITED 237 AT PD**

- **VERIFY ALT**
- **VERIFY ASSIGN**
- **VERIFY AT ALT**
- **REQ ALT CHG**
- **EXPECT HI/LOW**
- **MAINTAIN BLOCK**
- **MAINTAIN ALT**
- **CLimb/DEC WHEN AT PD**

**UNITED 237 AT PD DESCEND & MAINTAIN**

**Figure 8: Altitude Menu Selections**

- **510**
- **470**
- **430**
- **390**
- **350**
- **310**
- **290**
- **260**
- **240**

- **530**
- **490**
- **450**
- **410**
- **370**
- **330**
- **290**
- **270**
- **250**
items will give a list of messages sent and received. The controller will be able to press the function key next to an unacknowledged message to retransmit the message.

3.2.3 Device-Host Dialogue

While in the main loop, the program will continually poll the host for data as long as a key is not pressed. The program sends a poll request and waits for a response or times out. There are four types of information that the host can send back. The host will check each type in a specific order to ensure that the most important is sent first. If any of the types of data is returned, the program will branch to a routine to handle it. If none of the types of data is waiting to be returned by the host, the host will return a message that it has no data to send. The program will remain in the main loop in such a circumstance.

The first type of information is message acknowledgments coming back from pilots. The program looks up the message number and flags it with an "A+" for a positive acknowledgment, an "A-" for a negative acknowledgment or an "A" for a neutral acknowledgment. A positive acknowledgment from a pilot is like a "yes" answer in which
he has complied with a clearance, seen conflicting traffic, etc. A negative acknowledgment is like a "no" answer in which he does not see the conflicting traffic, reached an altitude, etc. A neutral acknowledgment is one in which the pilot is simply acknowledging that a message was received. The program will clear the screen and display the aircraft's call sign and acknowledgment type. If the message number cannot be found, an error message to that effect will also be displayed. The buzzer will sound to alert the controller that an acknowledgment has arrived. The program will wait for the controller to press any key to allow time to read the message. Once a key is pressed, the screen is cleared and control is passed to the main loop where the main menu is displayed.

The second type of data is an air-to-ground message initiated by a pilot. The host will know which aircraft are assigned to a controller's sector and will ensure that the message gets directed to the right controller. The program looks up the message, decodes it, clears the screen and displays the message with the aircraft's call sign. If the message is undecodable, the aircraft's call sign and error message will be displayed. The buzzer will sound, in any event, to alert the controller, and the program will wait for a key to be pressed. The message will then be stored in the message list with an "I" to indicate an
incoming message. The message area will be erased and a branch done back to the main loop after the controller presses any key.

The third type of data that could come back from the host computer is the memory load transaction. There are 10 (with the capability of more) user programmable menus in the menu hierarchy. The data can be prepared either in the ATC main computer or in a personal computer (PC), or in both. If done in the ATC main computer, someone other than the controller using the device will set up the new menu and send it to the device. If done on a PC, the device is plugged into a serial communications port and the PC program simply responds to poll requests with a memory load transaction. A program to set up, store and send the programmable menus would be written by FAA programmers. When a memory load transaction is received, the device program displays the menu about to be loaded and waits for either the CLEAR or XMIT key to be pressed. The menu is not loaded by the device if the CLEAR key is pressed but is loaded if the XMIT key is pressed. Any other key will cause an error condition. Program control then goes back to the main loop.

The fourth type of data that could be received is aircraft identification data. It consists of the CID, call sign,
altitude, groundspeed, track and flight mode. The data is sent on each poll request as long as the cursor on the radar display is over one of the controller’s aircraft. The call sign is displayed in the message area and the data is stored. If the target moves away from the cursor or the controller moves the cursor away from the target, the device will clear the aircraft data on the next poll request.

Other data not related to poll requests is passed to and from the host as a result of certain activities by the controller. A completed message to be transmitted to a pilot is one. The device sends the message to the host and does not expect a response other than the stop-n-wait protocol dialogue. If the controller is building a message to send NAVAID data to a pilot, the device will send a NAVAID request to the host and wait for a response with the NAVAID data. The same dialogue occurs if the controller is building a message to send PIREP or altimeter data to a pilot or desires to view his AC list.

Appendix E shows the format of the messages being passed between the host and the device. The pseudo code to support the design specifications for the device software and menu hierarchy is listed in Appendix C. The code is presented in a high level format which shows the basic logic of how the device will function.
3.3 Host Computer Software

The host computer provides data for the device and routes messages to and from the Mode S system. The host software will provide the data structures required to hold data that is to be transmitted to the device.

The host main routine is executed when an interrupt occurs as a result of data entering the communications port to which the device is attached. There are nine types of data that can arrive in the communications port: a communications acknowledgment (ACK), a negative communications acknowledgment (NACK), a poll request, a CID request, an outbound message, a NAVAID status request, an altimeter request, an AC list request, or a PIREP request.

If an ACK is received, nothing is done and control is given back to the operating system. If a NACK is received, the last communication sent to the device is re-transmitted and control is given back to the operating system. The host will timeout and warn the computer operator if a device fails to respond to an ACK or NACK. The computer operator will then inform the controller that his device may have failed or been disconnected.
A poll request causes the host to begin checking four types of data that may be waiting to be sent to the device. Three of these data types will have been stored in FIFO queues for each type. The queues will be checked in a specific order based on the importance of the data waiting to be sent. The host will send the first member of the first queue that has data and then give control back to the operating system. If all three queues are empty, it will check and determine if aircraft identification data is to be returned to the device. If no data is present for a poll request, it will send a no data response.

The first queue checked holds message acknowledgments sent from pilots. The host will place the message data into the queue when it is received from the Mode S processor. It will also place acknowledgment characters in the aircraft’s data block on the radar display as an extra visual signal to alert the controller that the message acknowledgment was received. An "A", "A+" or "A−" will be used for the neutral, positive or negative acknowledgments.

The next queue checked is the air-to-ground message queue. This data is handled the same way as message acknowledgments except the host will put an "I" in the aircraft’s data block.
The last queue checked is the memory load queue. This queue holds menu data to be loaded into nonvolatile RAM. The host software responsible for loading the queue is also responsible for error checking and formatting. This software will reside in the ATC computer so a menu can be loaded while the controller is using the device. The person manning the data position would enter the new menu through a terminal and direct the host computer to put the menu data into the queue when the radar controller is ready for it. The software to load menus could also reside in a PC. The controller could plug the device into the PC prior to taking the shift and load the programmable menus with data associated with the sector about to be worked. With this method, a controller can keep a personalized copy of the menu system on a diskette.

The host checks if aircraft identification data is waiting to be sent by testing the position of the cursor on the radar display. If the cursor is over the target of one of the controller's aircraft, it sends the aircraft's identification data to the device.

The host computer will receive a poll request every second as long as the controller is not building a message on the device. This timing is based on experimentation with the simulator described in chapter 4. The queues will normally
receive data on a much less frequent basis. The message acknowledge queue will receive data only as often as the controller transmits messages. The air-to-ground queue will receive data only as often as pilots transmit messages to the controller. The memory load queue might receive data once or twice during a shift. Therefore the cursor position on the radar display will be tested on a majority of the poll requests. For most of those, the cursor will not be over an aircraft’s target. The majority of poll requests will therefore be responded to with a no data response.

The next item that might come from the device into the communications port is a CID request. This will cause the host to look up the CID in its tables. If a CID is found for an aircraft under the control of the controller, the host would return the aircraft’s identification data. If the CID is not found an invalid CID message will be returned. If the aircraft is in another controller’s sector, a message stating that will be returned. If the aircraft is in another controller’s sector, a message stating that will be returned.

An outbound message arriving in the communications port from the device will cause the host to encode it and format it for Mode S transmission. The host will place a "message out" character in the aircraft’s data block on the radar display as an additional visual reference that a message
was sent to the aircraft. A "U" symbol will be used for an unacknowledged message and an "M" symbol for a message not requiring an acknowledgment. The host computer in an enroute facility will also use the data in the message to update the appropriate flight plan data fields.

The next item that could arrive in the communications port is a NAVAID status request. NAVAID status data is stored in the ATC computer. A NAVAID status request will cause the host send the status of the selected NAVAID.

The next item that could arrive in the communications port is an altimeter data request. Altimeter data is stored in the ATC computer for all facilities in the country. An altimeter data request will cause the host send the data of the selected station.

Another item that could arrive from the device is an AC list request. The host computer holds data about all of the aircraft in the controller's sector. The call sign, flight mode, CID, altitude, heading, groundspeed and destination of each aircraft will be returned to the device on an AC list request.

The last item that could arrive in the communications port is a PIREP request. PIREPs (pilot weather reports) are already stored in the ATC computer. A PIREP request will
cause the host to build and send a menu of PIREPs associated with the controller's sector.

The pseudo code to support the design specifications for the host software is listed in Appendix D. The code is presented in a high level format which shows the basic logic of how the host routine will function.
4.1 Description of the Simulator

A simulator was developed to demonstrate how the device works and provides a mechanism for an air traffic controller to use and evaluate it. The simulator was implemented on a pair of IBM XT PC's. Each PC had 640K of memory, a CGA monitor and was running the MS-DOS 3.3 operating system. One PC simulated the device and the other PC simulated the host computer. The two computers were connected via the serial communications port and placed side-by-side. Two people were required to operate the simulator. One sat at the device PC and used it as the communications device. The other person sat at the host PC and accomplished the host computer functions that were simulated.

The software is written and compiled in Microsoft Quick Basic 4.5. See Appendix F for a listing of the device and host source code. A "stop-and-wait" protocol with a block check character was used for communications between the
host and the device. The device face was simulated by
drawing the menu, 18 function keys, CLEAR key, XMIT key and
PREV MENU key on the screen. A mouse was used to select
the keys on the screen. The XMIT, CLEAR and PREV MENU keys
were put on the screen to reduce the number of transitions
from the mouse to the keyboard. The PC keyboard was used
for the remaining alpha and numeric keys.

The host was simulated with minimum automation. The
operator of the host PC had the capability to input data to
be sent to the device. The host software responded to CID
requests from the device with three "hard coded" CID's.
The operator was given the capability to input the
following: AC information to simulate the controller
identifying an aircraft with the track ball, pilot
acknowledgments to tactical instructions sent from the
device, pilot-to-controller messages and an altitude menu
memory load function.

Three out of the 18 main menu functions of the prototype
were partially simulated on the device PC: altitude,
vectoring and retransmit. The altitude function included
the "pilot's discretion" modifier and the altitude
selection menu. The vectoring function was simulated only
with the "turn right/left" menu. The retransmit function
allowed the controller to select a message to retransmit.
It retained up to nine messages including both outbound and pilot-to-controller messages. The software would only allow the controller to retransmit unacknowledged messages.

A software timer was implemented to sound the buzzer if ten seconds had passed since the controller had begun building a message. It started as soon as any valid key was pressed and stopped when a message was sent or the CLEAR key was pressed. The "communications out" software timer was not implemented on the simulator. The device was programmed to poll the host every second. Originally, the device polled the host on every pass through the main program loop. This, however, caused the host PC to spend nearly all of its time processing to poll requests (which were interrupt driven) which effectively did not give the host operator an opportunity to key in data. The one second interval worked out best in that poll requests were frequent enough not to cause perceptible delays and did not have any detrimental effect on keyed input at the host PC.

The device PC software used a routine that tests the mouse button and the keyboard. When a key was pressed, its ASCII code was stored for use by the calling routine. If the mouse was clicked over the XMIT, CLEAR or PREV MENU keys on the screen, the ASCII equivalent was returned.
If the first key pressed was a digit, the device accepted two more digits and sent them to the host PC as a CID request. The request was sent after the third digit was entered and did not require the XMIT key to be pressed. This provided a smooth process in building a message that required the XMIT key to be pressed only once to transmit the message to the pilot.

Tactical messages sent to the host PC from the device PC were displayed on the host PC screen. The host PC operator used this data to send a simulated acknowledgment back to the device PC. The simulated acknowledgment messages were displayed on the device PC screen indicating the call sign and whether it is a positive, negative or neutral acknowledgment. The controller was required to press any key to continue. Pilot-to-controller messages were handled the same way showing the call sign and the message text.

Aircraft data was entered into the host PC and included the CID, call sign, altitude, heading, airspeed and type of aircraft. The completion of entering the data simulated the track ball being placed over the aircraft on the controller's radar screen. The aircraft data was transmitted on the next poll request and the call sign was displayed on the device PC.
The memory load function was simulated with the capability to load the altitude option menu. The new menu was sent to the device when the host PC operator completed entering the menu and the host PC was polled. The menu was displayed on the device PC and the controller could accept it using the XMIT key or reject it using the CLEAR key.

4.2 Results of the Simulation

An air traffic controller from the Jacksonville Center facility volunteered to use and evaluate the simulator. The controller was given a general overview of the concept and a brief explanation of the prototype system. The controller was then shown how to use the device as simulated on the PC while the author acted as the host PC operator.

The scenario was set up as follows: The controller was controlling three aircraft, United 235, Delta 2011 and Cessna N4019T with CID's of 123, 456 and 789 respectively. The controller was asked to send vectoring and altitude messages to his aircraft.

The controller proceeded to build and send messages with little help. The ten second input timer seemed to be
sufficient. Most of the messages were completed between 5 and 10 seconds. A shorter timer would not be recommended since it would cause the buzzer to sound too frequently and become annoying. A longer interval is also not recommended because the object of the timer is to alert the controller that the device is not receiving messages from the host computer.

The methodology was easy for the volunteer to learn and use. He would mostly key in the CID and use the mouse to select a menu item. The identification method simulating the use of the track ball was also demonstrated. Acknowledgment messages were keyed into the host PC and sent back to the device PC. The controller was also sent a pilot-to-controller message and the memory load function was demonstrated by sending a new altitude menu. The simulation was run for about a half an hour.

The simulation was rather cumbersome due to the transition required between using the mouse and the keyboard and due to the lack of automatic functionality in the host PC. It was also difficult to simulate the air traffic control environment without a radar display. A third PC could be used to display a simulated radar display with the three aircraft on it.
The simulation did foster some good discussion concerning the functionality of the system. The following changes and enhancements to the design of the device emerged:

1) The controller should also have the capability to retransmit acknowledged messages. Pilots sometimes acknowledge a message but forget to act.

2) The text of acknowledged messages should be displayed along with the call sign and type of acknowledgment. This would help the controller remember what message is being acknowledged.

3) The right/left selections on the turn menu should be optional. If not used, the computer will calculate "right" or "left" based on the shortest direction from the current heading to the assigned heading. There are situations, however, where the controller may wish to turn the aircraft in the longest direction for spacing and separation.

4) Controllers frequently ask pilots for abbreviated PIREP's. These include requests for ride conditions at a specified position or altitude, cloud tops, etc.
The test controller was impressed with the functionality of the device. He liked the idea of being able to key in and send messages while talking to other aircraft. One favorable aspect of the simulator was its ability to build a message with a minimum of keystrokes. He stated that with the ever increasing amount of traffic and a predicted shortage of controllers, a device such as this would be of great value. The most important result of the simulation was that the methodology was found to be easy to learn and use.
CONCLUSION

This prototype device, if implemented, would reduce the workload of air traffic controllers and enhance the safety of air traffic. It would provide a mechanism for controllers to communicate with more than one aircraft at a time. It would ensure that tactical air traffic messages are delivered error free in a timely manner and to the intended aircraft. It would provide pilots with easily understood messages and eliminate the confusion that sometimes occurs in ATC communications.

Air traffic controllers interviewed are in agreement that the device would benefit them. The enroute controllers stated that they would benefit most from the device's ability to automatically update the computer flight data. Both terminal and enroute controllers stated that they would benefit from the ability to communicate with multiple aircraft simultaneously. Three enroute controllers and one terminal controller were given a description of the device. They all had mostly positive remarks. The only exception was the terminal controller who is not required to key data and indicated that he can usually talk fast enough. But
even he agreed that it would be expedient to have the ability to send messages via a data link when the air traffic situation gets congested.

When the air traffic situation gets congested, a digital communications system will be an excellent tool to help controllers keep track of the tactical messages they are giving to pilots. It will provide a mechanism for them to quickly view the last few messages and ascertain if they have been acknowledged. It will also give them the ability to retransmit a message without having to rekey or say it.

This methodology and design will greatly alleviate the controller’s voice communication workload. It will not replace voice communication, but it will ensure that messages get sent to the intended aircraft expeditiously and correctly.
RECOMMENDATIONS

To fully evaluate the feasibility of this device and the methodology, a fully functioning prototype should be built next. It should be designed to attach to an air traffic controller's training simulator. These simulators are set up in each enroute and terminal facility for ongoing training. Some method would need to be developed to simulate the pilot's side of the digital communications system on the training simulator.

The pilot's side of the digital communication system is another area that can investigated in depth. A prototype of the pilot's device could be developed and used in conjunction with the controller's prototype in the training simulator. This arrangement would be ideal to fully develop the algorithms and hardware to implement a tactical air traffic control digital communication system.
REFERENCES


[Leeper79]

[Orlando89]
APPENDIX A

Controller Phrases

This appendix is a list of phrases that could be used by controllers in the radar environment. Data in braces "<>" is variable data selected from menus or keyed in by the controller. Data in brackets "[]" is optional. The "id" is the aircraft’s call sign.

Alert

1) <id> LOW ALTITUDE ALERT, CHECK YOUR ALTITUDE IMMEDIATELY.
   (THE <MEA | MVA | MOCA | MIA> IN YOUR AREA IS <alt>) |
   (THE <MDA | DH> IS <alt>)

2) <id> TRAFFIC ALERT <position of traffic>
   (ADVISE YOU TURN <RIGHT | LEFT> <hdg>) |
   (<CLIMB | DESCEND> <alt>) IMMEDIATELY

Communication

1) <id> CONTACT <facility> <freq> [AT <time | fix | alt>]

2) <id> CHANGE TO MY FREQUENCY <freq>

3) <id> REMAIN THIS FREQUENCY

4) <id> CHANGE TO ADVISORY FREQUENCY APPROVED

5) <id> FREQUENCY CHANGE APPROVED

Routing Clearances

1) <id> APPROVED AS REQUESTED

2) <id> <operation> APPROVED

3) <id> UNABLE <operation> [<reason> | <additional instructions>]

4) <id> CHANGE <portion of route> TO READ <amended route>

5) <id> CLEARED <amended route> REST OF ROUTE UNCHANGED

6) <id> CLEARED DIRECT <fix>
TRAFFIC

1) TRAFFIC, <number> O’CLOCK, <number> MILES,
   [<direction> BOUND, <relative mvnt>, <type aircraft>,
   (<alt> | ALTITUDE UNKNOWN)]

2) TRAFFIC, NUMEROUS TARGETS VICINITY OF <location>

3) TRAFFIC NO LONGER A FACTOR

4) <number> O’CLOCK TRAFFIC NO LONGER A FACTOR

5) <direction> TRAFFIC NO LONGER A FACTOR

WEATHER

1) ATTENTION ALL AIRCRAFT, (SIGMET | CONVECTIVE SIGMET | CENTER
   WEATHER ADVISORY), <ident> <description>

2) WEATHER AREA BETWEEN <number> O’CLOCK AND <number>
   O’CLOCK <number> MILES

3) REQUEST FLIGHT CONDITIONS
   [(AT PRESENT POSITION) |
   (OVER <fix>) |
   (ALONG PRESENT ROUTE) |
   (BETWEEN <fix> AND <fix>)]

4) <pirep data>

5) MONITOR HIWAS <freq>

ALTIMETER

1) THE <facility> [<time>] ALTIMETER <setting>

ALTITUDE

1) MAINTAIN <alt>

2) MAINTAIN <alt> UNTIL
   (<time>) |
   (PAST <fix>) |
   (<number> (MILES | MINUTES) PAST <fix>)

3) CROSS <fix>
   (AT <alt>) |
   (AT OR (ABOVE | BELOW) <alt>)
4) <id> INTERCEPT <route>
   (AT <alt>) |  
   (AT OR (ABOVE | BELOW) <alt>)

5) <id> (CLIMB | DESCEND) AND MAINTAIN <alt>
   [(AFTER PASSING <fix>) | (AT <time>)]

6) <id> (CLIMB | DESCEND) AND MAINTAIN <alt> WHEN ESTABLISHED AT
   LEAST <number> (MILES | MINUTES) PAST <fix> ON THE <navaid>
   <number> RADIAL

7) <id> (CLIMB | DESCEND) TO REACH <alt>
   (AT <time | fix>) |  
   (AT A POINT <number> MILES <direction> OF <navaid>)

8) <id> (CLIMB | DESCEND) AT PILOTS DISCRETION

9) <id> MAINTAIN <alt> THROUGH <alt>

10) <id> EXPECT (CLIMB | DESCENT) CLEARANCE
    (IN <number> (MILES | MINUTES)) |  
    (AT <fix>)

11) <id> REQUEST ALTITUDE CHANGE FROM <facility>
    [AT <time | fix | alt]}

12) <id> VERIFY AT <alt>

13) <id> VERIFY ASSIGNED ALTITUDE <alt>

14) <id> AFFIRMATIVE <alt>

15) <id> NEGATIVE,
    ((CLIMB | DESCEND) AND MAINTAIN <alt>) |  
    (MAINTAIN <alt>)

16) <id> SAY ALTITUDE

17) <id> VERIFY ALTITUDE AND ALTIMETER SETTING

18) <id> STOP ALTITUDE SQUAWK. ALTITUDE DIFFERS BY <number> FEET

HOLDING

1) <id> EXPECT FURTHER CLEARANCE VIA <route>

2) <id> CLEARED TO <fix>, (HOLD <direction> AS PUBLISHED) |  
   (NO DELAY EXPECTED)
3) <id> EXPECT FURTHER CLEARANCE <time>
   [ANTICIPATE ADDITIONAL <number> (MINUTE | HOUR)
   ((DELAY AT <fix>) |)
   (ENROUTE DELAY) |
   (TERMINAL DELAY))]

4) <id> HOLD <direction> OF <fix> ON <radial | course | bearing |
   azimuth | airway | route>
   [<number> (MINUTE | MILE) LEG], [(LEFT | RIGHT) TURNS]

ARRIVALS
1) <id> EXPECT <type> APPROACH TO RUNWAY <number>
2) <id> CLEARED FOR <type> [RUNWAY <number>] APPROACH
3) <id> CLEARED FOR STRAIGHT-IN <type> [RUNWAY <number>] APPROACH
4) <id> CLEARED FOR APPROACH
5) <id> <approach clearance> CIRCLE TO LAND RUNWAY <number>
6) <id> EXPECT VECTORS ACROSS FINAL APPROACH COURSE FOR SPACING
7) <id> POSITION <number> MILES FROM <fix> TURN (RIGHT | LEFT)
   HEADING <number>. MAINTAIN <alt> UNTIL ESTABLISHED ON THE
   LOCALIZER. CLEARED FOR ILS RUNWAY <number> APPROACH.
   [ILS RUNWAY <number> LOCALIZER FREQUENCY IS <number>]
8) <id> YOU HAVE CROSSED THE FINAL APPROACH COURSE. TURN (RIGHT |
   LEFT) IMMEDIATELY AND RETURN TO THE LOCALIZER COURSE.

RADAR
1) <id> (OVER | PASSING) <fix>
2) <id> <number> MILES FROM <fix>
3) <id> <number> MILES <direction> OF <fix | airway | location>
4) <id> (CROSSING | JOINING | DEPARTING) <airway | route>
5) <id> (INTERCEPTING | CROSSING) <navaid> <number> RADIAL
6) <id> RADAR SERVICE TERMINATED
7) <id> RADAR CONTACT [<position>]

- 70 -
8) <id> RADAR CONTACT LOST [<position>]

VECTORING

1) <id> TURN (RIGHT | LEFT) HEADING <number>
   [(FOR VECTOR TO <fix | airway>) |
    (FOR VECTOR TO INTERCEPT <navaid> <number> RADIAL) |
    (VECTOR FOR SPACING) |
    (FOR VECTOR TO [<approach name>] FINAL APPROACH COURSE)]

2) <id> FLY HEADING <number>
   [(FOR VECTOR TO <fix | airway>) |
    (FOR VECTOR TO INTERCEPT <navaid> <number> RADIAL) |
    (VECTOR FOR SPACING) |
    (FOR VECTOR TO [<approach name>] FINAL APPROACH COURSE)]

3) <id> FLY PRESENT HEADING
   [(FOR VECTOR TO <fix | airway>) |
    (FOR VECTOR TO INTERCEPT <navaid> <number> RADIAL) |
    (VECTOR FOR SPACING) |
    (FOR VECTOR TO [<approach name>] FINAL APPROACH COURSE)]

4) <id> DEPART <fix> HEADING <number>
   [(FOR VECTOR TO <fix | airway>) |
    (FOR VECTOR TO INTERCEPT <navaid> <number> RADIAL) |
    (VECTOR FOR SPACING) |
    (FOR VECTOR TO [<approach name>] FINAL APPROACH COURSE)]

5) <id> TURN <number> DEGREES (RIGHT | LEFT)
   [(FOR VECTOR TO <fix | airway>) |
    (FOR VECTOR TO INTERCEPT <navaid> <number> RADIAL) |
    (VECTOR FOR SPACING) |
    (FOR VECTOR TO [<approach name>] FINAL APPROACH COURSE)]

6) <id> RESUME OWN NAVIGATION

7) <id> FLY HEADING <number>. WHEN ABLE PROCEED DIRECT <fix>

SPEED

1) <id> SAY AIRSPEED

2) <id> SAY MACH NUMBER

3) <id> (INCREASE | REDUCE) SPEED
   ((TO <number>) |
    (TO MACH <number>) |
    (<number> KNOTS))
4) <id> DO NOT EXCEED (<number> KNOTS) | (MACH <number>))

5) <id> IF PRACTICAL, (REDUCE | INCREASE) SPEED
   ((TO <number>) | 
   (TO MACH <number>) | 
   (<number> KNOTS))

6) <id> REDUCE SPEED ((TO <number>) | (<number> KNOTS)), THEN
   DESCEND AND MAINTAIN <alt>

7) <id> DESCEND AND MAINTAIN <alt>, THEN REDUCE SPEED
   ((TO <number>) | 
   (TO MACH <number>) | 
   (<number> KNOTS))

8) <id> CROSS <fix> AT AND MAINTAIN <alt> AT <number> KNOTS

9) <id> IF PRACTICAL, MAINTAIN <number> KNOTS

10) <id> RESUME NORMAL SPEED
APPENDIX B

Menu Hierarchy

This appendix shows the hierarchy of the menu system followed by each menu in the system. The screens are numbered for easy reference from the hierarchy diagrams.

List of Screens

<table>
<thead>
<tr>
<th>Number</th>
<th>Screen</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Main Menu</td>
<td>88</td>
</tr>
<tr>
<td>2</td>
<td>O’Clock Menu</td>
<td>89</td>
</tr>
<tr>
<td>3</td>
<td>Direction Menu</td>
<td>90</td>
</tr>
<tr>
<td>4</td>
<td>Enter Data Screen</td>
<td>91</td>
</tr>
<tr>
<td>5</td>
<td>NAVAID Menu</td>
<td>92</td>
</tr>
<tr>
<td>6</td>
<td>Altitude Options Menu</td>
<td>93</td>
</tr>
<tr>
<td>7</td>
<td>Fix Menu</td>
<td>94</td>
</tr>
<tr>
<td>8</td>
<td>Minutes Menu</td>
<td>95</td>
</tr>
<tr>
<td>9</td>
<td>Miles Menu</td>
<td>96</td>
</tr>
<tr>
<td>10</td>
<td>Airway Menu</td>
<td>97</td>
</tr>
<tr>
<td>11</td>
<td>Facility Menu</td>
<td>98</td>
</tr>
<tr>
<td>12</td>
<td>Location Menu</td>
<td>99</td>
</tr>
<tr>
<td>13</td>
<td>Routing Menu</td>
<td>100</td>
</tr>
<tr>
<td>14</td>
<td>Routing Options Menu</td>
<td>101</td>
</tr>
<tr>
<td>15</td>
<td>Communication Menu</td>
<td>102</td>
</tr>
<tr>
<td>16</td>
<td>Communication Menu When Options</td>
<td>103</td>
</tr>
<tr>
<td>17</td>
<td>Alert Menu</td>
<td>104</td>
</tr>
<tr>
<td>18</td>
<td>Altitude Alert Options Menu</td>
<td>105</td>
</tr>
<tr>
<td>19</td>
<td>Traffic Alert Options Menu</td>
<td>106</td>
</tr>
<tr>
<td>20</td>
<td>Traffic Menu</td>
<td>107</td>
</tr>
<tr>
<td>21</td>
<td>Traffic Menu Bound/Movement Options</td>
<td>108</td>
</tr>
<tr>
<td>22</td>
<td>Traffic Menu Aircraft Type Options</td>
<td>109</td>
</tr>
<tr>
<td>23</td>
<td>Traffic Menu Relative Altitude Options</td>
<td>110</td>
</tr>
<tr>
<td>24</td>
<td>Weather Menu</td>
<td>111</td>
</tr>
<tr>
<td>25</td>
<td>Weather Menu Solicit PIREP Options</td>
<td>112</td>
</tr>
<tr>
<td>26</td>
<td>Weather Menu Send PIREP Options</td>
<td>113</td>
</tr>
<tr>
<td>27</td>
<td>Altimeter Menu</td>
<td>114</td>
</tr>
<tr>
<td>28</td>
<td>Altimeter Menu Time Option</td>
<td>115</td>
</tr>
<tr>
<td>29</td>
<td>Altitude Menu</td>
<td>116</td>
</tr>
<tr>
<td>30</td>
<td>Altitude Menu Feet Options</td>
<td>117</td>
</tr>
<tr>
<td>31</td>
<td>Negative Altitude Options</td>
<td>118</td>
</tr>
<tr>
<td>32</td>
<td>Request Altitude Change From Options</td>
<td>119</td>
</tr>
<tr>
<td>33</td>
<td>Expect Higher/Lower Options</td>
<td>120</td>
</tr>
<tr>
<td>34</td>
<td>Expect Higher/Lower When Options</td>
<td>121</td>
</tr>
<tr>
<td>Number</td>
<td>Screen</td>
<td>page</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>35</td>
<td>Maintain Block Option</td>
<td>122</td>
</tr>
<tr>
<td>36</td>
<td>Climb/Descend to Reach Option Menu</td>
<td>123</td>
</tr>
<tr>
<td>37</td>
<td>Climb/Descend When at Least Option Menu</td>
<td>124</td>
</tr>
<tr>
<td>38</td>
<td>Climb/Descend After Options Menu</td>
<td>125</td>
</tr>
<tr>
<td>39</td>
<td>Cross/Intercept at Options Menu</td>
<td>126</td>
</tr>
<tr>
<td>40</td>
<td>Climb/Descend Options Menu</td>
<td>127</td>
</tr>
<tr>
<td>41</td>
<td>Maintain Until Options Menu</td>
<td>128</td>
</tr>
<tr>
<td>42</td>
<td>Holding Menu</td>
<td>129</td>
</tr>
<tr>
<td>43</td>
<td>Holding Instructions Place Options Menu</td>
<td>130</td>
</tr>
<tr>
<td>44</td>
<td>Holding Instructions Leg Options Menu</td>
<td>131</td>
</tr>
<tr>
<td>45</td>
<td>Expect Further Options Menu</td>
<td>132</td>
</tr>
<tr>
<td>46</td>
<td>Where Delay Options Menu</td>
<td>133</td>
</tr>
<tr>
<td>47</td>
<td>Arrivals Menu</td>
<td>134</td>
</tr>
<tr>
<td>48</td>
<td>Left/Right Options</td>
<td>135</td>
</tr>
<tr>
<td>49</td>
<td>Type Approach Options Menu</td>
<td>136</td>
</tr>
<tr>
<td>50</td>
<td>Runway Options Menu</td>
<td>137</td>
</tr>
<tr>
<td>51</td>
<td>ILS Instructions Gate Menu</td>
<td>138</td>
</tr>
<tr>
<td>52</td>
<td>Circle Option</td>
<td>139</td>
</tr>
<tr>
<td>53</td>
<td>Radar Menu</td>
<td>140</td>
</tr>
<tr>
<td>54</td>
<td>Radar Options Menu</td>
<td>141</td>
</tr>
<tr>
<td>55</td>
<td>Vectoring Menu</td>
<td>142</td>
</tr>
<tr>
<td>56</td>
<td>Vectoring Purpose Options Menu</td>
<td>143</td>
</tr>
<tr>
<td>57</td>
<td>Speed Menu</td>
<td>144</td>
</tr>
<tr>
<td>58</td>
<td>Speed Increase/Decrease Options Menu</td>
<td>145</td>
</tr>
<tr>
<td>59</td>
<td>RE-XMIT Menu</td>
<td>146</td>
</tr>
<tr>
<td>60</td>
<td>AC List</td>
<td>147</td>
</tr>
<tr>
<td>61</td>
<td>Incoming Message Display</td>
<td>148</td>
</tr>
<tr>
<td>62</td>
<td>Message Acknowledged Display</td>
<td>149</td>
</tr>
<tr>
<td>63</td>
<td>Other Functions Menu</td>
<td>150</td>
</tr>
</tbody>
</table>
Figure 9: Main Menu Hierarchy
Figure 10: Vectoring Menu Hierarchy
Figure 11: Altitude Menu Hierarchy
Figure 12: Speed Menu Hierarchy
Figure 13: Holding Menu Hierarchy
Figure 15: Arrivals Menu Hierarchy
Figure 16: Altimeter Menu Hierarchy
Figure 17: Traffic Menu Hierarchy
Figure 19: Alert Menu Hierarchy
Figure 20: Radar Menu Hierarchy
Figure 21: Weather Menu Hierarchy
Screen 1: Main Menu
<table>
<thead>
<tr>
<th></th>
<th></th>
<th>12 O’CLOCK</th>
<th></th>
<th></th>
<th>1 O’CLOCK</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>F2</td>
<td>11 O’CLOCK</td>
<td>F3</td>
<td>F4</td>
<td>2 O’CLOCK</td>
</tr>
<tr>
<td>F5</td>
<td>F6</td>
<td>10 O’CLOCK</td>
<td>F7</td>
<td>F8</td>
<td>3 O’CLOCK</td>
</tr>
<tr>
<td>F9</td>
<td>F10</td>
<td>9 O’CLOCK</td>
<td>F11</td>
<td>F12</td>
<td>4 O’CLOCK</td>
</tr>
<tr>
<td></td>
<td>F13</td>
<td>8 O’CLOCK</td>
<td>F14</td>
<td>F15</td>
<td>5 O’CLOCK</td>
</tr>
<tr>
<td></td>
<td>F16</td>
<td>7 O’CLOCK</td>
<td>F17</td>
<td>F18</td>
<td>6 O’CLOCK</td>
</tr>
</tbody>
</table>

Screen 2: O’Clock Menu
Screen 3: Direction Menu
Screen 4: Enter Data Screen
Screen 5: NAVAID Menu
(programmable)
Screen 6: Altitude Options Menu
(programmable)
Screen 7: Fix Menu
(programmable)
Screen 8: Minutes Menu
<table>
<thead>
<tr>
<th>F1</th>
<th>50 MILES</th>
<th>45 MILES</th>
<th>F10</th>
</tr>
</thead>
<tbody>
<tr>
<td>F2</td>
<td>60 MILES</td>
<td>40 MILES</td>
<td>F11</td>
</tr>
<tr>
<td>F3</td>
<td>70 MILES</td>
<td>35 MILES</td>
<td>F12</td>
</tr>
<tr>
<td>F4</td>
<td>80 MILES</td>
<td>30 MILES</td>
<td>F13</td>
</tr>
<tr>
<td>F5</td>
<td>100 MILES</td>
<td>25 MILES</td>
<td>F14</td>
</tr>
<tr>
<td>F6</td>
<td>4 MILES</td>
<td>20 MILES</td>
<td>F15</td>
</tr>
<tr>
<td>F7</td>
<td>3 MILES</td>
<td>15 MILES</td>
<td>F16</td>
</tr>
<tr>
<td>F8</td>
<td>2 MILES</td>
<td>10 MILES</td>
<td>F17</td>
</tr>
<tr>
<td>F9</td>
<td>1 MILE</td>
<td>5 MILES</td>
<td>F18</td>
</tr>
</tbody>
</table>

Screen 9: Miles Menu
Screen 10: Airway Menu
(programmable)
| F1 | JAX CENTER 118.7 | JAX APPROACH 121.3 |
| F2 | JAX CENTER 123.5 | JAX APPROACH 122.7 |
| F3 | JAX CENTER 125.9 | DAYTONA APPRCH 124.5 |
| F4 | JAX CENTER 125.1 | DAYTONA APPRCH 127.5 |
| F5 | ATLANTA CENTER 117.6 | MACON APPROACH 125.3 |
| F6 | ATLANTA CENTER 118.3 | TALAHASSEE APCH 132.5 |
| F7 | MIAMI CENTER 109.7 | SAVANAH APPROCH 131.2 |
| F8 | MIAMI CENTER 109.3 | |
| F9 | |

Screen 11: Facility Menu
(programmable)
Screen 12: Location Menu
(programmable)
Screen 13: Routing Menu
Screen 14: Routing Options Menu
Screen 15: Communication Menu
Screen 16: Communication Menu When Options
Screen 17: Alert Menu
Screen 18: Altitude Alert Options Menu
Screen 19: Traffic Alert Options Menu
Screen 20: Traffic Menu
Screen 21: Traffic Menu Bound/Movement Options

<table>
<thead>
<tr>
<th>F1</th>
<th>CLIMBING</th>
<th>MANEUVERING</th>
<th>F10</th>
</tr>
</thead>
<tbody>
<tr>
<td>F2</td>
<td>LEVEL</td>
<td>OPPOSITE DIR</td>
<td>F11</td>
</tr>
<tr>
<td>F3</td>
<td>DESCENDING</td>
<td>CONVERGING</td>
<td>F12</td>
</tr>
<tr>
<td>F4</td>
<td>NORTH BOUND</td>
<td></td>
<td>F13</td>
</tr>
<tr>
<td>F5</td>
<td>NORTHWEST BOUND</td>
<td>NORTHEAST BOUND</td>
<td>F14</td>
</tr>
<tr>
<td>F6</td>
<td>WEST BOUND</td>
<td>EAST BOUND</td>
<td>F15</td>
</tr>
<tr>
<td>F7</td>
<td>SOUTHWEST BOUND</td>
<td>SOUTHEAST BOUND</td>
<td>F16</td>
</tr>
<tr>
<td>F8</td>
<td>DIRECTION UNKNOWN</td>
<td>SOUTH BOUND</td>
<td>F17</td>
</tr>
<tr>
<td>F9</td>
<td></td>
<td></td>
<td>F18</td>
</tr>
<tr>
<td>F1</td>
<td>HELICOPTER</td>
<td>767</td>
<td>F10</td>
</tr>
<tr>
<td>-----</td>
<td>------------</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>F2</td>
<td>CESSNA</td>
<td>757</td>
<td>F11</td>
</tr>
<tr>
<td>F3</td>
<td>BEACHCRAFT</td>
<td>747</td>
<td>F12</td>
</tr>
<tr>
<td>F4</td>
<td>LEAR</td>
<td>737</td>
<td>F13</td>
</tr>
<tr>
<td>F5</td>
<td>CITATION</td>
<td>727</td>
<td>F14</td>
</tr>
<tr>
<td>F6</td>
<td>L1011</td>
<td>707</td>
<td>F15</td>
</tr>
<tr>
<td>F7</td>
<td>A300</td>
<td>DC10</td>
<td>F16</td>
</tr>
<tr>
<td>F8</td>
<td>DH125</td>
<td>DC9</td>
<td>F17</td>
</tr>
<tr>
<td>F9</td>
<td>MD80</td>
<td>TYPE UNKNOWN</td>
<td>F18</td>
</tr>
</tbody>
</table>

Screen 22: Traffic Menu Aircraft Type Options (programmable)
Screen 23: Traffic Menu Relative Altitude Options
Screen 24: Weather Menu
Screen 25: Weather Menu Solicit PIREP Options
727 reported moderate chop at FL370

767 reported rime icing over JAX VOR at 170 at 0130Z
Screen 27: Altimeter Menu
(programmable)
Screen 28: Altimeter Menu Time Option
<table>
<thead>
<tr>
<th>F1</th>
<th>VERIFY ALT &amp; ALTIMETER</th>
<th>STOP ALT SQUAWK</th>
<th>F10</th>
</tr>
</thead>
<tbody>
<tr>
<td>F2</td>
<td>VERIFY ASSIGNED ALT</td>
<td>AFFIRMATIVE ALT</td>
<td>F11</td>
</tr>
<tr>
<td>F3</td>
<td>VERIFY AT ALT</td>
<td>NEGATIVE ALT</td>
<td>F12</td>
</tr>
<tr>
<td>F4</td>
<td>REQUEST ALT CHG FROM</td>
<td>SAY ALT</td>
<td>F13</td>
</tr>
<tr>
<td>F5</td>
<td>EXPECT HIGHER/LOWER IN/AT</td>
<td>INTERCEPT AT ALT</td>
<td>F14</td>
</tr>
<tr>
<td>F6</td>
<td>MAINTAIN BLOCK</td>
<td>CROSS FIX AT ALT</td>
<td>F15</td>
</tr>
<tr>
<td>F7</td>
<td>MAINTAIN ALT</td>
<td>MAINTAIN UNTIL</td>
<td>F16</td>
</tr>
<tr>
<td>F8</td>
<td>CLIMB/DESCEND WHEN AT LEASE</td>
<td>CLIMB/DESCEND TO REACH</td>
<td>F17</td>
</tr>
<tr>
<td>F9</td>
<td>AT PILOT’S DISCRETION</td>
<td></td>
<td>F18</td>
</tr>
</tbody>
</table>

Screen 29: Altitude Menu
Screen 30: Altitude Menu Feet Options
Screen 31: Negative Altitude Options
Screen 32: Request Altitude Change From Options
Screen 33: Expect Higher/Lower Options
Screen 34: Expect Higher/Lower When Options
Screen 35: Maintain Block Option
Screen 36: Climb/Descend to Reach Options Menu
Screen 37: Climb/Descend When at Least Options Menu
Screen 38: Climb/Descend After Options Menu
Screen 39: Cross/Intercept at Options Menu
Screen 40: Climb.Descend Options Menu
Screen 41: Maintain Until Options Menu
Screen 42: Holding Menu
Screen 43: Holding Instructions Place Options Menu
Screen 44: Holding Instructions Leg Options Menu
Screen 45: Expect Further Options Menu
Screen 46: Where Delay Options Menu
Screen 47: Arrivals Menu
Screen 48: Left/Right Options
Screen 49: Type Approach Options Menu
Screen 50: Runway Options Menu (programmable)
Screen 51: ILS Instructions Gate Menu
(programmable)
Screen 52: Circle Option
<table>
<thead>
<tr>
<th>F1</th>
<th>INTERCEPTING RADIAL</th>
<th>F10</th>
</tr>
</thead>
<tbody>
<tr>
<td>F2</td>
<td>CROSSING RADIAL</td>
<td>F11</td>
</tr>
<tr>
<td>F3</td>
<td>MILES FROM FIX OVER FIX</td>
<td>F12</td>
</tr>
<tr>
<td>F4</td>
<td>PASSING FIX</td>
<td>F13</td>
</tr>
<tr>
<td>F5</td>
<td></td>
<td>F14</td>
</tr>
<tr>
<td>F6</td>
<td></td>
<td>F15</td>
</tr>
<tr>
<td>F7</td>
<td></td>
<td>F16</td>
</tr>
<tr>
<td>F8</td>
<td>RADAR CONTACT LOST</td>
<td>F17</td>
</tr>
<tr>
<td>F9</td>
<td>RADAR CONTACT TERMINATED</td>
<td>F18</td>
</tr>
</tbody>
</table>

Screen 53: Radar Menu
Screen 55: Radar Options Menu
Screen 55: Vectoring Menu
Screen 56: Vectoring Purpose Options Menu
Screen 57: Speed Menu

- SAY AIRSPEED
- SAY MACH
- CROSS FIX AT ALT AND SPEED
- DESCEND THEN REDUCE SPEED
- REDUCE SPEED THEN DESCEND
- IF PRACTICAL REDUCE SPEED
- REDUCE SPEED

- RESUME NORMAL SPEED
- DO NOT EXCEED KNOTS
- DO NOT EXCEED MACH
- IF PRACTICAL MAINTAIN SPEED
- IF PRACTICAL INCREASE SPEED
- INCREASE SPEED
Screen 58: Speed Increase/Decrease Options Menu
I UNITED 237 REQUEST HIGHER

A+ NORTHWEST 856 TRAFFIC 12 O'CL OCK 3MI TP UNK

A- DELTA 252 TRAFFIC 3 O'CLOCK 10 MI S BND 727 FL240

A UNITED 1251 CTC JAX CTR 125.6

O CESSNA N5432X CLRD DIR JAX

A DELTA 252 TURN R 270

O UNITED 237 DESC FL270

PRESS F KEY NEXT TO MESSAGE TO RE-XMIT IT

Screen 59: RE-XMIT Menu
<table>
<thead>
<tr>
<th>Destination</th>
<th>Mode</th>
<th>CID</th>
<th>Altitude (Feet)</th>
<th>Heading (Degrees)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORTHWEST 856</td>
<td>ATL</td>
<td>L</td>
<td>125</td>
<td>008</td>
</tr>
<tr>
<td>UNITED 237</td>
<td>JFK</td>
<td>L</td>
<td>318</td>
<td>210</td>
</tr>
<tr>
<td>DELTA 252</td>
<td>ATL</td>
<td>M</td>
<td>271</td>
<td>270</td>
</tr>
<tr>
<td>UNITED 1251</td>
<td>JAX</td>
<td>C</td>
<td>146</td>
<td>240</td>
</tr>
<tr>
<td>CESSNA 5432X</td>
<td>CRG</td>
<td>D</td>
<td>235</td>
<td>170</td>
</tr>
</tbody>
</table>

**Screen 60: AC List**
**INCOMING MESSAGE**

CESSNA 5432X REQUEST LOWER

PRESS ANY KEY TO GO BACK TO MAIN MENU

Screen 61: Incoming Message Display
Screen 62: Message Acknowledged Display
Screen 63: Other Functions Menu
This pseudo code represents how the logic of the software will basically flow. Note that whenever the user is required to press a key (WAIT FOR KEYPRESS), the following can occur, the logic of which is omitted: If the CLEAR key is pressed, control branches back to the main loop and the aircraft ID is erased. If the PREV MENU is pressed, the last displayed menu is restored and control will branch back to that point.

PROGRAM DEVICE
BEGIN
  INITIALIZE DATA STRUCTURES
  LOOP
    DISPLAY MAIN MENU
    CALL POLLHOST
    IF KEYPRESSED
      CALL PROCESSINPUT
    END IF
  END LOOP
END

SUBROUTINE POLLHOST
BEGIN
  SEND POLL REQUEST
  IF NO RESPONSE IN .5 SECONDS
    SIGNAL POSSIBLE DISCONNECT
  END IF
  IF RESPONSE = AC DATA
    STORE AC DATA
    DISPLAY CALL SIGN
    RETURN
  END IF
  IF RESPONSE = MSG ACK
    SEARCH FOR CID
    IF FOUND
      IF POSITIVE ACK
        FLAG MSG WITH "A+"
      ELSE
        IF NEGATIVE ACK
          FLAG MSG WITH "A-"
        ELSE
          FLAG MSG WITH "A"
        END IF
      END IF
  END IF
END IF
  DISPLAY ACK MSG
ELSE
  DISPLAY NO MSG PENDING ACK
END IF
SOUND BUZZER
WAIT FOR KEYPRESS
RETURN
END IF
IF RESPONSE = AC MSG
  SEARCH FOR MSG TYPE
  IF FOUND
    DISPLAY MSG
    STORE MSG
  ELSE
    DISPLAY ERROR MSG
END IF
SOUND BUZZER
WAIT FOR KEYPRESS
RETURN
END IF
IF RESPONSE = MEMORY LOAD
  DISPLAY MENU
  SOUND BUZZER
  IF MEM LOCATION INVALID
    DISPLAY INVALID MSG
    WAIT FOR KEYPRESS
    RETURN
  ELSE
    DISPLAY READY TO LOAD MSG
    WHILE KEY NOT = XMIT OR CLEAR
      WAIT FOR KEYPRESS
      IF KEY = XMIT
        LOAD NEW MENU
      ELSE
        IF KEY = CLEAR
          RETURN
        ELSE
          SOUND BUZZER
        END IF
      END IF
    END WHILE
  END IF
END IF
END POLLHOST

SUBROUTINE PROCESSINPUT
BEGIN
  CASE KEY
    VALUE = NUMERIC
      CALL PROCESSNUMBERS
    VALUE = F1
      CALL F1KEY
VALUE = F2
CALL F2KEY
VALUE = F3
CALL F3KEY
VALUE = F4
CALL F4KEY
VALUE = F5
CALL F5KEY
VALUE = F6
CALL F6KEY
VALUE = F7
CALL F7KEY
VALUE = F8
CALL F8KEY
VALUE = F9
CALL F9KEY
VALUE = F10
CALL F10KEY
VALUE = F11
CALL F11KEY
VALUE = F12
CALL F12KEY
VALUE = F13
CALL F13KEY
VALUE = F14
CALL F14KEY
VALUE = F15
CALL F15KEY
VALUE = F16
CALL F16KEY
VALUE = F17
CALL F17KEY
VALUE = F18
CALL F18KEY
OTHERWISE
SOUND BUZZER
END CASE
END PROCESSINPUT

SUBROUTINE PROCESSNUMBERS
BEGIN
GET SECOND AND THIRD DIGITS FROM KEYBOARD
IF CLEAR IS PRESSED
RETURN
END IF
SEND CID REQUEST TO HOST
IF RETURNED CID = "000"
DISPLAY INVALID CID MSG
ELSE
DISPLAY CALL SIGN
STORE AC DATA
ENDIF
RETURN
SUBROUTINE F1KEY (VECTORING)
BEGIN
  IF AC ID NOT STORED
    SOUND BUZZER
    DISPLAY "NO AC IDENTIFIED"
    WAIT FOR KEYPRESS
    RETURN
  END IF
  DISPLAY VECTOR MENU
  WAIT FOR KEYPRESS
  IF CLEAR PRESSED
    CLEAR AC DATA
    RETURN
  END IF
  IF PREV MENU PRESSED
    RETURN
  END IF
  IF F6 PRESSED
    (MSG 7: FLY HDG, PROCEED DIRECT)
    DISPLAY "FLY HDG"
    GET 3 DIGITS AND DISPLAY THEM
    DISPLAY " WHEN ABLE PROCEED DIRECT "
    DISPLAY FIX MENU
    WAIT FOR KEYPRESS
    DISPLAY FIX
    WAIT FOR KEYPRESS
    IF XMIT PRESSED
      BUILD MESSAGE
      SEND MSG TO HOST
    END IF
    CLEAR AC DATA
    RETURN
  ELSE
    IF F14 PRESSED
      (MSG 6: RESUME OWN NAVIGATION)
      DISPLAY "RESUME OWN NAV"
      WAIT FOR KEYPRESS
      IF XMIT PRESSED
        BUILD MESSAGE
        SEND MSG TO HOST
      END IF
      CLEAR AC DATA
      RETURN
    ELSE
      IF F7 PRESSED
        (MSG 2: FLY HDG)
        DISPLAY "FLY HDG "
        GET 3 DIGITS AND DISPLAY THEM
      ELSE
        IF F8 PRESSED
          (MSG 5: TURN DEGREES LEFT)
          DISPLAY "TURN L "
          GET 2 DIGITS AND DISPLAY THEM
          DISPLAY " DEGREES"
        ELSE
          ...
        END IF
      END IF
    END IF
  END IF
END
IF F9 PRESSED  (MSG 1: TURN LEFT TO HDG)
   DISPLAY "TURN L HDG"
   GET 3 DIGITS AND DISPLAY THEM
ELSE
   IF F15 PRESSED  (MSG 4: DEPART FIX ON HDG)
      DISPLAY "DEPART"
      DISPLAY FIX MENU
      WAIT FOR KEYPRESS
      DISPLAY FIX AND "HEADING"
   ELSE
      IF F16 PRESSED  (MSG 3: FLY PRESENT HDG)
         DISPLAY "FLY PRESENT HEADING"
      ELSE
         IF F17 PRESSED  (MSG 5: TURN DEGREES R)
            DISPLAY "TURN R"
            GET 2 DIGITS
            DISPLAY 'DEGREES"
         ELSE
            IF F18 PRESSED (MSG 1: TURN R TO HDG)
               DISPLAY "TURN R HDG"
               GET 3 DIGITS
            END IF
         END IF
      END IF
   END IF
END IF
DISPLAY PURPOSE MENU
WAIT FOR KEYPRESS
IF XMIT PRESSED
   BUILD MESSAGE
   SEND MSG TO HOST
   RETURN
ELSE
   IF F7 PRESSED  (FOR VECTOR TO APPROACH OPTION)
      DISPLAY "FOR VECTOR TO"
      DISPLAY APPROACH MENU
      WAIT FOR KEYPRESS
      IF XMIT PRESSED
         ADD "FINAL APPROACH COURSE" TO MSG
         BUILD MESSAGE
         SEND MSG TO HOST
         RETURN
      END IF
   END IF
   DISPLAY APPROACH NAME
   DISPLAY RUNWAY MENU
   WAIT FOR KEYPRESS
   DISPLAY RUNWAY
   ADD "FINAL APPROACH COURSE" TO MSG
   BUILD MESSAGE
ELSE

IF F8 PRESSED (FOR VECTOR TO FIX OPTION)
   DISPLAY "FOR VECTOR TO"
   DISPLAY FIX MENU
   WAIT FOR KEYPRESS
   DISPLAY FIX
   BUILD MESSAGE
ELSE
   IF F9 PRESSED (FOR VECTOR TO AIRWAY OPTION)
      DISPLAY "FOR VECTOR TO"
      DISPLAY AIRWAY MENU
      WAIT FOR KEYPRESS
      DISPLAY AIRWAY
      BUILD MESSAGE
   ELSE
      IF F17 PRESSED (VECTORING FOR SPACING OPTION)
         DISPLAY "VECTOR FOR SPACING"
         BUILD MESSAGE
      ELSE
      ELSE
      END IF
      END IF
   END IF
   END IF
END IF
END IF
END IF
END IF
WAIT FOR KEYPRESS
IF XMIT PRESSED
   SEND MSG TO HOST
END IF
CLEAR AC DATA
RETURN
END F1KEY

SUBROUTINE F2KEY (ALTITUDE)
BEGIN
   IF AC ID NOT STORED
      SOUND BUZZER
      DISPLAY "NO AC IDENTIFIED"
      WAIT FOR KEYPRESS
      RETURN
   END IF
   DISPLAY ALTITUDE MENU
   WAIT FOR KEYPRESS
   IF CLEAR PRESSED
      CLEAR AC DATA
      RETURN

- 156 -
END IF
IF PREV MENU PRESSED
  RETURN
END IF
IF F1 PRESSED  (MSG 17: VERIFY ALT AND ALTIMETER)
  DISPLAY "VERIFY ALT AND ALTIMETER"
  BUILD MESSAGE
END IF
IF F2 PRESSED  (MSG 13: VERIFY ASSIGNED ALT)
  DISPLAY "VERIFY ASSIGNED ALT"
  DISPLAY ALT OPTIONS MENU
  WAIT FOR KEYPRESS
  DISPLAY ALTITUDE
  BUILD MESSAGE
END IF
IF F3 PRESSED  (MSG 12: VERIFY AT ALT)
  DISPLAY "VERIFY AT ALT"
  DISPLAY ALTITUDE OPTIONS MENU
  WAIT FOR KEYPRESS
  DISPLAY ALTITUDE
  BUILD MESSAGE
END IF
IF F4 PRESSED  (MSG 11: REQUEST ALT CHG FROM)
  DISPLAY "REQUEST ALT CHANGE FROM"
  DISPLAY FACILITY MENU
  WAIT FOR KEYPRESS
  DISPLAY FACILITY
  DISPLAY REQ CHANGE FROM OPTIONS MENU
  WAIT FOR KEYPRESS
  IF XMIT PRESSED
    BUILD MESSAGE
  ELSE
    IF F9 PRESSED  (AT ALT OPTION)
      DISPLAY "AT"
      DISPLAY ALTITUDE OPTIONS MENU
      WAIT FOR KEYPRESS
      DISPLAY ALTITUDE
      BUILD MESSAGE
    ELSE
      IF F17 PRESSED  (AT TIME OPTION)
        DISPLAY "AT"
        GET AND DISPLAY TIME
        BUILD MESSAGE
      ELSE
        IF F18 PRESSED  (AT FIX OPTION)
          DISPLAY "AT"
          DISPLAY FIX MENU
          WAIT FOR KEYPRESS
          DISPLAY FIX
          BUILD MESSAGE
        END IF
      END IF
    END IF
  END IF
END IF
END IF
END IF
END IF
IF F5 PRESSED  (MSG 10: EXPECT HIGHER/LOWER IN/AT)
   DISPLAY "EXPECT"
   DISPLAY HIGHER/LOWER OPTIONS MENU
   WAIT FOR KEYPRESS
   IF F9 PRESSED
      DISPLAY "CLIMB"
   ELSE
      DISPLAY "DESCENT"
   END IF
   DISPLAY HIGHER/LOWER WHEN OPTIONS MENU
   WAIT FOR KEYPRESS
   IF F8 PRESSED  (AT FIX OPTION)
      DISPLAY "AT"
      DISPLAY FIX MENU
      WAIT FOR KEYPRESS
      DISPLAY FIX
      BUILD MESSAGE
   ELSE
      IF F9 PRESSED  (IN MILES OPTION)
         DISPLAY "IN"
         DISPLAY MILES MENU
         WAIT FOR KEYPRESS
         DISPLAY MILES AND "MILES"
         BUILD MESSAGE
      ELSE
         IF F18 PRESSED  (IN MINUTES OPTION)
            DISPLAY "IN"
            DISPLAY MINUTES MENU
            WAIT FOR KEYPRESS
            DISPLAY MINUTES AND "MINUTES"
            BUILD MESSAGE
         END IF
      END IF
   END IF
END IF
IF F6 PRESSED  (MSG 9: MAINTAIN BLOCK)
   DISPLAY "MAINTAIN BLOCK"
   DISPLAY ALTITUDE MENU
   WAIT FOR KEYPRESS
   DISPLAY ALTITUDE AND "THRU"
   DISPLAY ALTITUDE MENU
   WAIT FOR KEYPRESS
   DISPLAY ALTITUDE
   BUILD MESSAGE
END IF
IF F7 PRESSED  (MSG 1: MAINTAIN ALT)
   DISPLAY "MAINTAIN ALT"
   DISPLAY ALTITUDE MENU
   WAIT FOR KEYPRESS
   DISPLAY ALTITUDE
   BUILD MESSAGE
END IF
IF F8 PRESSED  (MSG 6: CLIMB/DESCEND WHEN AT LEAST)

IF AC’S ALT NOT STORED
   DISPLAY CLIMB/DESCEND OPTION MENU
   WAIT FOR KEYPRESS
   IF F17 PRESSED
      SET CLIMB MODE ON
   ELSE
      IF F18 PRESSED
         SET DESCEND MODE ON
   END IF
END IF
END IF
IF AC ALT > ALT OR DESCEND MODE ON
   DISPLAY "DESCEND AND MAINTAIN"
ELSE
   DISPLAY "CLIMB AND MAINTAIN"
END IF
DISPLAY ALTITUDE MENU
WAIT FOR KEYPRESS
DISPLAY ALTITUDE
DISPLAY "WHEN ESTABLISHED AT LEAST"
DISPLAY MILES/MINUTES OPTIONS MENU
WAIT FOR KEYPRESS
IF F9 PRESSED
   DISPLAY MILES MENU
   WAIT FOR KEYPRESS
   DISPLAY MILES "MILES PAST"
ELSE
   IF F18 PRESSED
      DISPLAY MINUTES MENU
      WAIT FOR KEYPRESS
      DISPLAY MINUTES "MINUTES PAST"
   END IF
END IF
DISPLAY FIX MENU
WAIT FOR KEYPRESS
DISPLAY FIX "ON THE"
DISPLAY NAVAID MENU
WAIT FOR KEYPRESS
DISPLAY NAVAID
GET AND DISPLAY 3 DIGIT RADIAL
DISPLAY "RADIAL"
BUILD MESSAGE
END IF
IF F9 PRESSED
   (MSG 8: AT PILOT’S DISCRETION)
   TURN PILOT’S DISCRETION FLAG ON
END IF
IF F10 PRESSED
   (MSG 18: STOP ALT SQUAWK)
   DISPLAY "STOP ALT SQUAWK, ALT DIFFERS BY"
   DISPLAY FEET MENU
   WAIT FOR KEYPRESS
   DISPLAY FEET "FEET"
   BUILD MESSAGE
END IF
IF F11 PRESSED
   (MSG 14: AFFIRMATIVE ALT)
DISPLAY "AFFIRMATIVE"
DISPLAY ALTITUDE OPTION MENU
WAIT FOR KEYPRESS
DISPLAY ALTITUDE
BUILD MESSAGE
END IF
IF F12 PRESSED
DISPLAY "NEGATIVE"
DISPLAY NEG OPTIONS MENU
WAIT FOR KEYPRESS
IF F16 PRESSED
DISPLAY "CLIMB AND MAINTAIN"
ELSE
IF F17 PRESSED
DISPLAY "DESCEND AND MAINTAIN"
ELSE
IF F18 PRESSED
DISPLAY "MAINTAIN"
END IF
END IF
END IF
DISPLAY ALT OPTIONS MENU
WAIT FOR KEYPRESS
DISPLAY ALTITUDE
BUILD MESSAGE
END IF
IF F13 PRESSED
DISPLAY "SAY ALTITUDE"
BUILD MESSAGE
END IF
IF F14 PRESSED
DISPLAY "INTERCEPT"
GET AND DISPLAY ROUTE TEXT FROM KEYBOARD
DISPLAY CROSS/INTERCEPT OPTIONS MENU
WAIT FOR KEYPRESS
IF F16 PRESSED
DISPLAY "AT OR ABOVE"
ELSE
IF F17 PRESSED
DISPLAY "AT"
ELSE
IF F18 PRESSED
DISPLAY "AT OR BELOW"
END IF
END IF
END IF
DISPLAY ALT OPTIONS MENU
WAIT FOR ALTKEYPRESS
DISPLAY ALTITUDE
BUILD MESSAGE
END IF
IF F15 PRESSED
DISPLAY "CROSS"
DISPLAY FIX MENU
WAIT FOR KEYPRESS
DISPLAY FIX
DISPLAY CROSS/INTERCEPT OPTIONS MENU
WAIT FOR KEYPRESS
IF F16 PRESSED
   DISPLAY "AT OR ABOVE"
ELSE
   IF F17 PRESSED
      DISPLAY "AT"
   ELSE
      IF F18 PRESSED
         DISPLAY "AT OR BELOW"
      END IF
   END IF
END IF
DISPLAY ALT OPTIONS MENU
WAIT FOR KEYPRESS
DISPLAY ALTITUDE
BUILD MESSAGE
END IF
IF F16 PRESSED
   (MSG 2: MAINTAIN UNTIL)
   DISPLAY "MAINTAIN"
   DISPLAY ALT OPTIONS MENU
   WAIT FOR KEYPRESS
   DISPLAY ALTITUDE AND "UNTIL"
   DISPLAY MAINTAIN UNTIL OPTIONS MENU
   WAIT FOR KEYPRESS
IF F8 OR F17 OR F18 PRESSED
   (PAST FIX OPTIONS)
   IF F8 PRESSED
      (MILES PAST OPTION)
      DISPLAY MILES MENU
      WAIT FOR KEYPRESS
      DISPLAY MILES AND "PAST"
   ELSE
      (MINUTES PAST OPTION)
      IF F17 PRESSED
         DISPLAY MINUTES MENU
         WAIT FOR KEYPRESS
         DISPLAY MINUTES AND "PAST"
      ELSE
         (PAST FIX OPTION)
         DISPLAY "PAST"
      END IF
   END IF
END IF
DISPLAY FIX MENU
WAIT FOR KEYPRESS
DISPLAY FIX
BUILD MESSAGE
ELSE
   IF F9 PRESSED
      (UNTIL TIME OPTION)
      GET AND DISPLAY 4 DIGIT TIME
      BUILD MESSAGE
   END IF
END IF
END IF
IF F17 PRESSED
   (MSG 7: CLIMB/DESCEND TO REACH)
   IF AC'S ALT NOT STORED
DISPLAY CLIMB/DESCEND OPTION MENU
WAIT FOR KEYPRESS
IF F17 PRESSED
  SET CLIMB MODE ON
ELSE
  IF F18 PRESSED
    SET DESCEND MODE ON
  END IF
END IF
END IF
IF AC ALT > ALT OR DESCEND MODE ON
  DISPLAY "DESCEND TO REACH"
ELSE
  DISPLAY "CLIMB TO REACH"
END IF
DISPLAY ALTITUDE MENU
WAIT FOR KEYPRESS
DISPLAY ALTITUDE "AT"
DISPLAY CLIMB/DESCEND TO REACH OPTIONS MENU
WAIT FOR KEYPRESS
IF F9 PRESSED (AT TIME OPTION)
  GET AND DISPLAY 4 DIGIT TIME
  BUILD MESSAGE
ELSE
  IF F18 PRESSED (AT FIX OPTION)
    DISPLAY FIX MENU
    WAIT FOR KEYPRESS
    DISPLAY FIX
    BUILD MSG
  ELSE
    IF F8 PRESSED (AT A POINT OPTION)
      DISPLAY "A POINT"
      DISPLAY MILES MENU
      WAIT FOR KEYPRESS
      DISPLAY MILES
      DISPLAY DIRECTION MENU
      WAIT FOR KEYPRESS
      DISPLAY DIRECTION
      DISPLAY "OF"
      DISPLAY NAVAID MENU
      WAIT FOR KEYPRESS
      DISPLAY NAVAID
      BUILD MESSAGE
    END IF
  END IF
END IF
END IF
END IF
IF F18 PRESSED (MSG 5: CLIMB/DESCEND)
  IF AC’S ALT NOT STORED
    DISPLAY CLIMB/DESCEND OPTION MENU
    WAIT FOR KEYPRESS
    IF F17 PRESSED
      SET CLIMB MODE ON
    ELSE
      - 162 -
IF F18 PRESSED
   SET DESCEND MODE ON
END IF
END IF
END IF
IF AC ALT > ALT OR DESCEND MODE ON
   DISPLAY "DESCEND AND MAINTAIN"
ELSE
   DISPLAY "CLIMB AND MAINTAIN"
END IF
DISPLAY ALTITUDE MENU
WAIT FOR KEYPRESS
DISPLAY ALTITUDE
DISPLAY CLIMB/DESCEND AFTER OPTIONS MENU
WAIT FOR KEYPRESS
IF XMIT PRESSED
   BUILD MESSAGE
ELSE
   IF F9 PRESSED
      DISPLAY "AFTER PASSING"
   END IF
   DISPLAY FIX MENU
   WAIT FOR KEYPRESS
   DISPLAY FIX
   BUILD MESSAGE
ELSE
   IF F18 PRESSED
      DISPLAY "AT"
   END IF
   GET AND DISPLAY 4 DIGIT TIME
   BUILD MESSAGE
END IF
END IF
END IF
WAIT FOR KEYPRESS
IF XMIT PRESSED
   SEND MSG TO HOST
END IF
CLEAR AC DATA
RETURN
END F2KEY

SUBROUTINE F3KEY
   BEGIN
      IF AC ID NOT STORED
         SOUND BUZZER
         DISPLAY "NO AC IDENTIFIED"
         WAIT FOR KEYPRESS
         RETURN
      END IF
      DISPLAY SPEED MENU
      WAIT FOR KEYPRESS
      IF CLEAR PRESSED
         CLEAR AC DATA
         RETURN
      END IF
   END F3KEY
RETURN
END IF
IF PREV MENU PRESSED
RETURN
END IF
IF F3 PRESSED (MSG 1: SAY AIRSPEED)
  DISPLAY "SAY AIRSPEED"
  BUILD MESSAGE
END IF
IF F4 PRESSED (MSG 2: SAY MACH)
  DISPLAY "SAY MACH NUMBER"
  BUILD MESSAGE
END IF
IF F5 PRESSED (MSG 8: CROSS FIX AT ALT & AIRSPEED)
  DISPLAY "CROSS"
  DISPLAY FIX MENU
  WAIT FOR KEYPRESS
  DISPLAY FIX "AT AND MAINTAIN"
  DISPLAY ALTITUDE MENU
  WAIT FOR KEYPRESS
  DISPLAY ALTITUDE AND "AT"
  ACCEPT AND DISPLAY 3 DIGITS
  DISPLAY "KNOTS"
  BUILD MESSAGE
END IF
IF F6 PRESSED (MSG 7: DESCEND THEN REDUCE SPEED)
  DISPLAY "DESCEND AND MAINTAIN"
  DISPLAY ALTITUDE MENU
  WAIT FOR KEYPRESS
  DISPLAY ALTITUDE AND "THEN REDUCE SPEED"
  DISPLAY INCREASE/DECREASE OPTIONS MENU
  WAIT FOR KEYPRESS
  IF F8 PRESSED (REDUCE TO MACH NUMBER OPTION)
    DISPLAY "TO MACH"
    GET AND DISPLAY DECIMAL AND 2 DIGITS
  ELSE
    IF F17 PRESSED (REDUCE NUMBER OF KNOTS OPTION)
      GET AND DISPLAY 2 DIGITS
      DISPLAY "KNOTS"
    ELSE
      IF F18 PRESSED (REDUCE TO n KNOTS OPTION)
        DISPLAY "TO"
        GET AND DISPLAY 3 DIGITS
      END IF
    END IF
  END IF
END IF
BUILD MESSAGE
END IF
IF F7 PRESSED (MSG 6: REDUCE SPEED THEN DESCEND)
  DISPLAY "REDUCE SPEED"
  DISPLAY INCREASE/DECREASE OPTIONS MENU
  WAIT FOR KEYPRESS
  IF F17 PRESSED (REDUCE NUMBER OF KNOTS OPTION)
    GET AND DISPLAY 2 DIGITS
DISPLAY "KNOTS, THEN DESCEND AND MAINTAIN"
ELSE
  IF F18 PRESSED  (REDUCE TO n KNOTS OPTION)
    DISPLAY "TO"
    GET AND DISPLAY 3 DIGITS
    DISPLAY ", THEN DESCEND AND MAINTAIN"
  END IF
END IF
DISPLAY ALTITUDE MENU
WAIT FOR KEYPRESS
DISPLAY ALTITUDE
BUILD MESSAGE
END IF
IF F8 PRESSED  (MSG 5: IF PRACTICAL REDUCE SPEED)
  DISPLAY "IF PRACTICAL, REDUCE SPEED"
  DISPLAY INCREASE/DECREASE OPTIONS MENU
  WAIT FOR KEYPRESS
  IF F8 PRESSED  (REDUCE TO MACH NUMBER OPTION)
    DISPLAY "TO MACH"
    GET AND DISPLAY DECIMAL AND 2 DIGITS
  ELSE
    IF F17 PRESSED  (REDUCE NUMBER OF KNOTS OPTION)
      GET AND DISPLAY 2 DIGITS
      DISPLAY "KNOTS"
    ELSE
      IF F18 PRESSED  (REDUCE TO n KNOTS OPTION)
        DISPLAY "TO"
        GET AND DISPLAY 3 DIGITS
      END IF
    END IF
  END IF
END IF
BUILD MESSAGE
END IF
IF F9 PRESSED  (MSG 3: REDUCE SPEED)
  DISPLAY "REDUCE SPEED"
  DISPLAY INCREASE/DECREASE OPTIONS MENU
  WAIT FOR KEYPRESS
  IF F8 PRESSED  (REDUCE TO MACH NUMBER OPTION)
    DISPLAY "TO MACH"
    GET AND DISPLAY DECIMAL AND 2 DIGITS
  ELSE
    IF F17 PRESSED  (REDUCE NUMBER OF KNOTS OPTION)
      GET AND DISPLAY 2 DIGITS
      DISPLAY "KNOTS"
    ELSE
      IF F18 PRESSED  (REDUCE TO n KNOTS OPTION)
        DISPLAY "TO"
        GET AND DISPLAY 3 DIGITS
      END IF
    END IF
  END IF
BUILD MESSAGE
END IF
IF F13 PRESSED  (MSG 10: RESUME NORMAL SPEED)
DISPLAY "RESUME NORMAL SPEED"
BUILD MESSAGE
END IF
IF F14 PRESSED (MSG 4: DO NOT EXCEED KNOTS)
DISPLAY "DO NOT EXCEED"
GET AND DISPLAY 3 DIGITS
DISPLAY "KNOTS"
BUILD MESSAGE
END IF
IF F15 PRESSED (MSG 4: DO NOT EXCEED MACH)
DISPLAY "DO NOT EXCEED MACH"
GET AND DISPLAY DECIMAL AND 2 DIGITS
BUILD MESSAGE
END IF
IF F16 PRESSED (MSG 9: IF PRACTICAL MAINTAIN)
DISPLAY "IF PRACTICAL, MAINTAIN"
GET AND DISPLAY 3 DIGITS
DISPLAY "KNOTS"
BUILD MESSAGE
END IF
IF F17 PRESSED (MSG 5: IF PRACTICAL INCREASE SPEED)
DISPLAY "IF PRACTICAL, INCREASE SPEED"
DISPLAY INCREASE/DECREASE OPTIONS MENU
WAIT FOR KEYPRESS
IF F8 PRESSED (INCREASE TO MACH NUMBER OPTION)
DISPLAY "TO MACH"
GET AND DISPLAY DECIMAL AND 2 DIGITS
ELSE
IF F17 PRESSED (INCREASE NUMBER OF KNOTS OPTION)
GET AND DISPLAY 2 DIGITS
DISPLAY "KNOTS"
ELSE
IF F18 PRESSED (INCREASE TO n KNOTS OPTION)
DISPLAY "TO"
GET AND DISPLAY 3 DIGITS
END IF
END IF
END IF
BUILD MESSAGE
END IF
IF F18 PRESSED (MSG 3: INCREASE SPEED)
DISPLAY "INCREASE SPEED"
DISPLAY INCREASE/DECREASE OPTIONS MENU
WAIT FOR KEYPRESS
IF F8 PRESSED (INCREASE TO MACH NUMBER OPTION)
DISPLAY "TO MACH"
GET AND DISPLAY DECIMAL AND 2 DIGITS
ELSE
IF F17 PRESSED (INCREASE NUMBER OF KNOTS OPTION)
GET AND DISPLAY 2 DIGITS
DISPLAY "KNOTS"
ELSE
IF F18 PRESSED (INCREASE TO n KNOTS OPTION)
DISPLAY "TO"
GET AND DISPLAY 3 DIGITS
END IF
END IF
END IF
BUILD MESSAGE
END IF
WAIT FOR KEYPRESS
IF XMIT PRESSED
SEND MSG TO HOST
END IF
CLEAR AC DATA
END F3KEY

SUBROUTINE F4KEY (HOLDING)
BEGIN
IF AC ID NOT STORED
SOUND BUZZER
DISPLAY "NO AC IDENTIFIED"
WAIT FOR KEYPRESS
RETURN
END IF
DISPLAY HOLDING MENU
WAIT FOR KEYPRESS
IF CLEAR PRESSED
CLEAR AC DATA
RETURN
END IF
IF PREV MENU PressED
RETURN
END IF
IF F7 PRESSED (MSG 3: EXPECT FURTHER CLEARANCE AT)
DISPLAY "EXPECT FURTHER CLEARANCE"
GET AND DISPLAY 4 DIGIT TIME
DISPLAY EXPECT FURTHER OPTIONS MENU
WAIT FOR KEYPRESS
IF XMIT PRESSED
BUILD MESSAGE
SEND MESSAGE TO HOST
CLEAR AC DATA
RETURN
END IF
DISPLAY "ANTICIPATE ADDITIONAL"
IF F9 PRESSED (EXPECT HOURS DELAY OPTION)
READ AND DISPLAY 1 DIGIT
DISPLAY "HOUR"
ELSE
IF F18 PRESSED
DISPLAY MINUTE MENU
WAIT FOR KEYPRESS
DISPLAY MINUTES
DISPLAY "MINUTE"
END IF
END IF
DISPLAY WHERE DELAY OPTIONS MENU
WAIT FOR KEYPRESS
IF F9 PRESSED  (DELAY AT FIX OPTION)
   DISPLAY "DELAY AT"
   DISPLAY FIX MENU
   WAIT FOR KEYPRESS
   DISPLAY FIX
   BUILD MESSAGE
ELSE
   IF F17 PRESSED  (TERMINAL DELAY OPTION)
      DISPLAY "TERMINAL DELAY"
      BUILD MESSAGE
   ELSE
      IF F18 PRESSED  (ENROUTE DELAY)
         DISPLAY "ENROUTE DELAY"
         BUILD MESSAGE
      END IF
   END IF
ENDIF
ENDIF
ENDIF
IF F9 PRESSED  (MSG 2: CLEARED TO FIX NO DELAY)
   DISPLAY "CLEARED TO "
   DISPLAY FIX MENU
   WAIT FOR KEYPRESS
   DISPLAY FIX
   DISPLAY "ANTICIPATE NO DELAY"
   BUILD MESSAGE
ENDIF
IF F16 PRESSED  (MSG 1: EXPECT FURTHER CLEARANCE VIA)
   DISPLAY "EXPECT FURTHER CLEARANCE VIA"
   ACCEPT AND DISPLAY ROUTE TEXT
   BUILD MESSAGE
ENDIF
IF F17 PRESSED  (MSG 4: HOLDING INSTRUCTIONS)
   DISPLAY "HOLD"
   DISPLAY DIRECTION MENU
   WAIT FOR KEYPRESS
   DISPLAY DIRECTION AND "OF"
   DISPLAY FIX MENU
   WAIT FOR KEYPRESS
   DISPLAY FIX AND "ON"
   DISPLAY HOLDING INSTRUCTIONS PLACE OPTIONS MENU
   WAIT FOR KEYPRESS
IF F7 PRESSED  (AIRWAY OPTION)
   DISPLAY AIRWAY MENU
   WAIT FOR KEYPRESS
   DISPLAY AIRWAY
ELSE
   IF F8 PRESSED  (BEARING OPTION)
      ACCEPT AND DISPLAY 3 DIGIT BEARING
      DISPLAY "BEARING"
   ELSE
      IF F9 PRESSED  (RADIAL OPTION)
         ACCEPT AND DISPLAY 3 DIGIT RADIAL
DISPLAY "RADIAL"
ELSE
IF F16 PRESSED (ROUTE OPTION)
ACCEPT AND DISPLAY ROUTE TEXT
DISPLAY "ROUTE"
ELSE
IF F17 PRESSED (AZIMUTH OPTION)
ACCEPT AND DISPLAY 3 DIGIT AZIMUTH
DISPLAY "AZIMUTH"
ELSE
IF F18 PRESSED (COURSE OPTION)
ACCEPT AND DISPLAY 3 DIGIT COURSE
DISPLAY "COURSE"
END IF
END IF
END IF
END IF
DISPLAY HOLDING INSTRUCTIONS LEG OPTIONS MENU
WAIT FOR KEYPRESS
IF XMIT PRESSED
BUILD MESSAGE
SEND MESSAGE TO HOST
CLEAR AC DATA
RETURN
ELSE
IF F7 PRESSED (LEFT TURN OPTION ONLY)
DISPLAY "LEFT TURNS"
BUILD MESSAGE
ELSE
IF F16 PRESSED (RIGHT TURN OPTION ONLY)
DISPLAY "RIGHT TURNS"
BUILD MESSAGE
ELSE
IF F9 PRESSED (MINUTE LEG OPTION)
DISPLAY MINUTE MENU
WAIT FOR KEYPRESS
DISPLAY MINUTES AND "MINUTE LEG"
ELSE
IF F18 PRESSED (MILE LEG OPTION)
DISPLAY MILES MENU
WAIT FOR KEYPRESS
DISPLAY MILES AND "MILE LEG"
END IF
END IF
DISPLAY HOLDING INSTRUCTIONS LEG OPTIONS MENU
WAIT FOR KEYPRESS
IF XMIT PRESSED
BUILD MESSAGE
SEND MESSAGE TO HOST
CLEAR AC DATA
RETURN
ELSE
IF F7 PRESSED (LEFT TURN OPTION ONLY)
    DISPLAY "LEFT TURNS"
    BUILD MESSAGE
ELSE
    IF F16 PRESSED (RIGHT TURN OPTION ONLY)
        DISPLAY "RIGHT TURNS"
        BUILD MESSAGE
    END IF
END IF
END IF
END IF
END IF
END IF
END IF
END IF
IF F18 PRESSED (MSG 2: CLEARED TO FIX HOLD AS PUBLISHED)
    DISPLAY "CLEARED TO"
    DISPLAY FIX MENU
    WAIT FOR KEYPRESS
    DISPLAY FIX AND "HOLD"
    DISPLAY DIRECTION MENU
    WAIT FOR KEYPRESS
    DISPLAY DIRECTION AND "AS PUBLISHED"
    BUILD MESSAGE
END IF
WAIT FOR KEYPRESS
IF XMIT PRESSED
    SEND MSG TO HOST
END IF
CLEAR AC DATA
END F4KEY

SUBROUTINE F5KEY (ROUTING)
BEGIN
    IF AC ID NOT STORED
        SOUND BUZZER
        DISPLAY "NO AC IDENTIFIED"
        WAIT FOR KEYPRESS
        RETURN
    END IF
    DISPLAY ROUTING MENU
    WAIT FOR KEYPRESS
    IF CLEAR PressED
        CLEAR AC DATA
        RETURN
    END IF
    IF PREV MENU PRESSED
        RETURN
    END IF
    IF F7 PRESSED (MSG 4: CHANGE ROUTE TO READ)
        DISPLAY "CHANGE"
        DISPLAY ROUTING OPTIONS MENU
        ACCEPT AND DISPLAY ROUTE TEXT
        WAIT FOR KEYPRESS
IF F9 PRESSED
  DISPLAY "TO READ"
  ACCEPT AND DISPLAY NEW ROUTE TEXT
  BUILD MESSAGE
END IF
END IF

IF F8 PRESSED (MSG 2: OPERATION APPROVED)
  DISPLAY ROUTING OPTIONS MENU
  ACCEPT AND DISPLAY OPERATION TEXT
  WAIT FOR KEYPRESS
  IF F16 PRESSED
    DISPLAY "APPROVED"
    BUILD MESSAGE
  END IF
END IF

IF F9 PRESSED (MSG 1: APPROVED AS REQUESTED)
  DISPLAY "APPROVED AS REQUESTED"
  BUILD MESSAGE
END IF

IF F16 PRESSED (MSG 5: AMEND ROUTE REST UNCHANGED)
  DISPLAY "CLEARED"
  DISPLAY ROUTING OPTIONS MENU
  ACCEPT AND DISPLAY ROUTE TEXT
  WAIT FOR KEYPRESS
  IF F17 PRESSED
    DISPLAY "REST OF ROUTE UNCHANGED"
    BUILD MESSAGE
  END IF
END IF

IF F17 PRESSED (MSG 3: UNABLE OPERATION)
  DISPLAY "UNABLE"
  DISPLAY ROUTING OPTIONS MENU
  ACCEPT AND DISPLAY OPERATION TEXT
  WAIT FOR KEYPRESS
  IF F8 PRESSED
    ACCEPT AND DISPLAY REASON TEXT
    BUILD MESSAGE
  ELSE
    IF F18 PRESSED
      ACCEPT AND DISPLAY ADDITIONAL INSTRUCTIONS TEXT
      BUILD MESSAGE
  END IF
END IF
END IF

IF F18 PRESSED (MSG 6: CLEARED DIRECT)
  DISPLAY "CLEARED DIRECT"
  DISPLAY FIX MENU
  WAIT FOR KEYPRESS
  DISPLAY FIX
  BUILD MESSAGE
END IF

WAIT FOR KEYPRESS
IF XMIT PRESSED
  SEND MSG TO HOST
END IF
CLEAR AC DATA
END F5KEY

SUBROUTINE F6KEY (ARRIVALS)
BEGIN
IF AC ID NOT STORED
  SOUND BUZZER
  DISPLAY "NO AC IDENTIFIED"
  WAIT FOR KEYPRESS
  RETURN
END IF
DISPLAY ARRIVALS MENU
WAIT FOR KEYPRESS
IF CLEAR PRESSED
  CLEAR AC DATA
  RETURN
END IF
IF PREV MENU PRESSED
  RETURN
END IF
IF F7 PRESSED (MSG 8: CROSSED FINAL, TURN IMMEDIATELY)
  YOU HAVE CROSSED THE FINAL APPROACH COURSE, TURN
  DISPLAY LEFT/RIGHT OPTIONS MENU
  IF F9 PRESSED
    DISPLAY "L IMMEDIATELY AND RETURN TO THE LOCALIZER"
    BUILD MESSAGE
  ELSE
    IF F18 PRESSED
      DISPLAY "R IMMEDIATELY AND RETURN TO THE LOCALIZER"
      BUILD MESSAGE
    END IF
  END IF
END IF
IF F15 PRESSED (MSG 1: EXPECT TYPE APPROACH)
  DISPLAY "EXPECT"
  DISPLAY TYPE APPROACH OPTIONS MENU
  WAIT FOR KEYPRESS
  DISPLAY TYPE AND "APPROACH TO RUNWAY"
  DISPLAY RUNWAY OPTIONS MENU
  WAIT FOR KEYPRESS
  DISPLAY RUNWAY
  BUILD MESSAGE
END IF
IF F17 PRESSED (MSG 6: EXPECT VECTOR CROSS FINAL)
  DISPLAY "EXPECT VECTOR ACROSS FINAL APPROACH COURSE"
  FOR SPACING"
  BUILD MESSAGE
END IF
IF F8 OR F9 OR F16 OR F18 PRESSED (QUALIFY FOR CIRCLE OPTION)
  IF F8 PRESSED (MSG 4: CLEARED FOR APPROACH)
    DISPLAY "CLEARED FOR APPROACH"
    BUILD MESSAGE
END IF
IF F9 PRESSED  (MSG 2: CLEARED FOR TYPE APPROACH)
  DISPLAY "CLEARED FOR"
  DISPLAY TYPE APPROACH OPTIONS MENU
  WAIT FOR KEYPRESS
  DISPLAY TYPE
  DISPLAY RUNWAY OPTIONS MENU
  WAIT FOR KEYPRESS
  IF XMIT PRESSED
    DISPLAY "APPROACH"
    BUILD MESSAGE
    SEND MESSAGE TO HOST
    CLEAR AC DATA
    RETURN
  ELSE
    DISPLAY "RUNWAY", RUNWAY, "APPROACH"
    BUILD MESSAGE
  END IF
END IF
END IF
IF F16 PRESSED  (MSG 7: POSITION FOR ILS APPROACH)
  DISPLAY ILS INSTRUCTIONS ARRIVAL GATE MENU
  WAIT FOR KEYPRESS
  DISPLAY MESSAGE BASED ON GATE SELECTION
  BUILD MESSAGE
END IF
IF F18 PRESSED  (MSG 3: CLEARED FOR STRAIGHT IN APPROACH)
  DISPLAY "CLEARED FOR STRAIGHT-IN"
  DISPLAY TYPE APPROACH OPTIONS MENU
  WAIT FOR KEYPRESS
  DISPLAY TYPE
  DISPLAY RUNWAY OPTION MENU
  WAIT FOR KEYPRESS
  IF XMIT PRESSED
    DISPLAY "APPROACH"
    BUILD MESSAGE
    SEND MESSAGE TO HOST
    CLEAR AC DATA
    RETURN
  ELSE
    DISPLAY "RUNWAY" RUNWAY "APPROACH"
    BUILD MESSAGE
  END IF
END IF
DISPLAY CIRCLE OPTION MENU
WAIT FOR KEYPRESS
IF F18 PRESSED
  DISPLAY "CIRCLE TO LAND RUNWAY"
  DISPLAY RUNWAY OPTION MENU
  WAIT FOR KEYPRESS
  DISPLAY RUNWAY
  BUILD MESSAGE
END IF
END IF
WAIT FOR KEYPRESS
IF XMIT PRESSED
    SEND MSG TO HOST
END IF
CLEAR AC DATA
END F6KEY

SUBROUTINE F7KEY  (ALTIMETER)
BEGIN
    IF AC ID NOT STORED
        SOUND BUZZER
        DISPLAY "NO AC IDENTIFIED"
        WAIT FOR KEYPRESS
        RETURN
    END IF
    DISPLAY ALTIMETER MENU
    WAIT FOR KEYPRESS
    IF CLEAR PRESSED
        CLEAR AC DATA
        RETURN
    END IF
    IF PREV MENU PRESSED
        RETURN
    END IF
    GET ALTIMETER DATA FROM HOST
    DISPLAY FACILITY, TIME, "ALTIMETER" VALUE
    BUILD MESSAGE
    WAIT FOR KEYPRESS
    IF XMIT PRESSED
        SEND MSG TO HOST
    END IF
    CLEAR AC DATA
END F7KEY

SUBROUTINE F8KEY  (ACKNOWLEDGE)
BEGIN
    IF AC ID NOT STORED
        SOUND BUZZER
        DISPLAY "NO AC IDENTIFIED"
        WAIT FOR KEYPRESS
        RETURN
    END IF
    DISPLAY "ROGER"
    BUILD MESSAGE
    WAIT FOR KEYPRESS
    IF CLEAR PRESSED
        CLEAR AC DATA
        RETURN
    END IF
    IF PREV MENU PRESSED
        RETURN
    END IF
    GET ALTIMETER DATA FROM HOST
DISPLAY FACILITY, TIME, "ALTIMITER" VALUE
BUILD MESSAGE
WAIT FOR KEYPRESS
IF XMIT PRESSED
   SEND MSG TO HOST
END IF
CLEAR AC DATA
END F8KEY

SUBROUTINE F9KEY (AC LIST)
BEGIN
GET DATA FOR ALL AIRCRAFT IN CONTROLLER'S SECTOR FROM HOST
DISPLAY FIRST 9 AIRCRAFT
LOOP
   WAIT FOR KEYPRESS
   IF CLEAR OR PREV MENU PRESSED
      RETURN
   END IF
   IF LEFT ARROW PRESSED
      DISPLAY PREVIOUS 9 AIRCRAFT
   ELSE
      IF RIGHT ARROW PRESSED
         DISPLAY NEXT 9 AIRCRAFT
      END IF
   END IF
END LOOP
END F9KEY

SUBROUTINE F10KEY (TRAFFIC)
BEGIN
   IF AC ID NOT STORED
      SOUND BUZZER
      DISPLAY "NO AC IDENTIFIED"
      WAIT FOR KEYPRESS
      RETURN
   END IF
   DISPLAY TRAFFIC MENU
   WAIT FOR KEYPRESS
   IF CLEAR PRESSED
      CLEAR AC DATA
      RETURN
   END IF
   IF PREV MENU PRESSED
      RETURN
   END IF
   IF F7 PRESSED (MSG 3: TRAFFIC NO LONGER A FACTOR)
      DISPLAY "TRAFFIC NO LONGER A FACTOR"
      BUILD MESSAGE
   END IF
   IF F8 PRESSED (MSG 2: NUMEROUS TARGETS)
      DISPLAY "TRAFFIC, NUMEROUS TARGETS VICINITY OF"
      DISPLAY DISPLAY LOCATION MENU

- 175 -
WAIT FOR KEYPRESS
DISPLAY LOCATION
BUILD MESSAGE
END IF
IF F9 PRESSED  (MSG 1: COMPUTER GENERATED TRAFFIC MSG)
WAIT FOR KEYPRESS (CONTROLLER WILL SLEW THE TRACK BALL OR
ENTER THE CID OF THE CONFLICTING
TRAFFIC TARGET AND THE HOST WILL SEND
THE AC ID AND DATA TO THE DEVICE)
IF F9 PRESSED
IF NO AC IDENTIFIED
DISPLAY "YOU MUST IDENTIFY THE CONFLICTING TARGET"
ELSE
BUILD AND DISPLAY TRAFFIC MESSAGE
END IF
END IF
END IF
IF F16 PRESSED  (MSG 4: O’CLOCK TRAFFIC NO LONGER A FACTOR)
DISPLAY O’CLOCK MENU
WAIT FOR KEYPRESS
DISPLAY CLOCK VALUE AND "O’CLOCK TRAFFIC NO LONGER FACTOR"
BUILD MESSAGE
END IF
IF F17 PRESSED  (MSG 5: DIRECTION TRAFFIC NO LONGER A FACTOR)
DISPLAY DIRECTION MENU
WAIT FOR KEYPRESS
DISPLAY DIRECTION AND "TRAFFIC NO LONGER A FACTOR"
BUILD MESSAGE
END IF
ELSE
IF XMIT PRESSED
DISPLAY "ALTITUDE UNKNOWN"
BUILD MESSAGE
SEND MESSAGE TO HOST
CLEAR AC DATA
RETURN
ELSE
IF MOVEMENT KEY PRESSED
DISPLAY RELATIVE MOVEMENT
ELSE
IF DIRECTION BOUND KEY PRESSED
DISPLAY DIRECTION AND "BOUND"
DISPLAY BOUND/MOVEMENT OPTIONS MENU
WAIT FOR KEYPRESS
IF XMIT PRESSED
DISPLAY "ALTITUDE UNKNOWN"
BUILD MESSAGE
SEND MESSAGE TO HOST
CLEAR AC DATA
RETURN
ELSE
   IF MOVEMENT KEY PRESSED
      DISPLAY RELATIVE MOVEMENT
   END IF
END IF
END IF

DISPLAY AC TYPE MENU
WAIT FOR KEYPRESS
IF XMIT PRESSED
   DISPLAY "ALTITUDE UNKNOWN"
   BUILD MESSAGE
   SEND MESSAGE TO HOST
   CLEAR AC DATA
   RETURN
ELSE
   DISPLAY AC TYPE
   DISPLAY RELATIVE ALTITUDE OPTIONS MENU
   WAIT FOR KEYPRESS
   IF F18 PRESSED
      DISPLAY "ALTITUDE UNKNOWN"
      BUILD MESSAGE
   ELSE
      IF F16 OR F17 PRESSED
         IF F16 PRESSED
            DISPLAY "REPORTED ALTITUDE"
         END IF
         DISPLAY ALTITUDE OPTIONS MENU
         WAIT FOR KEYPRESS
         DISPLAY ALTITUDE
         BUILD MESSAGE
      END IF
   END IF
END IF
WAIT FOR KEYPRESS
IF XMIT PRESSED
   SEND MSG TO HOST
END IF
CLEAR AC DATA
END F10KEY

SUBROUTINE F11KEY
COMMUNICATION
BEGIN
   IF AC ID NOT STORED
      SOUND BUZZER
      DISPLAY "NO AC IDENTIFIED"
WAIT FOR KEYPRESS
RETURN
END IF
DISPLAY COMMUNICATION MENU
WAIT FOR KEYPRESS
IF CLEAR PRESSED
CLEAR AC DATA
RETURN
END IF
IF PREV MENU PRESSED
RETURN
END IF
IF F8 PRESSED  (MSG 4: CHANGE TO ADVISORY FREQ APPROVED)
DISPLAY "DISPLAY ADVISORY FREQUENCY APPROVED"
BUILD MESSAGE
END IF
IF F9 PRESSED  (MSG 2: CHANGE TO MY FREQUENCY)
DISPLAY "CHANGE TO MY FREQ"
DISPLAY FACILITY MENU
WAIT FOR KEYPRESS
IF DIGIT PRESSED
  ACCEPT AND DISPLAY FREQUENCY
ELSE
  IF FUNCTION KEY PRESSED
    DISPLAY FREQUENCY
  END IF
END IF
END IF
BUILD MESSAGE
END IF
IF F16 PRESSED  (MSG 3: REMAIN THIS FREQUENCY)
DISPLAY "REMAIN THIS FREQUENCY"
BUILD MESSAGE
END IF
IF F17 PRESSED  (MSG 5: FREQUENCY CHANGE APPROVED)
DISPLAY "FREQUENCY CHANGE APPROVED"
BUILD MESSAGE
END IF
IF F18 PRESSED  (MSG 1: CONTACT FACILITY ON FREQUENCY)
DISPLAY "CONTACT"
DISPLAY FACILITY MENU
WAIT FOR KEYPRESS
DISPLAY FACILITY AND FREQUENCY
DISPLAY WHEN OPTIONS MENU
WAIT FOR KEYPRESS
IF DIGIT PRESSED
  ACCEPT AND REPLACE FREQUENCY ON DISPLAY
  WAIT FOR KEYPRESS
END IF
IF XMIT PRESSED
  BUILD MESSAGE
  SEND MESSAGE TO HOST
  CLEAR AC DATA
  RETURN
ELSE
IF F9 PRESSED  (AT ALTITUDE OPTION)
   DISPLAY "AT"
   DISPLAY ALTITUDE OPTIONS MENU
   WAIT FOR KEYPRESS
   DISPLAY ALTITUDE
   BUILD MESSAGE
ELSE
   IF F17 PRESSED  (AT FIX OPTION)
      DISPLAY FIX MENU
      WAIT FOR KEYPRESS
      DISPLAY FIX
      BUILD MESSAGE
   ELSE
      IF F18 PRESSED  (AT TIME OPTION)
         ACCEPT AND DISPLAY 4 DIGIT TIME
         BUILD MESSAGE
   END IF
END IF
END IF
END IF
WAIT FOR KEYPRESS
IF XMIT PRESSED
   SEND MSG TO HOST
END IF
CLEAR AC DATA
END F11KEY

SUBROUTINE F12KEY  (ALERT)
BEGIN
   IF AC ID NOT STORED
      SOUND BUZZER
      DISPLAY "NO AC IDENTIFIED"
      WAIT FOR KEYPRESS
      RETURN
   END IF
   DISPLAY ALERT MENU
   WAIT FOR KEYPRESS
   IF CLEAR PRESSED
      CLEAR AC DATA
      RETURN
   END IF
   IF F9 PRESSED  (MSG 1: LOW ALTITUDE ALERT)
      DISPLAY "LOW ALTITUDE ALERT, CHECK ALTITUDE IMMEDIATELY"
      DISPLAY ALTITUDE ALERT OPTIONS MENU
      WAIT FOR KEYPRESS
      DISPLAY TYPE ALTITUDE SELECTION AND "IN YOUR AREA IS"
      ACCEPT AND DISPLAY 3 DIGIT ALTITUDE
      BUILD MESSAGE
   END IF
   IF F18 PRESSED  (MSG 2: TRAFFIC ALERT)
      DISPLAY "TRAFFIC ALERT, TRAFFIC"
      DISPLAY O’CLOCK MENU
WAIT FOR KEYPRESS
DISPLAY O’CLOCK POSITION
DISPLAY TRAFFIC ALERT OPTIONS MENU
WAIT FOR KEYPRESS
IF F8 PRESSED (TURN LEFT OPTION)
   DISPLAY "ADVISE YOU TURN LEFT"
   ACCEPT AND DISPLAY 3 DIGIT HEADING
ELSE
   IF F17 PRESSED (TURN RIGHT OPTION)
      DISPLAY "ADVISE YOU TURN RIGHT"
      ACCEPT AND DISPLAY 3 DIGIT HEADING
   ELSE
      IF F9 PRESSED (CLIMB OPTION)
         DISPLAY "CLIMB"
      ELSE
         IF F18 PRESSED (DESCEND OPTION)
            DISPLAY "DESCEND"
         END IF
      END IF
   END IF
DISPLAY ALTITUDE OPTIONS MENU
WAIT FOR KEYPRESS
DISPLAY ALTITUDE
END IF
END IF
DISPLAY "IMMEDIATELY"
BUILD MESSAGE
END IF
WAIT FOR KEYPRESS
IF XMIT PRESSED
   SEND MSG TO HOST
END IF
CLEAR AC DATA
END F12KEY

SUBROUTINE F13KEY (EMERGENCY)
BEGIN
   DISPLAY STORED EMERGENCY MESSAGE
   WAIT FOR KEYPRESS
   IF XMIT PRESSED
      SEND MSG TO HOST
   END IF
END F13KEY

SUBROUTINE F14KEY (RADAR)
BEGIN
   IF AC ID NOT STORED
      SOUND BUZZER
      DISPLAY "NO AC IDENTIFIED"
   END IF
   RETURN
END IF
DISPLAY RADAR MENU
WAIT FOR KEYPRESS
IF CLEAR PRESSED
   CLEAR AC DATA
   RETURN
END IF
IF PREV MENU PRESSED
   RETURN
END IF
IF F9 PRESSED  (MSG 8: RADAR CONTACT LOST)
   DISPLAY "RADAR CONTACT LOST"
   WAIT FOR KEYPRESS
ELSE
   IF F17 PRESSED  (MSG 6: RADAR SERVICE TERMINATED)
      DISPLAY "RADAR SERVICE TERMINATED"
      BUILD MESSAGE
   ELSE
      IF F18 PRESSED  (MSG 7: RADAR CONTACT)
         DISPLAY "RADAR CONTACT"
         WAIT FOR KEYPRESS
   END IF
END IF
END IF
IF F1 PRESSED  (MSG 5: INTERCEPTING RADIAL)
   DISPLAY "INTERCEPTING"
   DISPLAY NAVAID MENU
   WAIT FOR KEYPRESS
   DISPLAY NAVAID
   GET AND DISPLAY 3 DIGIT RADIAL
   DISPLAY "RADIAL"
   BUILD MESSAGE
END IF
IF F2 PRESSED  (MSG 5: CROSSING RADIAL)
   DISPLAY "CROSSING"
   DISPLAY NAVAID MENU
   WAIT FOR KEYPRESS
   DISPLAY NAVAID
   GET AND DISPLAY 3 DIGIT RADIAL
   DISPLAY "RADIAL"
   BUILD MESSAGE
END IF
IF F3 PRESSED  (MSG 2: MILES FROM FIX)
   DISPLAY MILES MENU
   WAIT FOR KEYPRESS
   DISPLAY MILES AND "MILES FROM"
   DISPLAY FIX MENU
   WAIT FOR KEYPRESS
   DISPLAY FIX
   BUILD MESSAGE
END IF
IF F4 PRESSED  (MSG 1: OVER FIX)
   DISPLAY "OVER"
   DISPLAY FIX MENU
   WAIT FOR KEYPRESS
   DISPLAY FIX

   - 181 -
BUILD MESSAGE
END IF
IF F5 PRESSED (MSG 1: PASSING FIX)
DISPLAY "PASSING"
DISPLAY FIX MENU
WAIT FOR KEYPRESS
DISPLAY FIX
BUILD MESSAGE
END IF
END IF
IF F10 PRESSED (MSG 4: CROSSING AIRWAY/ROUTE)
DISPLAY "CROSSING"
DISPLAY RADAR OPTIONS MENU
WAIT FOR KEYPRESS
IF F17 PRESSED (AIRWAY OPTION)
DISPLAY AIRWAY MENU
WAIT FOR KEYPRESS
DISPLAY AIRWAY
BUILD MESSAGE
ELSE
IF F18 PRESSED (ROUTE OPTION)
ACCEPT AND DISPLAY ROUTE TEXT
BUILD MESSAGE
END IF
END IF
END IF
END IF
IF F11 PRESSED (MSG 4: JOINING AIRWAY/ROUTE)
DISPLAY "JOINING"
DISPLAY RADAR OPTIONS MENU
WAIT FOR KEYPRESS
IF F17 PRESSED (AIRWAY OPTION)
DISPLAY AIRWAY MENU
WAIT FOR KEYPRESS
DISPLAY AIRWAY
BUILD MESSAGE
ELSE
IF F18 PRESSED (ROUTE OPTION)
ACCEPT AND DISPLAY ROUTE TEXT
BUILD MESSAGE
END IF
END IF
END IF
IF F12 PRESSED (MSG 4: DEPARTING AIRWAY/ROUTE)
DISPLAY "DEPARTING"
DISPLAY RADAR OPTIONS MENU
WAIT FOR KEYPRESS
IF F17 PRESSED (AIRWAY OPTION)
DISPLAY AIRWAY MENU
WAIT FOR KEYPRESS
DISPLAY AIRWAY
BUILD MESSAGE
ELSE
IF F18 PRESSED (ROUTE OPTION)
ACCEPT AND DISPLAY ROUTE TEXT
BUILD MESSAGE
BUILD MESSAGE
END IF
END IF
END IF
END IF
END IF
END IF
END IF
END IF
IF F13 PRESSED (MSG 3: MILES IN DIRECTION OF)
  DISPLAY MILES MENU
  WAIT FOR KEYPRESS
  DISPLAY MILES AND "MILES"
  DISPLAY DIRECTION MENU
  WAIT FOR KEYPRESS
  DISPLAY DIRECTION AND "OF"
  DISPLAY RADAR OPTIONS MENU
  WAIT FOR KEYPRESS
IF F17 PRESSED (AIRWAY OPTION)
  DISPLAY AIRWAY MENU
  WAIT FOR KEYPRESS
  DISPLAY AIRWAY
  BUILD MESSAGE
ELSE
  IF F8 PRESSED (LOCATION OPTION)
    DISPLAY LOCATION MENU
    WAIT FOR KEYPRESS
    DISPLAY LOCATION
    BUILD MESSAGE
  ELSE
    IF F9 PRESSED (FIX OPTION)
      DISPLAY FIX MENU
      WAIT FOR KEYPRESS
      DISPLAY FIX
      BUILD MESSAGE
  END IF
END IF
END IF
WAIT FOR KEYPRESS
IF XMIT PRESSED
  SEND MSG TO HOST
END IF
CLEAR AC DATA
END F14KEY

SUBROUTINE F15KEY (WEATHER)
BEGIN
  IF F7, F8 OR F9 PRESSED (REQUIRE AC ID)
    IF AC ID NOT STORED
      SOUND BUZZER
      DISPLAY "NO AC IDENTIFIED"
      WAIT FOR KEYPRESS
      RETURN
  END IF
END IF
DISPLAY WEATHER MENU
WAIT FOR KEYPRESS
IF CLEAR PRESSED
CLEAR AC DATA
RETURN
END IF
IF PREV MENU PRESSED
RETURN
END IF
IF F7 PRESSED (MSG 3: SOLICIT PIREP)
DISPLAY "REQUEST FLIGHT CONDITIONS"
DISPLAY SOLICIT PIREP MENU
WAIT FOR KEYPRESS
IF XMIT PRESSED
BUILD MESSAGE
SEND MESSAGE TO HOST
CLEAR AC DATA
ELSE
IF F8 PRESSED (BETWEEN FIXES OPTION)
DISPLAY "BETWEEN"
DISPLAY FIX MENU
WAIT FOR KEYPRESS
DISPLAY FIX AND "AND"
DISPLAY FIX MENU
WAIT FOR KEYPRESS
DISPLAY FIX
BUILD MESSAGE
ELSE
IF F9 PRESSED (ALONG PRESENT ROUTE OPTION)
DISPLAY "ALONG PRESENT ROUTE"
BUILD MESSAGE
ELSE
IF F17 PRESSED (OVER FIX OPTION)
DISPLAY "OVER"
DISPLAY FIX MENU
WAIT FOR KEYPRESS
DISPLAY FIX
BUILD MESSAGE
ELSE
IF F18 PRESSED (AT PRESENT POSITION OPTION)
DISPLAY "AT PRESENT POSITION"
BUILD MESSAGE
END IF
END IF
END IF
END IF
END IF
END IF
END IF
END IF
END IF
IF F8 PRESSED (MSG 4: SEND PIREP)
GET PIREP MENU FROM HOST
DISPLAY FIRST 9 PIREPS
LOOP UNTIL VALID F KEY PRESSED
WAIT FOR KEYPRESS
IF LEFT ARROW PRESSED
DISPLAY PREVIOUS 9 PIREPS
ELSE
IF RIGHT ARROW PRESSED
DISPLAY NEXT 9 PIREPS
END IF
END IF
END LOOP
BUILD MESSAGE
END IF
IF F9 PRESSED (MSG 2: WEATHER AT A POSITION)
DISPLAY "WEATHER AREA BETWEEN"
DISPLAY O’CLOCK MENU
WAIT FOR KEYPRESS
DISPLAY CLOCK VALUE AND "O’CLOCK AND"
DISPLAY O’CLOCK MENU
WAIT FOR KEYPRESS
DISPLAY CLOCK VALUE AND "O’CLOCK’
DISPLAY MILES MENU
WAIT FOR KEYPRESS
DISPLAY MILES AND "MILES"
BUILD MESSAGE
END IF
IF F15 PRESSED (MSG 5: MONITOR HIWAS)
GET HIWAS FREQUENCY FROM HOST
DISPLAY "MONITOR HIWAS" FREQUENCY
BUILD MESSAGE
END IF
IF F16 PRESSED (MSG 1: CENTER WEATHER ADVISORY)
GET CENTER WEATHER ADVISORY FROM HOST
DISPLAY "ATTENTION ALL AIRCRAFT, CENTER WEATHER ADVISORY"
DISPLAY ID AND TEXT
BUILD MESSAGE
END IF
IF F17 PRESSED (MSG 1: CONVECTIVE SIGMET)
GET CONVECTIVE SIGMET DATA FROM HOST
DISPLAY "ATTENTION ALL AIRCRAFT, CONVECTIVE SIGMET"
DISPLAY ID AND TEXT
BUILD MESSAGE
END IF
IF F18 PRESSED (MSG 1: SIGMET)
GET SIGMET DATA FROM HOST
DISPLAY "ATTENTION ALL AIRCRAFT, SIGMET"
DISPLAY ID AND TEXT
BUILD MESSAGE
END IF
WAIT FOR KEYPRESS
IF XMIT PRESSED
SEND MSG TO HOST
END IF
CLEAR AC DATA
END F15KEY

SUBROUTINE F16KEY (NAVAIDS)
BEGIN
IF AC ID NOT STORED
SOUND BUZZER
DISPLAY "NO AC IDENTIFIED"
WAIT FOR KEYPRESS
RETURN
END IF
DISPLAY NAVAID MENU
WAIT FOR KEYPRESS
IF CLEAR PRESSED
CLEAR AC DATA
RETURN
END IF
IF PREV MENU PRESSED
RETURN
END IF
GET NAVAID STATUS FROM HOST
DISPLAY NAVAID DATA
BUILD MESSAGE
WAIT FOR KEYPRESS
IF XMIT PRESSED
SEND MSG TO HOST
END IF
CLEAR AC DATA
END F16KEY

SUBROUTINE F17KEY (RE-XMIT)
BEGIN
DISPLAY FIRST 9 MESSAGES
LOOP
WAIT FOR KEYPRESS
IF CLEAR PRESSED
CLEAR AC DATA
RETURN
ELSE
IF PREV MENU PRESSED
RETURN
ELSE
IF LEFT ARROW PRESSED
DISPLAY PREVIOUS 9 MESSAGES
ELSE
IF RIGHT ARROW PRESSED
DISPLAY NEXT 9 MESSAGES
ELSE
IF F KEY PRESSED
IF MESSAGE IS INBOUND TYPE
SOUND BUZZER
ELSE
IF MESSAGE IS ACKNOWLEDGED
SOUND BUZZER
ELSE
SEND MESSAGE TO HOST
CLEAR AC DATA
RETURN
END IF
END IF

- 186 -
END IF
END IF
END IF
END IF
END IF
END LOOP
END F17KEY

SUBROUTINE F18KEY (OTHER FUNCTIONS)
BEGIN
DISPLAY OTHER FUNCTIONS MENU
WAIT FOR KEYPRESS
IF CLEAR PRESSED
CLEAR AC DATA
RETURN
ELSE
IF PREV MENU PRESSED
RETURN
ELSE
IF F KEY PRESSED
ACCOMPLISH APPROPRIATE FUNCTION
END IF
END IF
END IF
END F18KEY
APPENDIX D

Host Pseudo Code

This pseudo code represents the basic functions required of the Host computer to handle the device. It describes the logic of the subroutine that would be invoked whenever data appears in the communications port. Logic not included is that to build and edit the programmable menus nor the logic required to interact with the other host processors.

SUBROUTINE HOST

.THIS ROUTINE IS INTERRUPT INITIATED UPON RECEIVING DATA IN
.THE COMMUNICATIONS PORT ATTACHED TO THE DEVICE.

BEGIN

READ COMM BUFFER
IF "ACK"
RETURN
ELSE
IF "NAK"
SEND LAST MESSAGE TO THE DEVICE
ELSE
IF POLL REQUEST
CALL POLLREQ
ELSE
IF CID REQUEST
CALL CIDREQ
ELSE
IF MESSAGE
CALL MESSAGE
ELSE
IF PIREP REQ
CALL PIREP
ELSE
IF ALTIMETER REQUEST
CALL ALTIMETER
ELSE
IF NAVAID REQUEST
CALL NAVAID
ELSE
IF AC LIST REQUEST
CALL ACLIST
END IF
END IF
END IF
END
END IF
END IF
END IF
END IF
END IF
RETURN
END HOST

SUBROUTINE POLLREQ
(HANDLES POLL REQUESTS FROM THE DEVICE)
BEGIN
IF TOP OF ACK STACK HAS DATA (ACKNOWLEDGMENT FROM PILOT)
POP THE STACK
SEND ACK MESSAGE TO DEVICE
MOVE ACK CHARACTER TO DATA BLOCK
ELSE
IF TOP OF PILOT MSG HAS DATA (UNSOLICITED PILOT MESSAGE)
POP THE STACK
SEND MESSAGE TO DEVICE
MOVE MESSAGE CHARACTER TO DATA BLOCK
ELSE
IF TOP OF MEM LOAD STACK HAS DATA (MENU TO BE SENT)
POP THE MEM LOAD STACK
SEND MENU TO THE DEVICE
ELSE
IF RADAR CURSOR IS OVER TARGET
IF TARGET IS IN CONTROLLER’S SECTOR
SEND AIRCRAFT DATA TO THE DEVICE
END IF
ELSE
SEND "NO DATA" RESPONSE
END IF
END IF
END IF
END IF
RETURN
END POLLREQ

SUBROUTINE CIDREQ
(RETURNS AIRCRAFT DATA BASED ON CID)
BEGIN
SEARCH DATABASE FOR CID
IF FOUND
IF IN CONTROLLER’S SECTOR
SEND AIRCRAFT DATA TO THE DEVICE
ELSE
SEND "000" CID BACK TO THE DEVICE
END IF
ELSE
SEND "000" CID BACK TO THE DEVICE
END IF
RETURN
SUBROUTINE MESSAGE
(PASSES TACTICAL MESSAGE BEING TRANSMITTED TO PILOT)
(TO THE MODE S DATA LINK SYSTEM)
BEGIN
CHECK MESSAGE FOR ERRORS
IF ANY
   SEND "NAK" BACK TO THE DEVICE
ELSE
   FORMAT MESSAGE FOR MODE S
   PASS MESSAGE TO MODE S FOR TRANSMISSION
   IF MESSAGE TYPE REQUIRES ACKNOWLEDGMENT
      MOVE "UNACKED" CHARACTER TO DATA BLOCK
   END IF
   UPDATE APPROPRIATE DATABASES IN HOST
END IF
RETURN
END MESSAGE

SUBROUTINE PIREP
(LOADS DEVICE WITH PIREP DATA)
BEGIN
   BUILD PIREP MENU
   SEND PIREP DATA TO THE DEVICE
RETURN
END PIREP

SUBROUTINE ALTIMETER
(SENS ALTIMETER DATA TO THE DEVICE)
BEGIN
   GET STATION'S ALTIMETER VALUE AND TIME OF VALUE
   SEND ALTIMETER DATA TO THE DEVICE
RETURN
END ALTIMETER

SUBROUTINE NAVAID
(SEND NAVAID DATA TO THE DEVICE)
BEGIN
   GET CURRENT STATUS OF NAVAID
   SEND NAVAID DATA TO THE DEVICE
RETURN
END NAVAID

SUBROUTINE ACLIST
(SEND AC LIST TO THE DEVICE)
BEGIN
   WHILE NOT AT END OF LIST
      SEND AC DATA WITH EOL FLAG = 0
END WHILE
SEND AC DATA WITH EOL FLAG = 1
RETURN
END ACLIST
APPENDIX E

Message Formats

The following are the message formats used in the communications between
the host and the device. The following notations are used: "(N3)"
means three digit number, "(A10)" means ten character string, "#" means
the message number.

Outbound Messages to the Host

<table>
<thead>
<tr>
<th>MESSAGE NUMBER</th>
<th>TYPE</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>NAK</td>
<td>none</td>
</tr>
<tr>
<td>001</td>
<td>ACK</td>
<td>none</td>
</tr>
<tr>
<td>002</td>
<td>POLL REQUEST</td>
<td>none</td>
</tr>
<tr>
<td>003</td>
<td>CID REQUEST</td>
<td>CID(N3)</td>
</tr>
<tr>
<td>004</td>
<td>PIREP REQUEST</td>
<td>none</td>
</tr>
<tr>
<td>005</td>
<td>ALTIMETER REQUEST</td>
<td>STATION(A3)</td>
</tr>
<tr>
<td>006</td>
<td>NAVIAD REQUEST</td>
<td>STATION(A3)</td>
</tr>
<tr>
<td>007</td>
<td>AC LIST REQUEST</td>
<td>none</td>
</tr>
<tr>
<td>008</td>
<td>EMERGENCY</td>
<td>TEXT(A50)</td>
</tr>
<tr>
<td>009</td>
<td>ALERT MSG 1</td>
<td>CID(N3), #(N3), CODE(A1), ALT(N3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CODE: 1=MEA, 2=MVA, 3=MOCX, 4=MIA,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5=MDA, 6=DH</td>
</tr>
<tr>
<td>010</td>
<td>ALERT MSG 2</td>
<td>CID(N3), #(N3), POSITION(N2), CODE(A1),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VALUE(N3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CODE: 1=TURN RIGHT, 2=TURN LEFT, 3=CLIMB,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4=DESCEND</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VALUE = HDG OR ALT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>POSITION = O’CLOCK NUMBER</td>
</tr>
<tr>
<td>011</td>
<td>COMM MSG 1</td>
<td>CID(N3), #(N3), FACILITY(A15), FREQ(A6), CODE(A1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VALUE(A4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CODE: 0=no data 1=TIME, 2=FIX, 3=ALT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VALUE = TIME, FIX, ALT OR blank</td>
</tr>
<tr>
<td>MESSAGE NUMBER</td>
<td>TYPE</td>
<td>DATA</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>012</td>
<td>COMM MSG 2</td>
<td>CID(N3), #(N3), FREQ(A6)</td>
</tr>
<tr>
<td>013</td>
<td>COMM MSG 3</td>
<td>CID(N3), #(N3)</td>
</tr>
<tr>
<td>014</td>
<td>COMM MSG 4</td>
<td>CID(N3), #(N3)</td>
</tr>
<tr>
<td>015</td>
<td>COMM MSG 5</td>
<td>CID(N3), #(N3)</td>
</tr>
<tr>
<td>016</td>
<td>ROUTING MSG 1</td>
<td>CID(N3), #(N3)</td>
</tr>
<tr>
<td>017</td>
<td>ROUTING MSG 2</td>
<td>CID(N3), #(N3), OPERATION(A30)</td>
</tr>
<tr>
<td>018</td>
<td>ROUTING MSG 3</td>
<td>CID(N3), #(N3), OPERATION(A30), REASON(A30)</td>
</tr>
<tr>
<td>019</td>
<td>ROUTING MSG 4</td>
<td>CID(N3), #(N3), ROUTE(A30), AMENDED(A30)</td>
</tr>
<tr>
<td>020</td>
<td>ROUTING MSG 5</td>
<td>CID(N3), #(N3), ROUTE(A30)</td>
</tr>
<tr>
<td>021</td>
<td>ROUTING MSG 6</td>
<td>CID(N3), #(N3), FIX(A15)</td>
</tr>
<tr>
<td>022</td>
<td>TRAFFIC MSG 1</td>
<td>CID(N3), #(N3), O’CLOCK(N2), MI(N2), DIR(A2), MVNT(A15), TYPE(A15), ALT(N3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DIR = N, NE, E, SE, S, SW, W, NW, U</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MVNT: C=CLIMBING, L=LEVEL, D=DESCENDING</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M=MANEUVERING, V=CONVERGING, O=OPPOSITE DIRECTION</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ALT: 000=ALT UNKNOWN</td>
</tr>
<tr>
<td>023</td>
<td>TRAFFIC MSG 2</td>
<td>CID(N3), #(N3), LOCATION(A15)</td>
</tr>
<tr>
<td>024</td>
<td>TRAFFIC MSG 3</td>
<td>CID(N3), #(N3)</td>
</tr>
<tr>
<td>025</td>
<td>TRAFFIC MSG 4</td>
<td>CID(N3), #(N3), O’CLOCK(N2)</td>
</tr>
<tr>
<td>026</td>
<td>TRAFFIC MSG 5</td>
<td>CID(N3), #(N3), DIRECTION(A2)</td>
</tr>
<tr>
<td>027</td>
<td>WEATHER MSG 1</td>
<td>CODE(A1), ID(N3), DATA(A90)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CODE: S=SIGMET, C=CONVECTIVE SIGMET, W=CENTER WX ADVISORY</td>
</tr>
<tr>
<td>028</td>
<td>WEATHER MSG 2</td>
<td>CID(N3), #(N3), O’CLOCK(N2), O’CLOCK(N2), MI(N3)</td>
</tr>
<tr>
<td>029</td>
<td>WEATHER MSG 3</td>
<td>CID(N3), #(N3), CODE(A1), FIX(A15), FIX(A15)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CODE: 1=AT PRES POSITION, 2=OVER fix, 3=ALONG PRES ROUTE, 4=BETWEEN FIXES</td>
</tr>
<tr>
<td>030</td>
<td>WEATHER MSG 4</td>
<td>CID(N3), #(N3), PI REP(A90)</td>
</tr>
<tr>
<td>031</td>
<td>WEATHER MSG 5</td>
<td>CID(N3), #(N3), FREQ(A6)</td>
</tr>
<tr>
<td>MESSAGE NUMBER</td>
<td>TYPE</td>
<td>DATA</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>032 ALTIMETER MSG 1</td>
<td>CID(N3),#,FACILITY(A15),TIME(A4),SETTING(N4)</td>
<td></td>
</tr>
<tr>
<td>033 ALTITUDE MSG 1</td>
<td>CID(N3),#,ALT(N3)</td>
<td></td>
</tr>
<tr>
<td>034 ALTITUDE MSG 2</td>
<td>CID(N3),#,ALT(N3),CODE(A1),FIX(A15),TIME(A4),MI(N3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CODE: 1=TIME, 2=PAST FIX, 3=MILES PAST FIX</td>
<td></td>
</tr>
<tr>
<td>035 ALTITUDE MSG 3</td>
<td>CID(N3),#,ALT(N3),CODE(A1),FIX(A15)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CODE: 1=AT ALT, 2=AT/ABOVE 3=AT/BELOW</td>
<td></td>
</tr>
<tr>
<td>036 ALTITUDE MSG 4</td>
<td>CID(N3),#,ALT(N3),CODE(A1),ROUTE(A15)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CODE: 1=AT ALT, 2=AT/ABOVE 3=AT/BELOW</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CODE: 0=no other data, 1=AFTER PASSING FIX</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2=AT TIME, 3=WHEN X MILES PAST, 4=WHEN X MINUTES PAST</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TYPE: C=CLIMB, D=DESCEND</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PDFLAG: AT PILOT'S DISCRETION</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CODE: 0=no other data, 1=AFTER PASSING FIX</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2=AT TIME, 3=WHEN X MILES PAST, 4=WHEN X MINUTES PAST</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TYPE: C=CLIMB, D=DESCEND</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PDFLAG: AT PILOT'S DISCRETION</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CODE: 1=AT TIME, 2=AT FIX, 3=AT A POINT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TYPE: C=CLIMB, D=DESCEND</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PDFLAG: AT PILOT'S DISCRETION</td>
<td></td>
</tr>
<tr>
<td>040 ALTITUDE MSG 9</td>
<td>CID(N3),#,FROMALT(N3),TOALT(N3)</td>
<td></td>
</tr>
<tr>
<td>041 ALTITUDE MSG 10</td>
<td>CID(N3),#,NUMBER(N3),FIX(A15)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CODE: 1=IN MILES, 2=IN MINUTES, 3=AT FIX</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TYPE: C=CLIMB, D=DESCEND</td>
<td></td>
</tr>
<tr>
<td>042 ALTITUDE MSG 11</td>
<td>CID(N3),#,CODE(A1),FACILITY(A15),TIME(A4),FIX(A15),ALT(N3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CODE: 0=no other data, 1=AT TIME, 2=AT FIX</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3=AT ALTITUDE</td>
<td></td>
</tr>
<tr>
<td>MESSAGE</td>
<td>TYPE</td>
<td>DATA</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>043 ALTITUDE MSG 12</td>
<td>CID(N3),#(N3),ALT(N3)</td>
<td></td>
</tr>
<tr>
<td>044 ALTITUDE MSG 13</td>
<td>CID(N3),#(N3),ALT(N3)</td>
<td></td>
</tr>
<tr>
<td>045 ALTITUDE MSG 14</td>
<td>CID(N3),#(N3),ALT(N3)</td>
<td></td>
</tr>
<tr>
<td>046 ALTITUDE MSG 15</td>
<td>CID(N3),#(N3),ALT(N3),CODE(A1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CODE: 1=CLIMB AND MAINTAIN ALT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2=DESCEND AND MAINTAIN ALT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3=MAINTAIN ALT</td>
<td></td>
</tr>
<tr>
<td>047 ALTITUDE MSG 16</td>
<td>CID(N3),#(N3)</td>
<td></td>
</tr>
<tr>
<td>048 ALTITUDE MSG 17</td>
<td>CID(N3),#(N3)</td>
<td></td>
</tr>
<tr>
<td>049 ALTITUDE MSG 18</td>
<td>CID(N3),#(N3),NUMBER(N4)</td>
<td></td>
</tr>
<tr>
<td>050 HOLDING MSG 1</td>
<td>CID(N3)#(N3),ROUTE(A30)</td>
<td></td>
</tr>
<tr>
<td>051 HOLDING MSG 2</td>
<td>CID(N3)#(N3),CODE(A1),FIX(A15),DIR(A2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CODE: 1=HOLD AS PUBLISHED,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2=NO DELAY EXPECTED</td>
<td></td>
</tr>
<tr>
<td>052 HOLDING MSG 3</td>
<td>CID(N3)#(N3),CODE(A1),FIX(A15),TIME(A4),NUMBER(N2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CODE: 0=no other data,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1=MINUTE DELAY AT FIX</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2=HOUR DELAY AT FIX</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3=MINUTE ENROUTE DELAY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4=HOUR ENROUTE DELAY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5=MINUTE TERMINAL DELAY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6=HOUR TERMINAL DELAY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CODE: 1=RADIAL, 2=COURSE, 3=BEARING,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4=AZIMUTH, 5=AIRWAY, 6=ROUTE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LEG: 1=MILE LEGS, 2=MINUTE LEGS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TURNS: L=LEFT TURNS, R=RIGHT TURNS</td>
<td></td>
</tr>
<tr>
<td>054 ARRIVALS MSG 1</td>
<td>CID(N3),#(N3),TYPE(A15),RUNWAY(A3)</td>
<td></td>
</tr>
<tr>
<td>055 ARRIVALS MSG 2</td>
<td>CID(N3),#(N3),TYPE(A15),RUNWAY(A3)CIRCLERW(A3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RUNWAY: blank means no runway specified</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CIRCLERW: not blank means circle to runway</td>
<td></td>
</tr>
<tr>
<td></td>
<td>specified</td>
<td></td>
</tr>
<tr>
<td>MESSAGE NUMBER</td>
<td>TYPE</td>
<td>DATA</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------</td>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>056</td>
<td>ARRIVALS MSG 3</td>
<td>CID(N3),#(N3),TYPE(A15),RUNWAY(A3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CIRCLERW(A3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RUNWAY: blank means no runway specified</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CIRCLERW: not blank means circle to runway</td>
</tr>
<tr>
<td></td>
<td></td>
<td>specified</td>
</tr>
<tr>
<td>057</td>
<td>ARRIVALS MSG 4</td>
<td>CID(N3),#(N3),CIRCLERW(A3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CIRCLERW: not blank means circle to runway</td>
</tr>
<tr>
<td></td>
<td></td>
<td>specified</td>
</tr>
<tr>
<td>058</td>
<td>ARRIVALS MSG 6</td>
<td>CID(N3),#(N3)</td>
</tr>
<tr>
<td>059</td>
<td>ARRIVALS MSG 7</td>
<td>CID(N3),#(N3),MI(N2),FIX(A15),TURN(A1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HDG(N3),ALT(N3),RUNWAY(A3),FREQ(A6)</td>
</tr>
<tr>
<td>060</td>
<td>ARRIVALS MSG 8</td>
<td>CID(N3),#(N3),TURN(A1)</td>
</tr>
<tr>
<td>061</td>
<td>RADAR MSG 1</td>
<td>CID(N3),#(N3),CODE(A1),FIX(A15)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CODE: 1=OVER, 2=PASSING</td>
</tr>
<tr>
<td>062</td>
<td>RADAR MSG 2</td>
<td>CID(N3),#(N3),MILES(N3),FIX(A15)</td>
</tr>
<tr>
<td>063</td>
<td>RADAR MSG 3</td>
<td>CID(N3),#(N3),CODE(A1),MILES(N3),DIR(A2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DATA(A15)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CODE: 1=FIX, 2=AIRWAY, 3=LOCATION</td>
</tr>
<tr>
<td>064</td>
<td>RADAR MSG 4</td>
<td>CID(N3),#(N3),CODE(A1),TYPE(A1),DATA(A15)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CODE: 1=AIRWAY, 2=ROUTE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TYPE: 1=CROSSING, 2=JOINING, 3=DEPARTING</td>
</tr>
<tr>
<td>065</td>
<td>RADAR MSG 5</td>
<td>CID(N3),#(N3),NAVAID(A3),RADIAL(N3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TYPE: 1=CROSSING, 2=INTERCEPTING</td>
</tr>
<tr>
<td>066</td>
<td>RADAR MSG 6</td>
<td>CID(N3),#(N3)</td>
</tr>
<tr>
<td>067</td>
<td>RADAR MSG 7</td>
<td>CID(N3),#(N3),POSITION(A15)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>POSITION=blank means none given</td>
</tr>
<tr>
<td>068</td>
<td>RADAR MSG 8</td>
<td>CID(N3),#(N3),POSITION(A15)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>POSITION=blank means none given</td>
</tr>
<tr>
<td>069</td>
<td>VECTORING MSG 1</td>
<td>CID(N3),#(N3),TURN(A1),HDG(N3),REASON(A1),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DATA(A15),NAVAID(A3),RADIAL(N3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TURN: R=RIGHT, L=LEFT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REASON: 0=none given</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1=FOR VECTOR TO FIX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2=FOR VECTOR TO INTERCEPT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3=VECTOR FOR SPACING</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4=FOR VECTOR TO APPROACH</td>
</tr>
</tbody>
</table>

- 196 -
<table>
<thead>
<tr>
<th>MESSAGE NUMBER</th>
<th>TYPE</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>070</td>
<td>VECTORING MSG 2</td>
<td>CID(N3),#,HDG(N3),REASON(A1), DATA(A15),NAVAID(A3),RADIAL(N3)</td>
</tr>
<tr>
<td></td>
<td>REASON: 0=none given</td>
<td>1=FOR VECTOR TO FIX 2=FOR VECTOR TO INTERCEPT 3=VECTOR FOR SPACING 4=FOR VECTOR TO APPROACH</td>
</tr>
<tr>
<td>071</td>
<td>VECTORING MSG 3</td>
<td>CID(N3),#,REASON(A1), DATA(A15),NAVAID(A3),RADIAL(N3)</td>
</tr>
<tr>
<td></td>
<td>REASON: 0=none given</td>
<td>1=FOR VECTOR TO FIX 2=FOR VECTOR TO INTERCEPT 3=VECTOR FOR SPACING 4=FOR VECTOR TO APPROACH</td>
</tr>
<tr>
<td>072</td>
<td>VECTORING MSG 4</td>
<td>CID(N3),#,FIX(A15),HDG(N3),REASON(A1), DATA(A15),NAVAID(A3),RADIAL(N3)</td>
</tr>
<tr>
<td></td>
<td>REASON: 0=none given</td>
<td>1=FOR VECTOR TO FIX 2=FOR VECTOR TO INTERCEPT 3=VECTOR FOR SPACING 4=FOR VECTOR TO APPROACH</td>
</tr>
<tr>
<td>073</td>
<td>VECTORING MSG 5</td>
<td>CID(N3),#,NUM(N3),TURN(A1),REASON(A1), DATA(A15),NAVAID(A3),RADIAL(N3)</td>
</tr>
<tr>
<td></td>
<td>REASON: 0=none given</td>
<td>1=FOR VECTOR TO FIX 2=FOR VECTOR TO INTERCEPT 3=VECTOR FOR SPACING 4=FOR VECTOR TO APPROACH</td>
</tr>
<tr>
<td>074</td>
<td>VECTORING MSG 6</td>
<td>CID(N3),#</td>
</tr>
<tr>
<td>075</td>
<td>VECTORING MSG 7</td>
<td>CID(N3),#,HDG(N3),FIX(A15)</td>
</tr>
<tr>
<td>076</td>
<td>SPEED MSG 1</td>
<td>CID(N3),#</td>
</tr>
<tr>
<td>077</td>
<td>SPEED MSG 2</td>
<td>CID(N3),#</td>
</tr>
<tr>
<td>078</td>
<td>SPEED MSG 3</td>
<td>CID(N3),#,TYPE(A1),CODE(A1),NUM(A3)</td>
</tr>
<tr>
<td></td>
<td>TYPE: 1=INCREASE, 2=REDUCE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CODE: 1=TO number</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2=TO MACH number</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3=number KNOTS</td>
<td></td>
</tr>
<tr>
<td>079</td>
<td>SPEED MSG 4</td>
<td>CID(N3),#,CODE(A1),NUM(A3)</td>
</tr>
<tr>
<td></td>
<td>CODE: 1=MACH number</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2=number KNOTS</td>
<td></td>
</tr>
<tr>
<td>MESSAGE NUMBER</td>
<td>TYPE</td>
<td>DATA</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>080</td>
<td>SPEED MSG 5</td>
<td>CID(N3), #(N3), TYPE(A1), CODE(A1), NUM(A3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TYPE: 1=INCREASE, 2=REDUCE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CODE: 1=TO number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2=TO MACH number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3=number KNOTS</td>
</tr>
<tr>
<td>081</td>
<td>SPEED MSG 6</td>
<td>CID(N3), #(N3), ALT(N3), CODE(A1), NUM(A3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CODE: 1=TO number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2=number KNOTS</td>
</tr>
<tr>
<td>082</td>
<td>SPEED MSG 7</td>
<td>CID(N3), #(N3), ALT(N3), CODE(A1), NUM(A3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CODE: 1=TO number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2=TO MACH number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3=number KNOTS</td>
</tr>
<tr>
<td>083</td>
<td>SPEED MSG 8</td>
<td>CID(N3), #(N3), ALT(N3), FIX(A15), NUM(A3)</td>
</tr>
<tr>
<td>084</td>
<td>SPEED MSG 9</td>
<td>CID(N3), #(N3), NUM(A3)</td>
</tr>
<tr>
<td>085</td>
<td>SPEED MSG 10</td>
<td>CID(N3), #(N3)</td>
</tr>
</tbody>
</table>
Inbound Messages from the Host

<table>
<thead>
<tr>
<th>MESSAGE NUMBER</th>
<th>TYPE</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>NAK</td>
<td>none</td>
</tr>
<tr>
<td>001</td>
<td>ACK</td>
<td>none</td>
</tr>
<tr>
<td>003</td>
<td>AC INFO</td>
<td>CID(N3), CALLSIGN(A15), ALT(N3), HDG(N3), GS(N3), MODE(A1), ACTYPE(A10)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MODE: 1=LEVEL, 2=CLIMBING, 3=DESCENDING, 4=MANEUVERING</td>
</tr>
<tr>
<td>004</td>
<td>MSG ACK</td>
<td>CID(N3), #(N3), TYPE(A1), CALLSIGN(A15)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TYPE: 0=NEGATIVE, 1=POSITIVE, 3=NEUTRAL</td>
</tr>
<tr>
<td>005</td>
<td>AC MSG</td>
<td>CID(N3), CALLSIGN(A15), TYPE(N2), DATA(A30)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TYPE=MSG NUMBER</td>
</tr>
<tr>
<td>007</td>
<td>PIREP DATA</td>
<td>DATA(A30), ENDFLAG(A1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ENDFLAG: control byte signaling last record in transmission</td>
</tr>
<tr>
<td>008</td>
<td>ALTIMETER DATA</td>
<td>STATION(A3), TIME(A4), VALUE(A4)</td>
</tr>
<tr>
<td>009</td>
<td>NAVAID DATA</td>
<td>STATION(A3), FREQ(A6), STATUS(A15)</td>
</tr>
<tr>
<td>010</td>
<td>AC LIST DATA</td>
<td>DEST(A3), CALLSIGN(A15), CID(N3), ALT(N3), HDG(N3), GS(N3), MODE(A1), EOL(A1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MODE: L=LEVEL, C=CLIMBING, D=DESCENDING, M=MANEUVERING</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EOL: 0=NOT END OF LIST, 1=END OF LIST</td>
</tr>
</tbody>
</table>
APPENDIX F

Simulator Source Code

The following pages are the source code for the simulator. The first part is the code for the device followed by the code for the host.
DECLARE SUB MOUSE (M1%, M2%, M3%, M4%)
DEFINT A-Z
DIM CURSOR%(15, 1)
DIM MSGNUM$(9)
DIM MSGCID$(9)
DIM MSGTEXT$(9)
DIM MSGCODE$(9)
DIM MSGTYPE$(9)
DIM MSGCALLS$(9)
DIM MSGDATA$(9)
DIM TALTMENU$(36)
DIM ALTMENU$(36)
FOR I = 0 TO 8
    MSGNUM$ = " "
    MSGCID$ = " "
    MSGTEXT$ = " "
    MSGCODE$ = " "
    MSGTYPE$ = " 
    MSGCALLS$ = " 
NEXT I
BLANK$ = " 
BLANK2$ = " 
FOR I = 0 TO 35
    READ ALTMENU$(I)
NEXT I
DATA "510 
DATA " 
DATA "470 
DATA " 
DATA "430 
DATA " 
DATA "390 
DATA " 
DATA "350 
DATA " 
DATA "310 
DATA " 
DATA "290 
DATA " 
DATA "270 
DATA " 
DATA "250 

DATA " 
DATA "490 "
DATA " 
DATA "450 "
DATA " 
DATA "410 "
DATA " 
DATA "370 "
DATA " 
DATA "330 "
DATA " 
DATA "300 "
DATA " 
DATA "280 "
DATA " 
DATA "260 "
DATA " 
DATA "240 "
DATA " 
OPEN "COM1:4800,N,8,1,CS,DS,CD" FOR RANDOM AS 1 LEN = 640
FIELD #1, 128 AS CBUFF$
FIELD #1, 1 AS CBYTE$
ON TIMER(10) GOSUB TIMEOUT
KEY OFF
ON KEY(9) GOSUB ENDPGM  'F9 TERMINATES PROGRAM
KEY(9) ON
OLDTIME1 = TIMER
SCREEN 2
CLS
BLANK = 0
LINENUM = 22: LINEIND = 23: MSGNO$ = "000"
REM ****************************************************************************
REM CHECK IF MOUSE DRIVER IS INSTALLED AND SET UP MOUSE
REM ****************************************************************************
DEF SEG = 0
MSEG = 256 * PEEK(51 * 4 + 3) + PEEK(51 * 4 + 2)
MSE = 256 * PEEK(51 * 4 + 1) + PEEK(51 * 4)
IF MSEG OR (MSE - 2) THEN
  PRINT " "
ELSE
  PRINT "MOUSE DRIVER NOT FOUND"
  PRINT "PRESS ANY KEY"
  DTA$ = ""
  WHILE DTA$ = ""
    DTA$ = INKEY$
  WEND
  SYSTEM
END IF
DEF SEG = MSEG: MSE = MSE + 2
IF PEEK(MSE - 2) = 207 THEN
    PRINT "MOUSE DRIVER NOT FOUND"
    PRINT "PRESS ANY KEY"
    DTA$ = ""
    WHILE DTA$ = ""
        DTA$ = INKEY$
    WEND
    SYSTEM
END IF
M1% = 0
CALL MOUSE(M1%, M2%, M3%, M4%)
M1% = 15: M3% = 8: M4% = 16
CALL MOUSE(M1%, M2%, M3%, M4%)
CURSOR%(O, 0) = &HFFFF
CURSOR%(1, 0) = &HFFFF
CURSOR%(2, 0) = &HFFFF
CURSOR%(3, 0) = &HFFFF
CURSOR%(4, 0) = &HFFFF
CURSOR%(5, 0) = &HFFFF
CURSOR%(6, 0) = &HFFFF
CURSOR%(7, 0) = &HFFFF
CURSOR%(8, 0) = &HFFFF
CURSOR%(9, 0) = &HFFFF
CURSOR%(10, 0) = &HFFFF
CURSOR%(11, 0) = &HFFFF
CURSOR%(12, 0) = &HFFFF
CURSOR%(13, 0) = &HFFFF
CURSOR%(14, 0) = &HFFFF
CURSOR%(15, 0) = &HFFFF
CURSOR%(O, 1) = &H8000
CURSOR%(1, 1) = &HE000
CURSOR%(2, 1) = &HF800
CURSOR%(3, 1) = &HFEO0
CURSOR%(4, 1) = &HD800
CURSOR%(5, 1) = &HCO0
CURSOR%(6, 1) = &H600
CURSOR%(7, 1) = &H300
CURSOR%(8, 1) = &HO
CURSOR%(9, 1) = &HO
CURSOR%(10, 1) = &HO
CURSOR%(11, 1) = &HO
CURSOR%(12, 1) = &HO
CURSOR%(13, 1) = &HO
CURSOR%(14, 1) = &HO
CURSOR%(15, 1) = &HO
M1% = 9: M2% = 0: M3% = 0
CALL MOUSE(M1%, M2%, M3%, CURSOR%(O, 0))
M1% = 4: M3% = 620: M4% = 160
CALL MOUSE(M1%, M2%, M3%, M4%)
M1% = 1
CALL MOUSE(M1%, M2%, M3%, M4%)
REM *******************************************************
REM DRAW DEVICE OUTLINE
REM *******************************************************
MAINMENU:
PI! = 3.141593
CIRCLE (100, 10), 20, , PI! / 2, PI!
CIRCLE (500, 10), 20, , 0, PI! / 2
CIRCLE (100, 190), 20, , PI!, 3 * PI! / 2
CIRCLE (500, 190), 20, , 3 * PI! / 2, 0
LINE (100, 2)-(500, 2)
LINE (100, 198)-(500, 198)
LINE (80, 10)-(80, 190)
LINE (520, 10)-(520, 190)
LINE (170, 13)-(430, 187), , B
LINE (170, 163)-(430, 163)
LINE (300, 13)-(300, 163)
LOCATE 22, 15, 0
PRINT "ERROR"
CIRCLE (133, 185), 5
LINE (550, 113)-(585, 124), , B
LOCATE 17, 68: PRINT "PREV MENU"
LINE (25, 138)-(60, 149), , B
LOCATE 20, 4: PRINT "CLEAR"
LINE (550, 138)-(585, 149), , B
LOCATE 20, 70: PRINT "XMIT"
LINE (130, 14)-(165, 25), , B: LOCATE 3, 18, 0: PRINT "F1"
LINE (130, 30)-(165, 41), , B: LOCATE 5, 18, 0: PRINT "F2"
LINE (130, 46)-(165, 57), , B: LOCATE 7, 18, 0: PRINT "F3"
LINE (130, 62)-(165, 73), , B: LOCATE 9, 18, 0: PRINT "F4"
LINE (130, 78)-(165, 89), , B: LOCATE 11, 18, 0: PRINT "F5"
LINE (130, 94)-(165, 105), , B: LOCATE 13, 18, 0: PRINT "F6"
LINE (130, 110)-(165, 121), , B: LOCATE 15, 18, 0: PRINT "F7"
LINE (130, 126)-(165, 137), , B: LOCATE 17, 18, 0: PRINT "F8"
LINE (130, 142)-(165, 153), , B: LOCATE 19, 18, 0: PRINT "F9"
LINE (435, 14)-(470, 25), , B: LOCATE 3, 56, 0: PRINT "F10"
LINE (435, 30)-(470, 41), , B: LOCATE 5, 56, 0: PRINT "F11"
LINE (435, 46)-(470, 57), , B: LOCATE 7, 56, 0: PRINT "F12"
LINE (435, 62)-(470, 73), , B: LOCATE 9, 56, 0: PRINT "F13"
LINE (435, 78)-(470, 89), , B: LOCATE 11, 56, 0: PRINT "F14"
LINE (435, 94)-(470, 105), , B: LOCATE 13, 56, 0: PRINT "F15"
LINE (435, 110)-(470, 121), , B: LOCATE 15, 56, 0: PRINT "F16"
LINE (435, 126)-(470, 137), , B: LOCATE 17, 56, 0: PRINT "F17"
LINE (435, 142)-(470, 153), , B: LOCATE 19, 56, 0: PRINT "F18"
REM ************* DISPLAY MAIN MENU *************
VIEW (171, 14)–(429, 186)
CLS
VIEW
MAIN = 1
LOCATE 3, 23: PRINT "VECTORING"
LOCATE 3, 39: PRINT "TRAFFIC"
LOCATE 4, 23: PRINT BLANK2$ 
LOCATE 4, 39: PRINT BLANK2$ 
LOCATE 5, 23: PRINT "ALTITUDE"
LOCATE 5, 39: PRINT "COMMUNICATION"
LOCATE 6, 23: PRINT BLANK2$
LOCATE 6, 39: PRINT BLANK2$
LOCATE 7, 23: PRINT "SPEED"
LOCATE 7, 39: PRINT "ALERT"
LOCATE 8, 23: PRINT BLANK2$
LOCATE 8, 39: PRINT BLANK2$
LOCATE 9, 23: PRINT "HOLDING"
LOCATE 9, 39: PRINT "EMERGENCY"
LOCATE 10, 23: PRINT BLANK2$
LOCATE 10, 39: PRINT BLANK2$
LOCATE 11, 23: PRINT "ROUTING"
LOCATE 11, 39: PRINT "RADAR"
LOCATE 12, 23: PRINT BLANK2$
LOCATE 12, 39: PRINT BLANK2$
LOCATE 13, 23: PRINT "ARRIVALS"
LOCATE 13, 39: PRINT "WEATHER"
LOCATE 14, 23: PRINT BLANK2$
LOCATE 14, 39: PRINT BLANK2$
LOCATE 15, 23: PRINT "ALTIMETER"
LOCATE 15, 39: PRINT "NAVAIDS"
LOCATE 16, 23: PRINT BLANK2$
LOCATE 16, 39: PRINT BLANK2$
LOCATE 17, 23: PRINT "ACKNOWLEDGE"
LOCATE 17, 39: PRINT "RE-XMIT"
LOCATE 18, 23: PRINT BLANK2$
LOCATE 18, 39: PRINT BLANK2$
LOCATE 19, 23: PRINT "AC LIST"
LOCATE 19, 39: PRINT "OTHER FUNCTIONS"
LOCATE 20, 23: PRINT BLANK2$
LOCATE 20, 39: PRINT BLANK2$
LOCATE 21, 23: PRINT BLANK$
LOCATE 22, 23: PRINT BLANK$
LOCATE 22, 23: PRINT BLANK$
LINE (170, 163)–(430, 163)
LINE (300, 13)–(300, 163)
REM ****************************************************
REM MAIN
REM ****************************************************
REM 205
MAINLOOP:
  NEWTIME! = TIMER
  IF (NEWTIME! - OLDTIME!) > 1 THEN
    GOSUB POLLHOST  'POLL HOST EVERY SECOND
    OLDTIME! = NEWTIME!
  END IF
  IF MAIN = 0 THEN
    GOTO MAINMENU   'DISPLAY MAIN MENU IF NOT DISPLAYED
  END IF
  GOSUB KEYMOUSE    'READ KEYBOARD AND MOUSE
  IF DTA$ = "" AND FKEY = 0 THEN
    GOTO MAINLOOP
  END IF
  TIMER ON
  IF DTA$ = "" THEN
    REM ****** CHECK WHICH PF KEY WAS PRESSED IF ANY ******
    IF FKEY = 1 THEN
      MAIN = 0
      GOSUB F1KEY
      TIMER OFF
      GOTO MAINLOOP
    END IF
    IF FKEY = 2 THEN
      MAIN = 0
      GOSUB F2KEY
      TIMER OFF
      GOTO MAINLOOP
    END IF
    IF FKEY = 3 THEN
      LOCATE 22, 23
      PRINT "SPEED FUNCTION NOT ACTIVE"
      LOCATE 23, 23
      PRINT "PRESS ANY KEY"
      GOSUB NOFUNCT
      TIMER OFF
      GOTO MAINLOOP
    END IF
    IF FKEY = 4 THEN
      LOCATE 22, 23
      PRINT "HOLDING FUNCTION NOT ACTIVE"
      LOCATE 23, 23
      PRINT "PRESS ANY KEY"
      GOSUB NOFUNCT
      TIMER OFF
      GOTO MAINLOOP
    END IF
    IF FKEY = 5 THEN
      LOCATE 22, 23
PRINT "ROUTING FUNCTION NOT ACTIVE"
LOCATE 23, 23
PRINT "PRESS ANY KEY"
GOSUB NOFUNCT
TIMER OFF
GOTO MAINLOOP
END IF
IF FKEY = 6 THEN
    LOCATE 22, 23
    PRINT "ARRIVALS FUNCTION NOT ACTIVE"
    LOCATE 23, 23
    PRINT "PRESS ANY KEY"
    GOSUB NOFUNCT
    TIMER OFF
    GOTO MAINLOOP
END IF
IF FKEY = 7 THEN
    LOCATE 22, 23
    PRINT "ALTIMETER FUNCTION NOT ACTIVE"
    LOCATE 23, 23
    PRINT "PRESS ANY KEY"
    GOSUB NOFUNCT
    TIMER OFF
    GOTO MAINLOOP
END IF
IF FKEY = 8 THEN
    LOCATE 22, 23
    PRINT "ACKNOWLEDGE FUNCTION NOT ACTIVE"
    LOCATE 23, 23
    PRINT "PRESS ANY KEY"
    GOSUB NOFUNCT
    TIMER OFF
    GOTO MAINLOOP
END IF
IF FKEY = 9 THEN
    LOCATE 22, 23
    PRINT "AC LIST FUNCTION NOT ACTIVE"
    LOCATE 23, 23
    PRINT "PRESS ANY KEY"
    GOSUB NOFUNCT
    TIMER OFF
    GOTO MAINLOOP
END IF
IF FKEY = 10 THEN
    LOCATE 22, 23
    PRINT "TRAFFIC FUNCTION NOT ACTIVE"
    LOCATE 23, 23
    PRINT "PRESS ANY KEY"
GOSUB NOFUNCT
TIMER OFF
GOTO MAINLOOP
END IF

IF FKEY = 11 THEN
LOCATE 22, 23
PRINT "COMMUNICATION FUNCTION NOT ACTIVE. PRESS ANY KEY"
LOCATE 23, 23
PRINT "ACTIVE."
GOSUB NOFUNCT
TIMER OFF
GOTO MAINLOOP
END IF

IF FKEY = 12 THEN
LOCATE 22, 23
PRINT "ALERT FUNCTION NOT ACTIVE"
LOCATE 23, 23
PRINT "PRESS ANY KEY"
GOSUB NOFUNCT
TIMER OFF
GOTO MAINLOOP
END IF

IF FKEY = 13 THEN
LOCATE 22, 23
PRINT "EMERGENCY FUNCTION NOT ACTIVE"
LOCATE 23, 23
PRINT "PRESS ANY KEY"
GOSUB NOFUNCT
TIMER OFF
GOTO MAINLOOP
END IF

IF FKEY = 14 THEN
LOCATE 22, 23
PRINT "RADAR FUNCTION NOT ACTIVE"
LOCATE 23, 23
PRINT "PRESS ANY KEY"
GOSUB NOFUNCT
TIMER OFF
GOTO MAINLOOP
END IF

IF FKEY = 15 THEN
LOCATE 22, 23
PRINT "WEATHER FUNCTION NOT ACTIVE"
LOCATE 23, 23
PRINT "PRESS ANY KEY"
GOSUB NOFUNCT
TIMER OFF
GOTO MAINLOOP

- 208 -
END IF
IF FKEY = 16 THEN
  LOCATE 22, 23
  PRINT "NAVAIDS FUNCTION NOT ACTIVE"
  LOCATE 23, 23
  PRINT "PRESS ANY KEY"
  GOSUB NOFUNCT
  TIMER OFF
  GOTO MAINLOOP
END IF
IF FKEY = 17 THEN
  MAIN = 0
  GOSUB F17KEY
  TIMER OFF
  GOTO MAINLOOP
END IF
IF FKEY = 18 THEN
  LOCATE 22, 23
  PRINT "OTHER FUNCTIONS NOT ACTIVE"
  LOCATE 23, 23
  PRINT "PRESS ANY KEY"
  GOSUB NOFUNCT
  TIMER OFF
  GOTO MAINLOOP
END IF
GOTO MAINLOOP
END IF
REM **** CHECK IF NUMERIC KEY IS PRESSED *****
IF ASC(DTA$) = 27 THEN
  TIMER OFF
  GOSUB CLEARPRESS
  GOTO MAINLOOP
END IF
GOSUB CLEARPRESS
GOSUB PRINTIT
IF ASC(DTA$) < 48 OR ASC(DTA$) > 57 THEN
  TIMER OFF
  BEEP 'NOT A DIGIT
  GOTO MAINLOOP
END IF
CID$ = DTA$
SECDIGIT:
  DTA$ = ""
  WHILE DTA$ = ""
    GOSUB KEYMOUSE
    IF FKEY <> 0 THEN
      BEEP
  END IF
WEND
IF ASC(DTA$) = 27 THEN
    TIMER OFF
    GOSUB CLEARPRESS
    GOTO MAINLOOP
END IF
GOSUB PRINTIT
IF ASC(DTA$) < 48 OR ASC(DTA$) > 57 THEN
    BEEP
    GOTO SECDIGIT
END IF
CID$ = CID$ + DTA$
TIRDDIGIT:
    DTA$ = "$"
    WHILE DTA$ = ""
        GOSUB KEYMOUSE
        IF FKEY <> 0 THEN
            BEEP
        END IF
    WEND
IF ASC(DTA$) = 27 THEN
    GOSUB CLEARPRESS
    TIMER OFF
    GOTO MAINLOOP
END IF
GOSUB PRINTIT
IF ASC(DTA$) < 48 OR ASC(DTA$) > 57 THEN
    BEEP
    GOTO SECDIGIT
END IF
TIMER OFF
CID$ = CID$ + DTA$
REM ***** SEND CID REQUEST **********
BC% = 0
MSG$ = "003" + CID$
FOR I = 4 TO 6
    BC% = BC% + ASC(MID$(MSG$, I, 1))
NEXT I
BCC$ = MKI$(BC%)
MSG$ = MSG$ + BCC$
SCIDLAB0:
    LSET CBUFF$ = MSG$
    PUT #1, 8
    N = LOC(1)
    WHILE N < 2
        N = LOC(1)
    WEND
    GET #1, 2
MSGT$ = LEFT$(CBUFF$, 2)
IF MSGT$ <> "02" THEN
   GOTO SCIDLABO
END IF
N = LOC(1)
WHILE N < 3
   N = LOC(1)
WEND
GET #1, 3
CID$ = LEFT$(CBUFF$, 3)
MSG$ = CID$
IF CID$ <> "000" THEN
   N = LOC(1)
   WHILE N < 35
      N = LOC(1)
   WEND
   GET #1, 35
   MSG$ = MSG$ + LEFT$(CBUFF$, 35)
END IF
N = LOC(1)
WHILE N < 2
   N = LOC(1)
WEND
GET #1, 2
BCC$ = LEFT$(CBUFF$, 2)
BCC% = CVI(BCC$)
BC% = 0
FOR I = 1 TO LEN(MSG$)
   BC% = BC% + ASC(MID$(MSG$, I, 1))
NEXT I
IF BCC% <> BC% THEN
   GOTO SCIDLABO
END IF
IF CID$ = "000" THEN
   BEEP
   LOCATE 23, 23
   PRINT " INVALID CID, PRESS ANY KEY"
   DTA$ = ""
   TIMER ON
   WHILE DTA$ = ""
      DTA$ = INKEY$
   WEND
   TIMER OFF
   GOSUB CLEARPRESS
   GOTO MAINLOOP
END IF
CALLS$ = MID$(MSG$, 4, 15)
ALT$ = MID$(MSG$, 19, 3)
HDG$ = MID$(MSG$, 22, 3)
GS$ = MID$(MSG$, 25, 3)
MODE$ = MID$(MSG$, 28, 1)
TYPEAC$ = MID$(MSG$, 29, 10)
LINENUM = 22: LINEIND = 23
FOR I = 1 TO LEN(CALLS$)
    DTA$ = MID$(CALLS$, I, 1)
    GOSUB PRINTIT
NEXT I
PRINT " ": LINEIND = LINEIND + 1
GOTO MAINLOOP
REM *****************************************************
REM ROUTINE TO WAIT FOR OPERATOR AND REDISPLAY CALLSIGN
REM *****************************************************
NOFUNCT:
    BEEP
    DTA$ = ""
    WHILE DTA$ = ""
        DTA$ = INKEY$
    WEND
    LOCATE 22, 23
    PRINT BLANK$
    LOCATE 23, 23
    PRINT BLANK$
    IF LEN(CALLS$) = 0 THEN
        RETURN
    END IF
    LINENUM = 22
    LINEIND = 23
    FOR I = 1 TO LEN(CALLS$)
        DTA$ = MID$(CALLS$, I, 1)
        GOSUB PRINTIT
    NEXT I
    RETURN
REM *****************************************************
REM F1 KEY ROUTINE - VECTORING
REM *****************************************************
F1KEY:
    IF CID$ = "" OR CID$ = "000" THEN
        LOCATE 22, 23: PRINT "NO AC IDENTIFIED"
        LOCATE 23, 23: PRINT BLANK$
        BEEP
        RETURN
    END IF
    FOR I = 3 TO 20
        IF I = 19 THEN
            LOCATE I, 23
            PRINT "TURN LEFT"
        END IF
    NEXT I
LOCATE I, 39
PRINT "TURN RIGHT   "
ELSE
  IF I = 20 THEN
    LOCATE I, 23
    PRINT " TO HEADING   "
    LOCATE I, 39
    PRINT " TO HEADING   "
  ELSE
    LOCATE I, 23
    PRINT BLANK2$ 
    LOCATE I, 39
    PRINT BLANK2$ 
  END IF
END IF
NEXT I
GOOD = 0
WHILE GOOD = 0
  FKEY = 0
  DTA$ = ""
  WHILE FKEY = 0 AND DTA$ = ""
    GOSUB KEYMOUSE
  WEND
  IF FKEY = 9 OR FKEY = 18 THEN
    GOOD = 1
  ELSE
    IF DTA$ <> "" THEN
      IF ASC(DTA$) = 27 THEN
        GOSUB CLEARPRESS
        RETURN
      ELSE
        IF ASC(RIGHT$(DTA$, 1)) = 83 THEN
          RETURN
        ELSE
          BEEP
        END IF
      END IF
    ELSE
      BEEP
    END IF
  END IF
END IF
WEND
IF FKEY = 9 THEN
  TURN$ = "L"
ELSE
  TURN$ = "R"
END IF
LOCATE LINENUM, LINEIND
PRTSTR$ = " TURN " + TURN$ + " "
FOR I = 1 TO LEN(PRTSTR$)
    DTA$ = MID$(PRTSTR$, I, 1)
    GOSUB PRINTIT
NEXT I
HDG$ = ""
X = LINEIND
Y = LINENUM
GOOD = 0
WHILE GOOD = 0
    FKEY = 0: DTA$ = ""
    WHILE FKEY = 0 AND DTA$ = ""
        GOSUB KEYMOUSE
    WEND
    IF DTA$ <> "" THEN
        IF ASC(DTA$) = 27 THEN
            GOSUB CLEARPRESS
            RETURN
        END IF
        IF ASC(RIGHT$(DTA$, 1)) = 83 THEN
            DTA$ = CHR$(8)
            FOR I = 1 TO (8 + LEN(HDG$))
                GOSUB PRINTIT
            NEXT I
            GOTO FKEY
        END IF
        IF ASC(DTA$) = 13 THEN
            IF LEN(HDG$) <> 3 THEN
                BEEP
            ELSE
                IF HDG$ < "000" OR HDG$ > "360" THEN
                    BEEP
                ELSE
                    GOOD = 1
                END IF
            END IF
        END IF
    END IF
    IF GOOD = 0 THEN
        IF (ASC(DTA$) < 48 OR ASC(DTA$) > 57) AND (ASC(DTA$) <> 8) THEN
            BEEP
            ELSE
                GOSUB PRINTIT
                IF ASC(DTA$) = 8 THEN
                    HDG$ = LEFT$(HDG$, (LEN(HDG$) - 1))
                ELSE
                    HDG$ = HDG$ + DTA$
                END IF
        END IF
REM ********************************************************************************
REM F2 KEY ROUTINE - ALTITUDE
REM ********************************************************************************
F2KEY:

   IF CID$ = "" OR CID$ = "000" THEN
      LOCATE 22, 23: PRINT "NO AC IDENTIFIED"
      LOCATE 23, 23: PRINT BLANK$ 
      BEEP
      RETURN
   END IF
   PD$ = "0"
   FOR I = 3 TO 20
      IF I = 19 THEN
         LOCATE I, 23
         PRINT "AT PILOT'S"
         LOCATE I, 39
         PRINT "CLIMB/DESCEND"
      ELSE
         IF I = 20 THEN
            LOCATE I, 23
            PRINT "DESCRIPTI ON"
            LOCATE I, 39
            PRINT BLANK2$
         ELSE
            LOCATE I, 23
            PRINT BLANK2$
            LOCATE I, 39
            PRINT BLANK2$
         END IF
      END IF
   NEXT I
GOOD = 0
WHILE GOOD = 0
  FKEY = 0: DTA$ = ""
  WHILE FKEY = 0 AND DTA$ = ""
    GOSUB KEYMOUSE
  WEND
  IF DTA$ <> "" THEN
    IF ASC(DTA$) = 27 THEN
      GOSUB CLEARPRESS
      RETURN
    ELSE
      IF ASC(RIGHT$(DTA$, 1)) = 83 THEN
        IF PDS <> "1" THEN
          RETURN
        ELSE
          DTA$ = CHR$(8)
          FOR I = 1 TO 6
            GOSUB PRINTIT
          NEXT I
          RETURN
      END IF
    ELSE
      BEEP
    END IF
  END IF
ELSE
  IF PKEY = 9 OR PKEY = 18 THEN
    GOOD = 1
  ELSE
    BEEP
  END IF
END IF
WEND
IF PKEY = 9 THEN
  IF PDS <> "1" THEN
    DTASTR$ = " AT PD"
    FOR I = 1 TO 6
      DTA$ = MIDS(DTASTR$, I, 1)
      GOSUB PRINTIT
    NEXT I
  END IF
  PDS = "1"
ELSE
  PDS = "0"
END IF
IF ALT$ <> "" THEN
  MANOUVER$ = "U"
ELSE
LOCATE 22, 23: PRINT "CLIMB"
LOCATE 22, 39: PRINT "DESCEND"
LOCATE 23, 23: PRINT BLANK2$
LOCATE 23, 39: PRINT BLANK2$
FKEY = 0
WHILE FKEY = 0
    FKEY = 0: DTA$ = ""
    WHILE FKEY = 0 AND DTA$ = ""
        GOSUB KEYMOUSE
    WEND
    IF FKEY = 0 THEN
        IF ASC(DTA$) = 27 THEN
            GOSUB CLEARPRESS
            RETURN
        END IF
        IF ASC(RIGHT$(DTA$, 1)) = 83 THEN
            IF PD$ <> "1" THEN
                GOTO F2KEY
            ELSE
                DTA$ = CHR$(8)
                FOR I = 1 TO 6
                    GOSUB PRINTIT
                NEXT I
                GOTO F2KEY
            END IF
            END IF
            BEEP
        ELSE
            IF FKEY <> 9 AND FKEY <> 18 THEN
                BEEP
                FKEY = 0
            END IF
        END IF
    WEND
    IF FKEY = 9 THEN
        MANOUVERS$ = "C"
    ELSE
        MANOUVERS$ = "D"
    END IF
END IF
J = 0
FOR I = 3 TO 20
    LOCATE I, 23
    PRINT ALTMENU$(J)
    J = J + 1
NEXT I
FOR I = 3 TO 20
    LOCATE I, 39
PRINT ALTMENU$(J)
  J = J + 1
NEXT I
TALT$ = ""
WHILE TALT$ = ""
  FKEY = 0: DTA$ = ""
  WHILE FKEY = 0 AND DTA$ = ""
    GOSUB KEYMOUSE
  END WHILE
  IF FKEY = 0 THEN
    IF ASC(DTA$) = 27 THEN
      GOSUB CLEARPRESS
      RETURN
    END IF
    IF ASC(RIGHT$(DTA$, 1)) = 83 THEN
      IF PD$ <> "1" THEN
        GOTO F2KEY
      ELSE
        DTA$ = CHR$(8)
        FOR I = 1 TO 6
          GOSUB PRINTIT
        NEXT I
        GOTO F2KEY
      END IF
    END IF
  END IF
  BEEP
ELSE
  J = (FKEY * 2) - 2
  IF ALTMENU$(J) = BLANK2$ THEN
    BEEP
  ELSE
    TALT$ = LEFT$(ALTMENU$(J), 3)
  END IF
END IF
WEND
IF MANOUVER$ = "U" THEN
  IF ALT$ > TALT$ THEN
    MANOUVER$ = "D"
  ELSE
    MANOUVER$ = "C"
  END IF
ENDIF
IF MANOUVER$ = "C" THEN
  DTASTR$ = " CLIMB & MAINTAIN "
  MSGT$ = "029"
ELSE
  DTASTR$ = " DESCEND & MAINTAIN "
  MSGT$ = "030"
END IF
IF TALT$ < "180" THEN
    DTASTR$ = DTASTR$ + TALT$ + "M"
ELSE
    DTASTR$ = DTASTR$ + "FL" + TALT$
END IF
FOR I = 1 TO LEN(DTASTR$)
    DTA$ = MIDS(DTASTR$, I, 1)
    GOSUB PRINTIT
NEXT I
DTA$ = ""
WHILE DTA$ = ""
    FKEY = 0
    WHILE DTA$ = "" AND FKEY = 0
        GOSUB KEYMOUSE
    WEND
    IF FKEY <> 0 THEN
        BEEP
    ELSE
        IF ASC(DTA$) = 27 THEN
            GOSUB CLEARPRESS
            RETURN
        END IF
        IF ASC(RIGHT$(DTA$, 1)) = 83 THEN
            IF PD$ = "1" THEN
                DTA$ = CHR$(8)
                FOR I = 1 TO 6
                    GOSUB PRINTIT
                NEXT I
            END IF
            PD$ = "0"
            DTA$ = CHR$(8)
            FOR I = 1 TO LEN(DTASTR$)
                GOSUB PRINTIT
            NEXT I
            GOTO F2KEY
        END IF
        IF ASC(DTA$) <> 13 THEN
            BEEP
            DTA$ = ""
        END IF
    END IF
    WEND
    GOSUB SHIFTMMSG
    GOSUB GETMSGNUM
    MSGNUM$(8) = MSGNO$
    MSGCID$(8) = CID$
    MSGCODE$(8) = "O "
END IF
MSGTYPE$(8) = MSGT$
MSGCALLS$(8) = CALLS$
IF PD$ = "1" THEN
    DTASTR$ = "PD" + DTASTR$
END IF
MSGTEXT$(8) = DTASTR$
MSG$ = MSGT$ + CID$ + MSGNO$ + TALT$ + PD$ + "0"
MSGDATAS$(8) = MSG$
GOSUB SENDMSG
GOSUB CLEARPRESS
RETURN

REM ⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋯
END IF
IF MSGCODE$(I) <> "0" THEN
    BEEP
ELSE
    GOOD = 1
END IF
END IF
WEND
MSG$ = MSGDATA$(I)
GOSUB SENDMSG
RETURN

REM **********************************************************
REM ROUTINE TO ALERT CONTROLLER THAT 10 SECONDS HAS PASSED
REM **********************************************************
TIMEOUT:
    BEEP
    RETURN

REM **********************************************************
REM SUBROUTINE TO POSITION CURSOR ON LINE AND PRINT CHARACTER
REM **********************************************************
PRINTIT:
    IF ASC(DTA$) = 13 OR ASC(DTA$) = 27 THEN
        RETURN
    END IF
    IF ASC(DTA$) = 8 THEN
        LINEIND = LINEIND - 1
        IF LINEIND < 23 THEN
            LINENUM = 22
            LINEIND = 53
        END IF
        LOCATE LINENUM, LINEIND: PRINT " \\r
        RETURN
    END IF
    IF (ASC(DTA$) = 46) OR (ASC(DTA$) = 32) OR
        (ASC(DTA$) > 47 AND ASC(DTA$) < 58) OR
        (ASC(DTA$) > 64 AND ASC(DTA$) < 91) THEN
        IF LINEIND > 53 THEN
            LINENUM = 23
            LINEIND = 23
        END IF
        IF ASC(DTA$) = 32 THEN
            BLANK = BLANK + 1
            IF BLANK < 2 THEN
                LOCATE LINENUM, LINEIND
                PRINT DTA$
                LINEIND = LINEIND + 1
            END IF
        ELSE
BLANK = 0  
LOCATE LINENUM, LINEIND  
PRINT DTA$  
LINEIND = LINEIND + 1  
END IF  
END IF  
RETURN  
REM **************************************************  
REM SUBROUTINE ALWAYS PERFORMED WHEN CLEAR IS PRESSED  
REM **************************************************  
CLEARPRESS:
LOCATE 22, 23: PRINT BLANK$  
LOCATE 23, 23: PRINT BLANK$  
MSG$ = ""  
CID$ = ""  
CALLS$ = ""  
ALT$ = ""  
HDG$ = ""  
GS$ = ""  
MODE$ = ""  
TYPEAC$ = ""  
LINENUM = 22: LINEIND = 23  
RETURN  
REM **************************************************  
REM SUBROUTINE TO DRIVE MOUSE AND CHECK IF KEY IS PRESSED  
REM **************************************************  
KEYMOUSE:
DTA$ = ""  
FKEY = 0  
DTA$ = INKEY$  
IF DTA$ <> "" THEN  
   RETURN  
END IF  
M1% = 3  
CALL MOUSE(M1%, BT%, MX%, MY%)  
IF BT = 0 THEN  
   RETURN  
END IF  
IF MX > 129 AND MX < 166 AND MY > 13 AND MY < 26 THEN  
   FKEY = 1  
ELSE  
IF MX > 129 AND MX < 166 AND MY > 29 AND MY < 42 THEN  
   FKEY = 2  
ELSE  
IF MX > 129 AND MX < 166 AND MY > 45 AND MY < 58 THEN  
   FKEY = 3  
ELSE  
IF MX > 129 AND MX < 166 AND MY > 61 AND MY < 74 THEN
FKEY = 4
ELSE
IF MX > 129 AND MX < 166 AND MY > 77 AND MY < 90 THEN
  FKEY = 5
ELSE
IF MX > 129 AND MX < 166 AND MY > 95 AND MY < 106 THEN
  FKEY = 6
ELSE
IF MX > 129 AND MX < 166 AND MY > 109 AND MY < 122 THEN
  FKEY = 7
ELSE
IF MX > 129 AND MX < 166 AND MY > 125 AND MY < 138 THEN
  FKEY = 8
ELSE
IF MX > 129 AND MX < 166 AND MY > 141 AND MY < 154 THEN
  FKEY = 9
ELSE
IF MX > 434 AND MX < 471 AND MY > 13 AND MY < 24 THEN
  FKEY = 10
ELSE
IF MX > 434 AND MX < 471 AND MY > 29 AND MY < 42 THEN
  FKEY = 11
ELSE
IF MX > 434 AND MX < 471 AND MY > 45 AND MY < 58 THEN
  FKEY = 12
ELSE
IF MX > 434 AND MX < 471 AND MY > 61 AND MY < 74 THEN
  FKEY = 13
ELSE
IF MX > 434 AND MX < 471 AND MY > 77 AND MY < 90 THEN
  FKEY = 14
ELSE
IF MX > 434 AND MX < 471 AND MY > 93 AND MY < 106 THEN
  FKEY = 15
ELSE
IF MX > 434 AND MX < 471 AND MY > 109 AND MY < 122 THEN
  FKEY = 16
ELSE
IF MX > 434 AND MX < 471 AND MY > 125 AND MY < 138 THEN
  FKEY = 17
ELSE
IF MX > 434 AND MX < 471 AND MY > 141 AND MY < 154 THEN
  FKEY = 18
ELSE
IF MX > 24 AND MX < 61 AND MY > 137 AND MY < 150 THEN
  DTA$ = CHR$(27)
ELSE
IF MX > 549 AND MX < 586 AND MY > 137 AND MY < 150 THEN
DTA$ = CHR$(13)
ELSE
IF MX > 549 AND MX < 586 AND MY > 112 AND MY < 125 THEN
   DTA$ = MKI$(21248)
   END IF
   END IF
   END IF
   END IF
   END IF
   END IF
   END IF
   END IF
   END IF
   END IF
   END IF
   END IF
   END IF
   END IF
   END IF
   END IF
   END IF
RETURN
REM ***************************************************
REM
REM SUBROUTINE TO POLL HOST
REM ***************************************************
POLLHOST:
   LSET CBUFF$ = "002"
   PUT #1, 3
POLLL2:
   N = 0
   WHILE N < 2
      N = LOC(1)
   WEND
   GET #1, 2
   MSGT$ = LEFT$(CBUFF$, 2)
   IF MSGT$ = "00" THEN
      GOTO POLLHOST
   END IF
   IF MSGT$ = "01" THEN 'NO DATA SENT FROM HOST
      RETURN
   END IF
   IF MSGT$ <> "02" THEN
      GOTO MSG03
   END IF
REM ************** MSG TYPE 02 - AC DATA **************
N = LOC(1)
WHILE N < 40
    N = LOC(1)
WEND
GET #1, 40
MSG$ = LEFT$(CBUFF$, 38)
BCC$ = MID$(CBUFF$, 39, 2)
BCC% = CVI(BCC$)
BC% = 0
FOR I = 1 TO 38
    BYTE$ = MID$(MSG$, I, 1)
    BC% = BC% + ASC(BYTE$)
NEXT I
IF BCC% <> BC% THEN
    LSET CBUFF$ = "000"
    PUT #1, 3
    GOTO POLLL2
END IF
CID$ = MID$(MSG$, 1, 3)
CALLS$ = MID$(MSG$, 4, 15)
ALT$ = MID$(MSG$, 19, 3)
HDGS$ = MID$(MSG$, 22, 3)
GS$ = MID$(MSG$, 25, 3)
MODE$ = MID$(MSG$, 28, 1)
TYPEAC$ = MID$(MSG$, 29, 10)
LOCATE 22, 23: PRINT BLANK$:
LOCATE 23, 23: PRINT BLANK$
LINENUM = 22: LINEIND = 23
FOR I = 1 TO LEN(CALLS$)
    DTA$ = MID$(CALLS$, I, 1)
    GOSUB PRINTIT
NEXT I
RETURN

REM ************** MSG TYPE 03 - PILOT ACKNOWLEDGE **************
MSG03:
IF MSGT$ <> "03" THEN
    GOTO MSG04
END IF
N = LOC(1)
WHILE N < 24
    N = LOC(1)
WEND
GET #1, 24
MSG$ = LEFT$(CBUFF$, 22)
BCC$ = MID$(CBUFF$, 23, 2)
BCC% = CVI(BCC$)
BC% = 0
FOR I = 1 TO 22
BC% = BC% + ASC(MID$(MSG$, I, 1))
NEXT I
IF BCC% <> BC% THEN
    LSET CBUFF$ = "000"
    PUT #1, 3
    GOTO POLLL2
END IF
CID$ = MID$(MSG$, 1, 3)
NUM$ = MID$(MSG$, 4, 3)
J = -1
FOR I = 0 TO 8
    IF CID$ = MSGCID$(I) AND NUM$ = MSGNUM$(I) THEN
        J = I
    END IF
NEXT I
CALLS$ = MID$(MSG$, 8, 15)
IF J < 0 THEN
    VIEW (171, 14)-(429, 186)
    CLS
    MAIN = 0
    LOCATE 20, 23: PRINT "ACKNOWLEDGEMENT RECEIVED FROM "
    LOCATE 21, 23: PRINT CALLS$
    LOCATE 22, 23: PRINT "NO MESSAGE PENDING"
    LOCATE 23, 23: PRINT "PRESS ANY KEY"
    BEEP
    DTA$ = ""
    TIMER ON
    WHILE DTA$ = ""
        DTA$ = INKEY$
    WEND
    TIMER OFF
    GOSUB CLEARPRESS
    RETURN
END IF
MAIN = 0
CODE$ = MID$(MSG$, 7, 1)
IF CODE$ = "0" THEN
    MSGCODE$(J) = "A-"
    ANS$ = "NEGATIVE ACKNOWLEDGE"
ELSE
    IF CODE$ = "1" THEN
        MSGCODE$(J) = "A+
        ANS$ = "POSITIVE ACKNOWLEDGE"
    ELSE
        MSGCODE$(J) = "A"
        ANS$ = "MSG ACKNOWLEDGE"
    END IF
END IF
VIEW (171, 14)-(429, 186)
CLS
LOCATE 20, 23: PRINT ANS$ + " FROM"
LOCATE 21, 23: PRINT CALLS$
LOCATE 23, 23: PRINT "PRESS ANY KEY"
BEEP
DTA$ = ""
TIMER ON
WHILE DTA$ = ""
    DTA$ = INKEY$
WEND
TIMER OFF
GOSUB CLEARPRESS
RETURN
REM *************** MSG TYPE 04 - MSG FROM PILOT ***************
MSG04:
   IF MSGT$ <> "04" THEN
       GOTO MSG05
   END IF
   N = LOC(1)
   WHILE N < 50
       N = LOC(1)
   WEND
   GET #1, 50
   MSG$ = LEFT$(CBUFF$, 48)
   BCC$ = MID$(CBUFF$, 49, 2)
   BCC% = CVI(BCC$)
   BC% = 0
   FOR I = 1 TO 48
       BYTE$ = MID$(MSG$, I, 1)
       BC% = BC% + ASC(BYTE$)
   NEXT I
   IF BCC% <> BC% THEN
       LSET CBUFF$ = "000"
       PUT #1, 3
       GOTO POLLL2
   END IF
GOSUB SHIFTMSG
CID$ = MID$(MSG$, 1, 3)
CALLS$ = MID$(MSG$, 4, 15)
MSG$ = MID$(MSG$, 19, 30)
MSGCID$(8) = CID$
MSGCALLS$(8) = CALLS$
MSGTEXT$(8) = MSG$
MSGSNUM$(8) = "000"
MSGCOD$(8) = "I "
MSGTYPE$(8) = " "
VIEW (171, 14)-(429, 186)
CLS
MAIN = 0
LOCATE 18, 23: PRINT "INBOUND MESSAGE FROM"
LOCATE 19, 23: PRINT CALLS$
LOCATE 20, 23: PRINT MSG$
LOCATE 22, 23: PRINT "PRESS ANY KEY"
BEEP
DTA$ = ""
TIMER ON
WHILE DTA$ = ""
   DTA$ = INKEY$
WEND
TIMER OFF
GOSUB CLEARPRESS
RETURN
REM *************** MSG TYPE 05 - MEMORY LOAD ***************
MSG05:
   IF MSGT$ <> "05" THEN
      RETURN
   END IF
   N = LOC(1)
   WHILE N < 545
      N = LOC(1)
   WEND
   GET #1, 3
   LOC$ = LEFT$(CBUFF$, 3)
   BC% = 0
   FOR I = 1 TO 3
      BC% = BC% + ASC(MID$(LOC$, I, 1))
   NEXT I
   FOR I = 0 TO 35
      GET #1, 15
      TALTMENU$(I) = LEFT$(CBUFF$, 15)
      FOR J = 1 TO 15
         BC% = BC% + ASC(MID$(TALTMENU$(I), J, 1))
      NEXT J
   NEXT I
   GET #1, 2
   BCC$ = LEFT$(CBUFF$, 2)
   BCC% = CVI(BCC$)
   IF BCC% <> BC% THEN
      LSET CBUFF$ = "000"
      PUT #1, 3
      GOTO POLLL2
   END IF
   IF LOC$ <> "001" THEN
      LOCATE 22, 23: PRINT "INVALID MEMORY LOAD NUMBER: ", LOC$
LOCATE 23, 23: PRINT "RCVD FROM HOST, PRESS ANY KEY 
DTA$ = ""
TIMER ON
WHILE DTA$ = ""
    DTA$ = INKEY$
WEND
TIMER OFF
RETURN
END IF
MAIN = 0
LOCATE 3, 23: PRINT TALTMENU$(0)
LOCATE 3, 39: PRINT TALTMENU$(18)
LOCATE 4, 23: PRINT TALTMENU$(1)
LOCATE 4, 39: PRINT TALTMENU$(19)
LOCATE 5, 23: PRINT TALTMENU$(2)
LOCATE 5, 39: PRINT TALTMENU$(20)
LOCATE 6, 23: PRINT TALTMENU$(3)
LOCATE 6, 39: PRINT TALTMENU$(21)
LOCATE 7, 23: PRINT TALTMENU$(4)
LOCATE 7, 39: PRINT TALTMENU$(22)
LOCATE 8, 23: PRINT TALTMENU$(5)
LOCATE 8, 39: PRINT TALTMENU$(23)
LOCATE 9, 23: PRINT TALTMENU$(6)
LOCATE 9, 39: PRINT TALTMENU$(24)
LOCATE 10, 23: PRINT TALTMENU$(7)
LOCATE 10, 39: PRINT TALTMENU$(25)
LOCATE 11, 23: PRINT TALTMENU$(8)
LOCATE 11, 39: PRINT TALTMENU$(26)
LOCATE 12, 23: PRINT TALTMENU$(9)
LOCATE 12, 39: PRINT TALTMENU$(27)
LOCATE 13, 23: PRINT TALTMENU$(10)
LOCATE 13, 39: PRINT TALTMENU$(28)
LOCATE 14, 23: PRINT TALTMENU$(11)
LOCATE 14, 39: PRINT TALTMENU$(29)
LOCATE 15, 23: PRINT TALTMENU$(12)
LOCATE 15, 39: PRINT TALTMENU$(30)
LOCATE 16, 23: PRINT TALTMENU$(13)
LOCATE 16, 39: PRINT TALTMENU$(31)
LOCATE 17, 23: PRINT TALTMENU$(14)
LOCATE 17, 39: PRINT TALTMENU$(32)
LOCATE 18, 23: PRINT TALTMENU$(15)
LOCATE 18, 39: PRINT TALTMENU$(33)
LOCATE 19, 23: PRINT TALTMENU$(16)
LOCATE 19, 39: PRINT TALTMENU$(34)
LOCATE 20, 23: PRINT TALTMENU$(17)
LOCATE 20, 39: PRINT TALTMENU$(35)
LOCATE 22, 23: PRINT "READY TO LOAD THIS SCREEN"
LOCATE 23, 23: PRINT "XMIT TO ACCEPT, CLEAR TO REJECT"
GOOD = 0
DTA$ = ""
TIMER ON
WHILE GOOD = 0
  WHILE DTA$ = ""
    GOSUB KEYMOUSE
  WEND
  IF ASC(DTA$) = 27 THEN
    GOOD = 1
    TIMER OFF
    GOSUB CLEARPRESS
    RETURN
  ELSE
    IF ASC(DTA$) = 13 THEN
      GOOD = 1
    ELSE
      BEEP
    END IF
  END IF
WEND
TIMER OFF
FOR I = 0 TO 35
  ALTMENU$(I) = TALTMENU$(I)
NEXT I
GOSUB CLEARPRESS
RETURN
REM ************************************************************
REM SUBROUTINE TO SEND MESSAGE TO HOST
REM ************************************************************
SENDMSG:
  BC% = 0
  FOR I = 4 TO LEN(MSG$)
    BC% = BC% + ASC(MID$(MSG$, I, 1))
  NEXT I
  BCC$ = MKI$(BC%)
%SML1:
  LSET CBUFF$ = MSG$ + BCC$
  PUT #1, (LEN(MSG$) + 2)
  N = LOC(1)
  WHILE N < 2
    N = LOC(1)
  WEND
  GET #1, 2
  MSGT$ = LEFT$(CBUFF$, 2)
  IF MSGT$ = "00" THEN
    GOTO SML1
  IF MSGT$ = "01" THEN

- 230 -
LOCATE 22, 23: PRINT "UNABLE TO DELIVER MESSAGE"
LOCATE 23, 23: PRINT "PRESS ANY KEY"
DTA$ = ""
TIMER ON
WHILE DTA$ = ""
   DTA$ = INKEY$
WEND
TIMER OFF
GOSUB CLEARPRESS
RETURN

REM ********************************************************
REM SUBROUTINE TO SHIFT MESSAGES UP IN STACK
REM ********************************************************
SHIFTMSG:
   FOR I = 1 TO 8
      J = I - 1
      MSGNUM$(J) = MSGNUM$(I)
      MSGCID$(J) = MSGCID$(I)
      MSGTEXT$(J) = MSGTEXT$(I)
      MSGCODE$(J) = MSGCODE$(I)
      MSGTYPE$(J) = MSGTYPE$(I)
      MSGDATA$(J) = MSGDATA$(I)
      MSGCALLS$(J) = MSGCALLS$(I)
   NEXT I
RETURN

REM ********************************************************
REM SUBROUTINE TO GET NEXT MESSAGE NUMBER
REM ********************************************************
GETMSGNUM:
   BYTE1$ = MID$(MSGNO$, 1, 1)
   BYTE2$ = MID$(MSGNO$, 2, 1)
   BYTE3$ = MID$(MSGNO$, 3, 1)
   A = ASC(BYTE1$)
   B = ASC(BYTE2$)
   C = ASC(BYTE3$)
   C = C + 1
   IF C = 58 THEN
      C = 48
      B = B + 1
   END IF
   IF B = 58 THEN
      B = 48
      A = A + 1
   END IF
   IF A = 58 THEN
      A = 48
C = 1
END IF
MSGNO$ = CHR$(A) + CHR$(B) + CHR$(C)
RETURN
ENDPGM:
SYSTEM
DECLARE SUB ERASESCR ()
DECLARE SUB GETDATA ()
DECLARE SUB ACDATA ()
DECLARE SUB PILOTACK ()
DECLARE SUB PILOTMSG ()
DECLARE SUB MEMLOAD ()
DECLARE SUB COMMSUB ()
DECLARE SUB SENDMSG ()
DEFINT A-Z
DIM SHARED ALTMENU$(36)
COMMON SHARED BLANK$, INDATA$, CID$, CALLS$, HDG$
COMMON SHARED GS$, MODE$, TYPE$, MSG02, MSG02$, MSG03, MSG03$
COMMON SHARED MSG04, MSG04$, MSG05, MSG05$, LINENUM, LINEIND
COMMON SHARED MSG, MSG$, CBUFF$, CBYTE$
BLANK$ = ""
OPEN "COM1:4800,N,8,1,CS,DS,CD" FOR RANDOM AS 1 LEN = 640
FIELD #1, 128 AS CBUFF$
FIELD #1, 1 AS CBYTE$
ON COM(1) GOSUB COMMSUB
COM(1) ON
CLS
LASTMSG = 0

MAINLOOP:
LOCATE 1, 1, 0
PRINT "COM PORT ACTIVITY:"
LOCATE 23, 1, 0
PRINT "A=A/C DATA B=PILOT ACK C=PILOT MSG D=MEM LOAD E=CLS
Q=QUIT "
LOCATE 21, 1, 1
PRINT "ENTER COMMAND:"
LINEIND = 16
LINENUM = 21
INDATA$ = ""
CALL GETDATA
IF INDATA$ = "A" THEN
 CALL ACDATA
ELSE"
 IF INDATA$ = "B" THEN
 CALL PILOTACK
 ELSE
 IF INDATA$ = "C" THEN
 CALL PILOTMSG
 - 233 -
ELSE
  IF INDATA$ = "D" THEN
    CALL MEMLOAD
  ELSE
    IF INDATA$ = "E" THEN
      CLS
    ELSE
      IF INDATA$ = "Q" THEN
        CLS
      END
      ELSE
        LOCATE 22, 1, 0
        PRINT "INVALID CHOICE"
      END
      END IF
    END IF
  END IF
END IF
END IF
END IF
END IF
GOTO MAINLOOP

COMMSUB:
  LOCATE 2, 1, 0
  PRINT BLANK$
  LOCATE 3, 1, 0
  PRINT BLANK$
  N = LOC(1)
  IF N = 0 THEN
    GOTO ENDCOM
  END IF
  WHILE N < 3
    N = LOC(1)
  WEND
  GET #1, 3
  MSGT$ = LEFT$(CBUFF$, 3)
  LOCATE 2, 1, 0
  PRINT "MSG TYPE: ", MSGT$
  LOCATE 3, 1, 0
  IF MSGT$ = "001" THEN
    PRINT "ACK RECEIVED"
  GOTO ENDCOM
  END IF
  IF MSGT$ = "000" THEN
    PRINT "NAK RECEIVED"
    CALL SENDMSG
  GOTO ENDCOM
  END IF
  IF MSGT$ = "002" THEN
PRINT "POLL REQUEST"
IF MSG02 = 1 THEN
    MSG$ = MSG02$
    MSG = 2
    CALL SENDMSG
    MSG02 = 0
    GOTO ENDCOM
END IF
IF MSG03 = 1 THEN
    MSG$ = MSG03$
    MSG = 3
    CALL SENDMSG
    MSG03 = 0
    GOTO ENDCOM
END IF
IF MSG04 = 1 THEN
    MSG$ = MSG04$
    MSG = 4
    CALL SENDMSG
    MSG04 = 0
    GOTO ENDCOM
END IF
IF MSG05 = 1 THEN
    MSG$ = MSG05$
    MSG = 5
    CALL SENDMSG
    MSG05 = 0
    GOTO ENDCOM
END IF
LSET CBUFF$ = "01"
PUT #1, 2
GOTO ENDCOM
END IF
IF MSGT$ = "003" THEN
    PRINT "CID REQUEST"
    N = LOC(1)
    WHILE N < 5
        N = LOC(1)
    WEND
    GET #1, 5
    CID$ = LEFT$(CBUFF$, 3)
    BCC$ = MID$(CBUFF$, 4, 2)
    BCC% = CVI(BCC$)
    BC% = 0
    FOR I = 1 TO 3
        BC% = BC% + ASC(MID$(CID$, I, 1))
    NEXT I
    IF BC% <> BCC% THEN
LSET CBUFF$ = "00"
PUT #1, 2
GOTO ENDCOM
END IF
IF CID$ = "123" THEN
MSG$ = "02123" + "UNITED 235 " + "370180450LDC9 "
ELSE
IF CID$ = "456" THEN
MSG$ = "02456" + "DELTA 2011 " + "290050420L757 "
ELSE
IF CID$ = "789" THEN
MSG$ = "02789" + "CESSNA N4019T " + "150235150LC182 ">
ELSE
MSG$ = "02000"
END IF
ENDIF
ENDIF
MSG = 2
CALL SENDMSG
GOTO ENDCOM
ENDIF
IF MSGT$ = "029" OR MSGT$ = "030" THEN
IF MSGT$ = "029" THEN
PRINT "CLIMB AND MAINTAIN MSG"
ELSE
PRINT "DESCEND AND MAINTAIN MSG"
ENDIF
CALL ERASESCR
N = LOC(1)
WHILE N < 13
  N = LOC(1)
WEND
GET #1, 13
CID$ = LEFT$(CBUFF$, 3)
MSGNO$ = MID$(CBUFF$, 4, 3)
ALT$ = MID$(CBUFF$, 7, 3)
PD$ = MID$(CBUFF$, 10, 1)
BCC$ = MID$(CBUFF$, 12, 2)
BC% = 0
FOR I = 1 TO 11
  BC% = BC% + ASC(MID$(CBUFF$, I, 1))
NEXT I
BCC% = CVI(BCC$)
IF BC% <> BCC% THEN
  LSET CBUFF$ = "00"
  PUT #1, 2"
  GOTO ENDCOM
ENDIF
LSET CBUFF$ = "01"
PUT #1, 2
LOCATE 4, 1, 0
IF MSGT$ = "029" THEN
   PRINT "CLIMB AND MAINTAIN MSG"
ELSE
   PRINT "DESCEND AND MAINTAIN MSG"
END IF
LOCATE 5, 1, 0
PRINT "CID: ", CID$
LOCATE 6, 1, 0
PRINT "MESSAGE #: ", MSGNO$
LOCATE 7, 1, 0
PRINT "ALTITUDE: ", ALT$
IF PD$ = "1" THEN
   LOCATE 8, 1, 0
   PRINT "AT PILOTS DISCRETION"
END IF
GOTO ENDCOM
END IF
IF MSGT$ = "063" THEN
PRINT "VECTOR MSG"
CALL ERASESCR
N = LOC(1)
WHILE N < 12
   N = LOC(1)
WEND
GET #1, 12
CID$ = LEFT$(CBUFF$, 3)
MSGNO$ = MID$(CBUFF$, 4, 3)
HDG$ = MID$(CBUFF$, 7, 3)
TURN$ = MID$(CBUFF$, 10, 1)
BCC$ = MID$(CBUFF$, 11, 2)
BC% = 0
FOR I = 1 TO 10
   BC% = BC% + ASC(MID$(CBUFF$, I, 1))
NEXT I
BCC% = CVI(BCC$)
IF BC% <> BCC% THEN
   LSET CBUFF$ = "00"
   PUT #1, 2
   GOTO ENDCOM
END IF
LSET CBUFF$ = "01"
PUT #1, 2
LOCATE 4, 1, 0
PRINT "VECTOR MESSAGE"
LOCATE 5, 1, 0"
PRINT "CID: ", CID$
LOCATE 6, 1, 0
PRINT "MESSAGE #: ", MSGNO$
LOCATE 7, 1, 0
PRINT "TURN ", TURN$, " HEADING ", HDG$
GOTO ENDCOM
END IF
LOCATE 3, 1, 0
PRINT "UNKNOWN DATA FROM DEVICE: ", MSGT$
IF LOC(I) > 0 THEN
  JUNK$ = INPUT$(LOC(1), #1)
END IF
ENDDO:
RETURN

REM*********************************************************
REM THIS LOADS AC DATA TO BE XMIT'D TO THE DEVICE
REM*********************************************************
SUB ACDATA
CLS
LOCATE 20, 1, 0
PRINT "MODE TO ENTER A/C IDENTIFICATION DATA"
LOCATE 24, 1, 0
PRINT BLANK$
CID$ ="
WHILE CID$ = ":
  LOCATE 23, 1, 1
  PRINT "ENTER CID (3 DIGITS):"
  "
  LINEIND = 23
  LINENUM = 23
  INDATA$ = ""
  CALL GETDATA
  IF INDATA$ = "" THEN GOTO EXITACDATA
  IF LEN(INDATA$) <> 3 THEN
    LOCATE 24, 1, 0
    PRINT "CID MUST BE 3 DIGITS"
  ELSE
    CID$ = INDATA$
  END IF
END WHILE
CALLS$ = ":
WHILE CALLS$ = ":
  LOCATE 24, 1
  PRINT BLANK$
  LOCATE 23, 1
- 238 -
PRINT "ENTER CALL SIGN (UP TO 15 CHARS): "
LINEIND = 35
LINENUM = 23
INDATA$ = ""
CALL GETDATA
IF INDATA$ = "" THEN GOTO EXITACDATA
IF LEN(INDATA$) > 15 THEN
  LOCATE 24, 1
  PRINT "CALL SIGN MUST BE LESS THAN 16 CHARACTERS"
ELSE
  FOR K = LEN(INDATA$) TO 14
    INDATA$ = INDATA$ + " "
  NEXT K
  CALLS$ = INDATA$
END IF

WEND
ALT$ = ""
WEND
HDG$ = ""
WEND
ELSE
    HDG$ = INDATA$
END IF
WEND
GS$ = ""
WHILE GS$ = ""
    LOCATE 24, 1
    PRINT BLANK$
    LOCATE 23, 1
    PRINT "ENTER GROUND SPEED (3 DIGITS):"
    LINEIND = 32
    LINENUM = 23
    INDATA$ = ""
    CALL GETDATA
    IF INDATA$ = "" THEN GOTO EXITACDATA
    IF LEN(INDATA$) <> 3 THEN
        LOCATE 24, 1
        PRINT "GROUND SPEED MUST BE 3 DIGITS"
    ELSE
        GS$ = INDATA$
    END IF
WEND
MODE$ = ""
WHILE MODE$ = ""
    LOCATE 24, 1
    PRINT BLANK$
    LOCATE 23, 1
    PRINT "ENTER FLIGHT MODE (L, C, D, M):"
    LINEIND = 33
    LINENUM = 23
    INDATA$ = ""
    CALL GETDATA
    IF INDATA$ = "" THEN GOTO EXITACDATA
    IF INDATA$ <> "L" AND INDATA$ <> "C" AND
        INDATA$ <> "D" AND INDATA$ <> "M" THEN
        LOCATE 24, 1
        PRINT "WRONG FLIGHT MODE"
    ELSE
        MODE$ = INDATA$
    END IF
WEND
TYPE$ = ""
WHILE TYPE$ = ""
    LOCATE 24, 1
    PRINT BLANK$
    LOCATE 23, 1
    PRINT "ENTER TYPE AIRCRAFT (10 CHAR$):
    LINEIND = 33
LINENUM = 23
INDATA$ = ""
CALL GETDATA
IF INDATA$ = "" THEN GOTO EXITACDATA
IF LEN(INDATA$) > 10 THEN
    LOCATE 24, 1
    PRINT "TYPE MUST BE LESS THAN 11 CHARACTERS"
ELSE
    FOR K = LEN(INDATA$) TO 9
        INDATA$ = INDATA$ + " "
    NEXT K
    TYPE$ = INDATA$
END IF
WEND
MSG02$ = "02" + CID$ + CALLS$ + ALT$ + HDG$ + GS$ + MODE$ + TYPE$
MSG02 = 1
EXITACDATA:
CLS
END

SUB ERASESCR
FOR I = 4 TO 20
    LOCATE I, 1, 0
    PRINT " "
NEXT I
END SUB

REM ****************************************************
REM THIS SUBROUTINE READS THE KEYBOARD AND PRINTS IT
REM ****************************************************
SUB GETDATA
LOCATE LINENUM, LINEIND, 1
DTA$ = ""
WHILE DTA$ = ""
    DTA$ = INKEY$
WEND
WHILE ASC(DTA$) <> 13 AND ASC(DTA$) <> 27
    IF ASC(DTA$) = 8 THEN
        LINEIND = LINEIND - 1
        LOCATE LINENUM, LINEIND, 1
        PRINT " "
    ELSE
        LOCATE LINENUM, LINEIND, 1
        PRINT DTA$
    END IF

- 241 -
LINEIND = LINEIND + 1
INDATA$ = INDATA$ + DTA$
END IF
LOCATE LINENUM, LINEIND, 1
DTA$ = ""
WHILE DTA$ = ""
   DTA$ = INKEY$
   WEND
END
IF ASC(DTA$) = 27 THEN INDATA$ = ""
END SUB

REM *************************************************
REM
REM THIS LOADS AN ALT MENU TO BE XMIT'D TO THE DEVICE
REM *************************************************
SUB MEMLOAD
CLS
LOCATE 20, 1
PRINT "MODE TO ENTER ALTITUDE MENU DATA"
LOCATE 24, 1
PRINT BLANK$
J = 0
FOR I = 1 TO 18
   ALTMENU$(J) = ""
   WHILE ALTMENU$(J) = ""
      LOCATE 23, 1
      PRINT "ENTER F", I, " FIRST LINE:"
      LINEIND = 42
      LINENUM = 23
      INDATA$ = ""
      CALL GETDATA
      IF INDATA$ = "" THEN GOTO EXITMEMLOAD
      IF LEN(INDATA$) > 15 THEN
         LOCATE 24, 1
         PRINT "DATA MUST BE LESS THAN 16 CHARs"
      ELSE
         FOR K = LEN(INDATA$) TO 14
            INDATA$ = INDATA$ + " "
         NEXT K
         ALTMENU$(J) = INDATA$
      END IF
   WEND
   J = J + 1
   ALTMENU$(J) = ""
   WHILE ALTMENU$(J) = ""
      LOCATE 23, 1
      PRINT "ENTER F", I, " SECOND LINE:"
      LINEIND = 43
   END IF
   WEND
- 242 -
LINENUM = 23
INDATA$ = ""
CALL GETDATA
IF INDATA$ = "" THEN GOTO EXITMEMLOAD
IF LEN(INDATA$) > 15 THEN
   LOCATE 24, 1
   PRINT "DATA MUST BE LESS THAN 16 CHAR" "
ELSE
   FOR K = LEN(INDATA$) TO 14
      INDATA$ = INDATA$ + " "
   NEXT K
   ALTMENU$(J) = INDATA$
END IF
WEND
J = J + 1
NEXT I
MSG05$ = "05" + "001"
MSG05 = 1
EXITMEMLOAD:
CLS
END SUB

REM ************************************************************************************************************
REM THIS LOADS AN ACK FROM A PILOT TO BE XMIT'D TO THE DEVICE
REM ************************************************************************************************************
SUB PILOTACK
CLS
LOCATE 20, 1
PRINT "MODE TO ENTER PILOT ACKNOWLEDGEMENT" "
LOCATE 24, 1
PRINT BLANK$
CID$ = ""
WHILE CID$ = ""
   LOCATE 23, 1
   PRINT "ENTER CID (3 DIGITS):" "
   LINEIND = 23
   LINENUM = 23
   INDATA$ = """
   CALL GETDATA
   IF INDATA$ = "" THEN GOTO EXITPILOTACK
   IF LEN(INDATA$) <= 3 THEN
      LOCATE 24, 1
      PRINT "CID MUST BE 3 DIGITS" "
   ELSE
      CID$ = INDATA$
   END IF
WEND
CALLS$ = ""

- 243 -
WHILE CALLS$ = ""
    LOCATE 24, 1
    PRINT BLANK$
    LOCATE 23, 1
    PRINT "ENTER CALL SIGN (UP TO 15 CHARs):"
    LINEIND = 35
    LINENUM = 23
    INDATA$ = ""
    CALL GETDATA
    IF INDATA$ = "" THEN GOTO EXITPILOTACK
    IF LEN(INDATA$) > 15 THEN
        LOCATE 24, 1
        PRINT "CALL SIGN MUST BE LESS THAN 16 CHARACTERS"
    ELSE
        FOR K = LEN(INDATA$) TO 14
            INDATA$ = INDATA$ + " 
        NEXT K
        CALLS$ = INDATA$
    END IF
WEND

MSGNO$ = ""
WHILE MSGNO$ = ""
    LOCATE 24, 1
    PRINT BLANK$
    LOCATE 23, 1
    PRINT "ENTER MESSAGE NUMBER (3 DIGITS):"
    LINEIND = 34
    LINENUM = 23
    INDATA$ = ""
    CALL GETDATA
    IF INDATA$ = "" THEN GOTO EXITPILOTACK
    IF LEN(INDATA$) <> 3 THEN
        LOCATE 24, 1
        PRINT "MESSAGE # MUST BE 3 DIGITS"
    ELSE
        MSGNO$ = INDATA$
    END IF
WEND

ACKTYPE$ = ""
WHILE ACKTYPE$ = ""
    LOCATE 24, 1
    PRINT BLANK$
    LOCATE 23, 1
    PRINT "ENTER ACK TYPE (0,1 OR 2):"
    LINEIND = 28
    LINENUM = 23
    INDATA$ = ""
    CALL GETDATA
IF INDATA$ = "" THEN GOTO EXITPILOTACK
IF INDATA$ = "0" OR INDATA$ = "1" OR INDATA$ = "2" THEN
   ACKTYPE$ = INDATA$
ELSE
   LOCATE 24, 1
   PRINT "WRONG ACK TYPE"
END IF
WEND
MSG03$ = "03" + CID$ + MSGNO$ + ACKTYPE$ + CALLS$
MSG03 = 1
EXITPILOTACK:
CLS
END SUB

REM ****************************************************************
REM THIS LOADS A MSG FROM A PILOT TO BE XMIT'D TO THE DEVICE
REM ****************************************************************
SUB PILOTMSG
CLS
LOCATE 20, 1
PRINT "MODE TO ENTER PILOT MESSAGE"
LOCATE 24, 1
PRINT BLANK$
CID$ = ""
WHILE CID$ = ""
   LOCATE 23, 1
   PRINT "ENTER CID (3 DIGITS):"
   LINEIND = 23
   LINENUM = 23
   INDATA$ = ""
   CALL GETDATA
   IF INDATA$ = "" THEN GOTO EXITPILOTMSG"
   IF LEN(INDATA$) <> 3 THEN
      LOCATE 24, 1
      PRINT "CID MUST BE 3 DIGITS"
   ELSE
      CID$ = INDATA$
   END IF
WEND
CALLS$ = ""
WHILE CALLS$ = ""
   LOCATE 24, 1
   PRINT BLANK$
   LOCATE 23, 1
   PRINT "ENTER CALL SIGN (UP TO 15 CHAR$:  
   LINEIND = 35
   LINENUM = 23
   INDATA$ = ""

CALL GETDATA
IF INDATA$ = "" THEN GOTO EXITPILOTMSG
IF LEN(INDATA$) > 15 THEN
    LOCATE 24, 1
    PRINT "CALL SIGN MUST BE LESS THAN 16 CHARACTERS"
ELSE
    FOR K = LEN(INDATA$) TO 14
        INDATA$ = INDATA$ + " "
    NEXT K
END IF
END IF
WEND
MSGDATA$ = ""
WHILE MSGDATA$ = ""
    LOCATE 24, 1
    PRINT BLANK$
    LOCATE 24, 1
    PRINT "ENTER MESSAGE DATA (UP TO 30 CHARS)"
    LINEIND = 38
    LINENUM = 23
    INDATA$ = ""
    CALL GETDATA
    IF INDATA$ = "" THEN GOTO EXITPILOTMSG
    IF LEN(INDATA$) > 30 THEN
        LOCATE 24, 1
        PRINT "MESSAGE MUST BE LESS THAN 31 CHAR"
    ELSE
        FOR K = LEN(INDATA$) TO 29
            INDATA$ = INDATA$ + " "
        NEXT K
        MSGDATA$ = INDATA$
    END IF
END IF
WEND
MSG04$ = "04" + CID$ + CALLS$ + MSGDATA$
MSG04 = 1
EXITPILOTMSG:
CLS
END SUB

REM ****************************************
REM THIS TRANSMITS A MESSAGE TO THE DEVICE
REM ****************************************
SUB SENDMSG
    BC% = 0
    IF MSG <> 5 THEN
        FOR I = 3 TO LEN(MSG$)
            BC% = BC% + ASC(MID$(MSG$, I, 1))
        NEXT I
    END IF
END SUB
BCC$ = MKI$(BC%)
MSGOUT$ = MSG$ + BCC$
LSET CBUFF$ = MSGOUT$
PUT #1, LEN(MSGOUT$)
CALL ERASESCR
LOCATE 10, 1, 0
PRINT "SENDING MESSAGE "
LOCATE 11, 1, 0
PRINT "LENGTH: ", LEN(MSGOUT$)
LOCATE 12, 1, 0
PRINT "BCC: ", BC%
LOCATE 13, 1, 0
PRINT MSGOUT$
ELSE
FOR I = 3 TO LEN(MSG$)
   BC% = BC% + ASC(MID$(MSG$, I, 1))
NEXT I
LOCATE 10, 1, 0
PRINT "SENDING MEM LOAD MENU"
LOCATE 11, 1, 0
PRINT "LENGTH: ", LEN(MSG$)
LSET CBUFF$ = MSG$
PUT #1, LEN(MSG$)
FOR J = 0 TO 35
   MSGOUT$ = ALTMENU$(J)
   FOR I = 1 TO 15
      BC% = BC% + ASC(MID$(MSGOUT$, I, 1))
   NEXT I
   LOCATE 12, 1, 0
   PRINT "LENGTH: ", LEN(MSGOUT$)
   LOCATE 12, 1, 0
   PRINT MSGOUT$
   LSET CBUFF$ = MSGOUT$
PUT #1, 15
NEXT J
BCC$ = MKI$(BC%)
LSET CBUFF$ = BCC$
PUT #1, 2
LOCATE 13, 1, 0
PRINT "BCC: ", BCC%
END IF
END SUB
VITA

Alan Mastin has a Bachelor of Business Administration degree in Behavioral Science from the University of Houston, 1973, a Bachelor of Science degree (Magna Cum Laude) in Mathematics from the University of North Florida, 1985 and expects to receive a Master of Science in Computer and Information Sciences from the University of North Florida, August 1990. Dr. Jack Leither of the University of North Florida is serving as Mr. Mastin’s thesis advisor. Mr. Mastin is currently employed as a project leader at Revlon Professional Products and has been with the company for about three years. He has been working as a programmer/analyst since 1981. Prior to that Mr. Mastin served as an Air Force officer in the career field of KC-135 navigator and achieved the rank of Captain.

Mr. Mastin has an ongoing interest in aviation and the computer sciences. He obtained his private pilot’s license and has over 1300 hours nav time and 120 hours pilot time. Mr. Mastin has been developing user friendly online systems at Revlon for which he has received tremendous positive feedback. Married for 18 years, Mr. Mastin has two daughters, 16 and 13.