

138

AN ELECTRONIC HORIZONTAL SITUATION INDICATOR  
AND DEVELOPMENT SYSTEM

by

JEFF D. LAGERBERG

B.S., Kansas State University, 1984

---

A MASTER'S THESIS

submitted in partial fulfillment of the  
requirements for the degree

MASTER OF SCIENCE

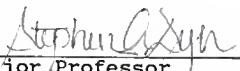
Department of Electrical and Computer Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1987

Approved by:

  
Major Professor

LD  
21662  
.T4  
EECE  
1987  
L33  
C. 2

TABLE OF CONTENTS

A11207 308824

CHAPTER ONE.

INTRODUCTION

Introduction to Digital Avionics . . . . . 1  
Statement of Problem . . . . . 4  
Thesis Overview . . . . . 5

CHAPTER TWO.

THE PROPOSED EHSI

Introduction . . . . . 7  
Navigational Instruments and Concepts . . . . . 9  
The EHSI and Display Pages . . . . . 24  
    Data Page . . . . . 25  
    NAV Page . . . . . 28  
    ILS Page . . . . . 31  
Conclusion . . . . . 40

CHAPTER THREE.

THE EHSI DEVELOPMENT SYSTEM HARDWARE

Introduction to Hardware . . . . . 41  
The HP-1345A Vector Graphics Display . . . . . 46  
The Control Keyboard . . . . . 49  
The Host Computer . . . . . 53  
The ATC-610 Flight Simulator . . . . . 55  
Data Acquisition and Communications Interface . . . . . 58  
    Introduction . . . . . 58  
    MC68000 Education Computer Board . . . . . 59  
    The Interface Board . . . . . 68  
Conclusion . . . . . 86

CHAPTER FOUR.

THE EHSI DEVELOPMENT SYSTEM SOFTWARE

Introduction to Software . . . . . 88  
The TUTOR Software . . . . . 89  
The Interface Board Software . . . . . 91  
    Support Subroutines . . . . . 93  
    Interrupt Processing . . . . . 99  
The Proposed Host Software . . . . . 104

TABLE OF CONTENTS CONTINUED

CHAPTER FIVE	CONCLUSIONS AND FUTURE WORK	
	Introduction . . . . .	106
	Future Considerations . . . . .	106
	Display Page Development . . . . .	107
	Ground Testing and Evaluation . . . . .	108
	Prototype Research and Design . . . . .	109
	Flight Testing and Evaluation . . . . .	110
	Marketing the EHSI . . . . .	111
	Conclusion . . . . .	112
REFERENCES		114
APPENDIX A.	INTERFACE BOARD DEVICE LIST . . . . .	116
APPENDIX B.	INTERFACE BOARD SCHEMATICS . . . . .	118
APPENDIX C.	INTERFACE SOFTWARE LISTINGS . . . . .	129

## LIST OF FIGURES

<u>Figure</u>		<u>Page</u>
1	VOR Navigational Aid Graphic	12
2	VOR Receiver	13
3	ADF Indicator	17
4	Determining Distance to an NDB	18
5	Waypoint Distance Example	20
6	Instrument Landing System ILS	21
7	The Proposed Data Page	26
8	The Proposed NAV Page	29
9	Runway Alignment Illusion	32
10	The Proposed ILS Page at 7.5 NM	35
11	The Proposed ILS Page at 5.5 NM	36
12	The Proposed ILS Page at 3.5 NM	37
13	The Proposed ILS Page at 1.3 NM	38
14	Development System Components Block Diagram	43
15	Development System Components Photo	44
16	HP-1345A I/O and Power Connector	50
17	Control Keyboard	51
18	Data Acquisition and Communication Interface	60
19	Educational Board Ports	62
20	Educational Board Functional Block Diagram	64
21	Address Decode Logic and Memory Segment Enable Signals	66

## LIST OF FIGURES CONTINUED

<u>Figure</u>		<u>Page</u>
22	Educational Board Memory Map	67
23	Interface Board (Photo)	69
24	Interface Board Functional Block Diagram	70
25	68000/PC Communications Protocol	78
26	Tutor Operational Flow Chart	90
27	Flow Diagram of Exception Processing	102

## LIST OF TABLES

<u>Table</u>		<u>Page</u>
1	Device Decode Addresses	72
2	Signal Ranges	87
3	Data Package Parameters	92

## ACKNOWLEDGEMENTS

The author would like to express his sincere appreciation for the assistance that has been provided to this manuscript by colleagues, friends and family. A special thanks to my major professor Dr. Stephen A. Dyer for his enlightening technical discussions and for his enthusiasm for the development of the EHSI.

The author is grateful to development team members Chuck Robertson and Dave Gruenbacher for their testing and general feedback of the operation of the development system. The author also acknowledges his Graduate Committee members Dr. Stephen A. Dyer, Dr. Donald R. Hummels and Dr. Maarten van Swaay for their time, guidance and thorough review of the original manuscript.

I am especially indebted to my family for their assistance and support through all my college years and of course for their many hours of drafting, typing and general encouragement on the manuscript. And finally, to my wife Laurie for her time, patience and encouragement in helping me see the manuscript through to completion.

## CHAPTER ONE

### INTRODUCTION

In the aviation industry, digital avionics have revolutionized air travel by saving fuel, reducing costs, reducing the pilot's workload and decreasing the chance for human error. Digital avionics first came onto the scene in 1970 along with the introduction of the wide-body transports. Delco and Litton Industries inertial navigation systems were the first to employ both digital computers and displays [1]. Rockwell Collins followed with the ANS-70 area navigation system for the DC-10. These systems provided pilots with a new generation of avionic instruments.

The breakthrough for digital avionics came in the mid 1970's with the advance of the semiconductor technology. Advances such as faster processors with more memory capacity provided the versatility that digital avionics required. Also, these advances allowed digital avionics to be less expensive and more reliable than previous analog designs. This type of versatility led Rockwell Collins to



develop its digital displays for the 757 / 767 Jumbo Jets [1]. These displays include a digital artificial horizon, attitude indicator, engine indication and crew alerting system (EICAS) and an electronic horizontal situation indicator (EHSI). Along with the flight computer, these instruments comprise the electronic flight instrument system (EFIS). The EFIS is responsible for total automatic flight control of the entire aircraft from just after takeoff to before landing. Since the EFIS has complete control over the flight of the aircraft, the Airbus Industrie's A310 uses triple redundancy of the important components [2].

In an effort to ease the transition and to help on early acceptance of digital avionics, Bendix decided to offer the airlines digital instruments that contained analog-to-digital conversion circuits on their front end. This enabled most equipment to be substituted directly into the aircraft without change to the aircraft wiring [1]. Due to these efforts and the efforts of many other digital avionics companies, pilots have now found the digital instruments more useful than initially expected. Pilots found that they rarely referred to navigation charts except at the start of the flight, but continuously referenced their EHSI and other digital displays [3]. This significantly reduced the pilot's workload by providing the pilot

with engine indication, flight control data, crew alerting system and navigational parameters.

Pilot's initial skepticism of digital avionics and their reliability existed in part because it was "going against the grain." Also, unions feared that digital avionics might reduce the number of pilots in the cockpit. Now that they have had a chance to become accustomed to using the EFIS components, the pilots tend to prefer them over electromechanical instruments. Over the next 2 to 3 years, pilots are expected to further exploit the capabilities of the digital avionics systems [4,5].

Three displays are common to most EFIS: An electronic attitude indicator (EAI), an electronic horizontal situation indicator (EHSI) and a display which groups various flight data. The EAI is essentially the same as its mechanical counterpart. It shows the relationship of the aircraft's attitude with respect to the horizon. The EHSI is an improvement over the mechanical HSI. It not only points to a VOR or an NDB, it combines the information about the plane's horizontal relationship to its surroundings and displays it pictorially for easier comprehension of the data. The EHSI depicts the VOR or NDB navigational fixes and the aircraft's position relative to the navigational fixes. It shows the recent, present and predicted

ground tracks. Also, heading information as well as air-speed, wind speed and wind direction are displayed on the EHSI. The EHSI provides an effective means of instantly orienting the pilot with the ground by displaying a "roadmap" of the aircraft's position relative to its planned course and the relationship to navigation aids in the vicinity [5].

The data to produce these displays is derived from the sensors and transducers onboard the aircraft. They include signals from navigational radios, distance measuring equipment, radar, transponders and even very low frequency Omega Global Positioning System stations. With the VLF/Omega stations it is possible to have positional accuracies to within 0.5 min in latitude and 0.1 min in longitude [6].

Small aircraft in the general-aviation community should be able to benefit from these digital avionics advances. The basic problem is that the number of instruments and the amount of information provided to the pilot is increasing. A single instrument that would integrate and produce a pictorial representation of the data would be very beneficial to aircraft operating under single-pilot instrument flight rule (IFR) conditions. The pictorial representation would reduce the pilot's workload thereby

making it easier for the pilot to comprehend the impact of the data even under high-stress situations.

Electronic flight information systems already exist for the large aircraft. However, no affordable system exists for the low-end general-aviation community. It is the purpose of this thesis to propose, describe and develop an EHSI by which the general-aviation community could benefit. This research will closely parallel and further develop the ideas presented in the paper entitled "A Proposed Electronic Horizontal Situation Indicator For Use In General-Aviation Aircraft" by S. A. Dyer [7].

The thesis will briefly describe the fundamentals of aircraft navigation in Chapter 2. The proposed operating modes of the EHSI are also discussed along with how these display modes can be generated from the various flight data. To develop such an avionic display, we must design some type of development system to provide initial testing, simulation and display technique development. In Chapters 3 and 4 such a working development system is described. Chapter 3 gives a detailed explanation of the development system hardware while Chapter 4 details the software used to control the development system components. Chapter 5 concludes the thesis by suggesting what work needs to be done to continue the development of the EHSI. These recom-

mendations cover areas which will need consideration as the EHSI matures toward flight testing.

Since the time frame involved for development of the EHSI from concept to flight testing is approximately 3 to 4 years, the development was divided up into manageable, thesis-length tasks. It was my task to introduce the subject matter and to propose several display pages for the EHSI. Also, it was my responsibility to design a development system that could be used to develop the EHSI pages and to incorporate simulated flight data. Using the development system, the display designers should be able to sit down at the computer console and through software acquire all system parameters needed to develop the initial EHSI displays.

## CHAPTER TWO

### THE PROPOSED EHSI

In this chapter the proposed EHSI and its operating modes will be defined. To aid the non-pilot reader various navigation terms, related aviation concepts and a basic description of the aircraft's navigational instruments will also be discussed. The proposed display modes (called pages) for the EHSI, along with a discussion of the usefulness and data needed to produce such displays will then be described. At this point in the discussion, the reader will have a basic understanding of the EHSI end product. Also, it will become apparent that an efficient development system will be needed to develop the EHSI.

The digital avionics revolution seems to have overlooked the need for a useful digital instrument for use by the general-aviation community. A similar increase in workload exists for the single pilot flying under Instrument Flight Rule conditions as for the pilots operating large airliners. The digital instruments of the large

jumbo jets are far too expensive and complex to be used on the small aircraft. Also, much of the data required to produce the existing digital avionic displays could not be acquired on a small aircraft. A new single instrument should be designed to take advantage of the digital avionic technology already developed for the large passenger airliners. The system must be able to acquire flight information from navigational radios and other sensors. This data must then be processed and presented to the pilot in a simple pictorial graphic on a digital display. The displays should show the results of the calculations made on the flight data and the data input during preflight. One basic display could show the aircraft's horizontal relationship to various navigation aids and chosen waypoints as though the observer were several miles above the aircraft. In addition to the pictorial relationship to navigation aids, a wealth of flight information can also be displayed.

Possibilities for digital avionics such as the EHSI are numerous. If the information is available to the EHSI system, it can be displayed in some manner to the pilot. However, it should be kept in mind that the goal of the instrument is not to overwhelm the pilot with redundant information but rather to reduce the pilot's workload by

providing a display which will aid in quicker assimilation of the available data. Before proceeding with a description of the EHSI proposed pages, I will introduce basic radio navigation concepts and electronic aids to instrument flying will be discussed.

### Navigation Instruments and Concepts

There are, in general, two types of conditions under which aircraft operate. If the weather conditions are better than the required minimums established by the Federal Aviation Administration (FAA), then Visual Flight Rules (VFR) are in affect. Aircraft operating in this type of weather are said to be operating under VFR conditions. When the weather conditions deteriorate and become worse than the prescribed VFR minimums, then Instrument Flight Rules (IFR) apply. An aircraft is said to be on an IFR flight if it is operating under these flight conditions [8]. A pilot flying under instrument conditions must ignore the senses of sight, sound, and feel associated with VFR flight and must rely solely on the instruments [9]. This reliance on instruments corresponds to a greater workload for the pilot flying under instrument conditions. Today, flying by instruments is considered the prime method



of aircraft control, regardless of the weather conditions [9]. Therefore, the workload from flying by instruments is present to some extent even under VFR conditions.

Several navigation systems are available to the instrument pilot that help answer the question, "Where am I?" These instruments, used either separately or in combination, provide the pilot with a geographical fix, or known position. This is accomplished by determining the distance and bearing from some known point. If only the bearings to two known points are available, a fix may still be obtained by the method of triangulation. These known points may be prominent obstacles or landmarks such as a sharp bend in a river or a bridge over the river. The horizontal direction from the aircraft to the landmark is the bearing. This is also called a line of position. By determining two line of positions and their intersection, one can obtain a geographical fix. Cross-country navigation can then be accomplished by the pilot continuously calculating a new fix from new lines of position. This technique is appropriate only for the pilot flying under VFR conditions.

When IFR conditions are in effect, different techniques must be used to establish a fix. These techniques can be classified as radio navigation. This term is

derived from the fact that a pilot can follow a predetermined flight path over the Earth's surface by utilizing different characteristics of radiowaves. Ground-based navigational facilities are located throughout the countryside at known geographical points. With the correct instruments, these facilities may be used to establish a navigational fix. One type of facility is a very high frequency omnidirectional range (VOR) navigation aid.

The VOR generates directional information and transmits it by ground equipment to the aircraft. It provides 360 magnetic courses to or from the VOR station. These courses are called radials and are by convention oriented from the station as shown in Figure 1 [9]. The VOR transmits two signals. One has a constant phase at all points around the VOR and is called the reference signal. The second signal is electrically rotated at 1800 revolutions per minute which produces a signal with variable phase around the VOR. In all directions other than magnetic north, the two signals are out of phase. The aircraft receiver can then measure phase difference to determine the bearing to or from the VOR.

The resulting bearing is displayed on a VOR receiver shown in Figure 2. The receiver includes an omnibearing course selector (OBS), a course deviation indicator

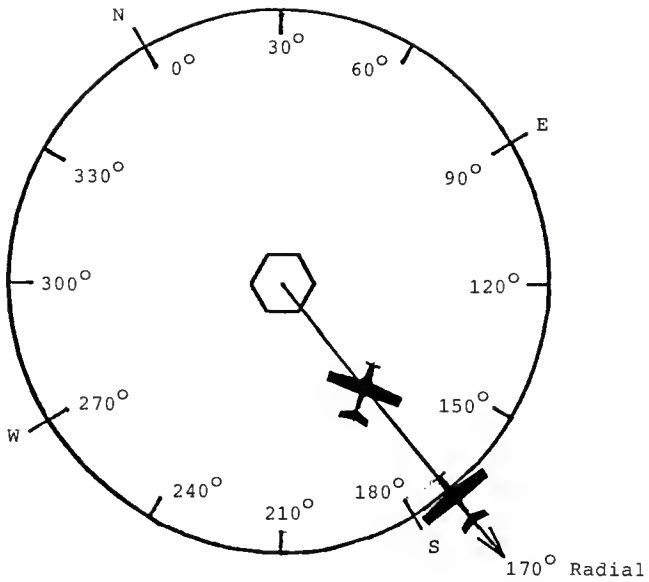


FIGURE 1. VOR NAVIGATIONAL AID GRAPHIC

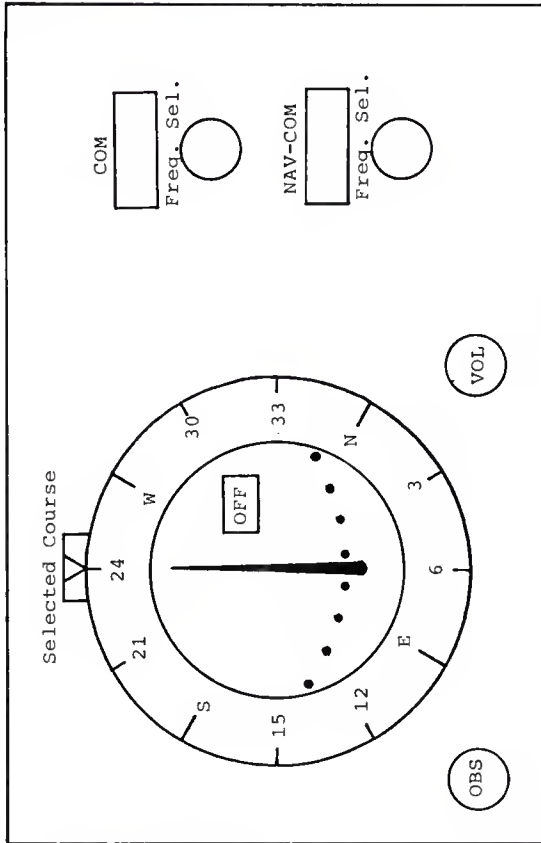


FIGURE 2. VOR RECEIVER

(CDI) and a frequency selector. The desired VOR frequency is first selected on the receiver. By turning the OBS, the pilot may select the desired course. The CDI is a needle hinged to move laterally across the interior of the compass dial. The CDI indicates the aircraft's deviation from the selected course (radial), but it shows nothing about the aircraft's heading. The CDI needle simply centers when the aircraft is on the selected radial. To fly to a VOR station, the pilot turns the OBS knob until the CDI needle centers. The radial shown represents either the heading or the reciprocal heading that must be maintained to fly over the VOR station. To determine which heading to fly, the pilot must reference the TO/FROM flag on the CDI instrument. If the TO flag is displayed and the CDI needle is centered then the OBS course is the heading that will take the aircraft to the station. However, if the FROM flag is displayed then the OBS course displayed is the reciprocal of the heading that will take the aircraft to the VOR. So the TO/ FROM flag indicates whether the OBS course will take the aircraft TO or FROM the station. It does not tell whether the aircraft is heading to or from the station [10].

Distance to a VOR can be determined by using the aircraft's distance measuring equipment (DME). The air-

craft's DME transmits a pair of RF pulses which are received by the ground facility. The ground facility then generates and transmits a second pair of pulses to the interrogating aircraft. The DME measures the round-trip elapsed time and converts this to a distance indication on the aircraft instrument panel. By using the directional information of a VOR and the distance information from the DME, the pilot can establish a geographical fix and a aircraft heading reference.

Another type of navigation facility is called TACAN or tactical air navigation system. With the appropriate equipment, it provides heading and distance information, simultaneously, to up to 100 aircraft. The aircraft receives this information, triggered by airborne interrogations, from the ground based facility. When VOR and TACAN facilities are located and operating simultaneously at the same geographical location, the navigation aid is called a VORTAC. Bearing information from the VOR is used and distance information from the TACAN is used to fix the aircraft's position.

A Non-Directional Homing Beacon (NDB) is a type of navigation facility which provides a non-directional radiation pattern. The bearing from the nose of the aircraft to the NDB is shown on an instrument in the cockpit called an

automatic direction finder (ADF) shown in Figure 3. Unlike the VOR CDI, the ADF needle points to the NDB regardless of aircraft heading or position. The relative bearing indicated is thus the angular relationship between the aircraft heading and the NDB location measured clockwise from the nose of the aircraft. This relative bearing must be related to aircraft heading to determine the magnetic bearing to or from the NDB station. The magnetic bearing to the station is simply the sum of the magnetic heading (compass) and the relative bearing (ADF).

The approximate distance to a NDB station may also be determined if the aircraft is not heading directly toward or away from the NDB. This is accomplished by the pilot recording the ADF bearing at the same instant the timing begins. By recording the new ADF bearing at any time thereafter and the elapsed time to the new bearing, the approximate distance to the station may be determined by the geometrical Law of Sines. Figure 4 shows the aircraft at Initial point I and terminal point T. The triangle formed is then used to compute the distance to the station. This distance is only an approximation and is only valid while the aircraft maintains constant heading and constant airspeed.

Waypoints are used to allow the pilot to fly to a

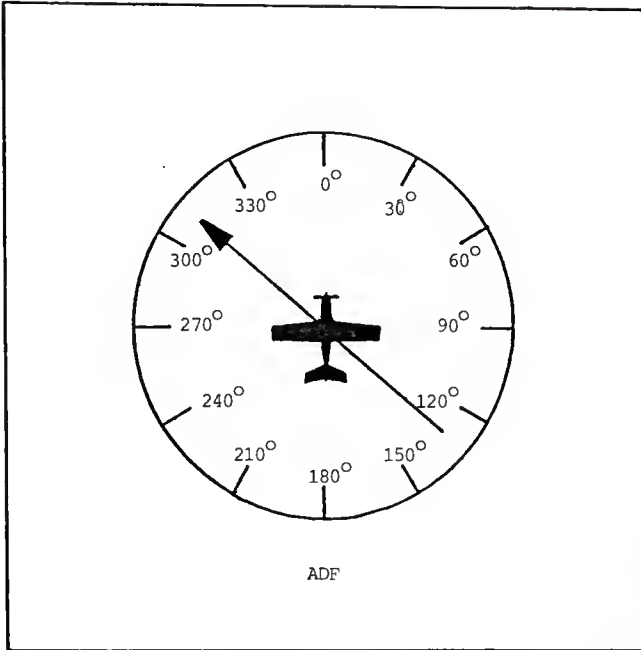
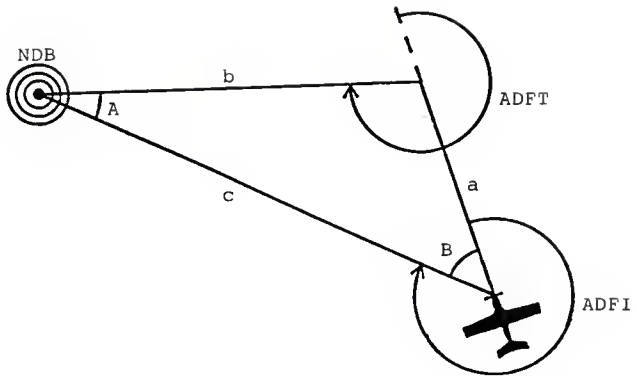


FIGURE 3. ADF INDICATOR





$$a = \frac{\text{Time to ADFT (min)}}{60} \quad (\text{Airspeed Nm/hr})$$

$$A = \text{ADFI} - \text{ADFT}$$

$$B = 360 - \text{ADFI}$$

$$b = a \frac{\sin (B)}{\sin (A)}$$

b = Approximate distance from ADFT to the NDB

FIGURE 4. DETERMINING DISTANCE TO AN NDB

predetermined point without the need to overfly ground-based navigation facilities. A waypoint is a predetermined geographical position on any radial offset from the navigation facility by a certain distance [8,9]. Therefore, the pilot can navigate to the waypoint instead of the VOR. This greatly reduces the congestion over a VOR. This type of navigation is called area navigation (RNAV). The advantage of the RNAV system stems from the ability to "locate" the navigation facility wherever it is convenient. Figure 5 shows that the distance flown from A to B, over a series of waypoints, is much shorter than overflying the actual stations.

When the weather conditions are IFR, Instrument Landing Systems (ILS) greatly facilitate the aircraft approach procedures. An ILS, graphically shown in Figure 6, is used to direct the aircraft toward the runway threshold during low or zero visibility [9]. This is done by providing feedback to the pilot pertaining to the aircraft's vertical and horizontal position with respect to a specified glide path and the runway centerline. An ILS system consists of three components. They are the localizer course, the glideslope course and VHF marker beacons.

The VHF localizer antenna array is located on the centerline at the opposite end of the instrument runway.

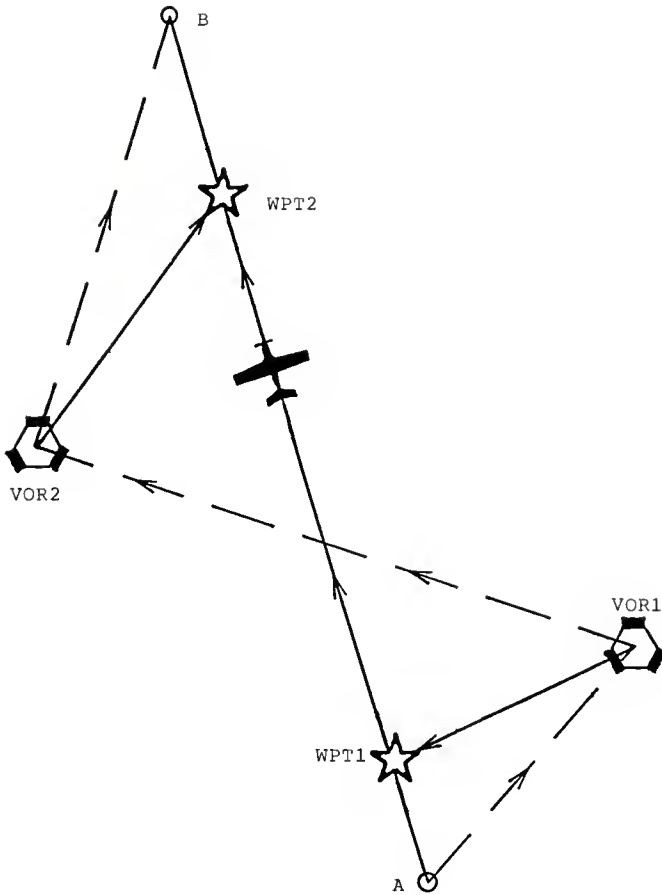


FIGURE 5. WAYPOINT DISTANCE EXAMPLE

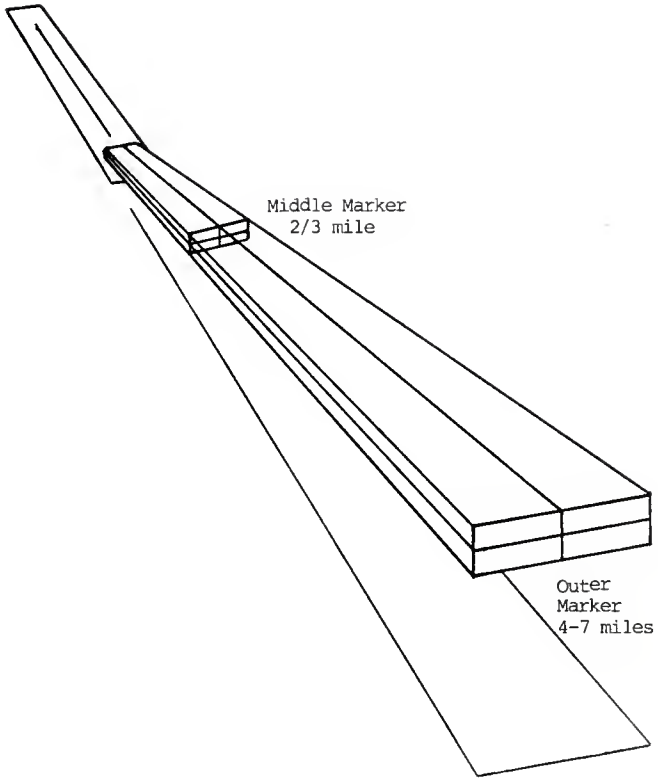


FIGURE 6. INSTRUMENT LANDING SYSTEM

The localizer unit generates a course reference down the centerline of the runway and extends approximately 18 nautical miles in both directions. It provides horizontal guidance to the airport runway. The glideslope furnishes vertical guidance along the proper descent angle to the touchdown point on the runway. Unlike the localizer, the glideslope transmitter radiates signals only in the direction of the final approach on the "front course." Both the localizer and the glideslope field patterns are modulated at two different frequencies. The right side of the localizer pattern, looking toward the runway from the front course, and the lower portion of the glideslope pattern are modulated at 150 Hertz. The left side of the localizer and the upper portion of the radiated glideslope pattern are modulated at 90 Hertz. The localizer signal produces an angular width of 5 degrees, typically. This produces localizer widths of roughly 3000 feet at the outer marker and 900 feet at the middle marker beacons. The glideslope projection angle above the horizon is typically 3 degrees and has a thickness of 1.4 degrees. This corresponds to a typical thicknesses of 475 feet at the outer marker and 75 feet at the middle marker. The intersection of the localizer and the glideslope at the outer marker occurs at an elevation of 1400 feet and 200 feet at the middle marker.

This information will be useful during the development of the ILS Page.

There are typically two VHF marker beacons, previously mentioned, that are located along the front course of the landing approach path. They are the outer marker (OM) and the middle marker (MM) beacons. The radiation patterns produced by these beacons are fan-shaped with an elliptical cross section. The OM is located 4 to 7 miles (typically 5 miles) from the runway threshold and indicates the point at which the glideslope will be intercepted on the localizer course. The MM is located approximately 3500 feet from the runway on the localizer front course. It provides another approach fix on the glide path to the threshold of the runway. It also typically coincides with the decision height (DH) established under the Federal Aviation Regulations by the FAA [9]. It represents the lowest altitude to which descent is authorized on final approach unless both the following conditions exist: the aircraft is in a position from which a normal approach can be made and the approach threshold or landing lights identifiable with the end of the runway are clearly visible. If upon arrival at the DH, any of the above requirements are not met, the pilot must immediately execute the missed approach procedure.

Compass locators, though not a specific component of the ILS may be incorporated into the system. They are low power radio beacons co-located with the OM and MM, and are used in conjunction with the ADF to guide the aircraft into the ILS system. Now that the reader has been introduced to the basic avionic and navigational concepts, the basic EHSI display pages can be discussed.

### The EHSI Pages

Three pages are proposed for display on the EHSI. They are the Data Page, the NAV Page and the ILS Page. There are many ways to present the information on the digital display. However, the current trend is to move away from duplicating the existing electromechanical needle-pointer instruments and to move instead toward pictorial presentations [5]. The proposed formats of these three pages form only one of many possible representations of the available data. Therefore, these pages allow the group who develops the end product a base from which to work. It should also be kept in mind that the EHSI is meant to co-exist with avionic components already on board general-aviation aircraft. That is, its purpose is not to eliminate the normal scan of the instruments by the pilot,

but rather to supplement this scan by aiding in the assimilation of the data and overall situational awareness.

Data Page The first page to be discussed is the Data Page. A vast amount of data is available to the pilot and much more information can be derived from calculations on this data. The purpose of the Data Page is to gather, organize and centrally locate this information on the EHSI to be viewed by the pilot. Also, the EHSI relieves the pilot from the laborious calculations by presenting the results of these calculations on the Data Page. A typical Data Page may appear as shown in Figure 7.

The data on the Data Page are organized by function and items having more importance are surrounded by a box for ease in scanning the page. There are groups for air-speed, timers for estimated time of arrival, communications and navigational frequencies. There are also indicators for RPM, manifold pressure, temperature and barometric pressure.

It is, of course, important not to get "carried away" with digital readouts. The Data Page would then appear to be a blurr of numbers. Pilots tend to prefer a mix of digital readouts and the "old" analog scales. The scales give the pilot a quick feeling of the value, while the digital readout provides a specific digital value that



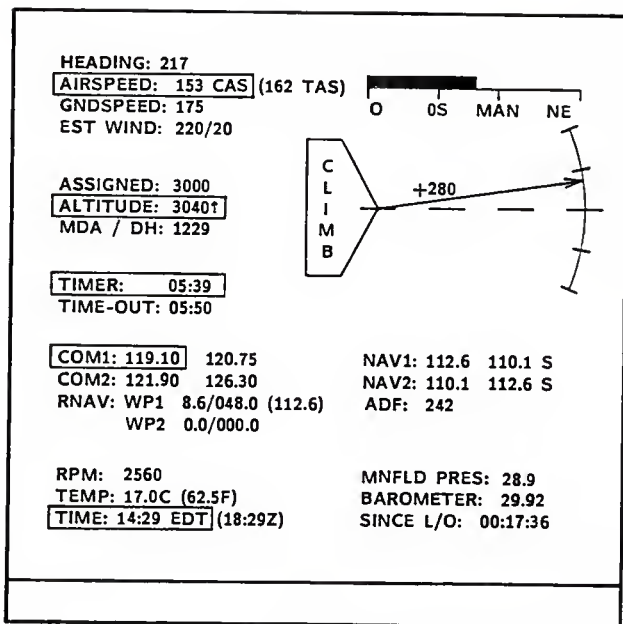


FIGURE 7. THE PROPOSED DATA PAGE

still requires reference to some maximum, minimum or ideal value [11]. The analog scales on the Data Page for climb rate and airspeed give the pilot an instant reference as to the desired condition.

The data for display on the Data Page are acquired from existing instruments and navigational radios while other information results from calculations and pilot input. Such data as magnetic heading, indicated airspeed, altitude, rate of climb, ADF bearing, RPM, manifold pressure and navigation frequencies are acquired directly from analog and digital signals of the instruments on board the aircraft. Items such as estimated wind speed and wind direction, assigned altitude, MDA/DH and waypoint locations are input to the EHSI by the pilot prior to or during flight. Finally, the EHSI performs computations on the existing data to give an indication of ground track, ground speed, distance to or from waypoints and estimated times of arrival.

In addition to performing calculations, the EHSI monitors possible alarm conditions and displays such conditions in the rectangular area at the bottom of the Data Page. Examples of such alarm conditions are: exceeding the maximum variation from the assigned altitude, approaching stall airspeed, approaching maximum structural airspeed

limits of the aircraft, timer timing out, or exceeding engine operating limits. The thresholds for these alarms and warnings can be set so that the warning appears early enough to avoid an alarm condition.

NAV Page The second proposed page is the NAV Page, an example of which appears in Figure 8. The purpose of this page is to present the aircraft and its surroundings in a plan representation with respect to the chosen navigational fixes. The fix may be a VOR, a VORTAC, an NDB or an active waypoint. This plan provides the pilot with a very effective means of relating the aircraft's position to its surroundings. This is accomplished by means of a graphic which portrays the aircraft and its horizontal relationship to navigational fixes as though the observer were a few miles above the aircraft. The EHSI then computes and displays the bearings to and from the navigational fix. In addition, distance information is displayed for VORTAC's, VOR's and waypoints. For a VORTAC, the distance is simply the distance given by the DME. For the VOR and the waypoints, the distance is approximated through geometrical calculations. Distance from an NDB is not easily or accurately determined but may be beneficial during periods when all other navigational fixes are out of range. This distance is determined by the technique previously men-

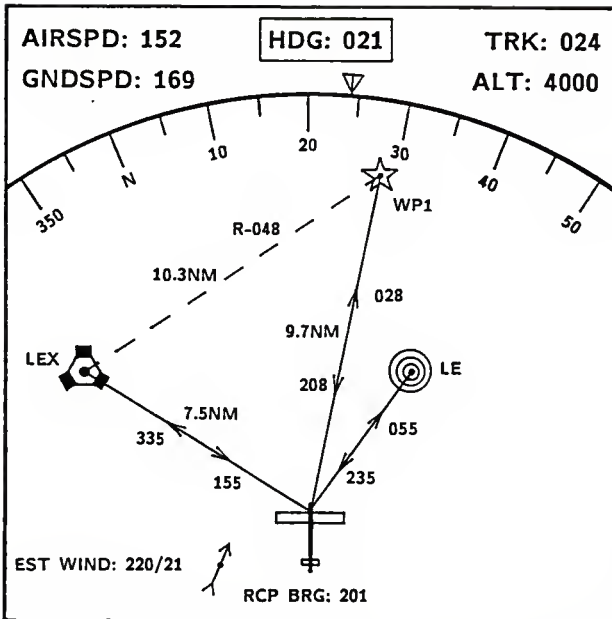


FIGURE 8. THE PROPOSED NAV PAGE

tioned on page 16.

The NAV Page also depicts the estimated wind speed and wind direction which is either input by the pilot or calculated from available data. The wind direction is indicated by a small arrow to the left of the aircraft symbol. The input wind vector is used to predict the aircraft's ground track. The resulting ground track heading is shown in the upper right corner of the page.

At the top center of the NAV Page appears the magnetic heading, enclosed by a square for ease of scanning. This heading information is supplied by the magnetic compass. A sixty-degree compass rose appears at the top of the page with the magnetic heading value centered in the middle of the rose. The compass rose will rotate so that the indicated heading always appears centered in the middle of the compass rose. Airspeed and altitude are also displayed at the top of the NAV Page.

The raw data for the NAV Page are acquired from the analog instruments and navigational radios. For example, the ADF indicator is used to establish the bearing to or from the NDB. The airspeed and altitude values are similarly acquired. Calculations on these raw data yield the present ground track and ground speed. The bearing to or from a VOR or a VORTAC is indirectly available from the

VOR receiver.

The space to the right of the aircraft symbol is reserved for alarm conditions. However, if no alarm conditions exist this area may be used for the NAV Page graphic. The same alarm conditions that were discussed for the Data Page also exist for the NAV Page.

ILS Page The third and final page to be discussed is the ILS Page. When on an ILS approach, the pilot continuously checks the position of the aircraft with respect to the localizer and glideslope by referencing the localizer / glideslope indicator. This instrument is slow to react to minor changes on the approach, giving little change in indication until there is a substantial deviation from the desired position. Also, the CDI can give the illusion that the aircraft is in perfect position on the approach path. If the aircraft is at the correct position on the localizer and glideslope, the CDI needles will be centered regardless of the attitude or heading of the aircraft as viewed in Figure 9. Therefore, the pilot needs to know the rate of change of position in addition to the instantaneous position with respect to the localizer and the glideslope. Also, checks should be constantly performed to ensure proper runway heading alignment previously described.

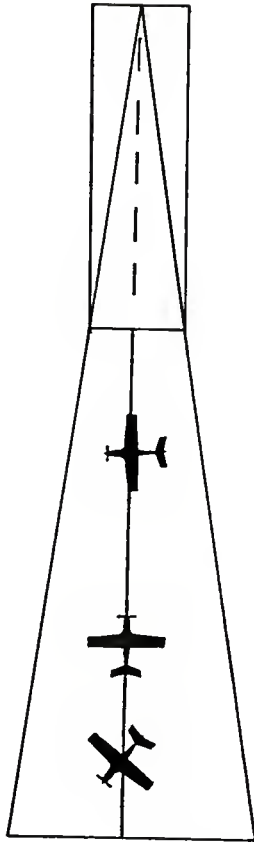


FIGURE 9. RUNWAY ALIGNMENT ILLUSION

It is obvious that the pilot is under a heavy workload. This may best be described by an excerpt from the Instrument Flying Handbook [9]. "The heaviest demand on the pilot occurs during descent from the outer marker to the middle marker, when you must maintain the localizer course, adjust pitch attitude to maintain the proper rate of descent, and adjust power to maintain proper airspeed. Simultaneously, the altimeter must be checked and preparation made for visual transition to land or for a missed approach. The need for accurate instrument interpretation and aircraft control can be appreciated, in the complete ILS, by noting the relationship between CDI/glide path needle indications and aircraft displacement from the localizer and glide path centerlines." The proposed ILS Page is designed to relieve an appreciable portion of this workload from the pilot by centralizing pertinent data and graphically representing the flight information.

The ILS Page will again use pictorial representation of the information gathered from the ILS receivers. This will allow the pilot to have a better feel of the aircraft's rate of change of position with respect to the localizer and the glideslope. The EHSI will also make the computations needed to relieve the pilot to perform more skilled tasks.



The ILS Page may be used as long as the aircraft is within ten nautical miles of the runway threshold. When the aircraft is at a greater distance than this, the NAV Page should be used to direct the aircraft into the ILS system. A typical ILS Page may appear as shown in Figure 10. This particular situation is for an aircraft 7.5 nautical miles from the runway threshold, on the localizer and at an altitude which will intercept the glideslope at the outer marker.

A tunnel to fly in is proposed. The left and right vertical walls of the tunnel correspond to maximum deviations from the localizer centerline, while the ceiling and floor of the tunnel correspond to the maximum deviation from the glideslope. As the reader can see from the rectangular tunnel, the glideslope deviation is less than the localizer deviation. The tunnel is depicted from approximately five nautical miles out to three fourths of a nautical mile. These distances correspond to the outer marker and middle marker respectively. Each rectangular box into the tunnel corresponds to one nautical mile and the last box in the tunnel corresponds to the middle marker or the DH window.

Figures 11, 12 and 13 illustrate different possible situations from when the aircraft enters the tunnel

**ALT: 2100**

**HDG: 230**

**DIST: 7.5 NM**

**DH: 1000**

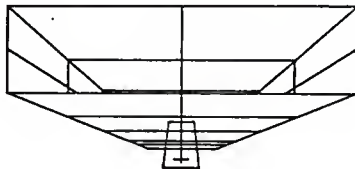


FIGURE 10. THE PROPOSED ILS PAGE AT 7.5 NM

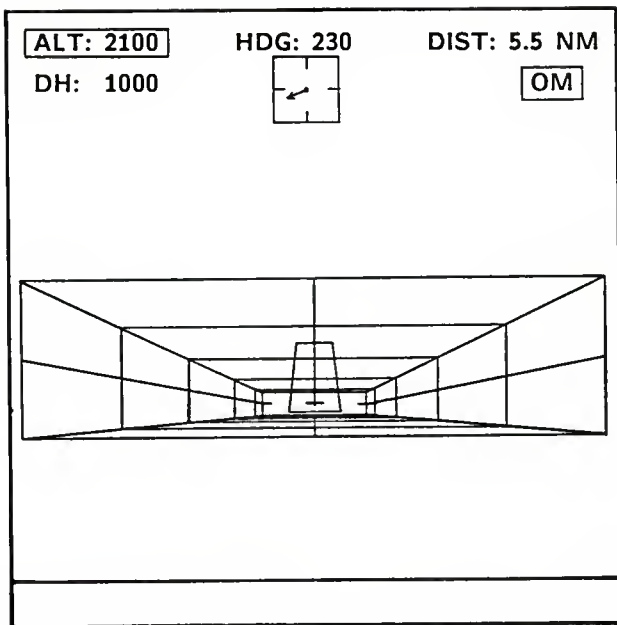


FIGURE 11. THE PROPOSED ILS PAGE AT 5.5 NM

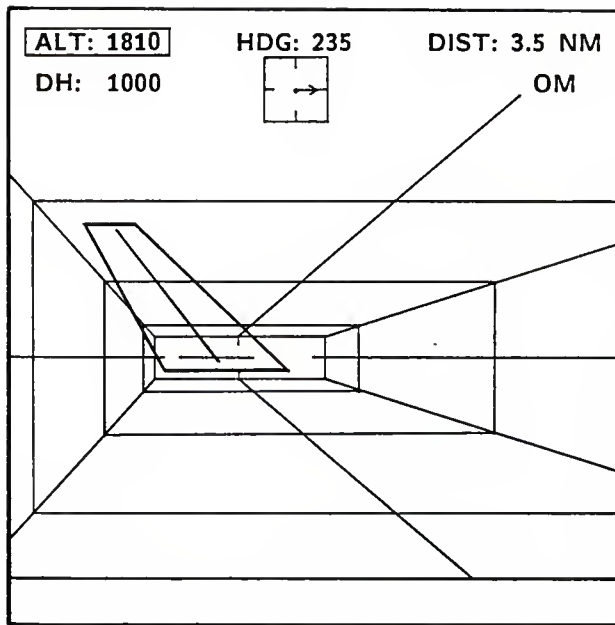


FIGURE 12. THE PROPOSED ILS PAGE AT 3.5 NM

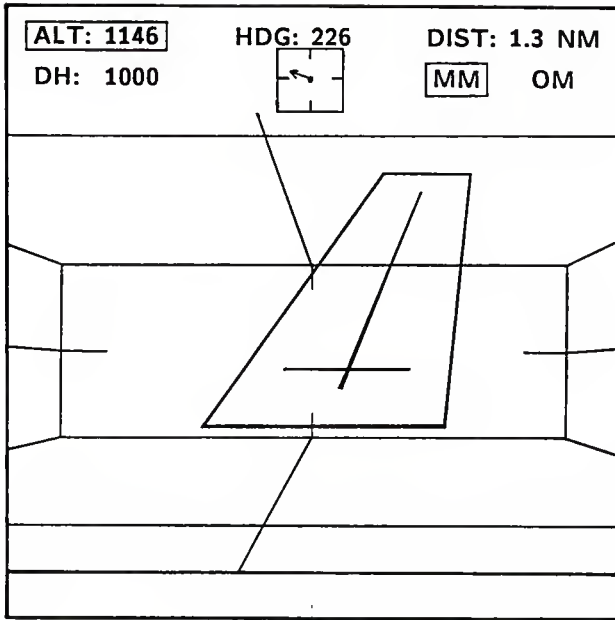


FIGURE 13. THE PROPOSED ILS PAGE AT 1.3 NM

until the aircraft is at decision height at the middle marker. Even though this pictorial representation inherently yields much trend information, a higher resolution of change of position has been provided, as shown by the change of position indicator at the top center of the ILS Page. This will give the pilot a quicker indication of the aircraft's trend. For example, the aircraft may be moving away from the localizer centerline at five feet per second but the change in the ILS graphic may not be discernible until the aircraft is several feet off the centerline. The change-of-position indicator will instantly show that there has been a change of position within the tunnel. This will be indicated by the arrow pointing in the direction of the change.

The bottom portion of the ILS Page is for alarm and warning conditions. Such conditions may be runway misalignment, approaching stall speed or approaching decision height. The top of the ILS Page shows digital values corresponding to the altitude, decision height, heading and distance. When the aircraft is over the outer marker, as in Figure 11, the symbol OM will appear, surrounded by a square. When the aircraft is no longer over the outer marker, the square will leave but the OM symbol will remain as in Figure 12, to remind the pilot of the progress of

the approach. The same sequence of events occurs over the middle marker as shown in Figure 13. At this point the pilot must make the decision to proceed with the approach or to declare a missed approach. If a missed approach is declared, then the pilot could change to the NAV Page and with the help of this page execute the missed-approach procedure.

Now that the proposed EHSI system and proposed pages have been discussed, it is time to actually develop the EHSI. To aid in this task, a development system was designed and built. The hardware associated with the EHSI Development System is described in Chapter 3.

## CHAPTER THREE

### THE EHSI DEVELOPMENT SYSTEM HARDWARE

To develop the EHSI, there must be a way to simulate an aircraft's instruments and navigational equipment. There also should be a host microcomputer to acquire and process this information and other pilot inputs to produce workable displays. The information should be pre-processed and available to the host in a convenient format to keep the host's overhead to a minimum. Here arises the need for an EHSI Development System.

The EHSI Development System will allow the acquisition of all pertinent flight data through the host software. It has a central controller that provides the communications with the various development system components including the host computer. The system will allow the host software and displays to be tested under simulated flight conditions. Eventually, the development system itself could be a component in the actual flight version of the EHSI.

The EHSI Development System is made up of five major



components, as shown in Figure 14 and in the photo of Figure 15. The first component is the display. The system will use an HP-1345A high resolution, vector graphics display. The display is used to present the various pages and to communicate with the pilot through messages on the display. The second component is the Control Keyboard. It consists of a six-by-six matrix keypad and a system ON/OFF switch. It is through the Control Keyboard that the pilot enters all preflight information and other parameters to control the EHSI display during the flight. The third component in the development system is the ATC-610 Flight Simulator. The simulator is a key component that provides flight simulation data for the host computer from avionic instruments, navigational radios, and from other sensors on board the simulator. The flight simulator also provides an effective means of real-time testing the EHSI system software running on the host computer which is the fourth component of the development system. The host computer is responsible for processing the data from the flight simulator and from the Control Keyboard to produce the display pages on the HP-1345A.

Each one of the four components described above provides an essential function to the overall EHSI Development System. However, they are intended to operate as separate

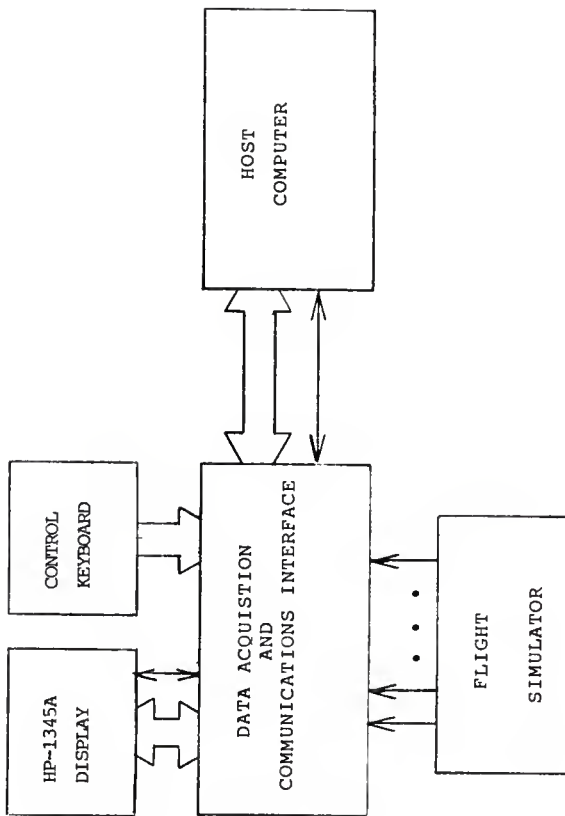


FIGURE 14. DEVELOPMENT SYSTEM COMPONENTS BLOCK DIAGRAM



FIGURE 15. DEVELOPMENT SYSTEM COMPONENTS

units and themselves provide no effective means of communicating with the other system components. Therefore, a MC68000 microprocessor based Data Acquisition and Communications Interface was built.

The 68000 based Data Acquisition and Communications Interface is the main controller and interface for all of the system components. It provides a means by which all of the system components can communicate with each other. The interface provides signal conditioning of all analog and binary signals from the flight simulator and may, on request, send these converted digital values to the host computer for further processing. The interface can send display commands to the HP-1345A or can provide a path by which the host can transfer graphic commands to and from the HP-1345A. In addition, the interface has a Real Time Clock and Calendar Module so the host can have continuous knowledge of the time and date. Also, the interface has a build-in piezo-electric alarm to alert the pilot that a dangerous condition exists and that he should take immediate action to eliminate the problem.

The interface controls the input of parameters through the Control Keyboard, and performs the EHSI initializations prior to the EHSI System Switch being turned on. Since the interface provides all of the commun-

ications with system components, performs all of the data gathering tasks, controls the display transfers to and from the HP-1345A, and completes the entire initialization process without the aid of the host, the interface will keep the host overhead processing time to an absolute minimum. The host may then concentrate on the processing of the data to produce the various displays for the HP-1345A.

Each one of these components plays an important part in the EHSI Development System. I will begin by describing each of the system components in detail and will then show how it is used and integrated into the system. I will also describe the Data Acquisition and Communications Interface in detail since it is the main controller for the entire development system.

#### HP-1345A Vector Graphics Display

The Hewlett-Packard 1345A is a high resolution vector graphics display that has a relatively low cost of \$4,000. Vector graphics implies it is able to plot any graphic within the 2048-by-2048 addressable point area using one or more of four basic commands. The viewing area is 9.5 cm vertical and 12.5 cm horizontal which gives 215 and 164 addressable points, per centimeter, in the vertical and

horizontal axes respectively. Because of its high resolution, the HP-1345A can draw both straight and curved lines. Curved lines are drawn as several straight vectors. The display weighs 4.4 kg and is capable of operating up to an altitude of 15,000 feet.

The HP-1345A produces vector graphics on its display screen in response to digital commands. It has four graphic commands. The first graphic command is the "Set Condition" command. It allows the user to set the vector graphic attributes by selecting from three line intensities, four line types, and four writing speeds. The second graphic command, the "Plot" command, allows random vector plotting within the 2048-by-2048 addressable point area. It allows the user to select the X and Y coordinates and the beam control, on or off (for solid or invisible line), while plotting the vector. The third graphic command is the "Graph" command. When the X coordinate of a plot will be a constant increment, this command allows the "Delta X" to be programmed once and incremented for each new Y coordinate. The Graph command may also be used with the beam control on or off. The last graphic command is the "Text" command. For character generation, the text command is used which allows the selection of a character from a modified ASCII character set. The characters can be generated in four

programmable orientations (0, 90, 180 and 270 degrees) and four programmable sizes.

The HP-1345A is used with the option 704, which is a 4k-by-16 bit Vector Memory. With the option installed, the display refresh is accomplished by the Vector Memory automatically. This eliminates the need for an external source providing the refresh. The option 704 recognizes two command groups for programming. The commands are for data transfer and memory address pointer manipulation.

A data transfer consists of the four graphic commands described above that are either a read or write to the vector memory. Address pointer operations are used for positioning the data in the vector memory. When the interface reads from or writes to Vector Memory, the memory location reference is determined by the current 16-bit contents of the Address Pointer. The Address Pointer command allows the user to program the Address Pointer with a new value in preparation for data transfer. It should be noted that the Vector Memory refresh has no effect on the Address Pointer, and the Address Pointer is auto-incremented after each read or write cycle. Another command that affects the Address Pointer is the "Internal Jump" command. It allows a jump to any other memory location in vector memory except a memory location that holds

another Internal Jump command. This command essentially loads the Address Pointer with the Internal Jump address so command execution will continue at the address specified by the Internal Jump command.

The HP-1345A is programmed through its 16-bit parallel port I/O Connector shown in Figure 16. The parallel command transfer is accomplished with a three wire handshake. The three handshake lines are Device Select (DS\*), Write (WR\*), and Read (RD\*). The asterisk (\*) symbol represents a signal that is enabled on a TTL low level. The Read and Write Cycle Timing diagrams and HP-1345A command reference may be found in the HP-1345A Digital Display Module Designers Manual [12].

### Control Keyboard

The Control Keyboard is made up of the EHSI System Switch and a 36 key command keypad. The Control Keyboard shown in Figure 17 allows the pilot to start or stop the EHSI flight system and to input various parameters or commands to control the system throughout the entire flight.

The command keypad provides the pilot access to enter navigation, communications, and ADF frequencies by using



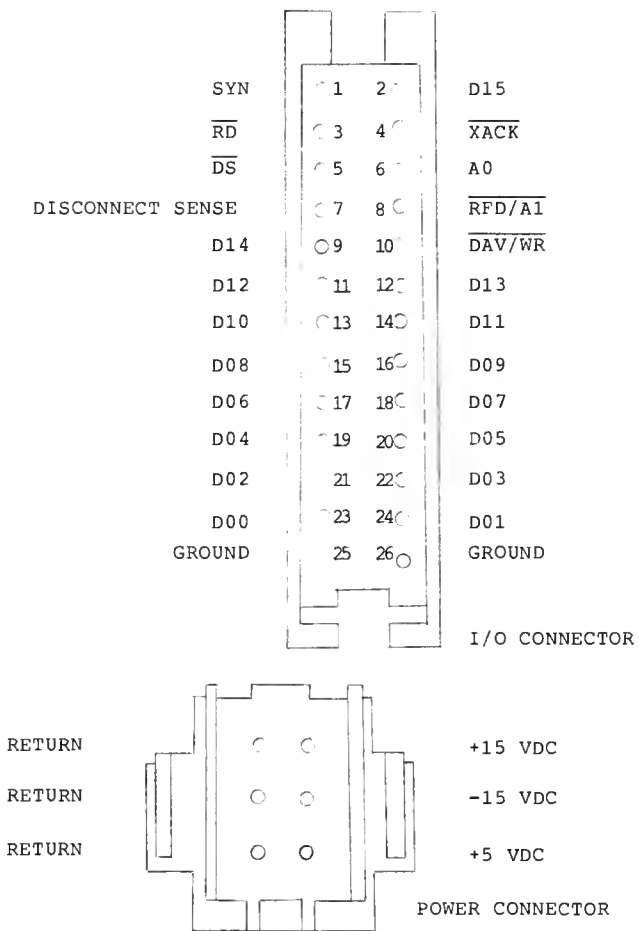


FIGURE 16. HP-1345A I/O AND POWER CONNECTOR

EHSI SYSTEM SWITCH



START TIMER	BRG/HOLD INBND		TIMER	RESET TIMER	SET CLOCK
ADF	FLIGHT DATA PAGE	MDA/DH	ASGN ALT	EST WIND	SET/RST ALRM
VOR1	NAV PAGE	—	<b>7</b>	<b>8</b>	<b>9</b>
VOR2	ILS PAGE	+	<b>4</b>	<b>5</b>	<b>6</b>
COM1	CLEAR	<b>x</b>	<b>1</b>	<b>2</b>	<b>3</b>
COM2	ENTER	÷	<b>0</b>	.	/

COMMAND KEYPAD

FIGURE 17. CONTROL KEYBOARD

the VOR1 & 2, COM1 & 2, and ADF keys. There is a standard calculator style keypad with the four basic arithmetic operations and ENTER and CLEAR keys. It can be used at any time to add, subtract, multiply or divide. The bottom line of the HP-1345A display may be used as a scratch pad for calculations performed on the keypad.

Three keys control which EHSI page is to be displayed. They are the ILS PAGE key, the FLIGHT DATA PAGE key, and the NAV PAGE (navigation) key. The pilot can move from one page to another by simply pressing the desired page key. A SET CLOCK key is also provided that permits the real time clock and calendar to be set at the precise time by prompting the user for the correct time and date information. Several other keys allow the pilot to set or input altitude minimums, minimum descent altitude / decision height, alarm limits, start or reset the timer, and to estimate the wind speed and wind direction. Additional keys could be added to provide the pilot with preflight and postflight checklists, and missed approach and emergency procedures.

The EHSI System Switch on the Control Keyboard is used to start and to stop the EHSI system. After the system is turned off, the system enters a wait state. When the system switch is turned back on, the whole system is re-

started, reinitialized, and is ready to continue with the current flight or to start a new flight.

#### Host Computer

The host computer is responsible for data processing and manipulation of all inputs and flight data to produce the EHSI pages for display on the HP-1345A. Almost any computer could be used as the host. The only requirements are that it have a parallel port, with two lines for handshaking, and be able to run a high level language such as C.

Originally a VAX 11/750 was considered to serve as the host computer. But access was limited by a baud rate of 9600 through a serial port, limiting the net throughput to approximately 1000 bytes per second. The parallel port of a PC compatible would provide a net throughput of approximately 10,000 bytes per second, hence, the PC compatible route was used. This will provide an adequate communications rate to allow the HP-1345A display to approximate a real-time graphic.

The host computer for the EHSI Development System will be the Zenith-158 Personal Computer. The Z-158 components consist of a composite monochrome video display, a 91

key ASCII keyboard, and the computing unit itself. The Z-158 comes standard with a RS-232C type serial asynchronous communications port, a Centronics standard parallel printer (I/O) port, a CPU clock speed select switch which allows the selection of 4.77 MHz or 8 MHz clock speeds, and six additional slots on the backplane for expansion. Also, a socket is provided for the optional 8087 Numeric Data processor.

The computer is configured with 256K bytes of RAM, expandable to 640K bytes, which reside on the Intel 8088 CPU motherboard. It uses one 360K byte, double-sided, double-density, 5.25 inch soft-sectored floppy disk drive. For mass storage, a 20 megabyte, Winchester Hard Disk System and Winchester controller board were added.

Since the Zenith is an "IBM PC Compatible," it will run almost all of the major software for the IBM PC. Microsoft's C, Version 4.0 has been chosen for the high level development language. The Centronics standard parallel port is used as a general purpose I/O port. The Centronics parallel port drivers limit the data transfer rate to 1000 characters per second. This transfer rate is insufficient for the development system communications. Therefore parallel port drivers were written in 8088 assembly language to communicate at 10,000 characters per second

with the Data Acquisition and Communications Interface. These drivers may be easily called by the C language programs.

The Zenith will be operating in the interrupt environment and will be taking advantage of the parallel port, autovectorred interrupt number seven. This interrupt service routine will be utilized by the Zenith software for recognizing what type of interrupt has occurred. More on this may be found later in the section on the host software.

#### ATC-610 Flight Simulator

The ATC-610 Flight Simulator is FAA-approved and simulates an IFR-instrument equipped aircraft. It will provide the host with actual flight simulation data.

The ATC-610 simulator uses analog and hybrid computer techniques to drive indicators which are similar to the flight instrumentation found in a typical IFR-equipped light aircraft [13]. All important analog signals on the flight simulator were tapped and fed to the analog port on the Interface Board. Signal-conditioning in the form of offset, buffering and gain circuitry was provided on the Interface Board to provide compatible signals to the ADCs.

Two eight bit, eight channel ADC's digitize the analog signals. The following instruments and signals are used by the development system:

- ATTITUDE INDICATOR - There are two signals that indicate bank and pitch attitudes. They show the attitude of the aircraft in relation to the natural horizon.
- VERTICAL SPEED - This instrument indicates vertical-velocity or rate-of-climb to a range of +/-2000 fpm.
- MANIFOLD PRESSURE - INDICATOR Indicates intake manifold pressure which will vary with RPM, throttle and mixture setting, and altitude.
- COURSE DEVIATION - INDICATOR The deviation indicator consists of a dial and a needle hinged to move laterally across the dial. The needle indicates deviation from the selected radial or its reciprocal.
- ALTIMETER - Standard altimeter which displays indicated altitude. This shows the approximate height of the aircraft above a mean sea level.
- AIRSPEED INDICATOR - This provides the pilot with Indicated Airspeed in knots and miles per hour.
- ADF BEARING - INDICATOR This display indicates the relative bearing from the nose of the aircraft to the Non Directional Beacon (NDB).
- DME INDICATOR - Distance Measuring Equipment is calibrated to 20 NM. It is automatically coupled to the selected VOR and indicates the distance to that VOR.
- RPM INDICATOR - Indicates propeller revolutions per minute.
- LANDING GEAR - POSITION INDICATOR Binary signal that indicates the gear position, down and locked, or gear in any other position.

- TO/FROM/OFF FLAG - This flag shows that the course selected by the omni bearing selector will take the aircraft to or from the VOR station. It does not tell that the aircraft is heading to / from the station. The OFF flag indicates the signal is unreliable.
- OUTER MARKER - Indicates the aircraft is over the outer marker beacon.
- MIDDLE MARKER - Indicates the aircraft is over the middle marker beacon.
- PERCENT POWER - This signal is generated within the flight simulator. It combines mixture and throttle settings to determine percent of operating power.
- DELTA X & Y - POSITION This signal is also generated within the flight simulator. Delta X is the difference between the X coordinates of the airplane and the selected VOR within the simulator's rectangular coordinate system. Delta Y is the difference between the Y coordinates. Together these signals can give the angular distance between the VOR and the aircraft which is used for current radial information of selected VOR.

These simulator signals are non-uniform, non-symmetric analog signals. For example, the airspeed indicator analog signal varies from 0 to +15 Volts DC, while the manifold pressure signal fluctuates between 6.7 to 8.3 Volts DC. There are two analog to digital converters (ADC) on board the Data Acquisition and Communications Interface. One ADC accepts unipolar inputs from 0 to 10 Volts. The second ADC is configured to accept bipolar signals at +/- 5



Volts. The flight simulator signals were conditioned by several analog techniques including amplification, attenuation and level shifting. These conditioned signals were then applied to the ADCs. In addition, to guarantee the flight simulator would not become "loaded" by these analog circuits, extremely high input impedance operational amplifier buffers were used on all signal inputs. The actual analog circuits will be described in the next section and schematics may be found in Appendix B.

#### Data Acquisition and Communications Interface

The Data Acquisition and Communications Interface is responsible for linking and controlling all of the EHSI Development System components. All "paths" between the HP-1345A, the flight simulator, and the Control Keyboard pass through the interface. Therefore, the interface provides the data communications and data transfer bridge between the host computer and the other development system components.

The host computer utilizes the interface to establish bidirectional communications with the HP-1345A. This data transfer occurs at a speed such that the pages on the HP-1345A appear to be operating in real time. The host is

aware of the state of all signals on the flight simulator. This is accomplished by the host simply requesting the information from the interface via a basic command. The interface also informs the host immediately when the Control Keyboard becomes active.

The Data Acquisition and Communications Interface performs all of the above tasks and provides as much data preprocessing as possible to keep the host's overhead to a minimum. This will allow the host to maintain real time processing. In addition, the host communication with the interface is accomplished with just four basic commands under an interrupt environment. In order for the reader to fully understand how the interface operates, I will describe the parts of the interface and also describe how to use the interface.

The Data Acquisition and Communications Interface consists of two major components. They are the 68000 Educational Board and the Interface Board shown in Figure 18. The Educational Board's 68000 microprocessor provides the interface with the intelligence. The Interface Board is used for all communications with the other system components and provides signal conditioning for the flight simulator analog signals.

68000 Educational Board This board is produced by

INTERFACE BOARD

- FLIGHT SIMULATOR ANALOG PORT
- CONTROL KEYBOARD PORT
- HOST PARALLEL PORT
- PARALLEL PORT FOR HP 1345A
- FLIGHT SIMULATOR SIGNAL CONDITIONING
- REAL TIME CLOCK
- ALARM HORN

BUS AND  
CONTROL SIGNALS

68000

EDUCATIONAL  
BOARD

FIGURE 18. DATA ACQUISITION AND COMMUNICATIONS INTERFACE

Motorola Inc. and is intended for educational use. It provides an effective means for developing hardware and software for a 68000 microprocessor based system. The MC68000 16-bit microprocessor, memory, Centronics standard parallel printer port, and serial I/O communications are located on a single printed circuit card [14]. The Educational Board has a resident firmware monitor called "TUTOR." The user must connect an RS-232C compatible "dumb" terminal to the educational board to access the TUTOR monitor. The dumb terminal is connected to port 1 of the Educational Board and provides a means of data entry and program monitoring. The ports of the Educational Board are shown in Figure 19. The terminal used with the Educational Board is the Zenith-29 Video Display Terminal. It has a 12 inch cathode ray tube, which displays 25 rows and 80 columns. The Z-29 terminal provides I/O capability between the user and the Educational Board. A summary of the Educational Board's features include:

- 4 megahertz M68000 MPU
- 32k bytes of DRAM
- 16k bytes of firmware in ROM "TUTOR"
- Two serial communication ports which are RS-232C compatible and have selectable baud rates.
- Centronics compatible parallel printer interface

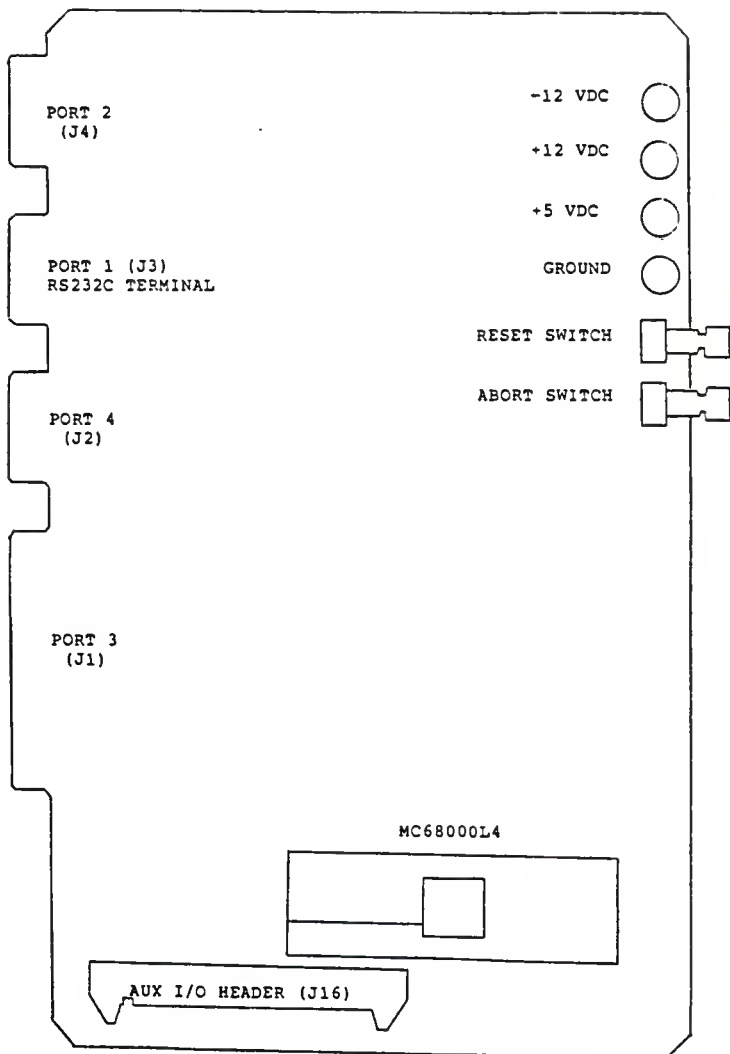


FIGURE 19. EDUCATIONAL BOARD PORTS

Audio tape serial I/O port.

24-bit programmable timer

Wire-wrap area provided for custom circuits

RESET and ABORT function switches.

The Educational Board has two function buttons that allow termination of a program or reinitialization of the board. The RESET button is the black button located on the lower edge of the board. Again refer to Figure 19. This button causes all processes to be halted, reinitializes the board, and restarts TUTOR. The ABORT button is the red button located next to the RESET button. The ABORT button causes a level seven interrupt which vectors control back to TUTOR. No registers are changed and the system does not get reinitialized. The ABORT button is used by the programmer to halt program execution during software debugging.

A functional block diagram of the educational board is shown in Figure 20. Even though the board has many features, the EHSI Development System will only use the board's 68000 and bus signals, user RAM and the TUTOR programming and operating monitor. This will make the transition to a stand-alone 68000 microprocessor system smooth once the EHSI development is completed.

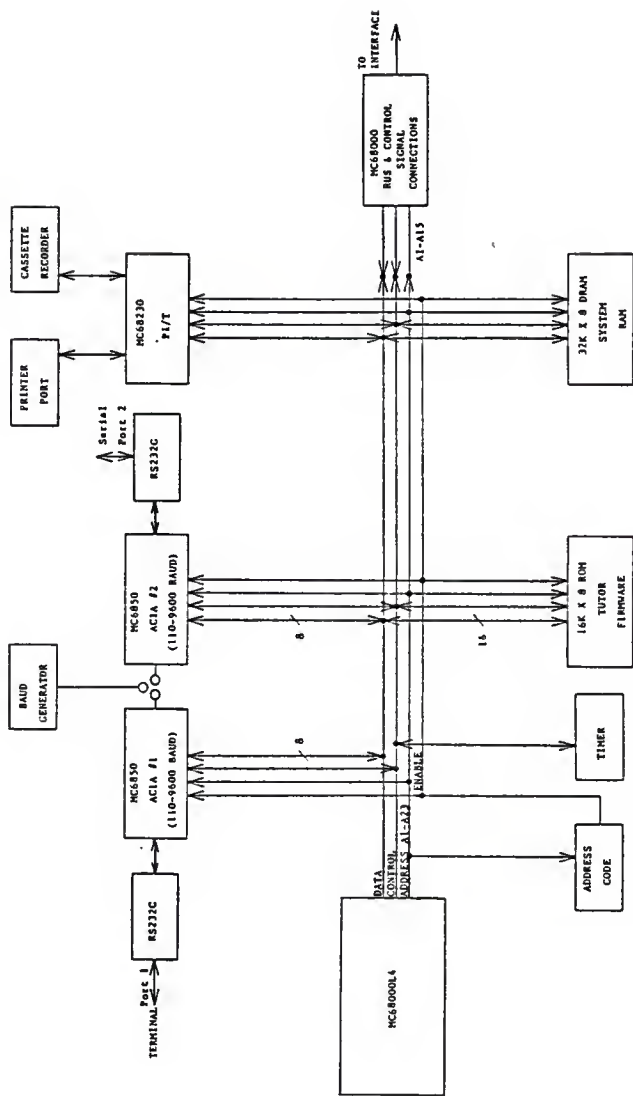


FIGURE 20. EDUCATIONAL BOARD FUNCTIONAL BLOCK DIAGRAM

The Educational Board contains a memory segment decoder, shown on page 181 of Appendix B (for convenience reproduced in Figure 21). This provides the user with address decode lines for several memory segments within the memory map shown in Figure 22. The Interface Board will use two of these signals, E1 and E6. Active low signal E1 is used to enable the two 8k X 8 bit EPROMs which will eventually hold the interface software. E1 signal is enabled for the memory segment \$20000 to \$2FFFF. E6 is an active high signal called M6800 Page Address Decode. This is a 64k byte segment of the system memory map reserved for M6800 type peripherals. Signal E6 is produced as follows. After the MC68000 asserts signal E6, it then waits for valid peripheral address signal line (VPA\*) to be taken low by an external device. The processor then synchronizes itself to the 6800 E clock and continues its bus cycles. Memory segment E6 will be used by all other devices on the Interface Board such as the PIA's, Real Time Clock, and analog to digital converters (ADC).

The 68000 Educational Board is used to control the Interface Board. The Educational Board provides an area for an "Auxiliary I/O Header" (designated J16) which is a 50 pin, right angle, wire-wrap header. It is through this wire-wrap header that the bus and control signals used by



ENABLE SIGNAL	ADDRESS SEGMENT	DESCRIPTION
E1*	\$20000 - \$2FFFF	EHSI EPROM AREA
E2*	\$40000 - \$4FFFF	USER SEGMENT
E3*	\$50000 - \$5FFFF	USER SEGMENT
E4*	\$60000 - \$6FFFF	USER SEGMENT
E5*	\$70000 - \$7FFFF	USER SEGMENT
E6	\$30000 - \$3FFFF	M6800 SYNCHRONOUS PERIPHERALS PAGE

ALL SIGNALS ARE ACTIVE LOW (\*) EXCEPT E6.

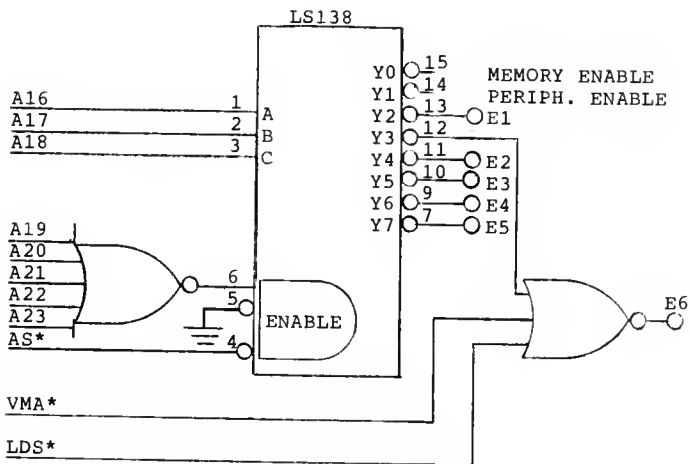


FIGURE 21. ADDRESS DECODE LOGIC AND MEMORY SEGMENT ENABLE SIGNALS

\$000 \$007	ROM RESET EXCEPTION VECTORS
\$008 \$3FF	RAM EXCEPTION VECTOR TABLE
\$900 \$1F9F	EHSI SCRATCHPAD
\$1FA0 \$1FFF	EHSI DATA PACKAGE (95 BYTES)
\$2000 \$37FF	EHSI SUBROUTINE AREA
\$3800 \$3FFF	EHSI LOOKUP TABLES
\$4000 \$5FFF	EHSI MAIN PROGRAM AREA
\$6000 \$7FFF	VECTOR MEMORY MIRROR
\$8000 \$BFFF	ROM TUTOR FIRMWARE MONITOR
\$C000 \$FFFF	NOT USED
\$10000 \$1FFFF	EDUCATIONAL BOARD I/O DEVICES
\$20000 \$2FFFF	E1* (2) 8k x 8 EHSI EPROMs
\$30000 \$3FFFF	E6 M6800 PERIPHERAL PAGE EHSI PIA's, A/D's, REAL TIME CLK
\$40000 \$4FFFF	E2*
\$50000 \$5FFFF	E3*
\$60000 \$6FFFF	E4*
\$70000 \$7FFFF	E5*
\$80000 \$FFFFF	NOT USED

\* DENOTES ACTIVE LOW SIGNAL

FIGURE 22. EDUCATIONAL BOARD MEMORY MAP

the Interface Board are acquired. A ribbon cable then connects these signal lines to the Interface Board. The header pin number with corresponding signal designation is shown in Appendix B, page 119.

Interface Board The Interface Board is the other major component of the Data Acquisition and Communications Interface. A photo of the Interface Board is shown in Figure 23. It is the interface for communications between the 68000 Educational Board, the ATC-610 Flight Simulator, the Control Keyboard, the HP-1345A display and the host computer. Therefore, the interface is used to link the EHSI Development System components together. A functional block diagram of the interface may be found in Figure 24.

The Interface Board has five ports which are used to control the development system components, a real time clock/ calendar, an alarm horn circuit to alert the pilot for emergency conditions and 16k bytes of EPROM space used for the final version of the interface software. The five ports on the Interface Board are the 50-pin signal port from the Educational Board, the 26-pin port for the HP-1345A display, the 20-pin port for the Command Keyboard, the 26-pin port for the flight simulator analog signals and the 26-pin parallel port for communications with the host. The Data Acquisition and Communications Interface

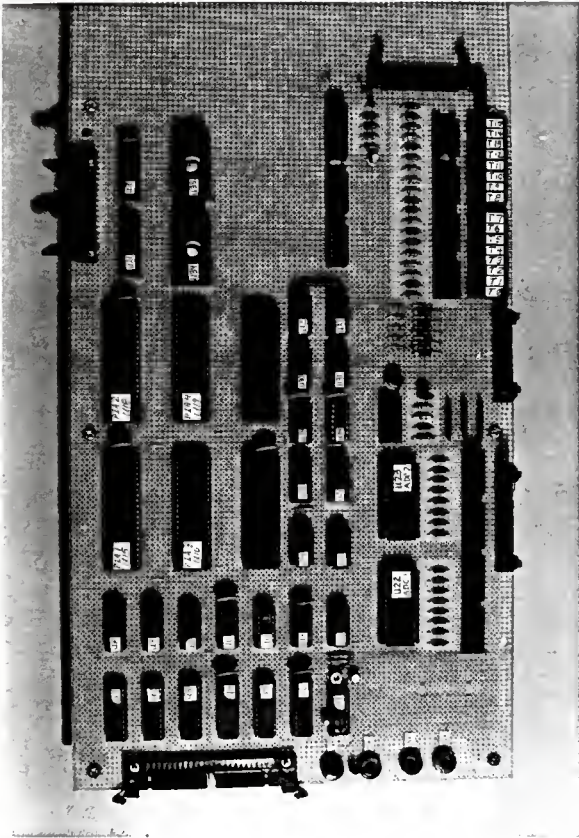


FIGURE 23. INTERFACE BOARD

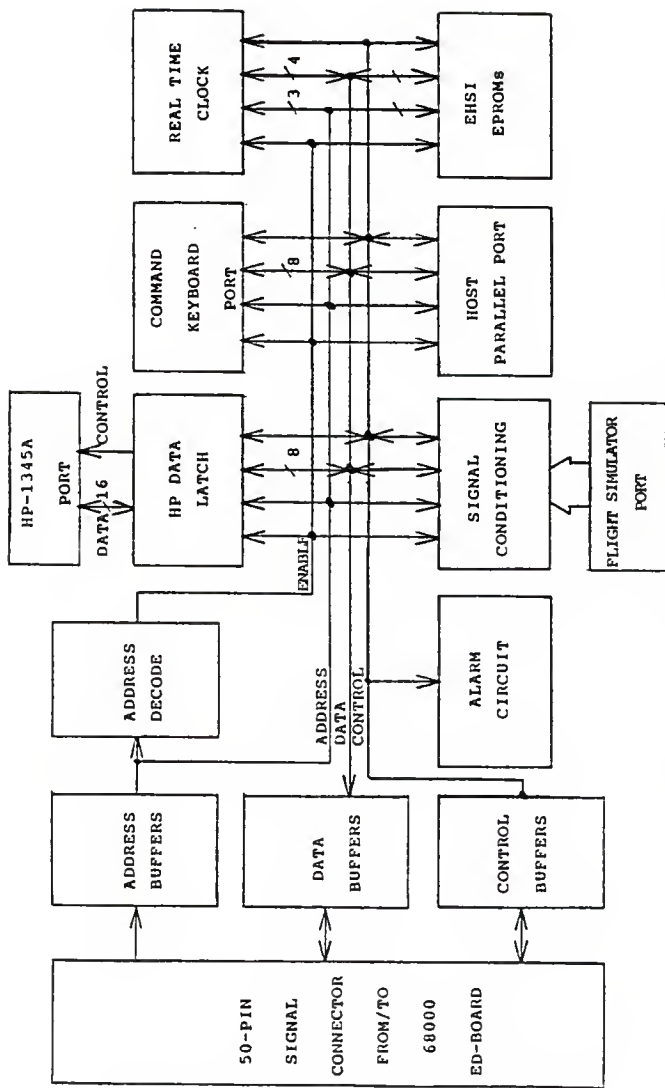


FIGURE 24. INTERFACE BOARD FUNCTIONAL BLOCK DIAGRAM

schematics may be found in Appendix B.

The Interface Board uses the 50-pin connector to access the bus and control signals on the 68000 Educational Board. These signals are the data lines, address lines, clock lines, 68000 interrupt request line, enable lines and control lines. The signals are first buffered by LS244 and LS245 (U1 - U5) bus driver integrated circuits before going on to the appropriate destinations on the Interface Board. The data lines are tri-state and enabled only when E1 or E6 are asserted. This indicates that the program is referencing either one of these memory segments. The direction of these data buffers is determined by the processor R/W control signal.

The address lines and Educational Board signal E6 are used by decoders U8 and U9 to provide address decode lines within the M6800 Address Decode Page (E6). A memory map for the E6 page is shown in Table 1. The table tabulates the decode addresses for the various devices on the Interface Board. The addresses are fully decoded, therefore, there are not multiple addresses within the E6 memory segment that may enable the same decoder output. Each decoder has eight output enable lines. The following notation will be used to describe the decoder output enable lines. EN14\* implies decoder 1 line 4 will go low when the

TABLE 1. DEVICE DECODE ADDRESSES

(DECODER #1)

ADDRESS	DEVICE	DESCRIPTION	FUNCTION
EN10*			
\$30001	PIA #1	PORT A DATA REGISTER	CONTROL
\$30003	PIA #1	PORT A CONTROL REGISTER	
\$30005	PIA #1	PORT B DATA REGISTER	IRQ INPUTS
\$30007	PIA #1	PORT B CONTROL REGISTER	
EN11*			
\$30009	PIA #2	PORT A DATA REGISTER	HP DATA
\$3000B	PIA #2	PORT A CONTROL REGISTER	LSB
\$3000D	PIA #2	PORT B DATA REGISTER	HP DATA
\$3000F	PIA #2	PORT B CONTROL REGISTER	MSB
EN12*			
\$30011	PIA #3	PORT A DATA REGISTER	KEY MATRIX
\$30013	PIA #3	PORT A CONTROL REGISTER	COLS.
\$30015	PIA #3	PORT B DATA REGISTER	KEY MATRIX
\$30017	PIA #3	PORT B CONTROL REGISTER	ROWS
EN13*			
\$30019	PIA #4	PORT A DATA REGISTER	BINARY I/P
\$3001B	PIA #4	PORT A CONTROL REGISTER	
\$3001D	PIA #4	PORT B DATA REGISTER	SPARE
\$3001F	PIA #4	PORT B CONTROL REGISTER	
EN14*			
\$30021	PIA #5	PORT A DATA REGISTER	PAR PORT
\$30023	PIA #5	PORT A CONTROL REGISTER	
\$30025	PIA #5	PORT B DATA REGISTER	SPARE
\$30027	PIA #5	PORT B CONTROL REGISTER	
EN15*			
\$30029	PIA #6	PORT A DATA REGISTER	FUTURE
\$3002B	PIA #6	PORT A CONTROL REGISTER	
\$3002D	PIA #6	PORT B DATA REGISTER	FUTURE
\$3002F	PIA #6	PORT B CONTROL REGISTER	
EN16*			
\$30031	A/D #1	ANALOG INPUT #1	BANK
\$30033	A/D #1	ANALOG INPUT #2	PITCH
\$30035	A/D #1	ANALOG INPUT #3	VERT SPEED
\$30037	A/D #1	ANALOG INPUT #4	DELTA X
EN17*			
\$30039	A/D #1	ANALOG INPUT #5	DELTA Y
\$3003B	A/D #1	ANALOG INPUT #6	MAN PRES.
\$3003D	A/D #1	ANALOG INPUT #7	CDI/ILS
\$3003F	A/D #1	ANALOG INPUT #8	GLIDESLOPE

TABLE 1. DEVICE DECODE ADDRESSES CONTINUED

(DECODER #2)

ADDRESS	DEVICE	DESCRIPTION	FUNCTION
EN20*			
\$30041	A/D #2	ANALOG INPUT #9	ALTIMETER
\$30043	A/D #2	ANALOG INPUT #10	AIRSPEED
\$30045	A/D #2	ANALOG INPUT #11	COMPASS
\$30047	A/D #2	ANALOG INPUT #12	ADF
EN21*			
\$30049	A/D #2	ANALOG INPUT #13	DME
\$3004B	A/D #2	ANALOG INPUT #14	% POWER
\$3004D	A/D #2	ANALOG INPUT #15	RPM
\$3004F	A/D #2	ANALOG INPUT #16	SPARE
EN22*			
\$30051	NONE	AVAILABLE ADDRESS DECODE LINE	
\$30053			
\$30055			
\$30057			
EN23*			
\$30059	NONE	AVAILABLE ADDRESS DECODE LINE	
\$3005B			
\$3005D			
\$3005F			
EN24*			
\$30061	CLK/CAL	TEST ONLY	
\$30063	CLK/CAL	TENTHS OF SECONDS	
\$30065	CLK/CAL	UNITS OF SECONDS	
\$30067	CLK/CAL	TENS OF SECONDS	
EN25*			
\$30069	CLK/CAL	UNITS OF MINUTES	
\$3006B	CLK/CAL	TENS OF MINUTES	
\$3006D	CLK/CAL	UNITS OF HOURS	
\$3006F	CLK/CAL	TENS OF HOURS	
EN26*			
\$30071	CLK/CAL	UNITS OF DAYS	
\$30073	CLK/CAL	TENS OF DAYS	
\$30075	CLK/CAL	DAY OF WEEK	
\$30077	CLK/CAL	UNITS OF MONTHS	
EN27*			
\$30079	CLK/CAL	TENS OF MONTHS	
\$3007B	CLK/CAL	YEARS	
\$3007D	CLK/CAL	STOP/START THE CLOCK	
\$3007F	CLK/CAL	INTERRUPT SYSTEM	



appropriate address is decoded.

The second Educational Board enable line used by the Interface Board is enable signal E1. E1 is asserted for an address in the range from \$20000 to \$2FFFF. This 64k byte E1 segment is used to enable the EPROMs (U34, U35) on the Interface Board. The two 8k x 8 bit EPROMs store the high and low order bytes of the interface software. The 16k-bytes of EPROM space reside at \$20000 to \$23FFF within the E1 memory segment. The EPROM circuit will be discussed in more detail later in this Chapter.

As mentioned earlier, the Interface Board has five ports from which the development system components are controlled. These ports allow the components to be controlled and allow a means to communicate information to the host computer. I will now describe the function of each port and the hardware associated with each port. Also, I will describe the hardware used for interrupt processing.

One of the most important ports on the interface is the HP-1345A display port. All communications with the HP display are handled through this port. The port is a bidirectional, 16-bit parallel port with three handshake lines and consists of two 8-bit ports of Peripheral Interface Adapter #2 (U18). The PIA's two 8-bit ports are capable of input or output. PIA #1 is used as a general

purpose control PIA. The three handshake lines for the parallel port originate from this PIA. All data lines are buffered by two bidirectional bus drivers U20 and U21. An LS245 (U6) buffer, is used to buffer the three handshake lines DS\*, RD\*, and WR\*. Please refer to page 120 of Appendix B for a schematic of the HP-1345A parallel port.

The data lines and handshake lines are supplied to the HP-1345A via a 26 pin, wire-wrap connector on the Interface Board. Four signal lines on the HP-1345A's parallel port that are unused. They are SYNC (used for external refresh), XACK\* (for very fast data transfers), A0 and A1. The HP-1345A has a test display that appears on power up whenever the "disconnect sense" line is not grounded. Since the display is used to show the EHSI pages, the disconnect sense line is tied to ground. The actual programming of the HP-1345A Vector Graphics Display will be explained in the software section of the next chapter.

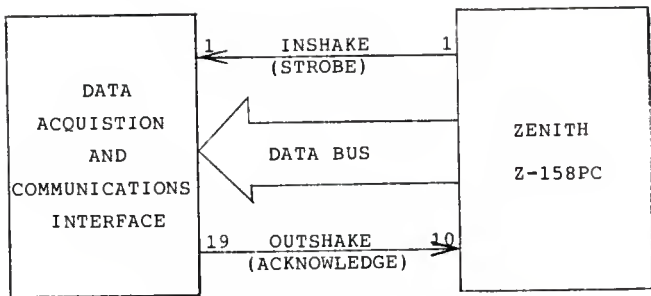
An 8-bit parallel port with a two wire handshake is used to communicate with the Zenith-158 Personal Computer. Port A of PIA # 5 is used as the 8-bit, bidirectional, parallel port. Refer to page 123 of Appendix B for a detailed schematic of the parallel port. The two handshake lines, used to control the data transfer to and from the

Interface Board, originate from PIA #1, the control PIA. These two handshake lines are of special interest and will be described in a moment.

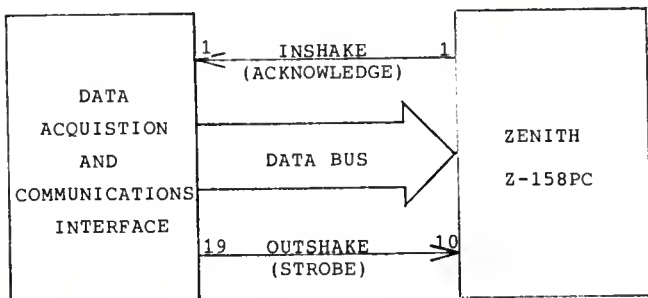
The Zenith's parallel port is used as a general purpose I/O port. The data transfers will be bidirectional. The Zenith User's Manual states that its parallel port may be used for input as well as output. However, there is a problem with the Zenith parallel port hardware. Since in the 8088 microprocessor system, the address and data lines are multiplexed on the same bus, the data lines must be latched by flip-flops. U261 (LS374) is used by the Zenith for the purpose of latching data at the output of the parallel port. This works well. However, when the port is used for input, U261 is still latched with the last character output to the port. Therefore, when the port is read through input buffer U260 (LS244), the latched output data is read instead of the input data at the port connector. The solution was to tri-state the output latch (U261) during a read cycle. This solution was easily implemented since the latched outputs are able to go to high impedance through application of a TTL high level to pin 1 of U260. The RPA\* signal, which enables the read buffers, was inverted and applied to pin 1 of U261. This will send the latched outputs to high impedance during a read cycle. The

inverter (LS04) rides "piggyback" on U261. The hardware modification has no effect on the parallel printer port operation.

A bidirectional, tri-state data buffer on the Interface Board is controlled by the Zenith-158 to isolate the Interface Board from the Zenith parallel port during non-transfer states. Only two handshake lines will be used for communications between the Interface Board and the Zenith-158. The two lines represent a data strobe and a data acknowledge. In order to keep the number of handshake lines to a minimum, the two lines will be used for bidirectional communications. Since the same handshake line represents a data acknowledge when transfer is in one direction and a data strobe when communication is in the other direction, it becomes confusing to use the names acknowledge and strobe. Therefore, the names Outshake and Inshake are used. Outshake represents a pulse output on the output handshake line by the Interface Board. Outshake will represent a data strobe or acknowledge, depending on the direction of communications. Please refer to Figure 25. Inshake is used to represent the input handshake line. A pulse accepted on this line by the Interface Board may represent a data strobe or acknowledge depending on the direction of the data transfer. When the Inshake pulse



TRANSFER TO 68000 FROM ZENITH



TRANSFER TO ZENITH FROM 68000

FIGURE 25. 68000/PC COMMUNICATIONS PROTOCOL

represents a data strobe for the Zenith-158, the 68000 uses the pulse to initiate interrupt processing.

A hardware interrupt is provided on the parallel port of the Zenith-158 PC. The interrupt is initiated when the Z-158 PC receives a pulse on its strobe or Inshake line. The interface software will utilize this hardware to interrupt the Zenith when the clock data has changed, when a key has been pressed on the Control Keyboard, or to signal the host software that the EHSI has been turned on or off. More about 68000 interrupt processing may be found in the EHSI Development System Software chapter that follows.

The Control Keyboard consists of a 36 key keypad and the EHSI System Switch. The keypad, shown on page 121 of Appendix B, is organized as a six-by-six matrix. The buttons are common by rows and common by columns. The common row wires are connected to Port B of PIA #3 and the column wires to Port A of PIA #3 via the 20-pin Control Keyboard connector. The rows are always left in the output logic low state. The three AND gates (U28) are wired to the columns. The columns will all read as TTL logic level high when no keys are depressed and the output of the AND gates will be high. When any key is pressed the column of the depressed key will be at a TTL low logic level and the

output of the AND gate will go low. This signal is called KEYIRQ\*. It is tied to the processor interrupt line and to a interrupt flip flop latch shown on page 122 of Appendix B.

This hardware interrupt logic is used to recognize and start key identity interrupt processing whenever a key is pressed on the keypad. The key identity is determined by the KEYIRQ\* interrupt service routine as follows. All of the row and column wires are pulled up to +5 volts through 4700 ohm resistor networks. In the normal state, the interface software will ground the rows by outputting logic low level on Port B of PIA#3. The columns are then read. If any columns are found to be a logic low level, a hardware interrupt is generated to the 68000. This will indicate to the processor that a key has been pressed somewhere in the column that went low. When this KEYIRQ interrupt is generated by a depressed key, the 68000 outputs a logic low level on Port A of PIA #3. The columns are now at a TTL logic low. The rows are then read through Port B of PIA #3. The row having a TTL low logic level will indicate the row of the depressed key. Now that the column and row of the depressed key are known, the key number can be determined.

The EHSI System Switch on the Control Keyboard is a

simple SPST switch that is grounded on one side, and has a 4700 ohm pull-up resistor to +5 Volts on the other side. The switch is used to start and to stop the EHSI system. When the switch is in the "OFF" position, the SPST switch forms an open circuit and the system switch input becomes a TTL high level. With the switch in the "ON" position, the input goes to a TTL low level. The EHSI System Switch input is on pin 13 of PIA #1, Port B.

A MM58174 (U7) Real Time Clock and Calendar is provided on the Interface Board and is used inform the host of current time and date information. The clock/calendar schematics are on page 124 of Appendix B. The clock is decoded at addresses \$30063 to \$3007F and the time and date registers are memory-mapped to the addresses shown in Table 1. The clock has a battery backup of three volts. This provides uninterrupted operation of the time register updates during power down. The clock device contains an interrupt timer which generates a logic low interrupt output upon timer time out. This interrupt is called CLKIRQ\*. Like KEYIRQ\*, it is tied to the processor interrupt line and to a interrupt flip-flop latch. The timer may be programmed to interrupt every 60, 5 or 1/2 seconds. The clock timebase is generated from a 32.768 kHz crystal-controlled oscillator and can be adjusted by tuning a



capacitor located next to the real time clock chip (U7).

There are two 8192 x 8 bit UV erasable programmable read only memories (MCM68764) onboard the Interface Board. They will be used to store the final version of the interface software. The 68000 has a 16-bit data bus with the memory organized as 8-bits or one byte. The two EPROMs are 8-bit devices. Therefore, one EPROM (U34) will store the low order byte of a word and the second EPROM (U35) will store the high order byte of the 16-bit word. The EPROMs will be enabled whenever address decode signal E1 (\$20000 - \$2FFFF) is low and when the processor is in a read cycle (signified by R/W being high).

The EPROMs operate from a single power supply and have maximum access time of 450 nanoseconds. A Data Transfer Acknowledge (DTACK\*) signal is generated and returned to the 68000 whenever the E1 signal is decoded. DTACK\* provides an asynchronous bus cycle to allow the 68000 to interface to devices with various access times. The DTACK\* signal is returned approximately 500 nanoseconds after the EPROMs are enabled. This will provide enough time for the EPROMs to access the data. The EPROM DTACK\* line is connected to a three-input AND gate with open collector output. The other inputs of the AND gate are tied high through a 4700 ohm resistor. These inputs are

provided as future DTACK\* inputs. The open collector AND gate output is connected to DTACK\* connection point E7 on the 68000 board, as shown on page 124 of Appendix B.

Three devices on the Interface Board can initiate exception or interrupt processing. They are the real time clock, the Command Keyboard and the host computer via the parallel port. Positive edge triggered flip-flops (U32, U33) are provided to latch these interrupts. As shown on page 122 of Appendix B, the outputs from the flip-flops go to bits 4-7 of PIA #1, Port B. The flip-flops used to latch the interrupts are provided for two reasons. The first reason is so the keyboard does not give multiple interrupts for one key closure and the second reason is to catch all interrupts in case an interrupt comes and goes while the software is servicing an interrupt of a higher priority. The flip-flops have clear inputs that enable the software to clear the interrupt latches, via bits 4-7 of PIA #1, Port A, once the interrupt service routines have finished.

The analog portion of the interface board consists of two analog-to-digital convertors, signal conditioning of the analog flight simulator signals for the analog-to-digital convertors and an alarm circuit. Two AD7581 analog-to-digital (ADC) convertors are used to digitize the analog

flight simulator signals. The AD7581 is a microprocessor compatible 8 bit, 8 channel memory buffered ADC. It consists of an 8 bit successive approximation ADC, an 8 channel analog multiplexer, an 8 x 8 bit dual-port RAM and microprocessor compatible control logic.

The conversion takes place on a continuous, channel sequencing basis using the M6800 E clock from the 68000 as the ADC clock. Converted digital data values are automatically transferred to the proper location within the 8 x 8 bit dual-port RAM onboard the ADC. Any channel of the dual-port RAM may be read at any time. The ADC may be used for unipolar (0 V to +10 V) operation or for bipolar (+5 to -5V) operation. ADC #1 has analog inputs 0 through 7 (AIN0-AIN7) and is used in the bipolar mode. ADC #2 is applied analog inputs 8 through 15 (AIN8-AIN15) and operates in the unipolar mode. An AD581 is used as a external precision voltage reference by the AD7581's shown on page 128 of Appendix B. The sixteen analog signals each have their own address within the M6800 address decode page E6. These addresses are shown in Table 1. Four decoder enable signals are used to select the ADC's. EN17\* and EN18\* select ADC #1 and EN20\* and EN21\* select ADC #2.

An alarm circuit, shown on page 122 of Appendix B, is used to sound a piezo-electric buzzer when a pilot gets

into a dangerous situation. The buzzer itself is rated to 15 mA at a maximum voltage of 20 Volts DC. At 15 mA and +15 Volts the alarm is extremely loud. A volume control potentiometer is used to vary the current through the piezo-electric buzzer from a minimum of 0.13 milliamps to a maximum of 1.1 milliamps. A monostable multivibrator is used to cycle the alarm buzzer on and off at a rate of 5 cycles per second. The multivibrator is enabled by bit 0 of PIA#1, Port B. The alarm will alert the pilot that an alarm condition exists and that immediate corrective action should be taken.

The actual analog signals from the flight simulator are non-uniform and non-symmetric signals. Some are unipolar or bipolar, others range from 6.7 V to 8.3 V while still others only have two possible states and can be classified as binary signals. Due to this uniqueness of each signal, analog circuitry was used to signal condition the analog signals before they could be applied to the ADCs.

Nineteen flight simulator signals are taken from the instruments of the flight simulator. Four are binary signals and fifteen are analog signals. The signals are acquired from within the flight simulator at points such as potentiometers, selector switches, avionic instruments,

etc. Table 2 shows the voltage range of each signal, the type of signal after conditioning and where the signal was acquired from within the flight simulator. All signals are first fed through various configurations of an operational amplifier circuit which has an input impedance on the order of Terra ohms. This ensures that the flight simulator is essentially isolated and does not "see" the interface board. Next, the operational amplifier circuits translate the analog voltage levels into the proper range before the signals can be applied to the ADCs. The signals are then digitized by the ADCs and are ready to be processed by the EHSI software.

In Chapter 4, the EHSI Development System Software will be described. This software is used to control the Data Acquisition and Communications Interface and the host computer. I will first describe the software that controls the Interface then the host software will be discussed.

TABLE 2. FLIGHT SIMULATOR SIGNAL INFORMATION

SIGNAL	VOLT. RANGE <sup>1</sup>	SIGNAL TYPE <sup>2</sup>	FLIGHT SIMULATOR EXTRACTION POINT
BANK	$\pm 8$ V	BIPOLAR	PIN 50 OF MAIN CONTROL CARD
PITCH	$\pm 5$ V	BIPOLAR	PIN 39 OF MAIN CONTROL CARD
VERT. SPEED	$\pm 12$ V	BIPOLAR	PIN 33 OF MAIN CONTROL CARD
DELTA X	$\pm 13$ V	BIPOLAR	PIN C OF FUEL/DME/GS CARD
DELTA Y	$\pm 13$ V	BIPOLAR	PIN D OF FUEL/DME/GS CARD
MANIFOLD PR.	$+6.7$ $\rightarrow$ $+8.3$ V	BIPOLAR	PIN 22 OF MAIN CONTROL CARD
CDI	$\pm 3$ V	BIPOLAR	PIN C OF CDI AT INSTRUMENT
GLIDESLOPE	$\pm 3$ V	BIPOLAR	PIN E OF CDI AT INSTRUMENT
ALTIMETER	$0$ $\rightarrow$ $-9$ V	UNIPOLAR	PIN 25 OF MAIN CONTROL CARD
AIRSPEED	$0$ $\rightarrow$ $+15$ V	UNIPOLAR	PIN 46 OF MAIN CONTROL CARD
COMPASS/GYRO	$0$ $\rightarrow$ $+9$ V	UNIPOLAR	POTENTIOMETER R9
ADF	$0$ $\rightarrow$ $+8.8$ V	UNIPOLAR	POTENTIOMETER R136
DME	$0$ $\rightarrow$ $+10$ V	UNIPOLAR	NAVIGATIONAL FREQ. SELECTOR SWITCH
PERCENT POWER	$0$ $\rightarrow$ $+3.7$ V	UNIPOLAR	PIN 7 OF MAIN CONTROL CARD
RPM	$0$ $\rightarrow$ $+1.7$ V	UNIPOLAR	PIN 20 OF MAIN CONTROL CARD
LAND. GEAR	$+15$ V or GND	BINARY	PIN 44 OF MAIN CONTROL CARD
TO/FROM FLAG	$+0.1$ /GND/ $-0.3$	BINARY	PIN A OF CDI AT INSTRUMENT
OUTER MARKER	$+15$ V or GND	BINARY	OUTER MARKER INDICATOR LIGHT
MIDDLE MARKER	$+15$ V or GND	BINARY	MIDDLE MARKER INDICATOR LIGHT

1 VOLTAGE RANGE OF SIGNAL AT SIMULATOR

2 TYPE OF SIGNAL AFTER SIGNAL CONDITIONING ON INTERFACE BOARD

3 SIGNAL IS POSITIVE FOR TO, NEGATIVE FOR FROM, GROUND FOR OFF

## CHAPTER FOUR

### THE EHSI DEVELOPMENT SYSTEM SOFTWARE

The development system software is used to control the entire development system. The software can be divided into three parts. These are the TUTOR software, the Interface Board software and the host software. This section will begin with a brief introduction to the EHSI development system software. The three major divisions of software listed above will then be discussed in detail.

The Tutor software is used as a monitor/operating system to provide a link between the user and the 68000 Educational Board. It is also used for 68000 assembly language program development and for debugging the assembly language software, in this case the interface software. The interface software provides control for the interface board's communication with the other development system components including the host. The host is responsible for the development of the software which produces the displays on the HP-1345A.

## The Tutor Software

The Educational Board's TUTOR firmware provides a self-contained programming and operating environment. It resides on two 8k x 8 ROM devices and interacts with user via commands entered into a dumb terminal. The commands allow the user to:

1. Display or modify memory
2. Display or modify internal 68000 registers
3. Execute a program
4. Control I/O resources.

The operational mode of TUTOR is shown in Figure 26. An additional function is available in TUTOR called TRAP 14 handler. It allows the user programs to utilize various routines within TUTOR. TUTOR is used to assemble, disassemble, debug the interface software, and acts as a system monitor. It will also be used to boot the interface software in the final version of the development system. For a detailed explanation of all TUTOR commands, please refer to Chapter 3 of the MC68000 Educational Computer Board User's Manual [14].



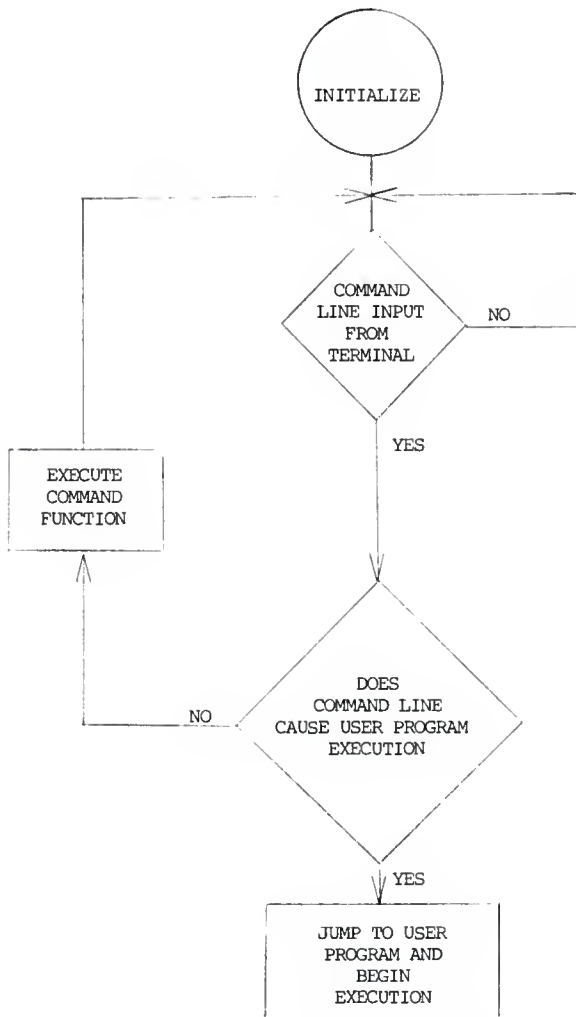


FIGURE 26. OPERATIONAL MODE OF TUTOR MONITOR

## The Interface Software

In order to provide communications between the development system components, the interface software was developed. The interface software runs under the TUTOR operating environment. The interface software consists of a main program, support subroutines and interrupt control/service routines. The software provides initialization of the interface board and other development system components. In addition, the interface software is responsible for continuously updating the data package with the digitized flight simulator signals, the clock time values and the last key pressed on the command keyboard. The data package is generated by the 68000 so the host can have a convenient and concise package of the system parameters. The data package parameters and location in RAM are shown in Table 3. The data package may be sent to the host on request using one of the four basic interface commands.

Four interface commands allow the host to communicate with the development system components and to access system parameters. The commands are:

Command #1) Transfer the data package to the host

Command #2) Transfer HP-1345A commands from the host  
to vector memory

TABLE 3. DATA PACKAGE OF SYSTEM PARAMETERS

RAM LOCATION	DATA PACKAGE ENTRY
\$1FA0	NUMBER OF ENTRIES IN DATA TABLE
\$1FA1	BANK
\$1FA2	PITCH
\$1FA3	VERTICAL SPEED
\$1FA4	DELTA X POSITION
\$1FA5	DELTA Y POSITION
\$1FA6	MANIFOLD PRESSURE
\$1FA7	COURSE DEVIATION INDICATOR
\$1FA8	GLIDESLOPE
\$1FA9	ALTIMETER
\$1FAA	AIRSPPEED
\$1FAB	COMPASS / DIRECTIONAL GYRO
\$1FAC	ADF
\$1FAD	DME
\$1FAE	PERCENT POWER
\$1FAF	RPM
\$1FB0	SPARE UNIPOLAR ANALOG INPUT
\$1FB1	BINARY INPUTS
	BIT 0 GEAR UP / DOWN
	BIT 1 TO / FROM / OFF FLAG
	BIT 2 TO / FROM / OFF FLAG
	BIT 3 OUTER MARKER
	BIT 4 MIDDLE MARKER
	BIT 5 SPARE
	BIT 6 SPARE
	BIT 7 SPARE
\$1FB2	LAST KEY PRESSED ON COMMAND KEYBOARD
\$1FB3	MONTH
\$1FB4	DAY OF THE WEEK (01 = SUNDAY)
\$1FB5	DATE
\$1FB6	HOURS
\$1FB7	MINUTES
\$1FB8	SECONDS

Command #3) Read a block of HP-1345A commands from  
the vector memory and send to the host

Command #4) Toggle the alarm on the interface board

The interface software was developed in a modular fashion using the "top-down" style of structured programming. The system is designed to operate under the interrupt environment. This implies that there are very few lines of code within the main program loop. Almost all of the work is accomplished by the interrupt service and support routines.

Support Subroutines Eleven subroutines are used by the main and interrupt programs. They involve initializing development system components, reading or sending HP-1345A commands to vector memory, setting the real time clock's time registers, finding which key is pressed on the command keyboard, getting the digitized values for the analog flight simulator signals, and performing communications with the host computer via the parallel port. The subroutines are all written to avoid destroying registers, if possible. Subroutines will aid in making the interface software more modular and structured. This will make the code more readable and easier to debug. Also, the use of subroutines instead of the same code will provide higher code densities. The subroutines used by the main program

will now be described functionally. For detailed documentation and actual 68000 code please refer to Appendix C.

The first two subroutines are INTPIA and INTVCM, which are used for initialization. INTPIA is used to initialize a port of a PIA. Each individual bit in both Port A and Port B, of the PIA, may be designated for input or output. To establish the direction of these bits the port control register must first be set to zero. If the bit is to be an output bit, then the bit in the data direction register must be a one. If the bit is to be an input bit, there must be a zero in the data direction register. After the direction is established, \$04 must be written to the port control register. Now the actual port pins will be referenced when writing to or reading from a PIA port.

INTVCM will initialize the HP-1345A's vector memory. When the HP-1345A is turned on the contents will be unknown since synchronous refresh can only occur after an "internal jump to 4095 (\$8FFF)" command. The memory is filled with internal jumps to 4095 during the development system initialization process. The subroutine also places a NOP (\$0000) command in location 4095 since an internal jump in memory can not jump to another internal jump command.

The next three subroutines are used to write/read

commands to/from vector memory. The three subroutines are SNDCMD, SNDLST, AND RDVCM.

SNDCMD is used to send a single HP-1345A command to vector memory. The 16-bit command may be either a graphic command, a vector memory address pointer command or an internal jump command. The proper handshake to transfer the command is controlled by this subroutine.

SNDLST allows vector memory to be sent an entire list of commands. The subroutine is given the starting address of the list and the number of commands in the list. Again, so that code is not repeated, subroutine SNDCMD is called to send each individual command.

RDVCM is used to read a block of HP-1345A commands from vector memory. The subroutine is sent the starting and ending addresses of the block to be read. The subroutine then sets the vector memory address pointer to the first command in the block to be read. The handshake lines are then used to give a read cycle handshake to the vector memory. This continues until the entire block is read. The commands are stored in RAM onboard the 68000 Educational Board. This RAM location is called vector memory mirror and resides from \$6000 to \$7FFF (4096 words). The block read from vector memory is stored in the same order and at the same location in the mirror RAM (offset by

\$6000) that it resided in HP vector memory. Subroutine SETCLK uses this subroutine to save the current graphic prior to prompting the user for time data.

GETKEY is called by the KEYIRQ service routine. It determines which key has been pressed on the control keyboard. The subroutine determines which column and row the pressed key is in. It then stores the pressed key number in the "data package", which is an area of memory reserved for simulator instrumentation data and associated information. If no key was found to be pressed, GETKEY returns a value of 37 in the key location of the data package.

SETCLK is used to set the real time clock and calendar on the interface board. The subroutine first saves the current HP-1345A display by storing it in the vector memory mirror RAM. It then takes control of the display and prompts the user for time and date information. The clock is then started by the user when the data becomes valid. The interrupted display is then restored to vector memory from the mirror RAM. The entire process is accomplished by the interface software. Again, this is to keep the host's overhead to a minimum.

ANALOG retrieves the digital values representing the analog flight simulator signals. The sixteen channels from ADC #1 and #2 are retrieved and stored in the Data Package.

This subroutine is called from the main program's main loop. Therefore, the Data Package will continuously (except during an interrupt) be updated with the current signal values.

Three subroutines are used for communications with the host computer. They are OUTSHAKE, INSHAKE AND ERROR.

OUTSHAKE is used to send a pulse on the output handshake line of the parallel port. The pulse may represent a data acknowledge or a data strobe depending on the direction of communications between the interface and the host. The pulse width can be varied by the programmer changing one instruction in this subroutine. It is currently set for a 25 microsecond pulse, which is more than adequate.

INSHAKE is also used for communications with the host. It waits for an input pulse on the input handshake line. This pulse may represent a data strobe or a data acknowledge depending on the direction of communication. The subroutine will wait for a maximum of 45 milliseconds for the line to go active and also 45 milliseconds for the line to go inactive. Therefore, the pulse width can vary from a minimum of 15 microseconds to a maximum of 45 milliseconds. Hence, it is very forgiving of a "sloppy" pulse. If the subroutine waits more than 45 milliseconds, either for active or inactive status, the subroutine ERROR is



called to trap the error.

ERROR is used to display an error condition on the Z-29 terminal so it may be brought to the attention of the user. There are four possible errors that may be flagged. They are Strobe Error, Acknowledge Error, Illegal Memory Error and Vector Memory Error. These are described in the documentation provided in Appendix C.

The main program starts out by initializing the system and the system parameters. It initializes all Peripheral Interface Adapters (PIA) and their ports and then initializes all interrupts. The Real Time Clock also gets initialized as well as the HP-1345A Vector Memory. The last initialization involves setting up the interrupt vectors for exception processing.

After all initializations are complete, the main program will display the message "INITIALIZATIONS COMPLETE" on the terminal and then instructs the pilot to "CONTINUE TO THE FLIGHT SIMULATOR." The main program then waits for the pilot to turn the EHSI System Switch on.

When the EHSI System Switch is turned on, the EHSI ready message will appear on the HP-1345A and all interrupts will be enabled. The 68000 will then inform the Host that the system is up and running and to start normal data processing. Now the main program enters its very short

main loop. Once in the main loop, the program goes about its task of continuously updating the "Data Package" and monitoring the EHSI System Switch. The main loop is the point that is interrupted when interrupt processing occurs.

The program constantly monitors the state of the EHSI System Switch. When the system is shutdown the main program disables all interrupts, displays a shutdown message and informs the Host of the shutdown. It then waits for the system switch to be turned back on. When it is turned back on, the system and the entire initialization process is restarted.

It is important to note that the main loop of the main program consists of just a few lines of code. Since the system is operating under an interrupt environment, most all of the work is done by the interrupt service routines.

Interrupt Processing Interrupt processing software design was chosen over the polling method for one important reason. Polling may be adequate now, however when the host software is compiled to run on the 68000, polling will not be adequate. Although interrupt programming code is more complex, it is a more efficient means of servicing periph-

erals since the overhead is small compared to polling. This will become important when the host software is added to the interface software.

The Interface Board hardware is designed to take advantage of the level 4, autovectorred interrupt #28 within the 68000. The clock interrupt, keyboard interrupt and host parallel port interrupt are wired so that any one interrupt may produce a level 4 interrupt.

When the 68000 receives a level 4 interrupt it completes the current task and then pushes the return address of the next instruction and the status register onto the stack. The processor then changes the interrupt priority mask to ignore interrupts equal to or less than level 4. The processor uses the interrupt vector to form an address by multiplying the vector (#28 or \$1C) by four. This will point to \$70 within the exception vector table. At this address in RAM, the INTERRUPT CONTROL ROUTINE (ICR) starting address is found (32 bits). The ICR is responsible for determining what device is requesting the interrupt and then executes the appropriate service routine. In order for the reader to understand the interrupt routines it is important to first understand the theory of the interrupt processing programs.

The 68000 may be interrupted by three devices, the

clock, the keyboard and the parallel port. The clock interrupt timer can be programmed to interrupt every 60, 5 or 1/2 seconds or not at all. The keyboard generates a level 4 interrupt whenever a key is pressed on the command keyboard. The Zenith-158 can interrupt the 68000 by sending a pulse (data strobe) on the 68000's input handshake line of the parallel port. Figure 27 shows the interrupt control routine's flow chart which will now be discussed.

When the 68000 receives an interrupt, it vectors control to the INTERRUPT CONTROL ROUTINE (ICR). The ICR determines which device is requesting the interrupt by checking the interrupt status bits of PIA #1, Port B, bits 7 through 4. After determining the device (which bit of the interrupt status bits) that is requesting the service, it then multiplies the bit number by four and uses this as a vector offset to point to the address of one of the three service routines. The three service routines are COMIRQ, KEYIRQ and CLKIRQ. (Note there are four available interrupt bits in the interrupt status bits but only the three are used. This allows for future addition of an additional interrupt).

COMIRQ service routine is initiated by the host sending a data strobe to the parallel port. When the 68000

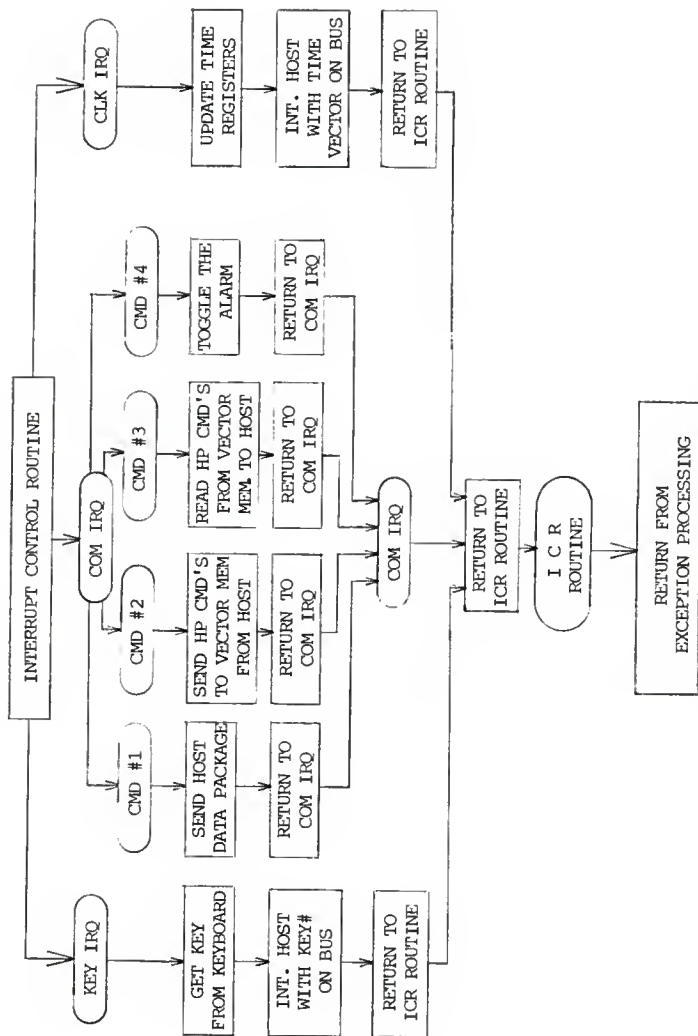


FIGURE 27. FLOW DIAGRAM OF EXCEPTION PROCESSING

is interrupted by the host, there will be an interface command number on the data bus of the port sent from the host. COMIRQ multiplies this vector by four and uses it as an offset to retrieve one of the four interface command service routine addresses. There are space available for seven interface command vectors, however only four commands are used at this time.

Interface Command #1 transfers the contents of the data package to the host. Interface Command #2 is used by the host to transfer HP-1345A commands to vector memory. Interface Command #3 is used to read a block of vector memory commands (specified by the host) and transfers them to the host. Interface Command #4 will toggle the alarm on the interface board. Detailed documentation of these interface commands may be found in Appendix C.

KEYIRQ is the second interrupt service routine. This routine is initiated on an interrupt caused by any key being pressed on the control keyboard. The routine calls subroutine GETKEY which returns the number of key pressed. If key number 36 is pressed, then subroutine SETCLK is called to set the real time clock/calendar. If the key is not number 36, then the key number is put in the data package and sent to the parallel port after which the host is interrupted via a pulse on the outshake line.

CLKIRQ is the third interrupt service routine. Whenever the real time clock interrupt timer times out (60, 5 or 1/2 second) this routine is executed via the ICR. The routine first reads the seconds, minutes, hours, days, date and month and stores this data in the data package. The routine then alerts the host that the time has changed by sending the arbitrary time change vector (\$60) on the parallel port. The host may then turn around and ask for the data package (using interface command #1) to get the new time values.

After an interrupt has been serviced, the service routine returns back to the ICR. The ICR then executes the RTE instruction which pulls the return address off the stack and then places it in the program counter. The exception processing is complete when the main program execution continues.

#### Proposed Host Software

The host software can be divided into two parts. The first part is communications software and the second part is EHSI display development software.

The communications software is responsible for transferring data back and forth between the host and the other

EHSI development system components. It must also be able to supply the EHSI development software with current flight simulation data. The development software should be responsible for using the information from the communications software to develop EHSI displays.

I propose to have the communications software run (keyed from some interrupt) as a background process. This will provide the host with communications and data package information from the 68000 Data Acquisition and Communications Interface. This information would be read by the communications software and stored at some location in RAM. This location should be agreed upon by the display development software and the communications software (background process).

The EHSI display development software will be free-running and not be part of the communications software interrupt processing. The host development software will generate displays from the data (data package information stored in the reserved memory location) and will be able to call the communications software to send the display data to vector memory.



CHAPTER FIVE  
RECOMMENDATIONS FOR FUTURE WORK  
AND  
CONCLUSION

Now that the proposed EHSI and display pages have been presented, it is time to use the EHSI Development System to develop and evaluate the initial phase of the EHSI. This development is only the first of many phases the EHSI will go through. There will be many areas that will need consideration during the EHSI development. In this chapter, I will give recommendations for work to be completed and also comment on areas that may need consideration as the EHSI matures.

Future Considerations

The phases of development of the EHSI fall into one of five general categories. The first task is to develop the display pages. Once developed, the group may begin ground testing of the raw EHSI system. After the perfor-

mance evaluations are complete the next step will be to research and design a flight test prototype. Extensive flight testing should occur during this phase of development. By constant evaluation and possible prototype redesign the EHSI should progress toward completion. Once the development group feels confident about the product, the question of whether to market the EHSI can be considered. In this chapter, I will discuss each one of these development phases, recommending work to be considered.

Display Page Development The first work is obviously to use the EHSI Development System to develop workable display pages. This is accomplished by using the host computer to acquire and process the needed information to produce display pages in the format described in Chapter 2. At this point it must also be determined how much information is to be entered by the pilot during preflight. This includes information such as navigation and communication frequencies, estimated wind speed and direction, waypoint locations etc. The development of a typical operational sequence would be helpful to maintain consistency and helpful for display operation.

Once the basic display page development software is operational, it should be compiled and assembled into 68000 machine code to be executed by the 68000. This will take

the slow host computer out of the processing loop. Also, any effort to move away from the dependency on the host will be beneficial during prototype design. After the initial pages have been developed, ground testing and evaluation of the EHSI system should begin.

Ground Testing and Evaluation Once the basic system is operational, ground testing and simulation should begin. This phase of development is for evaluating and refining the display pages. Also, it should be determined what details or portions of the pages are not possible to produce. Are these problems due to the poor accuracies of the flight simulator and will the actual avionics yield higher accuracies? Questions such as these should be evaluated.

The EHSI performance should continuously be evaluated. This evaluation may be by actual IFR rated pilots or by someone within the development group. The progress of the EHSI end product should always be evaluated and certain questions should be asked about the direction of the EHSI. These questions should include: Is the EHSI remaining basic or is there too much information cluttering the display pages? Does the EHSI reduce the pilot's workload or is the pilot having to juggle more data and make more decisions? Is the information presented in a format that

will aid in comprehension of the data? Have we highlighted so much data that the essential parameters don't stand out? Do the pages include all the appropriate information for the particular phase of flight? The answers to these questions should not be the opinion of one individual but should come from evaluations of the group members and from several IFR rated pilots. The results of these evaluations should be used to refine the EHSI to the point where it is possible to begin building an EHSI prototype.

EHSI Prototype Research and Design Once the EHSI has been refined, it is time to consider a prototype. By this time in the development it will be known what data will be required to produce the display pages. The same data must somehow be acquired from the instruments onboard the actual aircraft. It is doubtful the data will be available in the same form as on the flight simulator. However, directly or indirectly the data needed can be acquired. The data may directly be available from the instruments via a digital bus or a analog signal or the raw data may have to be indirectly produced by sensors or transducers. This area will require a great deal of research into the manufacturer's avionic components and how the EHSI will interface to them.

Other areas must be considered before the EHSI

prototype is designed. Redundancy of the vital components should be considered. Power requirements and reliability will certainly need consideration. The Federal Aviation Administration's rules and guidelines for electronic avionic components should be researched and incorporated into the design. These are just a few of many areas to be considered before the actual prototype is designed. Once these questions have been answered and once the prototype has been designed, it is time for flight testing and evaluation.

Flight Testing and Performance Evaluation When the EHSI prototype is completed it should be ready for the next phase of the EHSI development, actual flight testing. This phase begins with the temporary installation of the EHSI unto the aircraft. The installation and interfacing to instruments should have no effect on the operation of existing instrumentation.

Once the system is installed, flight testing and evaluation may lead to further refinement. The refinements or modifications should proceed as smoothly as possible. The refinements could be as simple as removing the EPROMs on the EHSI interface and simply reprogramming them on the ground to be reinstalled in the aircraft. Again, the same type of questions should be posed as during ground testing

and simulation. In addition, it should be determined if the pilots are depending on the EHSI too much or not enough. The testing and evaluation loop should continue until the EHSI development group is satisfied with the performance of the system. The next step includes research into the possibility of marketing the EHSI.

Marketing the EHSI If the EHSI development group decides that there does indeed exist a niche in the market for this digital avionic instrument, the next step would be to market the EHSI. Research should be conducted to establish the market where the EHSI could benefit the most. The EHSI should then be targeted at this market. Cost should be determined to allow a profit margin and to allow it to remain affordable to the small aircraft owners. The EHSI design should be protected against copyright infringement and also the EHSI design group should be protected against possible loss of benefits and time due to a duplicate system appearing on the market.

Research should determine the best method of market approach. Should the group sell the complete design to an avionic manufacturer or just sell the right to production? Should it be promoted as an add on to an existing electronic instrument system or be part of the standard equipment on new aircraft?

## Conclusion

Instrument rated pilots, flying under IFR conditions, have a wealth of information available to them. The problem exists that this information is not getting utilized because there is too much data to be processed by the pilot and not enough time to process it. Also, the data presented by the electromechanical instruments is in some cases not informative or is presented in a format that is not immediately comprehensible.

The general-aviation community would benefit from a digital avionic instrument that would truly provide a man-machine interface by relating the flight information to the pilot in much the same way as the digital avionics in large commercial airliners.

In this thesis an Electronic Horizontal Situation Indicator was proposed. Through three simple display page graphics, the pilot is presented with information about his horizontal relationship with the surroundings, a centralized place for pertinent flight data and a unique presentation of the aircraft's position while on an ILS approach. The Data Page provides information about the characteristics of the flight and of the aircraft. The NAV Page

presents the aircraft's horizontal relationship to the surroundings in a plan presentation. The ILS Page provides the pilot with an effective indication of the aircrafts position and trend while on an instrument approach.

To develop such an instrument as the EHSI, it was necessary to design and build an EHSI Development System. The development system consists of an ATC610 Flight Simulator, a Zenith-158 "IBM" compatible computer, an HP-1345A Vector Graphics display, a Command Keyboard and a Data Acquisition and Communications Interface that ties it all together. The host computer processes the data acquired and produces workable display pages for the HP-1345A. The development system hardware is discussed in Chapter 3. A discussion of the development system software and the proposed host software is given in Chapter 4. A great deal of work remains to be completed to continue the development of the EHSI. Chapter 5 gives recommendations for future work to be considered as the EHSI matures. This work involves display development, ground testing, prototype design, flight testing and product market evaluation.



## REFERENCES

- [1] Technical Survey, "New Avionic Systems Offer Efficiency, Safety Benefits," Aviation Week & Space Technology, Vol. 116/16, April 19, 1982, p-52.
- [2] Lerner, J.L., "The Automated Cockpit," IEEE Spectrum, Feb. 1983, p-57.
- [3] "Cockpit Crew," Aviation Week & Space Technology, July 5, 1982, p-37.
- [4] Scott, W.B., "Avionics Firms Invest in Digital Designs," Aviation Week & Space Technology, Oct. 3, 1983, p-103.
- [5] "Digital Avionics Unaffected by Turndown," Aviation Week & Space Technology, Nov. 9, 1981, p-181.
- [6] Stein, J.S., "Navigation System Capabilities Increase," Aviation Week & Space Technology, Dec. 21, 1981, Vol. 115, p-70.
- [7] Dyer, S.A., "A Proposed Electronic Horizontal Situation Indicator for use in General-Aviation Aircraft," Proceedings of 1982 Position, Location and Navigation Symposium, pp. 198-205.
- [8] U.S. Navy Hydrographic Office, Air Navigation, Ch. 4 and Ch. 6.
- [9] U.S. Dept. of Transportation FAA, Instrument Flying Handbook, 1980.
- [10] Kershner, W.K., Instrument Flight Manual, Iowa State University Press, Second Edition, 1969.
- [11] "Computers in the Cockpit," Oshkosh Air Show Seminar, Oshkosh WI., 1985.
- [12] Hewlett-Packard Inc., HP-1345A Digital Display Module Designers Manual, 1981.

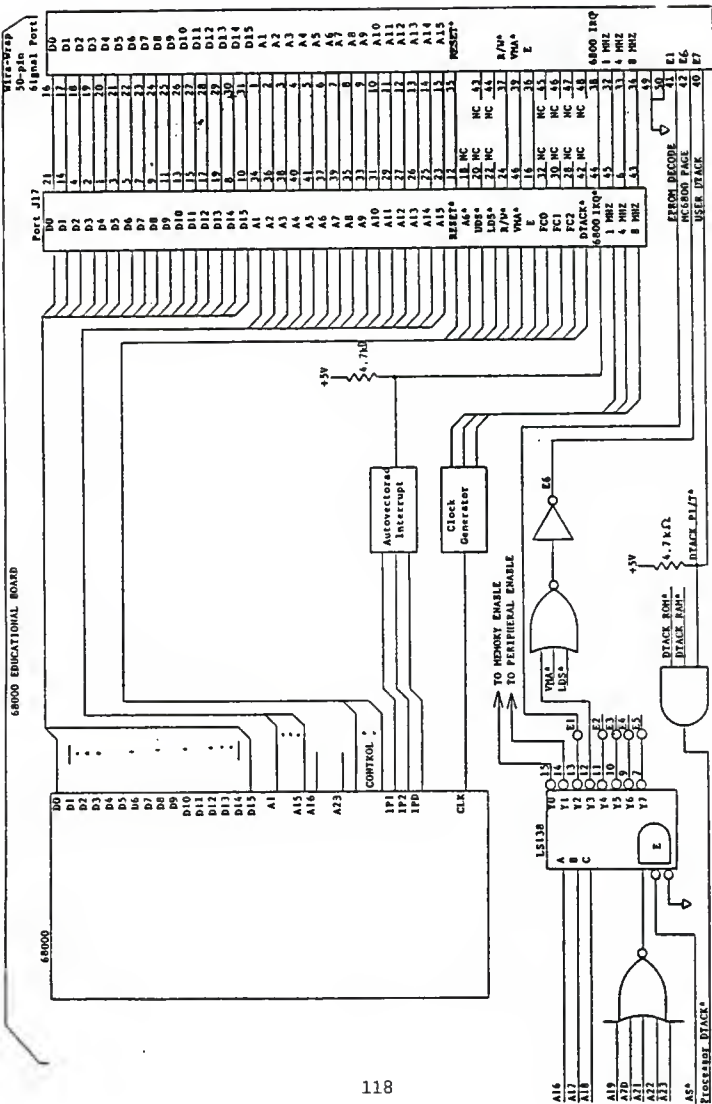
REFERENCES CONTINUED

- [13] ATC Division of Electronic Assoc. Inc., ATC 610 /710 Flight Simulator Service Manual.
- [14] Motorola Inc., MC68000 Educational Computer Board User's Manual, Second Edition, 1982.
- [15] Stern, M., "757-767 Cockpit," Radio Electronics, Part 1, Feb. 1983, p-39, Part 2, March 1983 p-43.
- [16] Stein, K.J., "Navy Evaluates Pictorial Cockpit Display," Aviation Week & Space Technology, Sept. 12, 1983, p-88.
- [17] Kocivar, B., "Revolution in the Cockpit: The New Jetliners," Popular Science, Nov. 1982, p-58.
- [18] Kershner, W.K., The Student Pilot's Flight Manual, Iowa State University Press, Third Edition, 1970.
- [19] Yannone, R.M., "Expert Systems in The Fighter of The 1990's," IEEE Aerospace and Electronic Systems Mag., February 1986.
- [20] Elson, B.M., "NASA Studies Business Aircraft Avionics," Aviation Week & Space Technology, April 26, 1982 p-119.
- [21] Julian, K., "Preventing Midair Collisions," High Technology, July 1985, p-48.
- [22] Scott, W.B., "New Electronic Instruments Displayed" Aviation Week & Space Technology, Sept. 28, 1981, p-77.
- [23] Donoghue, J.A., "Airbus A320 Cockpit: An Electric Experience," Air Transport World, March 1986, p-38.

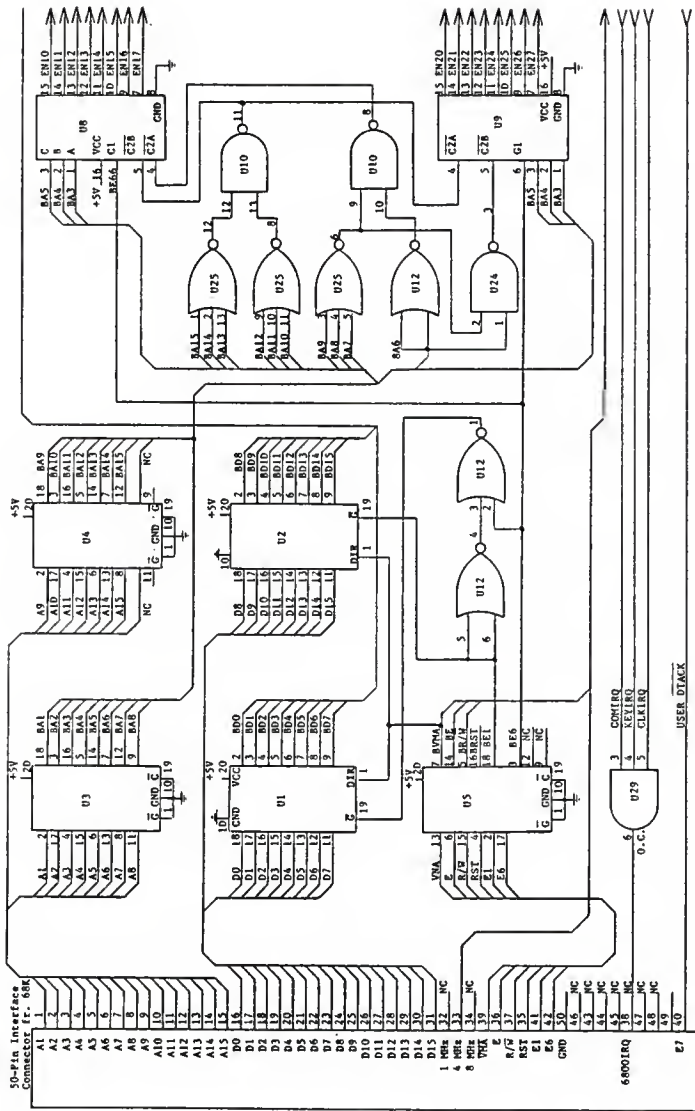
APPENDIX A  
INTERFACE BOARD DEVICE LIST

<u>DEVICE</u>	<u>DEVICE DESCRIPTION</u>	<u># PINS</u>
U1	74LS245	20
U2	74LS245	20
U3	74LS244	20
U4	74LS244	20
U5	74LS244	20
U6	74LS244	20
U7	MM58174 REAL TIME CLOCK	16
U8	74LS138	16
U9	74LS138	16
U10	74LS00	14
U11	74LS08	14
U12	74LS28	14
U13	74LS04	14
U14	74LS32	14
U15	MC6821 PIA #1 CONTROL	40
U16	MC6821 PIA #3 KEYBOARD	40
U17	MC6821 PIA #5 PARALLEL PORT	40
U18	MC6821 PIA #2 HP-DATA	40
U19	MC6821 PIA #4 BINARY INPUTS	40
U20	74LS245 HP-DATA BUS LSB	20
U21	74LS245 HP-DATA BUS MSB	20
U22	AD7581 ADC #1	28
U23	AD7581 ADC #2	28
U24	74LS00	14
U25	74LS27	14
U26	74LS175 DTACK DELAY	16
U27	SPARE SOCKET	16
U28	74LS11	14
U29	74LS15	14
U30	74LS08	14
U31	74LS32	14
U32	SPARE SOCKET	16
U33	SPARE SOCKET	16
U34	MCM68764 8K x 8 EPROM LSB	24
U35	MCM68764 8K x 8 EPROM MSB	24

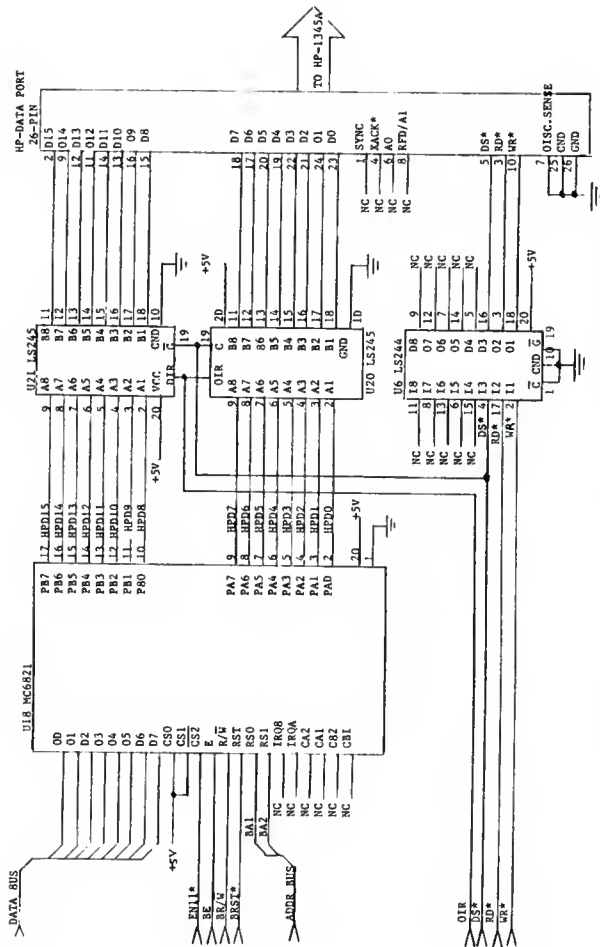
<u>Device</u>	<u>Device Description</u>	<u># Pins</u>
U36	LF353 DUAL OP AMP	8
U37	LF353 DUAL OP AMP	8
U38	LF353 DUAL OP AMP	8
U39	LF353 DUAL OP AMP	8
U40	LF353 DUAL OP AMP	8
U41	LF353 DUAL OP AMP	8
U42	LF353 DUAL OP AMP	8
U43	LF353 DUAL OP AMP	8
U44	LF347 QUAD OP AMP	14
U45	LF347 QUAD OP AMP	14
U46	LF347 QUAD OP AMP SPARE	14
U47	LF347 QUAD OP AMP SPARE	14
U48	LF353 DUAL OP AMP	8
U49	LF353 DUAL OP AMP SPARE	8
U50	74HC32 CMOS OR GATE	14
U51	SPARE SOCKET	14
U52	74LS245	20



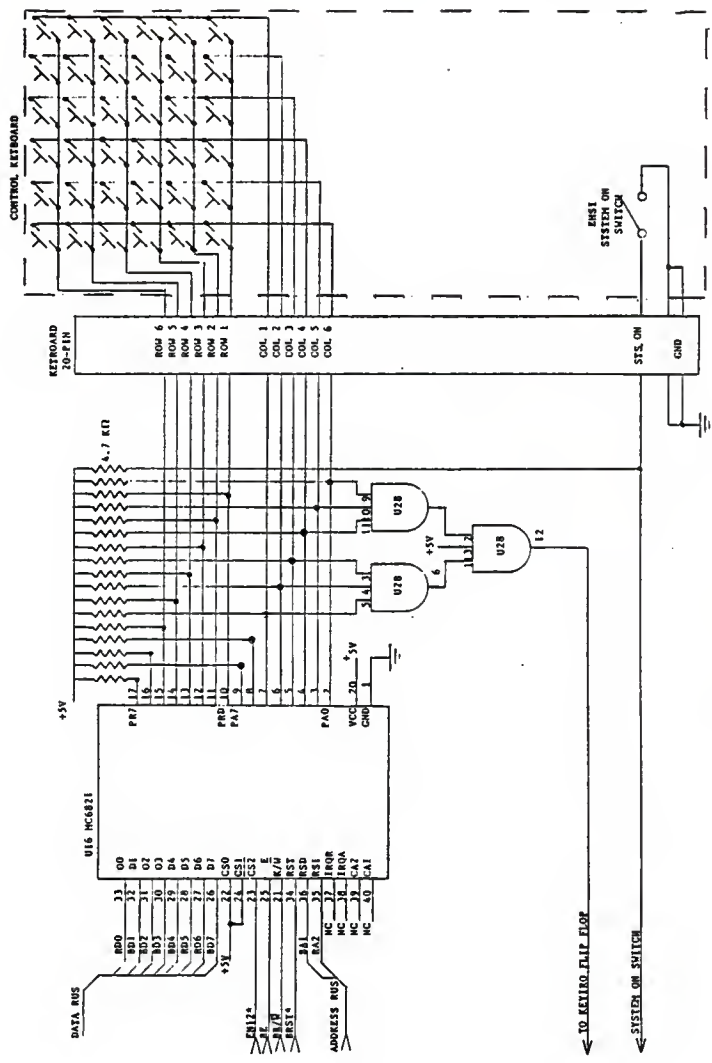
EDUCATIONAL BOARD BUS SIGNALS WITH MEMORY SEGMENT DECODER



INTERFACE BOARD DATA, ADDRESS AND CONTROL BUFFERS, DECODERS 1 & 2

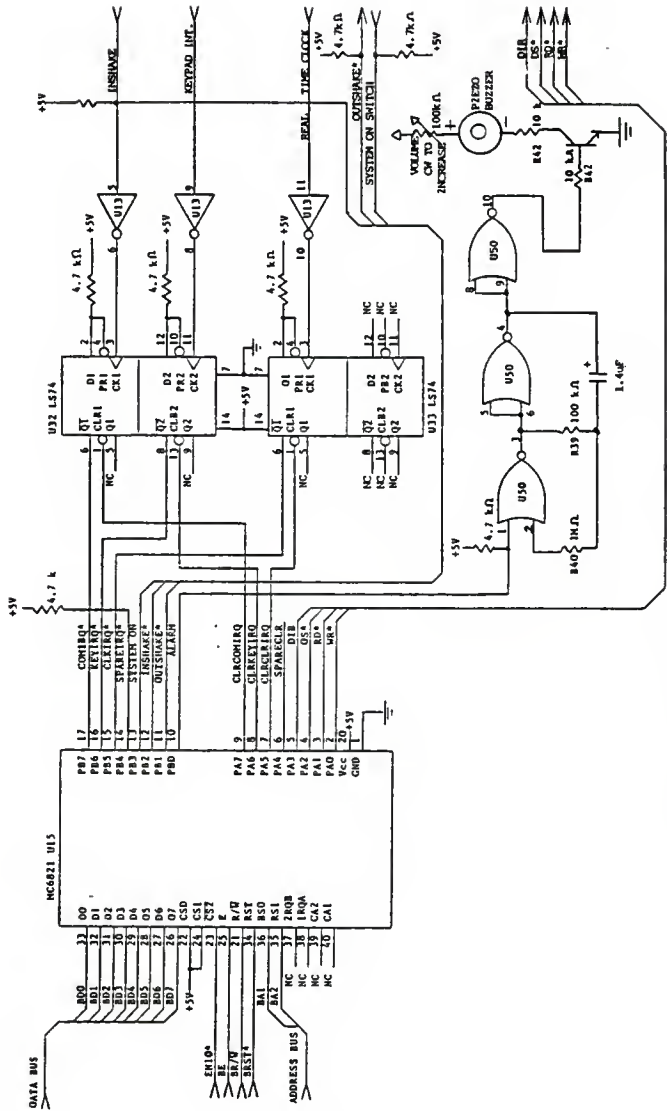


INTERFACE BOARD TO HP-1345A PARALLEL PORT WITH CONTROL LINES (PIA #2)

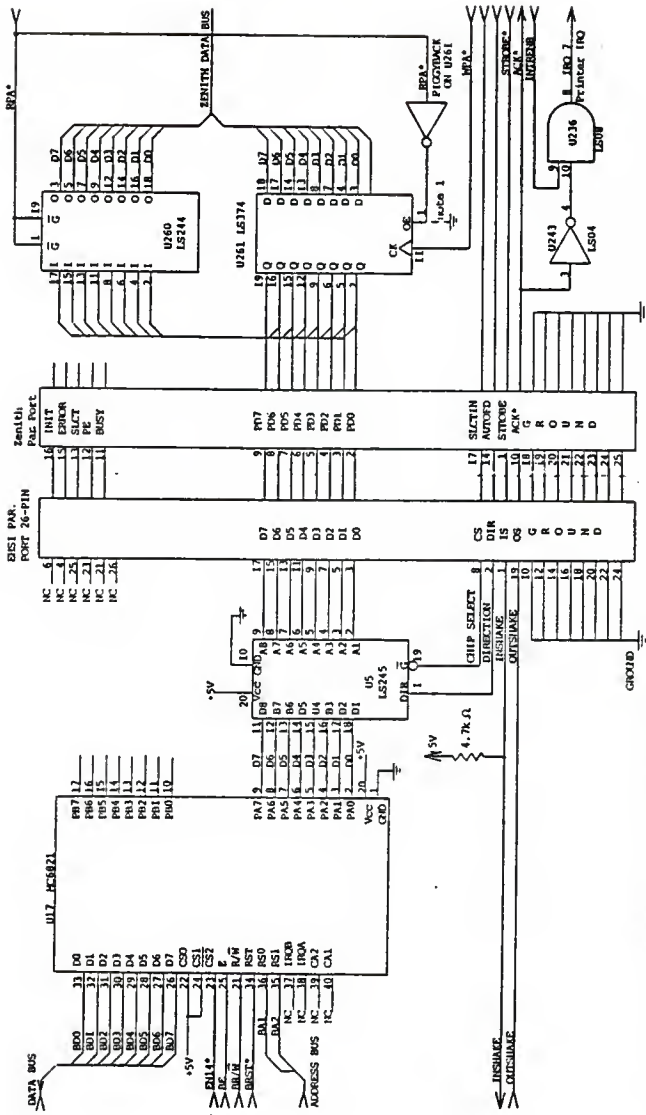


CONTROL KEYBOARD (PIA #3)



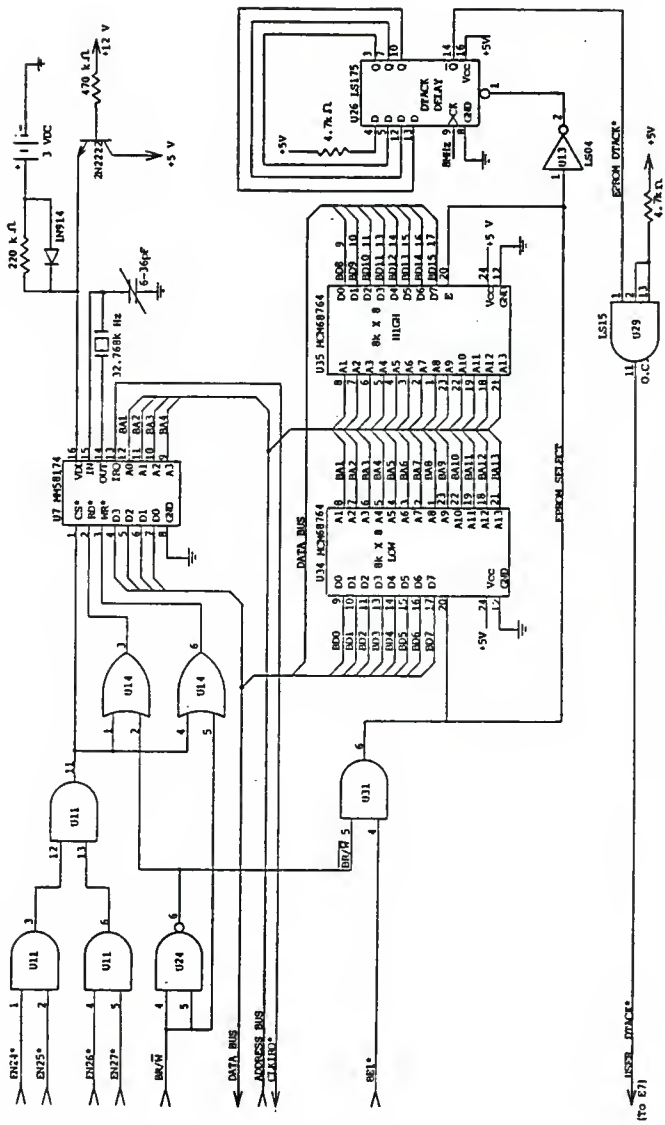


CONTROL PIA AND ALARM HORN (PIA #1)

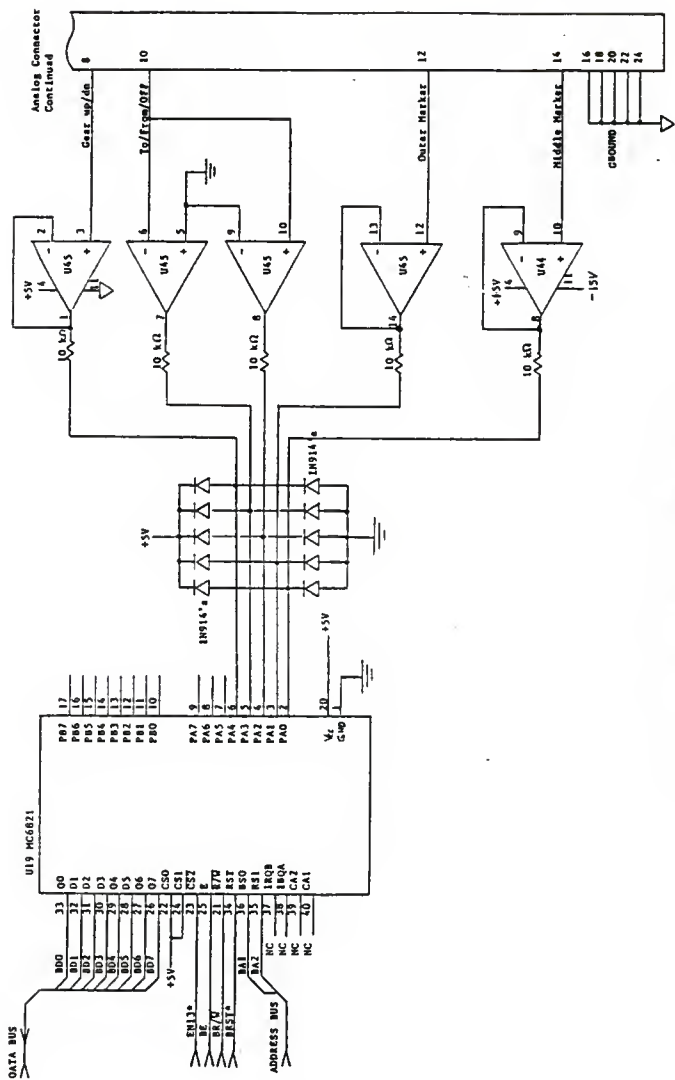


NOTE 1. PIN 11 OF U261 WAS GROUNDED. AN INVERTER WAS ADDED PICTORIAL CH U261.

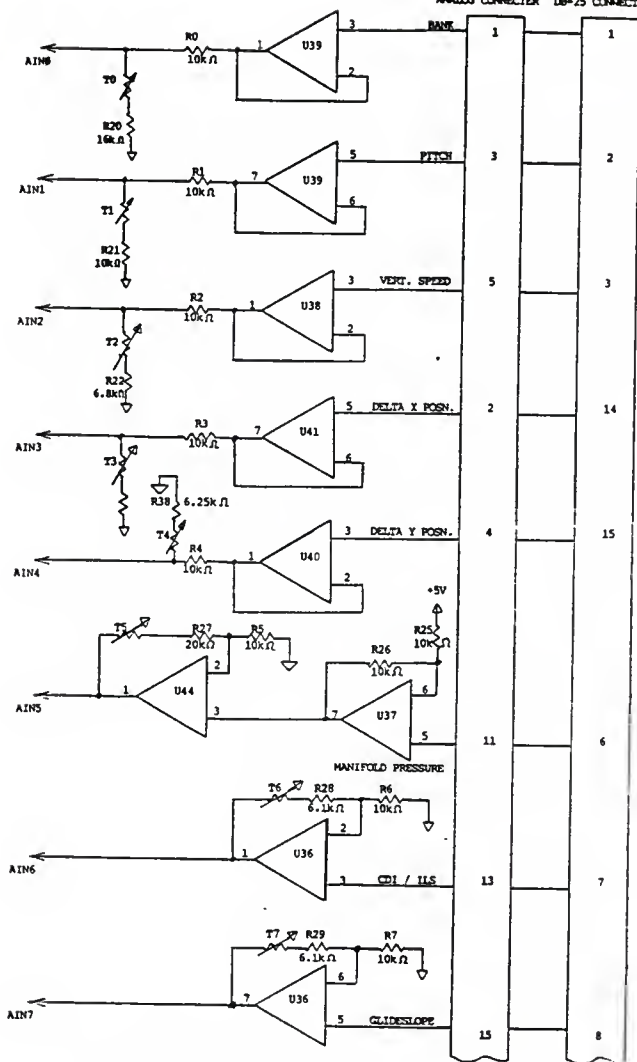
ZENITH-158 AND INTERFACE BOARD PARALLEL PORTS WITH HANDSHAKE LINES (PIA #5)



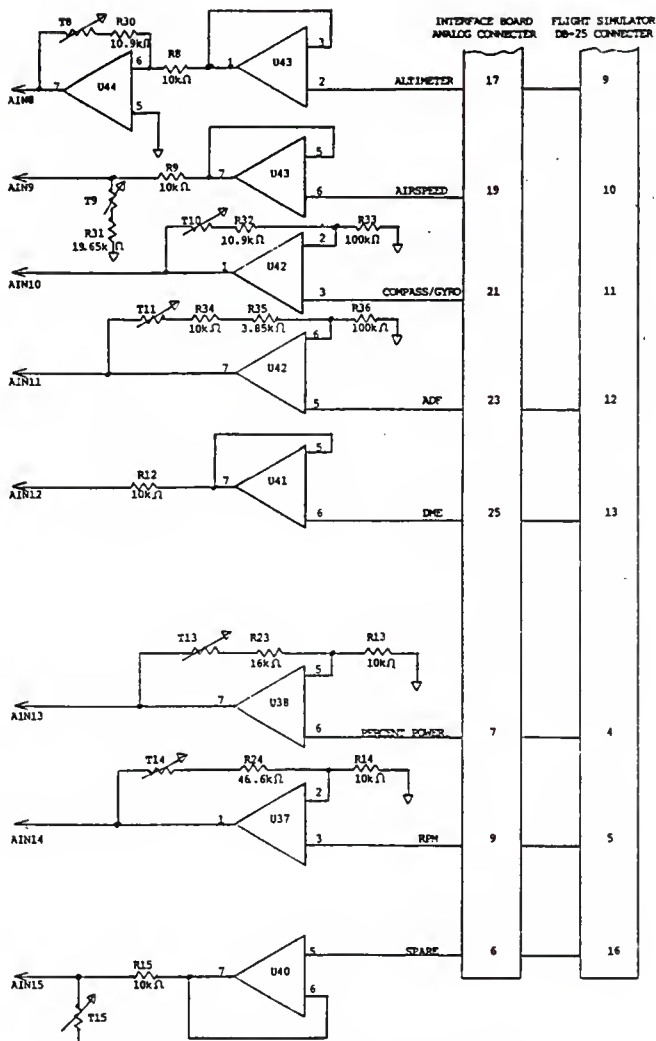
EPROM'S AND REAL TIME CLOCK



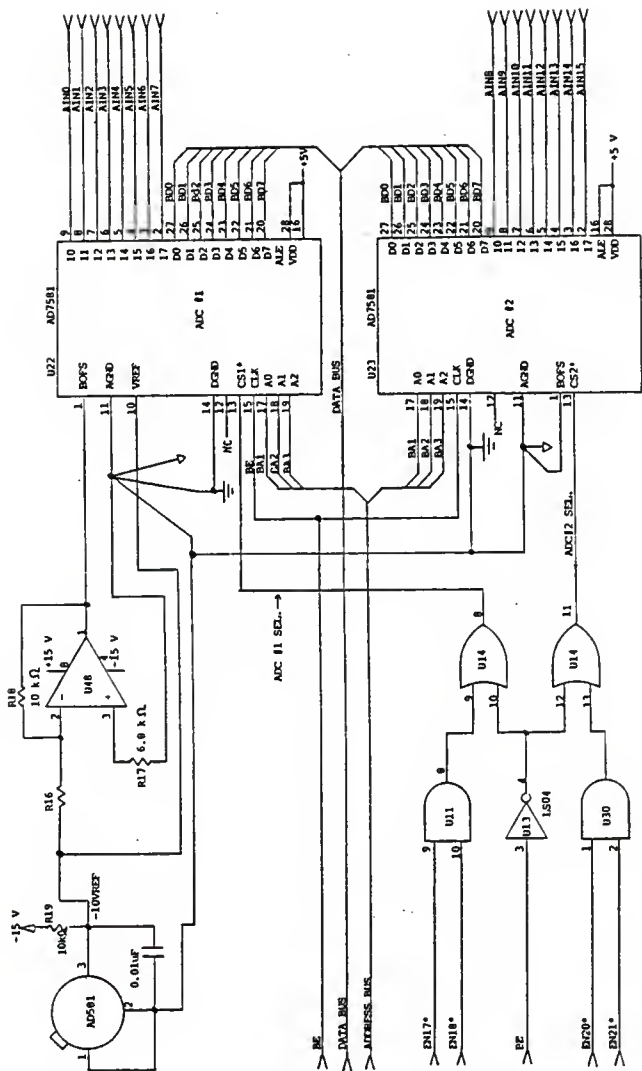
BINARY INPUTS WITH SIGNAL CONDITIONING



ANALOG INPUTS 0-7 SIGNAL CONDITIONING



ANALOG INPUTS 8-15 SIGNAL CONDITIONING



ANALOG TO DIGITAL CONVERTERS #1 AND #2

```

*****
*****
*
*                               MAIN PROGRAM
*
*                               EHSI DEVELOPMENT SYSTEM
*
*****
*****
*
*                               REVISION      DATE      PROGRAMMER
*                               1.45      AUGUST 28, 1986  JEFF LAGERBERG
*
*****

```

PURPOSE

To control the EHSI Development System by initializing the system and controlling the start-up and shutdown of the system. The main program also is responsible for updating the "Data Package" for data processing by the Host.

DESCRIPTION

The main program starts out by initializing the system and the system parameters. It initializes all Peripheral Interface Adapters (PIA), their outputs, and then initializes all interrupts. The Real Time Clock also gets initialized as well as the HP-1345A Vector Memory. The last initialization involves setting up the interrupt vectors for exception processing.

After all initializations are complete, the main program will display the message "INITIALIZATIONS COMPLETE" on the terminal and then instructs the pilot to "CONTINUE TO THE FLIGHT SIMULATOR." The main program then waits for the pilot to turn the EHSI System Switch on.

When the EHSI System Switch is turned on, the EHSI ready logo will appear on the HP-1345A and all interrupts will be enabled. The 68000 will then inform the Host that the system is up and running and to start normal data processing.

The main program now enters its very short main loop. Once in the main loop, the program goes about its task of continuously updating the "Data Package" and monitoring the EHSI System Switch. The main loop is the point that is interrupted when interrupt processing occurs.



\*\*\*\*\*

DESCRIPTION CONTINUED

The program constantly monitors the state of the EHSI System Switch. When the system is shutdown the main program disables all interrupts, displays a shutdown message and informs the Host of the shutdown. It then waits for the system switch to be turned back on. When it is turned back on, the system and the entire initialization process is restarted.

It is important to note that the main loop of the main program consists of just a few lines of code. Since the system is operating under an interrupt environment, most all of the work is done by the interrupt service routines.

\*\*\*\*\*

	ORG.	\$20900	
START	MOVE.L	#8,D3	SEND EIGHT LINE FEEDS
LF1	MOVE.B	#\$A,D0	TO THE TERMINAL
	MOVE.B	#248,D7	OUTPUT CHARACTER HANDLER
	TRAP	#14	
	DBRA	D3,LF1	
	MOVE.B	#\$D,D0	SEND CARRIAGE RETURN
	MOVE.B	#248,D7	OUTPUT CHARACTER HANDLER
	TRAP	#14	
	MOVE.L	#30,D3	SEND THIRTY SPACES
SP1	MOVE.B	#\$20,D0	TO THE TERMINAL
	MOVE.B	#248,D7	OUTPUT CHARACTER HANDLER
	TRAP	#14	
	DBRA	D3,SP1	
	MOVE.L	#\$39D4,A5	DISPLAY "EHSI SYSTEM RUNNING"
	MOVE.L	#\$39E8,A6	ON THE TERMINAL
	MOVE.B	#227,D7	OUTPUT STRING HANDLER
	TRAP	#14	
PIA#1PA	MOVE.L	#\$30001,A0	INIT PIA #1 PORT A OUTPUT
	MOVE.B	#\$FF,D0	
	JSR	INTPIA	
	MOVE.B	#\$0F,\$30001	INIT CONTROL BITS
PIA#1PB	MOVE.L	#\$30005,A0	INIT PIA #1 PORT B O/P I/P
	MOVE.B	#\$3,D0	
	JSR	INTPIA	
	MOVE.B	#3,\$30005	INIT CONTROL BITS

MAIN PROGRAM

\*\*\*\*\*  
 \*\*\*\*\*

MAIN PROGRAM CONTINUED

\*\*\*\*\*  
 \*\*\*\*\*

```

PIA#2PA  MOVE.L  #$30009,A0      INIT PIA #2 PORT A OUTPUT
          MOVE.B  #$FF,D0
          JSR     INTPIA

PIA#2PB  MOVE.L  #$3000D,A0      INIT PIA #2 PORT B OUTPUT
          JSR     INTPIA

PIA#3PA  MOVE.L  #$30011,A0      INIT PIA #3 PORT A INPUT
          MOVE.B  #0,D0
          JSR     INTPIA

PIA#3PB  MOVE.L  #$30015,A0      INIT PIA #3 PORT B OUTPUT
          MOVE.B  #$FF,D0
          JSR     INTPIA
          MOVE.B  #0,$30015      OUTPUT LOW ON ALL ROWS

PIA#4PA  MOVE.L  #$30019,A0      INIT PIA #4 PORT A INPUT
          MOVE.B  #0,D0
          JSR     INTPIA

PIA#5PA  MOVE.L  #$30021,A0      INIT PIA #5 PORT A OUTPUT
          MOVE.B  #$FF,D0
          JSR     INTPIA

CLK INIT MOVE.B  #0,$3007F      CLEARS INTERRUPT TIMER
          MOVE.B  $3007F,D0      THREE READS CLEARS
          MOVE.B  $3007F,D0      INTERRUPT OUTPUT LOGIC
          MOVE.B  $3007F,D0
          MOVE.B  #0,$30061      CLEAR CLOCK TEST MODE
          MOVE.B  #1,$3007D      MAKE SURE CLOCK IS RUNNING

INIT VCM JSR     INTVCM          INIT HP VECTOR MEMORY

CHK VCM  MOVE.B  #0,D3           CHECK VCM COMMUNICATIONS
RECHECK MOVE.W  #$COFF,D0        SET HP ADDR POINTER TO $FF
          JSR     SND CMD
          MOVE.W  #$7818,D0      SEND AN HP COMMAND
          JSR     SND CMD
          MOVE.L  #$FF,D0        SET UP TO READ BACK THE CMD
          MOVE.L  #$FF,D1        READ BACK ONLY ONE WORD CMD
          JSR     RDVCM          PUT CMD IN VCM MIRROR RAM
  
```

MAIN PROGRAM

\*\*\*\*\*  
\*\*\*\*\*

MAIN PROGRAM CONTINUED

\*\*\*\*\*  
\*\*\*\*\*

	CMPI.W	#\$7818,\$61FE	IS THE COMMAND THERE?
	BEQ	IRQ VTR	IF SAME COMMAND, CONTINUE
	CMPI.B	#0,D3	IS IT THE FIRST TIME
	BNE	RECHECK	IF NOT FIRST TIME RECHECK
	MOVE.B	#4,D2	SET ERROR FLAG TO VCMERR
	JSR	ERROR	SEND ERROR TO TERMINAL
	DBRA	D3,RECHECK	RECHECK COMMUNICATIONS W/ VCM
IRQ VTR	JSR	INTVCM	REINITIALIZE VECTOR MEMORY
	MOVE.L	#\$2500,\$70	PUT VECTOR IN \$70=\$00002500
	MOVE.W	#\$2300,SR	SET IRQ PRIORITY TO LEVEL 3
LF2	MOVE.L	#3,D3	SEND THREE LINE FEEDS
	MOVE.B	#\$A,D0	TO THE TERMINAL
	MOVE.B	#248,D7	OUTPUT CHARACTER HANDLER
	TRAP	#14	
	DBRA	D3,LF2	
SP2	MOVE.L	#28,D3	SEND 28 SPACES
	MOVE.L	#\$20,D0	TO THE TERMINAL
	MOVE.B	#248,D7	OUTPUT CHARACTER HANDLER
	TRAP	#14	
	DBRA	D3,SP2	
	MOVE.L	#\$39E8,A5	DISPLAY "INITIALIZATIONS
	MOVE.L	#\$3A00,A6	COMPLETE" TO THE TERMINAL
	MOVE.B	#227,D7	OUTPUT STRING HANDLER
	TRAP	#14	
LF3	MOVE.L	#3,D3	SEND THREE LINE FEEDS
	MOVE.L	#\$A,D0	TO THE TERMINAL
	MOVE.B	#248,D7	OUTPUT CHARACTER HANDLER
	TRAP	#14	
	DBRA	D3,LF3	
SP3	MOVE.L	#26,D3	SEND 26 SPACES
	MOVE.B	#\$20,D0	TO THE TERMINAL
	MOVE.B	#248,D7	OUTPUT CHARACTER HANDLER
	TRAP	#14	
	DBRA	D3,SP3	
	MOVE.L	#\$3A00,A5	DISPLAY "CONTINUE TO FLIGHT
	MOVE.L	#\$3A20,A6	SIMULATOR" TO THE TERMINAL
	MOVE.B	#227,D7	OUTPUT STRING HANDLER
	TRAP	#14	
LF4	MOVE.L	#6,D3	SEND SIX LINE FEEDS
	MOVE.L	#\$A,D0	TO THE TERMINAL
	MOVE.B	#248,D7	OUTPUT CHARACTER HANDLER
	TRAP	#14	
	DBRA	D3,LF4	

MAIN PROGRAM

\*\*\*\*\*  
 \*\*\*\*\*

MAIN PROGRAM CONTINUED

\*\*\*\*\*  
 \*\*\*\*\*

LOGO	CLR.L	D1	SET UP LOGO FOR DISPLAY
	MOVE.L	#\$3A90,A0	ON THE HP DISPLAY
	MOVE.W	(A0)+,D1	
	JSR	SNDLST	SEND THE LOGO TO VECTOR MEM.
	MOVE.W	#\$C000,D0	SET UP ADDRESS POINTER
SYSON	BTST	#3,\$30005	WAIT FOR SYSTEM ON SWITCH
	BEQ	SYSON	LOW OFF, HIGH ON
	JSR	SNDCMD	SET ADDRESS POINTER TO \$0000
	MOVE.W	#0,D0	SEND A NOP WHERE JUMP WAS SO
	JSR	SNDCMD	LOGO CAN BE DISPLAYED
SYSUP	MOVE.B	#\$65,\$30021	SEND SYSTEM UP VECTOR TO HOST
	JSR	OUTSHAKE	SEND STROBE TO HOST
	MOVE.B	#\$2,D2	SET ERROR FLAG TO ACKERR
	JSR	INSHAKE	WAIT FOR ACKNOWLEDGE
	MOVE.L	#\$30021,A0	MAKE PARALLEL PORT INPUT
	MOVE.B	#0,D0	
	JSR	INTPIA	
ENIRQS	OR.B	#\$F0,\$30001	ENABLE ALL INTERRUPT REQUESTS
ENCLKIRQ	MOVE.B	#\$0C,\$3007F	ENABLE INTERRUPT INTERVAL
MAIN	MOVE.B	#24,\$1FA0	PUT NUM OF ENTRIES IN DATA PAK
	JSR	ANALOG	PUT ANALOG VALUES IN DATA PAK
	MOVE.B	\$30019,\$1FB1	PUT BINARY INPUTS IN DATA PAK
	BTST	#3,\$30005	CHECK TO SEE IF SYSTEM IS ON
	BNE	MAIN	IF ON, GO THRU MAIN AGAIN
SYSDN	AND.B	#\$0F,\$30001	DISABLES INTERRUPTS
	MOVE.W	#\$2700,SR	DISABLE ALL INTERRUPT LEVELS
	BSET	#0,\$30005	TURN ALARM OFF IF ON
SHUTDN	MOVE.L	#\$3AA4,A0	DISPLAY SYSTEM SHUTDOWN MSG
	CLR.L	D1	
	MOVE.W	(A0)+,D1	
	JSR	SNDLST	
	MOVE.L	#\$30021,A0	MAKE PARALLEL PORT OUTPUT
	MOVE.B	#\$FF,D0	
	JSR	INTPIA	
	MOVE.B	#\$66,(A0)	SEND SHUT DN VECTOR TO HOST
	JSR	OUTSHAKE	STROBE THE HOST
	MOVE.B	#2,D2	SET ERROR FLAG TO ACKERR
	JSR	INSHAKE	WAIT FOR ACKNOWLEDGE

MAIN PROGRAM

\*\*\*\*\*  
\*\*\*\*\*

MAIN PROGRAM CONINTUED

\*\*\*\*\*  
\*\*\*\*\*

WAITON	BTST	#3,\$30005	WAIT FOR SYSTEM TO BE
	BEQ	WAITON	TURNED BACK ON
	MOVE.L	#4000,D0	DEBOUNCE SYSTEM ON SWITCH
SELF	DBRA	D0,SELF	
	BTST	#3,\$30005	SYSTEM STILL ON ?
	BEQ	WAITON	
	BRA	START	RESTART SYSTEM

MAIN PROGRAM

\*\*\*\*\*

SUBROUTINE INTPIA

PURPOSE: To initialize the direction of a port of a PIA

INITIAL CONDITIONS: The address of the port data register to be initialized is passed in register A0.L. The port direction byte is passed in register D0.B when the subroutine is called. If a bit in D0.B is a one, then the direction of that bit will be output. If the bit is a zero, then it will be an input bit.

ACTION: The port data direction register and port control registers are initialized according to the direction specified.

REGISTER USAGE: No register are affected. A0.L and D0.B are returned undisturbed.

\*\*\*\*\*

	ORG.	\$20000	
START	MOVE.L	A1,\$900	SAVE A1 REGISTER
	MOVE.L	A0,A1	COPY A0 TO A1. A0=PORT DATA.
	ADDA.L	#2,A1	A1=PORT CONTROL REG.
	CLR.B	(A1)	CLEAR PORT CONTROL REG.
	MOVE.B	D0,(A0)	SET DIRECTION OF PORT DATA
	MOVE.B	#4,(A1)	SET BIT #2 OF PORT CONTROL
	MOVEA.L	\$900,A1	RESTORE A1
END	RTS		RETURN TO CALLING ROUTINE

SUBROUTINE INTPIA

\*\*\*\*\*

### SUBROUTINE INTVCM

PURPOSE: To initialize the HP-1345A Vector Memory

INITIAL CONDITIONS: This subroutine requires no parameters to be passed to it.

ACTION: Vector Memory is initialized by placing an "Internal Jump to 4095" command in all memory locations except locaion 4095 which will hold a NOP. This initialization process is performed because synchronous refresh of the HP-1345A's display can only occur after an internal jump command has been executed. There must be a NOP in location 4095 because internal jumps are not allowed to branch to another jump command.

REGISTER USAGE: No registers are affected.

\*\*\*\*\*

	ORG.	\$2001A	
START	MOVEM.L	D0-D1/A0-A2,\$904	SAVE REGISTERS
	MOVE.L	#\$30001,A0	PIA #1 PORT A DATA (CONTROL)
	MOVE.L	#\$30009,A1	PIA #2 PORT A DATA (LSB OF HP)
	MOVE.L	#\$3000D,A2	PIA #2 PORT B DATA (MSB OF HP)
	MOVE.B	(A0),D0	GET THE CONTROL BITS
	ORI.B	#\$0B,D0	SET UP HP DEVICE SELECT
	ANDI.B	#\$FB,D0	SET UP DIRECTION OF BUFFERS
	MOVE.B	D0,(A0)	
	CLR.B	(A1)	SET ADDR. PTR. TO \$0000 OF VCM
	MOVE.B	#\$C0,(A2)	
	ANDI.B	#\$FE,D0	TAKE HP WRITE LINE LOW
	MOVE.B	D0,(A0)	(STROBE WR LINE)
	ORI.B	#\$01,D0	TAKE HP WRITE LINE HIGH
	MOVE.B	D0,(A0)	
	MOVE.L	#4094,DI	INIT. VECTOR MEMORY LOOP CTR.
	MOVE.B	#\$FF,(A1)	SEND \$FF TO HP LSB PORT
	MOVE.B	#\$8F,(A2)	SEND \$8F TO HP MSB PORT
LOOP	ANDI.B	#\$FE,D0	TAKE HP WRITE LINE LOW
	MOVE.B	D0,(A0)	
	ORI.B	#01,D0	TAKE HP WRITE LINE HIGH
	MOVE.B	D0,(A0)	
	DBRA	D1,LOOP	POINT TO NEXT VCM. LOCATION
	CLR.B	(A1)	SEND \$00 TO HP LSB PORT (NOP)
	CLR.B	(A2)	SEND \$00 TO HP MSB PORT

SUBROUTINE INTVCM

\*\*\*\*\*

SUBROUTINE INTVCM CONTINUED

\*\*\*\*\*

```

                ANDI.B   #$FE,DO           TAKE HP WRITE LINE LOW
                MOVE.B   DO,(A0)
                ORI.B    #05,DO           TAKE HP WRITE LINE HIGH
                MOVE.B   DO,(A0)
RESTORE        MOVEM.L  $904,DO-D1/A0-A2  RESTORE REGISTERS
END            RTS                RETURN TO CALLING ROUTINE
```

SUBROUTINE INTVCM



\*\*\*\*\*

SUBROUTINE SNDCMD

PURPOSE: To send a HP-1345A command to vector memory

INITIAL CONDITIONS: The 16-bit HP-1345A command is passed to the subroutine in register DO.W.

ACTION: The command is sent to the location pointed to by the current vector memory address pointer or if the command is an address pointer command, then it will be sent to the vector memory address pointer.

REGISTER USAGE: No registers are affected.

\*\*\*\*\*

```
ORG.          $20086
START  MOVEM.L  DO-D1/A0-A2,$914   SAVE REGISTERS
      MOVEA.L  #$30001,A0          PIA #1 PORT A DATA (CONTROL)
      MOVEA.L  #$30009,A1          PIA #2 PORT A DATA (LSB OF HP)
      MOVEA.L  #$3000D,A2          PIA #2 PORT B DATA (MSB OF HP)
      MOVE.B   (A0),D1             GET HP CONTROL BITS
      ORI.B    #$0B,D1             SET DIRECTION
      ANDI.B   #$FB,D1             SET HP DEVICE SELECT
      MOVE.B   D1,(A0)             PUT BACK TO HP CONTROL PORT
      MOVE.B   DO,(A1)             SEND LSB OF COMMAND TO HP PORT
      LSR.W    #8,DO               MOVE MSB TO LSB
      MOVE.B   DO,(A2)             SEND MSB OF COMMAND TO HP PORT
      ANDI.B   #$FE,D1             TAKE HP WRITE LINE LOW
      MOVE.B   D1,(A0)
      ORI.B    #$05,D1             TAKE HP WRITE & SELECT LINE HI
      MOVE.B   D1,(A0)
      MOVEM.L  $914,DO-D1/A0-A2   RESTORE REGISTERS
END    RTS                          RETURN TO CALLING ROUTINE
```

SUBROUTINE SNDCMD

\*\*\*\*\*

SUBROUTINE ANALOG

PURPOSE: To perform analog to digital conversion of the sixteen analog inputs and send the digital values to the "Data package"

INITIAL CONDITIONS: This subroutine requires no parameters to be passed to it.

ACTION: The sixteen digital values, corresponding to various flight instruments and controls are passed to the "Data Package." The data package is located at \$1FA0 to \$1FB9. The sixteen digital values corresponding to the analog inputs are located at \$1FA1 to \$1FB0 within the data package.

REGISTER USAGE: No registers are affected.

\*\*\*\*\*

	ORG.	\$200C4	
START	MOVEM.L	D0/A0-A1,\$928	SAVE REGISTERS
	MOVE.L	#\$1FA1,A0	SET UP THE DATA PACKAGE PTR.
	MOVE.L	#15,D0	SET LOOP CTR.
	MOVEA.L	#\$30031,A1	SET ADC PTR.
AGAIN	MOVE.B	(A1),(A0)+	PUT VALUE IN DATA PACKAGE
	ADDA.L	#2,A1	POINT TO NEXT ANALOG ADDRESS
	DBRA	D0,AGAIN	GET NEXT VALUE
	MOVEM.L	\$928,D0/A0-A1	RESTORE REGISTERS
END	RTS		RETURN TO CALLING ROUTINE

SUBROUTINE ANALOG

\*\*\*\*\*

### SUBROUTINE GETKEY

PURPOSE: To determine what key has been pressed on the command keyboard

INITIAL CONDITIONS: This subroutine requires no parameters to be passed to it.

ACTION: GETKEY determines the number of the key pressed in the six-by-six command keyboard matrix. There are thirty-six keys on the keyboard. The value of the key pressed is returned in register D2.B and in the data package at location \$1FB2. A value of thirty-seven is returned if no key was found pressed by GETKEY.

REGISTER USAGE: Register D2.B is used to hold the key value on return. No other registers are affected.

\*\*\*\*\*

	ORG.	\$200F0	
START	MOVEM.L	D0-D1/A0-A1,\$934	SAVE REGISTERS
	MOVE.L	#\$30011,A0	KEYBOARD COLUMNS
	MOVE.L	#\$30015,A1	KEYBOARD ROWS
	MOVE.L	#\$200,D2	SETUP KEY DELAY
COLUMNS	MOVE.B	(A0),D1	READ COLUMNS
	CMP.B	#\$FF,D1	ANY KEY PRESSED
	BNE	CONTINUE	IF KEY IS PRESSED CONTINUE
	DBRA.L	D2,COLUMNS	
	MOVE.B	#37,D2	RETURN NOKEY VALUE
	JMP	RETURN	
CONTINUE	MOVE.B	#\$FF,D0	MAKE COLUMNS PORT OUTPUT
	JSR	INTPIA	
	MOVE.B	#0,D0	MAKE ROWS PORT INPUT
	MOVE.L	A1,A0	
	JSR	INTPIA	
	MOVE.L	#\$30011,A0	KEYBOARD COLUMNS PORT
	MOVE.B	D0,(A0)	OUTPUT ZEROS ON COLUMNS
	MOVE.L	#\$200,D2	SET UP WAIT FOR KEY DELAY
ROWS	MOVE.B	(A1),D0	READ THE ROWS
	CMP.B	#\$FF,D0	IS A KEY PRESSED
	BNE	GOON	IF A KEY IS PRESSED, GO ON
	DBRA	D2,ROWS	
	MOVE.B	#37,D2	RETURN NOKEY VALUE
	JMP	RETURN	
GOON	MOVE.W	#-1,D2	INITIALIZE KEY COUNT

SUBROUTINE GETKEY

\*\*\*\*\*

SUBROUTINE GETKEY CONTINUED

\*\*\*\*\*

FINDROW	ADDQ.W	#1,D2	FIND WHICH ROW
	LSR.B	#1,D0	CHECK NEXT ROW
	BCS	FINDROW	
	MULU.W	#6,D2	SIX KEYS PER ROW
FINDCOL	ADDQ.W	#1,D2	FIND WHICH COLUMN
	LSR.B	#1,D1	CHECK NEXT COLUMN
	BCS	FINDCOL	
RETURN	MOVE.B	D2,\$1FB2	RETURN KEY IN DATA PACKAGE
	MOVE.B	#0,D0	CHANGE COLS PORT TO INPUT
	JSR	INTPIA	
	MOVE.B	#\$FF,D0	CHANGE ROWS PORT TO OUTPUT
	MOVE.L	A1,A0	
	JSR	INTPIA	
	MOVE.L	#\$30011,A0	
WAIT	MOVE.B	#0,(A1)	OUTPUT ZEROS ON THE ROWS
	CMP.B	#\$FF,(A0)	IS KEY STILL PRESSED
	BNE	WAIT	
	MOVE.L	#\$1000,D0	WAIT FOR KEY TO SETTLE
SELF	DBRA	D0,SELF	
	CMP.B	#\$FF,(A0)	MAKE SURE KEY IS RELEASED
	BNE	WAIT	
RESTORE	MOVEM.L	\$934,D0-D1/A0-A1	RESTORE REGISTERS
END	RTS		RETURN TO CALLING ROUTINE

SUBROUTINE GETKEY

\*\*\*\*\*

### SUBROUTINE RDVCM

PURPOSE: To read a block of HP-1345A commands from Vector Memory

INITIAL CONDITIONS: The starting address of the block to be read from vector memory is passed in register D0.L and the ending address of the block is passed in register D1.L.

ACTION: The block of HP-1345A commands are read from vector memory and moved to the vector memory mirror RAM located at \$6000 to \$7FFF. The block read from vector memory will reside at the same location in the vector memory mirror as it did in vector memory except it will be offset by \$6000.

REGISTER USAGE: Registers D0.L and D1.L are destroyed. No other registers are affected.

\*\*\*\*\*

START	ORG.	\$201C6	
	MOVEM.L	D2-D3/A0-A2, \$944	SAVE REGISTERS
	CLR.L	D2	PREPARE D2 FOR START ADDR COPY
	MOVE.W	D0, D2	COPY STARTING ADDR INTO D2
	MOVE.W	D0, D3	COPY STARTING ADDR INTO D3
	MOVE.L	#\$30001, A1	CONTROL DATA REG. PTR.
	ORI.W	#\$C000, D0	FORMAT ADDR. PTR. COMMAND
	JSR	SND CMD	SEND THE COMMAND
	LSL.L	#1, D2	MULTIPLY START ADDR BY TWO
	ADDI.L	#\$6000, D2	ADD VCM MIRROR BASE ADDR.
	MOVE.L	D2, A2	A2 HOLDS VCM MIRROR BASE ADDR.
	SUB.W	D3, D1	D1 HOLDS NUMBER OF VCM READS
	MOVE.B	(A1), D0	
	ORI.B	#3, D0	SET CONTROL FOR DIR, DS, WR, RD
	ANDI.B	#\$F3, D0	
	MOVE.B	D0, (A1)	
	MOVE.B	#0, D0	
	MOVE.L	#\$30009, A0	CHANGE HP-DATA PORT LSB TO I/P
	JSR	INTPIA	
	MOVE.L	#\$3000D, A0	CHANGE HP-DATA PORT MSB TO I/P
	JSR	INTPIA	
READ	MOVE.B	(A1), D0	GET CONTROL BYTE
	ANDI.B	#\$FD, D0	BRING RD LOW
	MOVE.B	D0, (A1)	
	MOVE.B	\$3000D, (A2)+	READ MSBYTE INTO VCM MIRROR
	MOVE.B	\$30009, (A2)+	READ LSBYTE INTO VCM MIRROR

SUBROUTINE RDVCM

\*\*\*\*\*

SUBROUTINE RDVCM CONTINUED

\*\*\*\*\*

```

        ORI.B   #2,D0           BRING RD HIGH
        MOVE.B  D0,(A1)
        DBRA    D1,READ
        MOVE.B  #$FF,D0
        MOVE.L  #$30009,A0      CHANGE HP-DATA PORT LSB TO O/P
        JSR     INTPIA
        MOVE.L  #$3000D,A0      CHANGE HP-DATA PORT MSB TO O/P
        JSR     INTPIA
        MOVE.B  (A1),D0
        ORI.B   #$F,D0         SET CONTROL
        MOVE.B  D0,(A1)
RESTORE  MOVEM.L $944,D2-D3/A0-A2  RESTORE REGISTERS
END      RTS                   RETURN TO CALLING ROUTINE
```

SUBROUTINE RDVCM

\*\*\*\*\*

SUBROUTINE SNDLST

PURPOSE: To send a list of HP-1345A commands to vector memory

INITIAL CONDITIONS: The starting address of the list to be sent is passed to the subroutine in register A0.L. The number of 16-bit HP-1345A commands in the list is passed in register D1.L.

ACTION: The list of commands are sent to vector memory starting at the current location of the vector memory address pointer. The list may contain a vector memory address pointer command to position the pointer for the list that follows.

REGISTER USAGE: Register A0.L will point to the byte after the last command in the list. No other registers are affected.

\*\*\*\*\*

START	ORG.	\$20256	
	MOVE.L	D0,\$958	SAVE D0
	DBRA	D1,CONTINUE	DECREMENT D1 AND CONTINUE
CONTINUE	MOVE.W	(A0)+,D0	GET COMMAND FROM LIST
	JSR	SND CMD	SEND COMMAND TO VECTOR MEM.
	DBRA	D1,CONTINUE	ARE ALL COMMANDS IN LIST SENT?
	MOVE.L	\$958,D0	RESTORE D0
END	RTS		RETURN TO CALLING ROUTINE

SUBROUTINE SNDLST

\*\*\*\*\*

### SUBROUTINE SETCLK

**PURPOSE:** To set the Real Time Clock/Calendar

**INITIAL CONDITIONS:** This subroutine requires no parameters to be passed. It may be executed by pressing the Set Clock key on the command keyboard at any time after the EHSI system is on. All time values are input by the user.

**ACTION:** The subroutine saves the current HP-1345A display and then takes control of the display to prompt the user for various time inputs.

When all time data has been input, the subroutine allows the user to start the clock at the instant the data is valid. The HP-1345A display is then restored to its setting prior to setting the clock.

**REGISTER USAGE:** No registers are affected.

\*\*\*\*\*

	ORG.	\$2026E	
START	MOVEM.L	DO-D3/A0-A2,\$95C	SAVE REGISTERS
	AND.B	#\$0F,\$30001	DISABLE INTERRUPTS
	CLR.L	DO	READ VEC MEM STARTING AT ZERO
	MOVE.L	#100,D1	READ 100 COMMANDS IN VEC. MEM.
	JSR	RDVCM	
	MOVE.L	#\$980,A1	POINTER FOR CLOCK CAL. DATA
	MOVE.L	#\$3824,A0	START ADDR. FOR PROMPT LISTS
	MOVE.L	#5,D0	FIVE + ONE CLOCK INPUTS
	MOVE.L	#\$37FF,A2	START ADDR OF KEY LOOKUP TABLE
NEXT	CLR.L	D1	
	MOVE.W	(A0)+,D1	GET NUMBER OF CHR. IN LIST
	JSR	SNDLST	SENT THE LIST TO VCM.
	MOVE.L	#1,D3	TWO DIGITS PER ENTRY
WAIT1	CMPI.B	#\$FF,\$30011	WAIT FOR A KEYSTROKE
	BEQ	WAIT1	
AGAIN	JSR	GETKEY	WHAT KEY WAS PRESSED
	CMPI.B	#25,D2	CHECK FOR VALID ROW
	BPL	INVALID	
	CMPI.B	#\$FA,\$30011	CHECK FOR VALID COLUMN
	BMI	INVALID	

SUBROUTINE SETCLK



\*\*\*\*\*

SUBROUTINE SETCLK CONTINUED

\*\*\*\*\*

	MOVE.B	0(A2,D2),D2	GET KEY FROM LOOK-UP TABLE
	CMPI.B	#\$A,D2	CHECK FOR PERIOD OR DIVIDE
	BPL	INVALID	
	MOVE.B	D2,(A1)+	HOLD CLOCK DATA
	DBRA	D3,WAIT1	GET SECOND OF TWO DIGIT ENTRY
	DBRA	D0,NEXT	NEXT CLOCK CALENDAR ENTRY
	BRA	CONTINUE	ENTRY COMPLETE. CONTINUE
INVALID	MOVE.L	A0,\$978	SAVE CURRENT LIST PTR.
	MOVE.L	D0,\$97C	SAVE LOOP PARAMETER CTR.
	MOVE.W	#\$C046,D0	COXX IS SET ADDR. TO FIRST NOP
	JSR	SNDCMD	
	MOVE.L	#\$399C,A0	START ADDR OF "INVALID KEY"
	CLR.L	D1	
	MOVE.W	(A0)+,D1	DISPLAY INVALID KEY, TRY AGAIN
	JSR	SNDLST	
WAIT2	CMPI.B	#\$FF,\$30011	WAIT FOR ANOTHER KEYSTROKE
	BEQ	WAIT2	
	MOVE.W	#\$C046,D0	POINT TO PLACE WHERE 8FFF WAS
	JSR	SNDCMD	SEND THE ADDR PTR COMMAND
	MOVE.W	#\$8FFF,D0	PUT 8FFF BACK IN THAT POSITION
	JSR	SNDCMD	SEND THE INTERNAL JUMP COMMAND
	MOVE.L	\$978,A0	RESTORE A0
	MOVE.L	\$97C,D0	RESTORE D0
	BRA	AGAIN	END OF INVALID KEY ROUTINE
CONTINUE	CLR.L	D1	
	MOVE.W	(A0)+,D1	DISPLAY THE LIST
	JSR	SNDLST	"PRESS ANY KEY TO START CLOCK"
WAIT3	CMPI.B	#\$FF,\$30011	WAIT FOR ANY KEY
	BEQ	WAIT3	
	MOVE.B	\$983,\$984	PREPARE TIME REGISTERS
	MOVE.B	\$982,\$983	PREPARE TIME REGISTERS
	MOVE.B	\$981,\$982	PREPARE TIME REGISTERS
	MOVE.L	#9,D1	NINE REGISTERS TO UPDATE
	MOVE.L	#\$982,A1	START ADDR OF TIME VALUES
	MOVE.L	#\$3007B,A2	START ADDR OF TIME REG'S
	MOVE.B	#0,\$3007F	CLEAR INTERRUPT TIMER CHAIN
	MOVE.B	\$3007F,D0	THREE READS CLEARS INTERRUPT
	MOVE.B	\$3007F,D0	OUTPUT LOGIC
	MOVE.B	\$3007F,D0	
	MOVE.B	#0,\$30061	CLEAR THE TEST MODE
	MOVE.B	#0,\$3007D	STOP THE CLOCK
TIME	MOVE.B	(A1)+,(A2)	UPDATE TIME REGISTER
	SUBA.L	#2,A2	POINT TO NEXT TIME REGISTER
	DBRA	D1,TIME	

SUBROUTINE SETCLK

\*\*\*\*\*

SUBROUTINE SETCLK CONTINUED

\*\*\*\*\*

	MOVE.B	#1,\$3007D	START CLOCK
	CLR.L	D1	DISPLAY "CLOCK/CALENDAR IS SET
	MOVE.W	(A0)+,D1	PRESS ANY KEY TO CONTINUE"
	JSR	SNDLST	SEND THE ABOVE PROMPT TO HP
WAIT4	CMPI.B	#\$FF,\$30011	WAIT FOR KEY RELEASE
	BNE	WAIT4	
	MOVE.L	#\$0FFFFFF,D0	DEBOUNCE DELAY
SELF	DBRA	D0,SELF	
WAIT5	CMPI.B	#\$FF,\$30011	WAIT FOR CONTINUE KEY
	BEQ	WAIT5	
	MOVE.W	#\$C000,D0	RESET VCM ADDR. PTR. TO \$0000
	JSR	SNDCMD	SEND THE ADDR. PTR. COMMAND
	MOVE.L	#\$6000,A0	RESTORE VCM TO PRIOR DISPLAY
	MOVE.L	#100,D1	100 COMMANDS WERE FIRST READ
	JSR	SNDLST	RESTORE VECTOR MEMORY
	ORI.B	#\$F0,\$30001	ENABLE INTERRUPT REQUESTS
RESTORE	MOVE.B	#\$0C,\$3007F	SET INTERRUPT TIMER TO 60 SEC.
END	MOVEM.L	\$\$\$95C,D0-D3/A0-A2	RESTORE REGISTERS
	RTS		RETURN TO CALLING ROUTINE

SUBROUTINE SETCLK

\*\*\*\*\*

SUBROUTINE OUTSHAKE

PURPOSE: To send out a pulse on the output handshake line of the parallel port

INITIAL CONDITIONS: This subroutine requires no parameter to be passed to it.

ACTION: The pulse on the output handshake line has a pulse width of approximately 25 microseconds. The actual pulsewidth can be varied according to the value in D0.L. The expression for the pulsewidth is

$$3 * (D0 + 1) + 4 \text{ microseconds.}$$

The pulse represents a Data Acknowledge when the Host is sending data to the 68000 interface and represents a Data Strobe when the 68000 is sending data to the Host.

REGISTER USAGE: No registers are affected.

\*\*\*\*\*

	ORG.	\$203F0	
START	MOVE.L	D0,\$9C8	SAVE REGISTER D0
	MOVE.L	#6,D0	SET PULSE LOW WIDTH
	BCLR	#1,\$30005	BRING OUTSHAKE LOW
			PULSE WIDTH =
			3 * (D0 + 1) + 4, MICRO SEC.
SELF	DBRA	D0,SELF	HOLD OUTSHAKE LOW
	BSET	#1,\$30005	TAKE OUTSHAKE HIGH
RESTORE	MOVE.L	\$9C8,D0	RESTORE REGISTER D0
END	RTS		RETURN TO CALLING ROUTINE

SUBROUTINE OUTSHAKE

\*\*\*\*\*

### SUBROUTINE INSHAKE

PURPOSE: To wait for an input handshake pulse from the Host through the parallel port

INITIAL CONDITIONS: This subroutine requires no parameters to be passed to it.

ACTION: The subroutine will wait for a pulse on the input handshake line of the parallel port. The pulse width should be greater than 15 microseconds. The subroutine will wait a maximum of approximately 45 milliseconds for the strobe to go low. Also, it will wait for 45 milliseconds for the strobe line to go back high.

If the Host does not respond within either one of these timeout times, the 68000 will display the appropriate error message on the terminal via the subroutine ERROR. It is then attempted to reestablish normal communications with the Host.

REGISTER USAGE: No registers are affected.

\*\*\*\*\*

	ORG.	\$20414	
START	MOVE.L	D0,\$9C8	SAVE REGISTER D0
	MOVE.L	#6000,D0	INITIALIZE TIMEOUT DELAY
HIGH	BTST	#2,\$30005	WAIT FOR INSHAKE TO GO LOW
	BEQ	CONTINUE	IF LOW CONTINUE
	DBRA	D0,HIGH	TRY AGAIN
	JSR	ERROR	TIMEOUT EXCEEDED DISPLAY MSG.
	BRA	RESTORE	
CONTINUE	MOVE.L	#6000,D0	INITIALIZE TIMEOUT DELAY
LOW	BTST	#2,\$30005	WAIT FOR INSHAKE HIGH
	BNE	RESTORE	IF INSHAKE IS HIGH RETURN
	DBRA	D0,LOW	TRY AGAIN
	JSR	ERROR	TIMEOUT EXCEEDED DISPLAY MSG.
RESTORE	MOVE.L	\$9C8,D0	RESTORE REGISTER D0
END	RTS		RETURN TO CALLING ROUTINE

SUBROUTINE INSHAKE

\*\*\*\*\*

### SUBROUTINE ERROR

PURPOSE: To trap and display the appropriate error condition on the terminal

INITIAL CONDITIONS: Subroutine error is sent an error flag in register D2.B. The possible error flags are:

D2=1 \*\*> STROBE ERROR <\*\* THE 68000 DID NOT RECEIVE THE EXPECTED STROBE FROM THE HOST IN THE ALLOTTED AMOUNT OF TIME.

D2=2\*\*> ACKNOWLEDGE ERROR <\*\* THE 68000 DID NOT RECEIVE THE DATA ACKNOWLEDGE FROM THE HOST IN THE ALLOTTED AMOUNT OF TIME.

D2=3 \*\*> ILLEGAL INTERFACE COMMAND <\*\* THE 68000 RECEIVED AN ILLEGAL INTERFACE COMMAND FROM THE HOST.

D2=4 \*\*> VECTOR MEMORY ERROR <\*\* THE 68000 WAS UNABLE TO ESTABLISH COMMUNICATIONS WITH THE HP-1345A.

ACTION: The subroutine determines what the error condition is and displays the error message on the terminal.

REGISTER USAGE: No registers are affected.

\*\*\*\*\*

	ORG.	\$20450	
START	MOVEM.L	D0-D1/D3/A0/A5-A6,\$9CC	SAVE REGISTERS
	MOVE.L	#7,D3	
HERE	MOVE.B	#\$A,D0	INSERT SEVEN LINE FEEDS
	MOVE.B	#248,D7	SEND LINE FEEDS TO TERMINAL
	TRAP	#14	
	DBRA	D3,HERE	
	MOVE.L	#\$3A20,A5	STARTING ADDRESS
	MOVE.L	#\$3A2B,A6	ENDING ADDRESS
	MOVE.B	#243,D7	DISPLAY "***>" ON TERMINAL
	TRAP	#14	
	CMPI.B	#1,D2	CHECK FOR STROBE ERROR
	BEQ	STRBERR	
	CMPI.B	#2,D2	CHECK FOR ACKNOWLEDGE ERROR
	BEQ	ACKERR	
	CMPI.B	#3,D2	CHECK FOR ILLIGAL COMMAND ERR
	BEQ	CMDERR	

SUBROUTINE ERROR

\*\*\*\*\*

SUBROUTINE ERROR CONTINUED

\*\*\*\*\*

	CMPI.B	#4,D2	CHECK FOR VECTOR MEMORY ERROR
	BEQ	VCMERR	
	BRA	RESTORE	
STRBERR	MOVE.L	#\$3A38,A5	SET UP POINTERS FOR ERROR MSG
	MOVE.L	#\$3A44,A6	
	BRA	DISPLAY	
ACKERR	MOVE.L	#\$3A44,A5	SET UP POINTERS FOR ERROR MSG
	MOVE.L	#\$3A55,A6	
	BRA	DISPLAY	
CMDERR	MOVE.L	#\$3A55,A5	SET UP POINTERS FOR ERROR MSG
	MOVE.L	#\$3A6E,A6	
	BRA	DISPLAY	
VCMERR	MOVE.L	#\$3A6E,A5	SET UP POINTERS FOR ERROR MSG
	MOVE.L	#\$3A81,A6	
	NOB		
DISPLAY	MOVE.B	#243,D7	CALL DISPLAY HANDLER
	TRAP	#14	
	MOVE.L	#\$3A2B,A5	SET UP POINTERS
	MOVE.L	#\$3A38,A6	
	MOVE.B	#227,D7	DISPLAY "<***" ON TERMINAL
	TRAP	#14	
RESTORE	MOVEM.L	\$9CC,D0-D1/D3/A0/A5-A6	RESTORE REGISTERS
END	RTS		RETURN TO CALLING ROUTINE

SUBROUTINE ERROR

\*\*\*\*\*

INTERRUPT CONTROL ROUTINE

PURPOSE: To determine, control and initiate service of an interrupting device

INITIAL CONDITIONS: There are three devices that may request service. The three interrupts and their priority are:

1. COMIRQ COMMUNICATIONS THROUGH PARALLEL PORT WITH HOST
2. KEYIRQ COMMAND KEYBOARD IS ACTIVE
3. CLKIRQ REAL TIME CLOCK

ACTION: This Interrupt Control Routine checks the interrupt status bits of PIA #1 Port B to determine which interrupt is requesting the service. The routine then fetches the address of the service routine from a look-up table and transfers control to that particular service routine. After the service routine has serviced the interrupt, the control is returned to the interrupted program via the Interrupt Control Routine.

REGISTER USAGE: No registers are affected.

\*\*\*\*\*

	ORG.	\$20500	
START	MOVEM.L	D0-D1/A0,\$98C	SAVE REGISTERS
	MOVE.L	#\$3AC0,A0	BASE ADDR. FOR ADDRESS OF SERVICE ROUTINE
	MOVE.W	#7,D0	INITIALIZE TEST BIT NUMBER
	MOVE.B	\$30005,D1	GET IRQ STATUS
NEXT	BTST	D0,D1	TEST BIT D0 OF IRQ STATUS
	BEQ	IRQ	BRANCH IF IRQ BIT FOUND
	CMPI.B	#4,D0	CHECK TO SEE IF NO IRQ
	BEQ	NOIRQ	
	DBRA	D0,\$NEXT	CHECK NEXT IRQ BIT D0
IRQ	BCLR	#7,\$30001	DISABLE PAR PORT INTERRUPTS
	LSL.W	#2,D0	MULTIPLY BY FOUR FOR OFFSET
	LEA.L	-16(A0,D0.W),A0	GET ADDR. OF ADDR. OF ROUTINE
	MOVE.L	(A0),A0	GET ADDRESS OF ROUTINE
	JSR	(A0)	EXECUTE SERVICE ROUTINE
	BSET	#7,\$30001	ENABLE PAR PORT FOR INTERRUPTS
NOIRQ	MOVEM.L	\$98C,D0-D1/A0	RESTORE REGISTERS
END	RTE		RETURN FROM INTERRUPT

INTERRUPT CONTROL ROUTINE

\*\*\*\*\*

SUBROUTINE COMIRQ

PURPOSE: To control the communications with the Host through the parallel port of the 68000 interface

INITIAL CONDITIONS: When the Host interrupts the 68000, it has the interface command vector number on the data bus of the parallel port. The 68000 uses this vector as an offset to fetch the address of the Interface Command routine. There are space available for seven interface commands, however only four commands are used at this time. The four commands are:

- CMD 1. THE HOST IS REQUESTING THE "DATA PACKAGE" BE SENT
- CMD 2. THE HOST REQUESTS THAT THE FOLLOWING HP-1345A COMMANDS BE SENT TO VECTOR MEMORY
- CMD3. THE HOST REQUESTS THAT A CERTAIN BLOCK OF HP-1345A COMMANDS BE READ BACK TO THE HOST FORM VECTOR MEMORY
- CMD4. THE HOST REQUESTS THE ALARM TO BE TOGGLED

ACTION: After it is determined which interface command is to be executed, the COMIRQ routine transfers control to that service routine. On completion of the service, this routine clears the COMIRQ interrupt and returns to the Interrupt Control Routine.

REGISTER USAGE: No registers are affected.

\*\*\*\*\*

ORG.	\$20546	
START MOVEM.L	D0-D2/A0,\$998	SAVE REGISTERS
MOVE.L	#\$3AD0,A0	A0 BASE ADDR FOR CMD ADDR'S
CLR.W	D0	CLEAR D0
MOVE.B	\$30021,D0	READ COMMAND AT PARALLEL PORT
CMPI.W	#8,D0	CHECK FOR ILLEGAL COMMAND
BMI	OK	EIGHT LEGAL COMMANDS
JSR	OUTSHAKE	ACKNOWLEDGE ILLEGAL COMMAND
MOVE.B	#3,D2	DISPLAY "ILLEGAL COMMAND, COMMAND IGNORED"

SUBROUTINE COMIRQ



\*\*\*\*\*

SUBROUTINE COMIRQ

\*\*\*\*\*

	JSR	ERROR	SEND COMMAND ERROR MSG.
	BRA	RESTORE	RESTORE AND RETURN
OK	LSL.W	#2,D0	MULTIPLY BY 4 FOR OFFSET
	LEA.L	-4(A0,D0.W),A0	FORM INTERFACE COMMAND ADDR.
	MOVE.L	(A0),A0	GET ADDRESS OF INT CMD ROUTINE
	JSR	(A0)	EXECUTE INTERFACE COMMAND
RESTORE	MOVEM.L	\$998,D0-D2/A0	RESTORE REGISTERS
END	RTS		RETURN TO CALLING ROUTINE

SUBROUTINE COMIRQ

\*\*\*\*\*

SUBROUTINE INTERFACE COMMAND #1

PURPOSE: To transfer the "Data Package" to the Host

INITIAL CONDITIONS: This subroutine requires no parameters be passed to it and is initiated by the Host sending a value of \$01 to the 68000 through the parallel port. This value represents Interface Command #1.

ACTION: The data package is sent to the Host via the parallel port and handshake lines. The data package consists of pertinent data such as digital values for the sixteen analog channels, last key pressed, time, date, a byte of binary inputs and the first entry in the package holds the number of entries in the data package.

This subroutine watches for error conditions such as the Host failing to acknowledge a data transfer. When an error occurs the subroutine "ERROR" is called. Subroutine ERROR will display the error condition on the terminal. This subroutine then tries to reestablish communications with the Host to continue data transfer. After the data package transfer is complete, the subroutine restores all registers and returns to the COMIRQ routine.

REGISTER USAGE: No registers are affected

\*\*\*\*\*

	ORG.	\$20590	
START	MOVEM.L	DO-D2/A0,\$9A8	SAVE REGISTERS
	JSR	OUTSHAKE	ACKNOWLEDGE INTERFACE CMD #1
	MOVE.L	#30021,A0	CHANGE PARALLEL PORT TO OUTPUT
	MOVE.B	#\$FF,DO	
	JSR	INTPIA	
	MOVE.L	#\$1FA0,A0	GET BASE ADDR OF DATA PACKAGE
	CLR.L	DO	
	MOVE.B	(A0),DO	GET NUMBER OF ENTRIES IN PACK.
	MOVE.B	#2,D2	SET ERROR FLAG TO ACKERR
NEXT	MOVE.B	(A0)+,\$30021	SEND ENTRY TO PORT
	JSR	OUTSHAKE	SEND STROBE PULSE
	JSR	INSHAKE	WAIT FOR ACKNOWLEDGE PULSE
	DBRA	DO,NEXT	POINT TO NEXT ENTRY

SUBROUTINE INTERFACE COMMAND #1

\*\*\*\*\*

SUBROUTINE INTERFACE COMMAND #1 CONTINUED

\*\*\*\*\*

	BSET	#7,\$30001	ENABLE PAR PORT FOR INTERRUPTS
	MOVE.L	#\$30021,A0	CHANGE PARALLEL PORT TO INPUT
	MOVE.B	#0,D0	
	JSR	INTPIA	
RESTORE	MOVEM.L	\$9A8,D0-D2/A0	RESTORE REGISTERS
END	RTS		RETURN TO CALLING ROUTINE

SUBROUTINE INTERFACE COMMAND #1

\*\*\*\*\*

SUBROUTINE INTERFACE COMMAND #2

PURPOSE: To transfer HP-1345A commands from the Host to vector memory

INITIAL CONDITIONS: This subroutine requires no parameters be passed and is initiated by the Host sending a value of \$02 to the 68000 through the parallel port. This value represents Interface Command #2.

ACTION: This subroutine reads an HP-1345A command from the Host and sends it to vector memory. The transfer continues until the most significant byte read has a value of \$FF. This value signifies that the Host has sent the last command. The subroutine terminates the data transfer, restores the registers and returns to the COMIRQ routine.

The subroutine watches for errors such as the Host failing to send another Data Strobe when the 68000 is expecting it. When an error occurs the subroutine "ERROR" is called. It will display the error condition on the terminal.

REGISTER USAGE: No registers are affected

\*\*\*\*\*

	ORG.	\$205EA	
START	MOVEM.L	D0/D2,\$9A8	SAVE REGISTERS
	JSR	OUTSHAKE	ACKNOWLEDGE INTERFACE CMD #2
	MOVE.B	#1,D2	SET ERROR FLAG TO STBERR
NEXT	JSR	INSHAKE	WAIT FOR STROBE PULSE
	MOVE.B	\$30021,D0	READ MSB OF HP COMMAND
	CMPI.B	#\$FF,D0	TRANSFER COMPLETE? (\$FF)
	BEQ	DONE	
	JSR	OUTSHAKE	SEND ACKNOWLEDGE PULSE FOR MSB
	LSL.W	#8,D0	MOVE DATA TO MSB OF D0
	JSR	INSHAKE	WAIT FOR DATA STROBE
	MOVE.B	\$30021,D0	READ LSB OF HP COMMAND
	JSR	SNDCMD	SEND CMD TO VECTOR MEMORY
	JSR	OUTSHAKE	SEND ACKNOWLEDGE PULSE FOR LSB
	BRA	NEXT	GET NEXT COMMAND
DONE	JSR	OUTSHAKE	ACKNOWLEDGE DONE SIGNAL
RESTORE	MOVEM.L	\$9A8,D0/D2	RESTORE REGISTERS
END	RTS		RETURN TO CALLING ROUTINE

SUBROUTINE INTERFACE COMMAND #2

\*\*\*\*\*

### SUBROUTINE INTERFACE COMMAND #3

PURPOSE: To read a block of HP-1345A commands from vector memory

INITIAL CONDITIONS: This subroutine requires no parameters be passed by the calling routine and is initiated by the Host sending a value of \$03 to the 68000 through the parallel port. This value represents Interface Command #3.

ACTION: The Host first sends two 16-bit words which represent the starting address and ending address of the block to be read from vector memory. The subroutine uses the starting address and formats a memory address pointer command to set-up the vector memory address pointer to the first memory location of the block to be read.

The block is then read, a byte at a time and sent back to the Host. When the transfer is complete, the ports are restored to their proper direction, the registers are restored and the control is transferred back to the COMIRQ routine.

REGISTER USAGE: No registers are affected

\*\*\*\*\*

START	ORG.	\$20630	SAVE REGISTERS
	MOVEM.L	D0-D3/A0,\$9A8	ACKNOWLEDGE INTERFACE CMD #3
	JSR	OUTSHAKE	SET ERROR FLAG TO STRBERR
	MOVE.B	#1,D2	WAIT FOR STROBE PULSE
	JSR	INSHAKE	READ MSB OF START ADDRESS
	MOVE.B	\$30021,D0	MOVE LSB OF D0 TO MSB
	LSL.W	#8,D0	SEND ACKNOWLEDGE
	JSR	OUTSHAKE	WAIT FOR STROBE
	JSR	INSHAKE	READ LSB OF START ADDRESS
	MOVE.B	\$30021,D0	SEND ACKNOWLEDGE
	JSR	OUTSHAKE	WAIT FOR STROBE
	JSR	INSHAKE	READ MSB OF ENDING ADDRESS
	MOVE.B	\$30021,D1	MOVE LSB OF D1 TO MSB
	LSL.W	#8,D1	SEND ACKNOWLEDGE
	JSR	OUTSHAKE	

SUBROUTINE INTERFACE COMMAND #3

\*\*\*\*\*

SUBROUTINE INTERFACE COMMAND #3 CONTINUED

\*\*\*\*\*

	JSR	INSHAKE	WAIT FOR STROBE
	MOVE.B	\$30021,D1	READ LSB OF ENDING ADDRESS
	JSR	OUTSHAKE	SEND ACKNOWLEDGE
	MOVE.W	D0,D3	COPY STARTING ADDRESS INTO D3
	MOVE.L	#\$30021,A0	CHANGE PARALLEL PORT TO OUTPUT
	MOVE.B	#\$FF,D0	
	JSR	INTPIA	
	SUB.W	D3,D1	D1 IS NUMBER OF VEC MEM READS
	MOVE.W	D3,D0	
	ORI.W	#\$C000,D0	FORMAT ADDRESS POINTER COMMAND
	JSR	SNDCMD	SET ADDR PTR TO STARTING ADDR
	MOVE.B	#0,D0	CHANGE HP LSB PORT TO INPUT
	MOVE.L	#\$30009,A0	
	JSR	INTPIA	
	MOVE.L	#\$3000D,A0	CHANGE HP MSB PORT TO INPUT
	JSR	INTPIA	
	MOVE.B	\$30001,D0	SET NEW DIR AND BUFFER CONTROL OF HP DATA PORT
	ORI.B	#3,D0	SET WR AND RD LINES HIGH
	AND.B	#\$F3,D0	SET HP DEVICE AND DIR LOW
	MOVE.B	D0,\$30001	
READ	MOVE.B	#2,D2	SET ERROR FLAG TO ACKERR
	BCLR	#1,\$30001	BRING HP READ LINE LOW
	MOVE.B	\$3000D,\$30021	SEND MSB TO PARALLEL PORT
	JSR	OUTSHAKE	SEND STROBE PULSE FOR MSB
	JSR	INSHAKE	WAIT FOR ACKNOWLEDGE PULSE
	MOVE.B	\$30009,\$30021	SEND LSB TO PARALLEL PORT
	BSET	#1,\$30001	BRING HP READ LINE HIGH
	JSR	OUTSHAKE	SEND STROBE PULSE FOR LSB
	JSR	INSHAKE	WAIT FOR ACKNOWLEDGE PULSE
	DBRA	D1,READ	READ NEXT VEC. MEM. LOCATION
	BSET	#7,\$30001	ENABLE PAR PORT INTERRUPTS
	MOVE.B	\$30001,D0	GET HP CONTROL BITS
	ORI.B	#\$0F,D0	SET DIR DEVICE WR AND RD HIGH
	MOVE.B	D0,\$30001	
	MOVE.B	#\$FF,D0	CHANGE HP LSB PORT TO OUTPUT
	MOVE.L	#\$30009,A0	
	JSR	INTPIA	
	MOVE.L	#\$3000D,A0	CHANGE HP MSB PORT TO OUTPUT
	JSR	INTPIA	
	MOVE.L	#\$30021,A0	CHANGE PARALLEL PORT TO INPUT
	MOVE.B	#0,D0	
	JSR	INTPIA	
RESTORE	MOVEM.L	\$9A8,D0-D3/A0	RESTORE REGISTERS
END	RTS		RETURN TO CALLING ROUTINE

SUBROUTINE INTERFACE COMMAND #3

\*\*\*\*\*

SUBROUTINE INTERFACE COMMAND #4

PURPOSE: To sound or silence the alarm

INITIAL CONDITIONS: This subroutine requires no parameters be passed and is initiated by sending the Host a value of \$04 to the 68000 through the parallel port. This value represents a Interface Command #4.

ACTION: This subroutine toggles the alarm on the interface board. The alarm is used to alert the pilot of an alarm condition such as impending stall, gear up on an approach etc. The Host sends Interface Command #4 to turn the alarm on and sends the command again to turn the alarm off.

REGISTER USAGE: No registers are affected

\*\*\*\*\*

START	ORG.	\$20750	ACKNOWLEDGE INTERFACE CMD #4
	JSR	OUTSHAKE	TOGGLE THE ALARM OUTPUT
	BCHG	#0,\$30005	
END	RTS		RETURN TO CALLING ROUTINE

SUBROUTINE INTERFACE COMMAND #4

\*\*\*\*\*

### SUBROUTINE KEYIRQ

**PURPOSE:** To determine and process the key pressed on the command keyboard

**INITIAL CONDITIONS:** This subroutine requires no parameters be passed and is initiated when any key on the command keyboard is pressed.

**ACTION:** This subroutine calls another subroutine called "GETKEY" which will return the value of the key pressed in register D2.B and in the Data Package. If a value of \$37 is returned from GETKEY, then no key was found pressed and this subroutine returns to the calling routine. If a valid key is returned from GETKEY, the key is sent to the parallel port and the Host is interrupted. When the Host acknowledges the key value, this subroutine will return to the calling routine.

**REGISTER USAGE:** No registers are affected

\*\*\*\*\*

	ORG.	\$20760	
START	MOVEM.L	D0/D2/A0,\$9BC	SAVE REGISTERS
	JSR	GETKEY	GET THE VALUE OF KEY PRESSED
	CMPI.B	#37,D2	WAS THERE A KEY FOUND
	BEQ	NOKEY	IF NO KEY THEN RETURN
	CMPI.B	#36,D2	CHECK FOR SET CLOCK KEY
	BNE	CONTINUE	IF NOT SET CLOCK KEY, CONTINUE
	JSR	SETCLK	SET THE CLOCK/CALENDAR
	BRA	NOKEY	AFTER CLOCK IS SET, RETURN
CONTINUE	MOVE.L	#\$30021,A0	CHANGE PARALLEL PORT TO OUTPUT
	MOVE.B	#\$FF,D0	
	JSR	INTPIA	
	MOVE.B	D2,\$30021	SEND VALID KEY TO PAR. PORT
	MOVE.B	#2,D2	SET ERROR FLAG TO ACKERR
	JSR	OUTSHAKE	STROBE THE HOST
	JSR	INSHAKE	WAIT FOR ACKNOWLEDGE
	BSET	#7,\$30001	ENABLE PAR PORT INTERRUPTS
	MOVE.L	#\$30021,A0	CHANGE PARALLEL PORT TO INPUT
	MOVE.B	#0,D0	
	JSR	INTPIA	
NOKEY	BCLR	#6,\$30001	CLEAR KEYBOARD INTERRUPT
	BSET	#6,\$30001	
RESTORE	MOVEM.L	\$9BC,D0/D2/A0	RESTORE REGISTERS
END	RTS		RETURN TO CALLING ROUTINE

SUBROUTINE KEYIRQ



\*\*\*\*\*

### SUBROUTINE CLKIRQ

PURPOSE: To read the Real Time Clock and Calendar and send the time data to the data package

INITIAL CONDITONS: This subroutine requires no parameters be passed to it and is initiated by the real time clock interrupt.

ACTION: The subroutine is initiated by the real time clock interrupt timer. The clock will interrupt at a predetermined interval such as one-half second. This subroutine will read the BCD numbers corresponding to the time and date and convert these numbers to binary numbers. The binary numbers are then supplied to the data package.

To alert the Host that the time has changed, the time vector (\$60) is placed on the parallel port data bus and the Host is interrupted. After receiving a Data Acknowledge from the Host, the subroutine clears the clock interrupt, restores registers and returns to the calling routine.

REGISTER USAGE: No registers are affected

\*\*\*\*\*

START	ORG.	\$207CA	
	MOVEM.L	D0/D2/A0,\$9BC	SAVE REGISTERS
	MOVE.B	\$30063,D0	DUMMY READ TO SETUP READ LOGIC
	MOVE.L	#\$1FB9,A0	SET UP DATA PACKAGE POINTER
	MOVE.B	\$30067,D0	READ TENS OF SECONDS
	AND.W	#\$F,D0	CLEAR DATA4 - DATA7
	MULU.W	#10,D0	
	MOVE.B	\$30065,D2	READ THE ONES OF SECONDS
	AND.B	#\$F,D2	CLEAR DATA4 - DATA7
	ADD.B	D2,D0	ADD TENS AND ONES OF SECONDS
	MOVE.B	D0,-(A0)	PUT IN THE DATA PACKAGE
	MOVE.B	\$3006B,D0	READ TENS OF MINUTES
	AND.W	#\$F,D0	CLEAR DATA4 - DATA7
	MULU.W	#10,D0	
	MOVE.B	\$30069,D2	READ THE ONES OF MINUTES
	AND.B	#\$F,D2	CLEAR DATA4 - DATA7
	ADD.B	D2,D0	ADD TENS AND ONES OF SECONDS
	MOVE.B	D0,-(A0)	PUT IN THE DATA PACKAGE

SUBROUTINE CLKIRQ

\*\*\*\*\*

SUBROUTINE CLKIRQ CONTINUED

\*\*\*\*\*

MOVE.B	\$3006F,D0	READ TENS OF HOURS
AND.W	#\$F,D0	CLEAR DATA4 - DATA7
MULU.W	#10,D0	
MOVE.B	\$3006D,D2	READ THE ONES OF HOURS
AND.B	#\$F,D2	CLEAR DATA4 - DATA7
ADD.B	D2,D0	ADD TENS AND ONES OF HOURS
MOVE.B	D0,-(A0)	PUT IN THE DATA PACKAGE
MOVE.B	\$30073,D0	READ TENS OF DAYS
AND.W	#\$F,D0	CLEAR DATA4 - DATA7
MULU.W	#10,D0	
MOVE.B	\$30071,D2	READ THE ONES OF DAYS
AND.B	#\$F,D2	CLEAR DATA4 - DATA7
ADD.B	D2,D0	ADD TENS AND ONES OF DAYS
MOVE.B	D0,-(A0)	PUT IN THE DATA PACKAGE
MOVE.B	\$30075,D0	READ DAY OF THE WEEK
AND.B	#\$F,D0	CLEAR DATA4 - DATA7
MOVE.B	D0,-(A0)	PUT IN THE DATA PACKAGE
MOVE.B	\$30079,D0	READ TENS OF MONTHS
AND.W	#\$F,D0	CLEAR DATA4 - DATA7
MULU.W	#10,D0	
MOVE.B	\$30077,D2	READ THE ONES OF MONTHS
AND.B	#\$F,D2	CLEAR DATA4 - DATA7
ADD.B	D2,D0	ADD TENS AND ONES OF MONTHS
MOVE.B	D0,-(A0)	PUT IN THE DATA PACKAGE
MOVE.B	\$30063,D0	DUMMY READ
MOVE.L	#\$30021,A0	MAKE PARALLEL PORT OUTPUT
MOVE.B	#\$FF,D0	
JSR	INTPIA	
MOVE.B	#\$60,\$30021	SEND THE TIME VECTOR TO PORT
JSR	OUTSHAKE	SEND DATA STROBE
MOVE.B	#2,D2	SET ERROR FLAG TO ACKERR
JSR	INSHAKE	WAIT FOR ACKNOWLEDGE
BSET	#7,\$30001	ENABLE PAR PORT FOR INTERRUPTS
MOVE.L	#\$30021,A0	CHANGE PARALLEL PORT TO INPUT
MOVE.B	#0,D0	
JSR	INTPIA	
MOVE.B	\$3007F,D0	THREE READS CLEARS CLOCK IRQ
MOVE.B	\$3007F,D0	
MOVE.B	\$3007F,D0	
BCLR	#5,\$30001	CLEAR CLOCK INTERRUPT LATCH
BSET	#5,\$30001	
RESTORE	MOVEM.L \$9BC,D0/D2/A0	RESTORE REGISTERS
END	RTS	RETURN TO CALLING ROUTINE

SUBROUTINE CLKIRQ

\*\*\*\*\*

MEMORY USAGE DEFINED

\*\*\*\*\*

SCRATCHPAD RAM

\$900-\$9E3      TEMPORARY STORAGE OF REGISTERS  
\$9E4-\$1F9F     AVAILABLE SCRATCHPAD RAM

KEYPAD LOOKUP TABLE

<u>Location</u>	<u>Value</u>	<u>Description</u>
\$3800	\$0A	KEYPAD LOOKUP TABLE
3801	0A	
3802	0A	
3803	00	KEY 0
3804	0A	
3805	0A	
3806	0A	
3807	0A	
3808	0A	
3809	01	KEY 1
380A	02	KEY 2
380B	03	KEY 3
380C	0A	
380D	0A	
380E	0A	
380F	04	KEY 4
3810	05	KEY 5
3811	06	KEY 6
3812	0A	
3813	0A	
3814	0A	
3815	07	KEY 7
3816	08	KEY 8
3817	09	KEY 9

HP-1345A GRAPHIC COMMANDS FOR SETCLK ROUTINE

<u>Location</u>	<u>Value</u>	<u>Description</u>
\$3824	\$0032	Number of commands in list
3826	C000	Set address ptr to zero
3828	00C0	Plot X
382A	1600	Plot Y
382C	4945	E
382E	494E	N

<u>Location</u>	<u>Value</u>	<u>Description</u>
3830	4954	T
3832	4945	E
3834	4952	R
3836	4920	SPACE
3838	4954	T
383A	4957	W
383C	494F	O
383E	4920	SPACE
3840	4944	D
3842	4949	I
3844	4947	G
3846	4949	I
3848	4954	T
384A	4953	S
384C	4920	SPACE
384E	4946	F
3850	494F	O
3852	4952	R
3854	4910	SPACE
3856	4945	E
3858	4941	A
385A	4943	C
385C	4948	H
385E	4910	SPACE
3860	4945	E
3862	494E	N
3864	4954	T
3866	4952	R
3868	4959	Y
386A	0200	Plot X
386C	1400	Plot Y
386E	4945	E
3870	494E	N
3872	4954	T
3874	4945	E
3876	4952	R
3878	4910	SPACE
387A	4959	Y
387C	4945	E
387E	4941	A
3880	4952	R
3882	8045	Internal Jump to Address \$45 (69)
3884	C045	Address Pointer to \$45 (69)
3886	0000	NOP
3888	8FFF	Internal Jump to \$FFF (4095)
388A	0007	Number of commands in list
388C	C029	Address Pointer to \$29 (41)

<u>Location</u>	<u>Value</u>	<u>Description</u>
388E	494D	M
3890	494F	O
3892	494E	N
3894	4954	T
3896	4948	H
3898	8045	Internal Jump to \$45 (69)
389A	0011	Number of commands in list
389C	C029	Address Pointer to \$29 (41)
389E	4944	D
38A0	4941	A
38A2	4959	Y
38A4	4910	SPACE
38A6	4918	(
38A8	4930	0
38AA	4931	1
38AC	493D	=
38AE	4953	S
38B0	4955	U
38B2	494E	N
38B4	4944	D
38B6	4941	A
38B8	4959	Y
38BA	4929	)
38BC	8045	Internal Jump to \$45 (69)
38BE	0006	Number of commands in list
38C0	C029	Address Pointer to \$29 (41)
38C2	4944	D
38C4	4941	A
38C6	4954	T
38C8	4945	E
38CA	8045	Internal Jump to \$45 (69)
38CC	0007	Number of commands in list
38CE	C029	Address Pointer to \$29 (41)
38D0	4948	H
38D2	494F	O
38D4	4955	U
38D6	4952	R
38D8	4953	S
38DA	8045	Internal Jump to \$45 (69)
38DC	0009	Number of commands in list
38DE	C029	Address Pointer to \$29 (41)
38E0	494D	M
38E2	4949	I
38E4	494E	N
38E6	4955	U
38E8	4954	T
38EA	4945	E

<u>Location</u>	<u>Value</u>	<u>Description</u>
38EC	4953	S
38EE	8045	Internal Jump to \$45 (69)
38F0	0020	Number of commands in list
38F2	C000	Address Pointer to \$00 (00)
38F4	0100	Plot X
38F6	1400	Plot Y
38F8	4950	P
38FA	4952	R
38FC	4945	E
38FE	4953	S
3900	4953	S
3902	4920	SPACE
3904	4941	A
3906	494E	N
3908	4959	Y
390A	4910	SPACE
390C	494B	K
390E	4945	E
3910	4959	Y
3912	4910	SPACE
3914	4954	T
3916	494F	O
3918	4920	SPACE
391A	4953	S
391C	4954	T
391E	4941	A
3920	4952	R
3922	4954	T
3924	4920	SPACE
3926	4943	C
3928	494C	L
392A	494F	O
392C	4943	C
392E	494B	K
3930	8FFF	Internal Jump to \$FFF (4095)
3932	0034	Number of commands in list
3934	C000	Address pointer to \$00 (00)
3936	0100	Plot X
3938	1300	Plot Y
393A	4943	C
393C	494C	L
393E	494F	O
3940	4943	C
3942	494B	K
3944	492F	/
3946	4943	C
3948	4941	A

<u>Location</u>	<u>Value</u>	<u>Description</u>
394A	494C	L
394C	4945	E
394E	494E	N
3950	4944	D
3952	4941	A
3954	4952	R
3956	4920	SPACE
3958	4949	I
395A	4953	S
395C	4920	SPACE
395E	4953	S
3960	4945	E
3962	4954	T
3964	0100	Plot X
3966	1200	Plot Y
3968	4950	P
396A	4952	R
396C	4945	E
396E	4953	S
3970	4953	S
3972	4920	SPACE
3974	4941	A
3976	494E	N
3978	4959	Y
397A	4920	SPACE
397C	494B	K
397E	4945	E
3980	4959	Y
3982	4920	SPACE
3984	4954	T
3986	494F	O
3988	4920	SPACE
398A	4943	C
398C	494F	O
398E	494E	N
3990	4954	T
3992	4949	I
3994	494E	N
3996	4955	U
3998	4945	E
399A	8FFF	Internal Jump to \$FFF (4095)
399C	001A	Number of commands in list
399E	0140	Plot X
39A0	1200	Plot Y
39A2	4949	I
39A4	494E	N
39A6	4956	V

<u>Location</u>	<u>Value</u>	<u>Description</u>
39A8	4941	A
39AA	494C	L
39AC	4949	I
39AE	4944	D
39B0	4920	SPACE
39B2	494B	K
39B4	4945	E
39B6	4959	Y
39B8	4920	SPACE
39BA	4920	SPACE
39BC	4920	SPACE
39BE	4954	T
39C0	4952	R
39C2	4959	Y
39C4	4920	SPACE
39C6	4941	A
39C8	4947	G
39CA	4941	A
39CC	4949	I
39CE	494E	N
39D0	8FFF	Internal Jump to \$FFF (4095)
39D2	XXXX	Not used

POWER UP AND ERROR MESSAGES DISPLAYED  
ON ZENITH TERMINAL

<u>Location</u>	<u>Value</u>	<u>Description</u>
\$39D4	07	Bell
39D5	45	E
39D6	48	H
39D7	53	S
39D8	49	I
39D9	20	SPACE
39DA	53	S
39DB	59	Y
39DC	53	S
39DD	44	T
39DE	45	E
39DF	4D	M
39E0	20	SPACE
39E1	52	R
39E2	55	U
39E3	4E	N
39E4	4E	N
39E5	49	I
39E6	4E	N
39E7	47	G



<u>Location</u>	<u>Value</u>	<u>Description</u>
39E8	49	I
39E9	4E	N
39EA	49	I
39EB	54	T
39EC	49	I
39ED	41	A
39EE	4C	L
39EF	49	I
39F0	5A	Z
39F1	41	A
39F2	54	T
39F3	49	I
39F4	4F	O
39F5	4E	N
39F6	53	S
39F7	20	SPACE
39F8	43	C
39F9	4F	O
39FA	4D	M
39FB	50	P
39FC	4C	L
39FD	45	E
39FE	54	T
39FF	45	E
3A00	43	C
3A01	4F	O
3A02	4E	N
3A03	54	T
3A04	49	I
3A05	4E	N
3A06	55	U
3A07	45	E
3A08	20	SPACE
3A09	54	T
3A0A	4F	O
3A0B	20	SPACE
3A0C	46	F
3A0D	4C	L
3A0E	49	I
3A0F	47	G
3A10	48	H
3A11	54	T
3A12	20	SPACE
3A13	53	S
3A14	49	I
3A15	40	M
3A16	55	U

<u>Location</u>	<u>Value</u>	<u>Description</u>
3A17	4C	L
3A18	41	A
3A19	54	T
3A1A	4F	O
3A1B	52	R
3A1C	07	Bell
3A1D	07	Bell
3A1E	07	Bell
3A1F	07	Bell
3A20	07	Bell
3A21	0D	CR
3A22	20	SPACE
3A23	20	SPACE
3A24	20	SPACE
3A25	20	SPACE
3A26	2A	*
3A27	2A	*
3A28	2A	*
3A29	3E	>
3A2A	20	SPACE
3A2B	20	SPACE
3A2C	3C	<
3A2D	2A	*
3A2E	2A	*
3A2F	2A	*
3A30	0A	LINE FEED
3A31	0A	LINE FEED
3A32	0A	LINE FEED
3A33	0A	LINE FEED
3A34	0A	LINE FEED
3A35	0A	LINE FEED
3A36	0A	LINE FEED
3A37	0A	LINE FEED
3A38	53	S
3A39	54	T
3A3A	52	R
3A3B	4F	O
3A3C	42	B
3A3D	45	E
3A3E	20	SPACE
3A3F	45	E
3A40	52	R
3A41	52	R
3A42	4F	O
3A43	52	R
3A44	41	A
3A45	43	C

<u>Location</u>	<u>Value</u>	<u>Description</u>
3A46	4B	K
3A47	4E	N
3A48	4F	O
3A49	57	W
3A4A	4C	L
3A4B	45	E
3A4C	44	D
3A4D	47	G
3A4E	45	E
3A4F	20	SPACE
3A50	45	E
3A51	52	R
3A52	52	R
3A53	4F	O
3A54	52	R
3A55	49	I
3A56	4C	L
3A57	4C	L
3A58	45	E
3A59	47	G
3A5A	41	A
3A5B	4C	L
3A5C	20	SPACE
3A5D	49	I
3A5E	4E	N
3A5F	54	T
3A60	45	E
3A61	52	R
3A62	46	F
3A63	41	A
3A64	43	C
3A65	45	E
3A66	20	SPACE
3A67	43	C
3A68	4F	O
3A69	4D	M
3A6A	4D	M
3A6B	41	A
3A6C	4E	N
3A6D	44	D
3A6E	56	V
3A6F	45	E
3A70	43	C
3A71	54	T
3A72	4F	O
3A73	52	R
3A74	20	SPACE

3A75	4D	M
3A76	45	E
3A77	4D	M
3A78	4F	O
3A79	52	R
3A7A	59	Y
3A7B	20	SPACE
3A7C	45	E
3A7D	52	R
3A7E	52	R
3A7F	4F	O
3A80	52	R
3A81-8F	XX	NOT USED

HP-1345A GRAPHIC COMMANDS FOR POWER UP  
AND SHUTDOWN MESSAGES

<u>Location</u>	<u>Value</u>	<u>Description</u>
\$3A90	\$0009	Number of commands in list
3A92	C000	Address pointer to \$00 (00)
3A94	8FFF	Internal Jump to \$FFF (4095)
3A96		Address Pointer
3A98	0398	Plot X
3A9A	12E8	Plot Y
3A9C	5945	E
3A9E	5948	H           POWER UP MESSAGE
3AA0	5953	S
3AA2	5949	I
3AA4	000C	Number of commands in list
3AA6	C000	Address pointer to \$00 (00)
3AA8	7818	Set HP-1345A mode
3AAA	0378	Plot X
3AAC	12E8	Plot Y
3AAE	5953	S
3AB0	5948	H
3AB2	5955	U
3AB4	5954	T           EHSI SHUTDOWN MESSAGE
3AB6	5944	D
3AB8	594F	O
3ABA	5957	W
3ABC	594E	N
3ABE	8FFF	Internal Jump to \$FFF (4095)

INTERRUPT VECTORS FOR SERVICE ROUTINES

<u>Location</u>	<u>Value</u>	<u>Description</u>
\$3AC0	XXXXXXXX	Spare Interrupt Request Line
3AC4	000027CA	Clock Interrupt Request Line
3AC8	00002760	Key Interrupt Request Line
3ACC	00002546	Communications Int. Req. Line

(Interrupt Vectors for COMIRQ Service Routines)

\$3AD0	00002590	Interface Command #1 Routine
3AD4	000025EA	Interface Command #2 Routine
3AD8	00002630	Interface Command #3 Routine
3ADC	00002750	Interface Command #4 Routine
3AE0	XXXXXXXX	Spare Interface Command Vector
3AE4	XXXXXXXX	Spare Interface Command Vector
3AE8	XXXXXXXX	Spare Interface Command Vector

\$3AF0 - \$3FFF Available Space

AN ELECTRONIC HORIZONTAL SITUATION INDICATOR  
AND DEVELOPMENT SYSTEM

by

JEFF D. LAGERBERG

B.S., Kansas State University, 1984

---

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the  
requirements for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1987

## ABSTRACT

Within the general-aviation community there exists a need for an affordable avionic instrument that would reduce the workload of a pilot flying under instrument flight conditions. An Electronic Horizontal Situation Indicator (EHSI) is described that would integrate navigational information and flight data to produce graphics on a high-resolution display module. Three graphic pages are suggested that present the flight information in a way that will greatly enhance the pilot's situational awareness of the aircraft's surrounding environment.

To develop the EHSI system and integrated displays it is necessary to have a development system that will simulate, gather and process the flight data required to produce the EHSI graphics. An EHSI Development System is presented which accomplishes the task of gathering and preprocessing the data from the pilot, from navigational radios and from conventional avionic instruments on board a flight simulator. The development system will have knowledge of all pertinent flight data and is able to transfer the data to a host computer via a two-wire handshake parallel port. The development system's host computer then processes the information to produce the graphic pages for the EHSI.