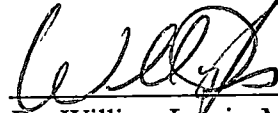


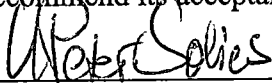
To the Graduate Council:

I am submitting herewith a thesis written by Mark Johnson entitled "Design and Construction of a Display Symbology Tester for General Aviation". I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Aviation Systems.

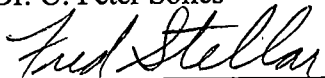


Dr. William Lewis, Major Professor

We have read this thesis and
recommend its acceptance:

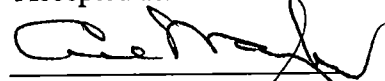


Dr. U. Peter Solies



Mr. Fred Stellar

Accepted for the Council:



Interim Vice Provost and
Dean of the Graduate School

DESIGN AND CONSTRUCTION OF A DISPLAY SYMBOLGY TESTER FOR GENERAL AVIATION

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Mark A. Johnson
May 2001

© Copyright 2001, Mark Allan Johnson

All rights reserved

Abstract

The objective of this research project was to design and construct a system to evaluate the effectiveness of diverse types of flight instrument symbology in transmitting information to the pilot. A structured systems engineering approach was used to select system components given the requirements to minimize cost and modifications to the test aircraft and maximize display flexibility while providing enough information to execute instrument flight tasks.

The system is composed of three major elements: flight data collection, data processing, and the display. Each of these elements was analyzed individually to establish requirements and implementation options. The options were compared using a weighted-array analysis to select the most appropriate solution. The final system configuration combined a three-axis inertial measurement unit from Watson Industries, a Motorola pressure sensor, a Pentium III laptop computer running custom software written in Microsoft Visual Basic, and an Earth Technologies 8.4 inch color liquid crystal display. The software was developed using an iterative process to design, test, and refine instrument appearance and function in a simulated flight environment.

During early testing, significant IMU deficiencies were noted and partially compensated for by the addition of a pressure sensor for direct altitude measurement and a variable velocity input circuit. Despite these enhancements, IMU performance remained poor and resulted in a significant number of aborted test points.

Three display layouts were used to conduct a limited evaluation of the potential of the system to meet the requirements. The first display was designed to closely replicate the standard GA aircraft "instrument-T". The second consisted of a gyroscope the size of

the entire display with digital readouts of other flight parameters superimposed upon it. The third display was designed to look similar to an F-16 heads up display. Three pilots were chosen from diverse backgrounds and were tasked with performing a simulated precision instrument approach while their performance was recorded. Error analysis was then conducted using commercial data analysis and plotting tools. Pilot performance varied widely between displays, with the "instrument-T" display producing the worst average performance.

This proof-of-principle evaluation was successful, but the basic system architecture should be refined one more time before conducting further symbology testing. The most significant recommendation is to replace the IMU with a more reliable data collection system, preferably one based upon the Global Positioning System. Also, before a rigorous symbology evaluation can be conducted, more detail needs to be added to the instrument types already implemented and new instruments, such as a horizontal situation indicator and turn-and-slip instrument, must be added. Then a sponsor should be sought to finance continuing display research projects.

Table of Contents

I. Introduction	1
Background	1
Avionics	2
Human Information Processing	3
Systems Engineering Process	5
II. Requirements Analysis	7
Input Requirements	7
Information Presented	7
Instrument Types	8
Additional Features	12
III. Functional Analysis	14
IV. Design Synthesis.....	16
Component Selection.....	16
Data Collection System.....	16
Control System.....	21
Display System	22
Integration Hardware.....	24
Software Development.....	24
Instruments	25
Control Input.....	29
IMU Communication Interface.....	31
Data Processing.....	32
Design Loop	34
Alignment.....	35
Altitude.....	36
Final System Architecture.....	39
V. System Verification.....	42
Overview	42
Data Analysis	44
VI. Project Summary	48
Conclusions	48
Recommendations	49
Hardware.....	49
Software.....	50
Future Projects	52
References.....	54
Appendices.....	55
Appendix A. List of Abbreviations.....	56
Appendix B. System Detail Photos.....	57
Appendix C. Tested Display Layouts	59
Appendix D. Display Test Plan	63
Appendix E. Sample Data	64
Appendix F.. Program Code	67
Vita	88

List of Tables

Table 1. Data Parameter Accuracy and Precision.....	8
Table 2. Data Collection Component Selection.....	20
Table 3. Data Processing Component Selection.....	22
Table 4. Display Component Selection.....	24
Table 5. Required Software Routines for Instruments.....	27
Table 6. Common Instrument Parameters	27
Table 7. Gauge Instrument Parameters.....	28
Table 8. Gyro Instrument Parameters	28
Table 9. Strip Instrument Parameters.....	29
Table 10. Software Command Functions.....	30
Table 11. IMU Data String	32
Table 12. Data Array.....	33
Table 13. System Cost and Weight	41
Table 14. Pilot Experience	43
Table 15. Average Pilot Deviation.....	46
Table D-1. Test Tasks	63

List of Figures

Figure 1. The Systems Engineering Process	5
Figure 2. Instrument Type Examples	10
Figure 3. System Functional Flow Block Diagram	14
Figure 4. Top-Level Functional Architecture	40
Figure 5. Pressure Sensor and Target Speed Circuit Schematics	41
Figure 6. Physical Architecture.....	41
Figure B-1. Front View of IMU and Circuits	57
Figure B-2. Top View of IMU and Circuits	58
Figure C-1. Basic Display.....	59
Figure C-2. Big Gyro Display.....	61
Figure C-3. Green Display.....	62

I. Introduction

Background

Pilot-Vehicle Interface (PVI) testing is becoming more important every year as demand for easier-to-use systems and displays increases. 88 percent of the general aviation (GA) accidents from 1980 to 1990 were attributed (at least in part) to pilot error, which could be linked to "...poor information transfer and information interfaces..." (FAA/NASA, 1990). Better displays can reduce the number of errors made in the information transfer process, but there is a significant amount of debate on what display format is the most effective for information presentation. This paper presents a simple but powerful system to test different instrument types and arrangements to measure their effectiveness at transmitting information to the pilot.

The first flight instrument used was an altimeter whose function was based upon the barometers of the day (Hawkins, 1987). Although other instruments (airspeed and heading indicators primarily) were developed over the first quarter-century of aviation, it was not until all-weather aircraft were required by the developing airmail system in the United States that full instrument-flight quality cockpits were developed. The instruments and cockpit layouts developed for that purpose did not change significantly until the mid 1970s. At that time, military aircraft experts began to incorporate video displays into cockpit designs to display flight and aircraft data in a variety of formats. Because the software could easily be changed, with the result that the information types and formats presented to the pilot would completely change, a dramatic increase in the amount of human factors testing began to take place.

Commercial aviation soon began to take advantage of the utility provided by multi-function video displays in cockpits, but the cost of these systems has hindered their introduction into general aviation aircraft. Smaller, cheaper systems are required before widespread acceptance. Nevertheless, an improved system for displaying aircraft flight data to the general aviation pilot will become important as the skies above the nation continue to become more crowded and air travelers demand a more efficient airways navigation system.

There has been a significant amount of research into cockpit display design for all aircraft, but especially for military and commercial aircraft. The high cost of accidents to these customers makes their significant investments in human factors research a necessity. The benefits that have been realized in those areas of the aviation industry can also be realized for general aviation, but only with a low-cost test system like the one described here.

Avionics

The aviation accident rate is dependent upon three "pillars": avionics, ergonomics, and operational experience (Hawkins, 1987). Improvements in any of these three pillars will advance aviation safety.

In the general aviation arena, there is little that can be done to improve the overall amount of operational experience since new pilots are entering the pool daily. The FAA is working hard to improve both initial and ongoing training, but there is significant resistance to raising requirements for pilot licenses. Therefore, it is up to aircraft design experts to improve the avionics and ergonomics of GA aircraft in order to improve safety.

The incredible advances in computer technology in the past 20 years have produced impressive improvements in avionics. Laptop computers are small, inexpensive, and powerful enough to coordinate inputs from a sophisticated data-collection system and produce high-resolution video output. A high-quality, color liquid crystal display (LCD) that is thin and lightweight enough for aircraft use and can easily be viewed at wide angles in full daylight conditions can now be purchased for under \$1000. Avionics improvements are not a limiting factor to improving cockpit displays.

Ergonomics is another matter. Because of the large amount of data to be presented, and numerous ways to present it, research must be conducted on which display formats are the most effective at getting information to the pilot. Perhaps no one format will prove to be the best in all cases, and displays that adapt their layout to the particular regime of flight or at the pilot's request will be the most effective.

Human Information Processing

When controlling an aircraft, the pilot is constantly checking his instruments and making actions based on the information obtained from those instruments. This is known as information processing. Several things can go wrong with this process, and an understanding of the basic principles of information processing is necessary to understand how display configuration can help reduce errors.

Whereas a "display" can refer to either a visual, auditory, or tactual presentation to a person, in the context of this paper it will be used to refer to visual displays only. Visual displays are the most complex of the three types and can be configured to carry almost any type of information. It is precisely this power and flexibility that make research into their configuration so important. Display format, layout, and configuration

will be used interchangeably to refer to the type, position, and style (color, font, shape) of instruments displayed.

Human information processing consists of several steps. In the case of visual displays, the first step is visually sensing the information (Hawkins, 1987). This is the act of seeing the data—making sure that it is presented within the pilots' field of regard, not obscured and not too low contrast to discern. The next step is perception or "pattern recognition" (Wiener, 1988), which is probably the most important stage of information processing. At this stage, symbols, numbers and words observed by the eye are converted into the ideas that have meaning to the operator.

These first two stages are followed by decision-making, response execution, and feedback. The display improvements that were investigated with the system in this paper focus primarily on the first two steps. The goal is to find the display that most reduces both the effort required and the number of errors made in sensing and perception. The system described in this paper allows researchers to experiment with different displays in search of the display that is most effective at converting raw information into meaning for the pilot.

A commonly used model of the complex interactions involved in aviation is the "SHEL" model developed by Edwards (1972). The four parts of the model are "Software", which consists of the written instructions and procedures for flying; the "Hardware" such as the displays and controls in the cockpit; the "Environment" in which flights are conducted; and the "Liveware", or the pilot. The most common failures in aviation happen at the junctions between these elements, and occasionally result in

disaster (Wiener, 1988). The system presented here will help the Hardware-Liveware interface become more efficient and less prone to errors.

Systems Engineering Process

The Systems Engineering Process (SEP) is a comprehensive, iterative and recursive problem solving process (Leonard, 1999). It provides a structured technique to transform requirements into functional and physical architectures. This process is diagrammed in Figure 1.

The process begins with an input that can either be the initial requirements for a new system, or the output of a previous iteration through the SEP. A requirements analysis is then conducted to determine what functions the project must fulfill and what bounds must be placed on the physical architecture (for instance: cost, weight or complexity).

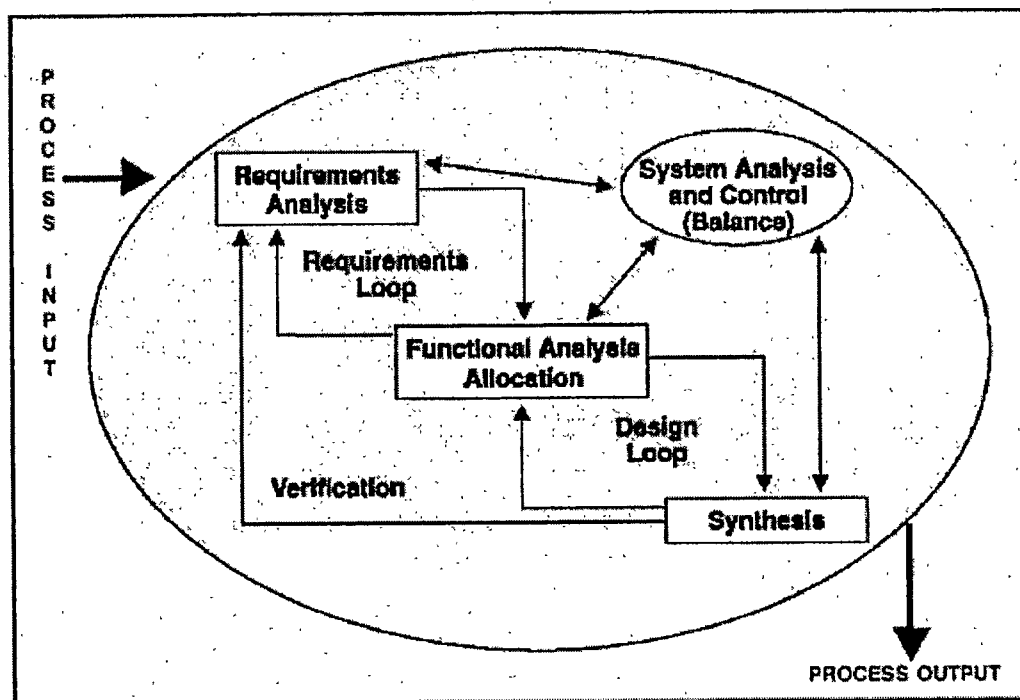


Figure 1. The Systems Engineering Process

These requirements are then used to determine a functional architecture. This begins as a high-level depiction of the basic steps to be performed by the system, and iterative loops are completed to break the high-level functions down into ever lower-level steps. These lower-level functions may then have requirements of their own and may produce a better understanding of the overall requirements, forcing a change to some of the previously established requirements. This SEP loop is called the requirements loop.

The next stage is Design Synthesis, where the functional architecture blocks are assigned to a physical architecture. In choosing particular components, the design loop ensures that each physical component of the system fulfills a specific functional block. If the component does not meet a functional block, and therefore a system requirement, it is eliminated.

The final step involves a system verification to ensure that the final system meets the input requirements as modified by the requirements analysis. The output can either be a complete system for the customer or may serve as input information for another loop through the entire SEP. In this manner a design can be continuously improved to meet changing requirements.

The next four chapters will cover how the steps of the SEP were applied to this project to create a finished product.

II. Requirements Analysis

Input Requirements

The first step of the SEP is an analysis of the system requirements. There was no defined customer for the project to set these requirements, but the idea had been discussed in several forums and some basic concepts established. The design philosophy could be summed up in a few fundamental requirements.

1. The system should provide enough information to effectively maneuver an aircraft under instrument flight conditions.
2. System cost should be as low as possible given the constraints of the first requirement.
3. The system should have as little impact on the test aircraft as possible, avoiding expensive modifications (STCs).
4. The system should be very adaptable and easy-to-use.

These principles established a starting point for planning the software portion of the project and choosing the hardware components that would be used. The next step was to determine what information was necessary to meet the first requirement, what additional information was desirable, and refine the system requirements definition.

Information Presented

To conduct successful instrument flight maneuvers, the pilot must have: aircraft attitude (pitch and roll), heading, airspeed, and altitude. Each of these parameters must have certain accuracy and precision, as listed in Table 1. The accuracy of the information must last for the length of time for one test maneuver. The accuracy may degrade beyond these limits after the test point is complete, as long as the parameters can be re-calibrated

Table 1. Data Parameter Accuracy and Precision

Parameter	Accuracy / Precision		
	Desired	Required	Test System
Attitude	0.5 deg / 0.2 deg	1.0 deg / 1.0 deg	0.2 deg / 0.03 deg
Heading	2.0 deg / 0.5 deg	30.0 deg / 2.0 deg	20.0 deg / 0.03 deg
Airspeed	2 knots / 0.5 knots	10 knots / 2 knots	*See note
Altitude	100 feet / 5 feet	2000 feet / 10 feet	60 feet / 10 feet
Vertical velocity	10 fps / 5 fps	100 fps / 20 fps	100 fps / 10 fps
Turn rate	0.3 deg/sec /	N/A	N/A
Course/GS dev	0.01 deg / 0.01 deg	0.1 deg / 0.1 deg	0.01 deg / 0.01 deg
Angle of attack	0.2 deg / 0.1 deg	N/A	N/A
Vertical accel	0.1 g / 0.01 g	N/A	0.02 g / 0.00025 g
Sideslip	0.2 deg / 0.1 deg	N/A	N/A
Long accel	0.5 KPS / 0.01 KPS	N/A	0.38 KPS / 0.005 KPS

Shaded cells represent required data. Other cells are additional optional data.

* Whereas the system was capable of presenting a calculated airspeed to the pilot, all tests were conducted using a fixed airspeed reference to preclude the fine-tuning required of the longitudinal acceleration integration function.

before the next test point. For the purpose of this project, five minutes was originally considered the minimum length of time to complete one test maneuver.

In addition to the required data, other information is usually presented to pilots. These are also listed in Table 1, with desired and required accuracy and precision values for each parameter. The final column shows the accuracy and precision of the parameters measured by the system used in this project.

Instrument Types

Continuing to refine the system requirements, the different ways in which the necessary information could be presented were considered. For this task, it was important to focus on the three different ways that flight instruments are used (Sanders, 1993).

1. Quantitative assessments. This is when the pilot needs to know the exact numerical value of a particular flight parameter. Examples include precise reading of altitude when leveling at an assigned flight level or airspeed when on final approach for landing.
2. Qualitative assessments. These are cases where the pilot just needs to know general trends or relative values. Examples include pilots' general sensation of the rate of change of altimeter on final approach and lack of change when on a level enroute leg.
3. Situational awareness. These instruments contribute to the pilots' knowledge of the plane's position in space. The "attitude gyro" is the most common example of this type.

Different instrument displays are used in different ways depending on the flight regime. Analysis of the required data and the ways in which that data would be used led to the conclusion that the display testing system needed to include the following different instrument types:

Attitude Gyro. A reliable attitude display is arguably the most important instrument in the cockpit. A "typical" gyroscope display indicates the instantaneous aircraft pitch and roll state. A normal ball-type gyroscope has blue to indicate sky and brown to indicate the ground with a pitch scale on the face. Other types may include a "pitch-ladder" type display as shown on military aircraft HUDs. There is usually a symbol in the center of the instrument to represent the aircraft and graduation marks around the perimeter to indicate specific roll angles. Additionally, course and glideslope deviation indicators are often placed on this instrument to indicate error from a desired

position when conducting a precision instrument approach. Two examples of this type of instrument are depicted in the upper left of Figure 2.

Gauges. These are circular displays with a pointer to indicate the current value. The two subtypes are “continuous” gauges that wrap-around to indicate higher values, and “non-continuous” gauges that have a starting angle for a minimum value and ending angle for a maximum value (i.e. the gauges “peg-out” at the minimum, maximum, or both). Gauges of either subtype may include a text display of the value on the face of the instrument as well. A scale consisting of graduations at certain intervals is usually included to help with interpretation of the value, sometimes including “major” graduations at large intervals and “minor” graduations at sub-intervals. The gauge may include special zones delineated by colored bands or highlighted numbers to indicate values of particular interest. Two examples of gauges are depicted in the center left side of Figure 2.

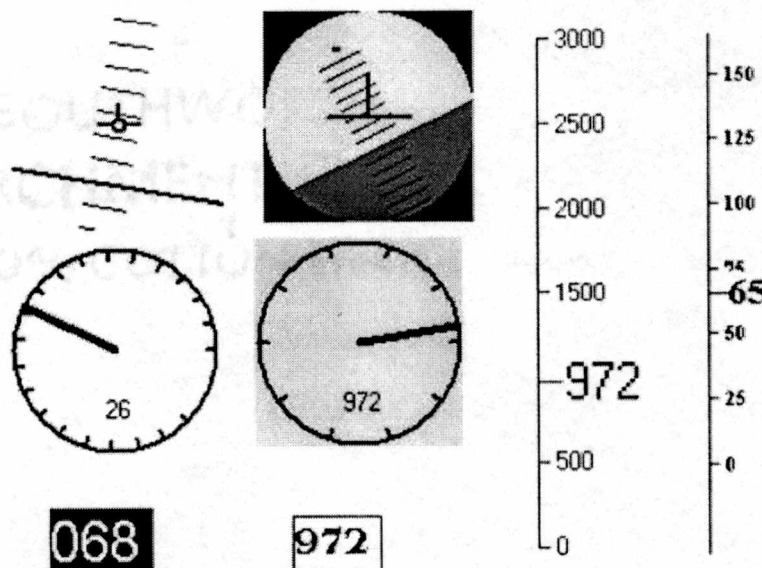


Figure 2. Instrument Type Examples

Text. Plain text digital displays may be used to present data. While this data may be very precise, as no interpretation of pointer position or interpolation is required, it may be difficult to detect trends in changing values. For this reason, text displays are often combined with other types of displays. Text displays should have the ability to take on a variety of fonts, sizes, and colors and may include features to enhance their contrast, such as a box around the number or a different colored background. Two examples of text instruments are shown on the bottom left of Figure 2.

Strips. These are linear scales with a pointer that indicates the current value. The two subtypes include moving-pointer/fixed-scale and moving-scale/fixed-pointer. The moving-pointer type displays the entire scale from minimum to maximum value at once and positions the pointer along the scale at the appropriate position. The moving-scale type displays only a portion of the full scale that "slides" back and forth around the non-moving pointer. Strip instruments may be either vertically or horizontally oriented. A digital text display of the actual value is often included adjacent to the pointer. Similar to gauges, they may contain special zones to indicate high or low values that are emphasized via special markings (like color). Two strip gauges (one of each subtype—the moving-pointer on the left) are depicted on the right side of Figure 2.

Horizontal Situation Indicator. The HSI is a special type of instrument used in navigation. It is primarily a situational awareness display and is used quantitatively to control aircraft heading. An HSI consists of a round "compass card" that has a heading scale marked out on the perimeter. At one position (usually top center) there is a pointer to indicate current aircraft heading. Around the outside of the moving compass card

there may also be additional pointers to indicate direction to a waypoint or radial from a navigation aid.

Turn and Slip Indicator: The turn and slip indicator contains a vertical needle that deflects right or left to indicate a turn in that direction. The amount of deflection indicates the rate of turn. The slip indicator consists of a ball that moves from side-to-side in a race to indicate an out of balanced flight condition (sideslip).

Many other specialized instrument types, such as three dimensional flight path displays, course deviation indicators, angle-of-attack indicators, and total energy cues, could be implemented. The adaptability of the hardware is such that only software changes would be required to add these new instrument types.

Additional Features

In addition to the different types of instruments, several other software features were considered critical to building a complete system. The first requirement was for a "simulation" mode that allowed testing displays on the ground. Using this feature, many potentially unacceptable display configurations could be rejected before spending valuable flight time. The most basic requirement was that the user needed command of pitch, roll, and airspeed adequate for executing basic instrument maneuvers. A control input method similar to that used in the aircraft would be optimal, but would not be required.

Additionally, a recording mode was deemed necessary to capture all instrument readings for subsequent analysis. These readings needed to be stored in a permanent medium that allowed conversion to a variety of formats for later analysis. The recording mode also needed to be able to capture data when the system is in simulation mode.

The last required feature was the ability to place a simulated runway with a precision approach anywhere in space, in order to practice instrument approaches without requiring an actual airfield. The field altitude, approach heading, and glideslope angle all needed to be variable to provide flexibility in planning test missions. Deviation indicators had to be included to provide the pilot a way to evaluate error and make a correction.

III. Functional Analysis

The next SEP step is to transform the identified requirements into a coherent description of system functions that can be used to guide the design synthesis activity to follow (Leonard, 1999). First, a high-level block diagram of system functions is drawn from the basic requirements. Then it is successively decomposed to ever lower level functions and those functions are analyzed to decide if the system requirements definition needs to be added to or modified.

The Functional Flow Block Diagram (FFBD) is a tool used to depict the system functions and their relationships. At the top of Figure 3 is the top-level FFBD for this project and immediately below that is the second level diagram.

The first level shows the system input (Aircraft State) and output (Pilot Interpretation) blocks in addition to the actual system functions of Data Collection, Data Processing, and Information Display.

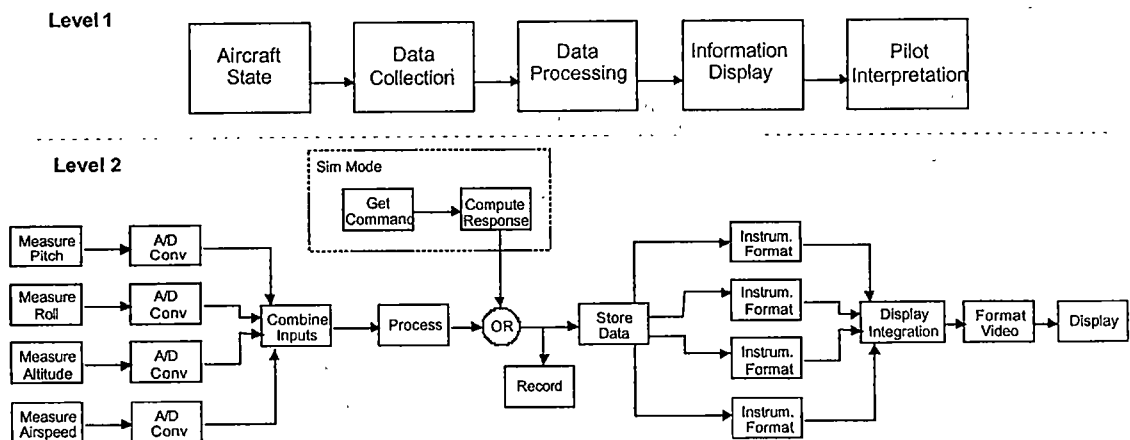


Figure 3. System Functional Flow Block Diagram

The second level just focuses on the system, decomposing Data Collection into the parameter measurement, analog to digital conversion, and data integration. The Data Processing block incorporates the lower-level functions of data processing (that can be further reduced to signal decoding, scale conversion, and result computation), recording, and storage as well as the functions of the simulation mode (getting a command and computing a response). The Information Display block can be devolved into instrument formatting, display integration, video signal formatting, and the actual display of information.

A few of the blocks can be further reduced, but only the Data Processing block has a significant amount of sub-parts. These blocks consist of the actual software subroutines that perform the steps shown. Appendix F contains the software code that is the third level FFBD for Data Processing.

IV. Design Synthesis

Component Selection

Once the initial requirements for the system (information, instruments, and additional features) were established, the process of selecting hardware began. Hardware solutions were sought for each of the top-level FFBD blocks (Data collection, Data processing and Information display). The options were weighed against each other using several measures of effectiveness (MOE) determined from the project requirements.

For each MOE, each option was ranked on a weighted scale with above average ratings being positive and below average ratings negative. The scores were added and the option with the highest score was the component chosen.

Data Collection System

Several possible alternatives were examined to collect the required information shown in Table 1 and comply with requirement 2 (minimize system cost).

It was immediately apparent that getting aircraft pitch and roll information was going to be the biggest problem. GA aircraft use vertical gyros to measure pitch and roll and display this information on the same mechanical instrument. However, there are no provisions for an electrical measurement signal from this information. Therefore, different options had to be investigated.

Litton makes an outstanding inertial navigation system using ring-laser gyroscopes designated the ASN-139. It is capable of providing information well under the desired tolerances listed in Table 1, but costs over \$100,000.

Systron-Donner manufactures a low-cost single-axis gyro known as the Gyrochip II. They are based upon vibrating quartz tuning fork technology to measure angular

acceleration and are available for approximately \$600 each. Three of these units would have to be mounted orthogonally in a single enclosure to measure pitch, roll, and yaw acceleration.

The analog output from the rate gyros would have to be converted to a digital signal before being used by the computer. A PCMCIA card for sale from Omega Engineering, the DAQP-12A, is capable of this A-D conversion. It can handle $\pm 5V$ DC sampling at rates up to 100KHz with 12-bit resolution and is approximately \$600.

A magnetic measurement unit would also be required to obtain magnetic heading information for the system. A Watson Industries unit, the MMU-310 could be used and the analog output could be converted by digital by the DAQP-12A card. The MU-310 cost \$800.

Although the total cost for this arrangement would be quite low (approximately \$3300), information about the long-term performance of the Systron-Donner gyros showed that the 1s error increases at approximately 2.5 degrees per minute (Hayward, 1997). The error would increase beyond the acceptable limits shown in Table 1 in 30 seconds, well below the minimum determined time of 5 minutes.

Fiber Optic Gyros are more precise than tuning fork based systems, and a FOG from KVH Industries, the E-Core 1000, was investigated. The E-Core 1000 is a single-axis gyro like the Gyrochip II, so would require 3 gyros at approximately \$800 each, but there is a digital output on these gyros so an A-D converter would not be required. A closer look at the specifications showed that the performance would still be below that specified in Table 1.

GPS-based attitude systems were investigated. Trimble manufactures a system called the TANS Vector that uses a four-antenna array to detect the GPS signals and calculate pitch, roll, and yaw angles, and provides GPS position and velocity information. The error in this system was not subject to gross deviation over time, but would require mounting antennas to the aircraft that would reduce system portability. Price was also a significant degrader, as the system cost approximately \$18,000.

The most promising GPS-based attitude system is the one described by Roger Hayward (1997) in collaboration with Seagull Technology. This system uses the Systron-Donner Horizon rate gyros, corrected via a signal from a GPS attitude system. The GPS portion uses a very short baseline antenna arrangement (36 cm x 50 cm), and estimated price was approximately \$12,000. Unfortunately, this was a developmental system and not available for use.

Complete Attitude and Heading Reference Systems using FOG or tuning fork gyros from Crossbow Technologies (FOG-AUTO), Systron-Donner (MotionPak), and Watson Industries (IMU-E604) were investigated next. The advantages of these systems are many:

- Rate gyros for all three axis
- Output of acceleration along each axis
- Internal hardware to perform basic error correction
- Conversion of rates to absolute pitch and roll angles
- Internal magnetic sensors for output of magnetic heading
- Digital output in RS-232 format for direct input to the control system
- Conversion from body axis coordinates to local level coordinates

All these systems had similar specifications for random angle walk and maximum rates and accelerations that were within the bounds set by Table 1. Unfortunately, they are also fairly expensive, between \$12,000 and \$14,000 each.

The options were summarized as: Build IMU (an IMU built from scratch using single-axis accelerometers like the Systron-Donner GyroChip II or KVH FOG and integration hardware), Civilian IMU (a complete IMU solution like the Watson E604 or Crossbow FOG-AUTO), Military IMU (a high-level solution like the Litton ASN-139 Ring Laser Gyro), GPS attitude (a GPS-based attitude determination system like the Trimble TANS Vector).

The five MOEs are: Data (includes amount of data and accuracy of data from the component), complexity (how much work is required to construct the component), intrusiveness (how much modification is required to the aircraft to accommodate the component), cost and availability. Each MOE is weighted as follows.

Data – Data adequate as defined in Table 1 is zero. Significantly better is +1, significantly worse is -1. If the system can also supply desired data (in addition to that required) then it earns an additional +1.

Complexity – If the system comes complete, with little interfacing or wiring involved, it earns +1. If it requires significant effort to construct and/or calibrate, then it earns -1.

Intrusiveness – If the system requires an aircraft STC, then it earns a -1. If slight modifications are needed (no STC involved) then it earns a 0, and if no modifications are required to the aircraft, it earns +1.

Cost – Each \$3,000 increment in cost is -1, with \$12,000 defined as zero.

Availability – Ease of acquiring the components required. +1 if the components are widely available, -1 if the components are hard to acquire (regardless of cost).

The weighted selection matrix is shown in Table 2:

The Civilian IMU solution scored the highest and a Watson Industries IMU-E604 was procured. In addition to the benefits mentioned previously, the IMU-E604 had inputs for four user channels that it would convert to digital and incorporate into the digital data stream. This would prove useful for inputs from pitot and static pressure sensors. Finally, Mr. Bill Watson was kind enough to loan one to this researcher for the flight test.

The IMU-E604 is a complete strap-down gyro system for aircraft operations with output of absolute pitch and roll angles, as well as linear acceleration in each axis and angular acceleration around each axis. The output is digital via an RS-232C serial interface. The yaw axis is slaved to an internal magnetic sensor for output of magnetic heading (Watson, 2000).

This unit provides direct output of attitude and heading, and the horizontal acceleration output could be integrated to get velocity. The output of vertical acceleration could be integrated once to get vertical velocity and a second time to get altitude. This means all the required information could come from one unit, greatly simplifying the system design.

Table 2. Data Collection Component Selection

	Data	Complex.	Intrus.	Cost	Avail	Total
Build IMU	-1	-1	+1	+2	0	+1
Civilian IMU	0	+1	+1	0	+1	+3
Military IMU	+2	+1	+1	-22	-1	-19
GPS attitude	+2	-1	0	-1	+1	+1

Test accuracy and precision are shown in Table 1. Note that the accuracy and precision of the computed parameters (Airspeed, vertical velocity, altitude) are mathematically derived from system performance under good alignment conditions and showed significant deviation in practice.

This component was by far the most expensive part of the project at just over \$13,000. While this is prohibitively expensive for most GA aircraft, it was considered acceptable for a developmental test system.

Control System

Once the data collection system was selected, the data processing system to interface to this data system was chosen. Based on the selection of the IMU, the data processing system had to be capable of input and output using a serial RS-232 port, converting that information into the data required by the pilot, formatting the display instruments, and supplying the appropriate signal to the display system.

The three options investigated were: Custom (a custom-built computer from specialized components), PC (a Windows-based laptop computer), Mac (a Macintosh compatible laptop computer). The MOEs were: Flexibility (which includes the number of software development tools for that option), Complexity and Cost. The MOEs were weighted as follows:

Flexibility - +1 if there were numerous tools available for software development, -1 if everything had to be done "from scratch".

Complexity - -1 if hardware construction was required, +1 if not.

Cost - Each \$500 increment in cost is +1, with \$2,500 defined as zero.

The weighted decision matrix is shown in Table 3. Both the PC and Mac laptops would be acceptable choices, with the PC preferred based on the Flexibility MOE.

A PC compatible laptop computer with a serial port and VGA display port was procured. PC compatible laptops are widely available and can be used for many other projects, amortizing the cost. Additionally, there are many software development tools available for the Windows operating systems that expedited project software development.

The computer used for this project was a 700 MHz Pentium III with 128 MB of RAM, although the minimum required is assessed to be a Pentium 90 with 16 MB RAM. The software is compatible with either Windows 95/98/ME, Windows NT 4.0, or Windows 2000 and is described in greater detail in the section titled Software Development.

Display System

The display needed to be large enough to display multiple instruments at a significant enough size for easy viewing in the cockpit, but could not be so large and heavy that it was difficult to mount in the cockpit. The display had to have high enough resolution so that the instruments could be rendered with enough fidelity to prevent significant distraction to the pilot. There was also a requirement to have a fairly wide viewing angle (for easy viewing by the co-pilot or test conductor), high brightness (for viewing in direct sunlight), and low power requirements (for use in GA aircraft).

Table 3. Data Processing Component Selection

	Flex.	Complex.	Cost	Total
Custom	-1	-1	+2	0
PC	+1	+1	+1	+3
Mac	0	+1	0	+1

Cathode ray tube displays have great brightness and viewing angles, and are fairly cheap, but have moderate power requirements and are generally too large for easy mounting in the cockpit. Plasma emission displays are small and bright with low power requirements, but are very expensive. Liquid Crystal Displays are generally small and light, and available with fairly bright backlights that have modest power requirements.

Therefore, the matrix options were: CRT (Cathode Ray Tube display), PE (Plasma Emission display), LCD (Liquid Crystal Display). The MOEs were Fidelity (includes resolution, color, and brightness), Power (low power preferred), Intrusiveness, and Cost. The MOEs were weighted as follows.

Fidelity – Excellent resolution, color, and brightness earns +1, adequate performance is 0.

Power – Each 1 amp increment in power draw is -1, with 2 Amps defined as zero.

Intrusiveness - If the system requires an aircraft STC, then it earns a -1. If slight modifications are needed (no STC involved) then it earns a 0, and if no modifications are required to the aircraft, it earns +1.

Cost – Each \$200 increment in cost is -1, with \$800 defined as zero.

Table 4 shows the display type selection matrix where the LCD is clearly favored option. The Earth Computer Technologies MTR-MBL-H8 Liquid Crystal Display (LCD) monitor was selected based on the component comparison. This is a color LCD measuring 8.4" diagonally with a high-brightness (500 NIT) backlamp for daytime viewing. The display resolution is 640x400 pixel with a 16-bit color depth (65,536 colors). It was controlled via a VGA signals from the laptop through an EarthVision VGA TFT LCD Controller (LQ9D03B) and mounted in front of the pilot. The display

Table 4. Display Component Selection

	Fidelity	Power	Intrus.	Cost	Total
CRT	+1	-2	0	+1	0
PE	+1	-1	+1	-1	0
LCD	+1	0	+1	0	+2

with controller and case weighed approximately 1.3 lbs and costs \$895 (Earth Tech, 1997).

Integration Hardware

Linking these three major components was relatively simple. The IMU was already cross-wired internally (normal RS-232C inputs were on output pins and vice-versa) so a null-modem adapter was not required to connect to the computer serial port. The display controller connected directly to the laptop monitor output with a standard monitor cable and interfaced directly with the LCD on the other side using a custom cable supplied by Earth Tech with the display.

Power and ground were supplied to the IMU and LCD display from a standard 12V DC car-cigarette lighter connector. For testing, a 115V AC to 12V DC adapter was used to plug the units into a standard US electrical outlet. Maximum power draw was a moderate 2.6 Amps, 2 Amps of which was required for the high-brightness lamp in the LCD display. The computer was powered by the internal batteries, which were sufficient for 4 hours of operation.

Software Development

Given these established design objectives and hardware capabilities, project software development was initiated. A PC compatible computer running Microsoft Windows had been selected as the data processing system, so a software development language for that configuration had to be selected. The software would only do a

minimal amount of computations and would not require exceptionally high speed. Most of the development effort would involve the graphical interfaces of the various instruments.

Microsoft Visual C++ was considered initially and rejected because the graphical programming features included with the package are less developed than those in other packages. Microsoft Visual Basic was selected because of its ease-of-use, rapid prototyping features, and object-oriented basis. There are numerous features to ease the development of graphic interfaces and an ActiveX control included to reduce the development effort required for serial port communication.

In order to avoid loading the computer too much when complicated display formats are designed, the display update rate was desired to be as low as acceptable to the pilots. Initially 20 Hz was tried, but testing showed a significant amount of flicker in complicated instruments (primarily gyros) that was minimized at slower update rates. Eventually, 5 Hz was determined to be an acceptably fast rate with minimum flicker. An attempt was made to completely eliminate the flicker, but was eventually abandoned as being too complicated for this stage of development.

Instruments

The instruments were created as separate user controls so that an unlimited number of any type could be created on any display and could each have their own individual parameters. This also made the debugging process simpler.

As part of the design loop, several compromises were made in selecting which instruments to implement to prevent the software portion of the project from becoming too complex and time-consuming. The first four instrument types (gyro, gauge, text, and

strip) were required for proof-of-concept, but the other instruments were not implemented. Additional compromises included:

- Gyro gauges do not include bank pointers or scales
- Only vertical strip instruments were implemented, not horizontal
- Gauges were simple, with only major graduations and simple line pointers
- Gauges and strips did not include "highlighted zones" or "red-lines"

These features can be implemented and other instrument types added in a later version of the project.

The gauge instruments were created first and served as the model for the design of the other instruments. In an object-oriented paradigm, each instrument contains the information regarding the data it should present and how it should appear. Features that each instrument needed to implement were the ability to turn this information about appearance and data into a graphical representation. Each instrument also needed to be able to store this data in a persistent medium and reload the information when required. In order to protect this information from possible corruption, the instruments needed interface routines so that the main program could set different parameters and get the values of those parameters when required. This meant that all instruments had a common set of routines to implement and these routines are listed in Table 5.

The parameters that controlled the instrument appearance varied depending on instrument type, but there was a basic set of parameters required for all instruments. These included the position, size, foreground and background colors, and what flight information this instrument was to display. These parameters are listed and described in Table 6.

Table 5. Required Software Routines for Instruments

Routine	Function
Initialize	Set default values for instrument parameters
Paint	Draw the instrument using the current parameters
MouseDown, MouseMove, MouseUp	Used in concert to re-position instruments as desired by user
GetType	Return the type of instrument when queried
Get/Set <i>Parameter</i>	Get or set the instrument parameter specified (color, font, pen width, etc.)
Update	Update the instrument value and re-draw
Save	Save the instrument parameters to persistent storage
Load	Load the instrument parameters from persistent storage

Table 6. Common Instrument Parameters

Parameter	Description
Left, Top	X, Y coordinates of instrument left top corner
Width, Height	The width and height of the instrument
Fore/Back Color	The foreground and background colors of the instrument
Line Width	The width of lines used to draw the instrument
Data Field	The index of the field in the data array that contains the data for this instrument

After the parameters common to all instruments were determined, the parameters specific to each type were added. Those parameters specific to gauge instruments are shown in Table 7. The “minV” and “maxV” parameters work in concert with the “stAng” and “endAng” parameters to allow the gauge to start and end anywhere around the face of the gauge with arbitrary values for start and end amounts. The “Continuous” parameter controls whether the gauge “pegs out” at the min and max, or continues to wrap around the gauge face (like the hundreds pointer on an altimeter).

Parameters specific to gyro instruments are shown in Table 8. The different aircraft symbols that are available include the “standard” inverted-T shape, the “W”

Table 7. Gauge Instrument Parameters

Parameter	Description
minV, maxV	The minimum and maximum values on the gauge
stAng, ending	The starting angle on the instrument face (where the minimum value is indicated) and the ending angle (maximum value)
TickInt	The interval between tick marks on the gauge face
PointerWidth	The width of the pointer
Continuous	A boolean value that indicates whether the gauge “wraps around” or stops at the minimum and maximum values.

Table 8. Gyro Instrument Parameters

Parameter	Description
nSkyColor, nGroundColor, nLineColor	The color of the gyro face for the Sky hemisphere, Ground hemisphere, and lines indicating pitch elevation
nSymbol	The type of aircraft symbol on the gyro face
nSymSize	The size of the aircraft symbol on the gyro face
nPitchScale	The relative scale of the pitch ladder
ILS	A Boolean value that determines whether ILS deviation needles are shown on the face of the instrument
Square	A Boolean value that determines whether the gyro appears square or round
nDataPitch	The index of the field in the data array that contains the data for pitch angle
nDataRoll	The index of the field in the data array that contains the data for roll angle

waterline symbol, and a “velocity-vector” style symbol. Examples of each of these are shown in the displays in Appendix B.

Other gyro parameters include the “nSymSize” parameter, which controls the percentage size of the symbol relative to the entire size of the gyro. The “nPitchScale” controls the spacing between the pitch ladder lines (and therefore how precisely a particular pitch attitude can be held). 1 means the entire pitch scale (+90 to -90) fits from

the top to the bottom of the instrument, 2 means only half the entire scale will fit on the instrument, and so on.

The only parameters specific to text instruments is the “nMinDigits” parameter, which controls the minimum number of digits to display. If the value to be shown has fewer digits, then leading zeros are added to the value until it meets the minimum requirement.

Parameters specific to strip instruments are shown in Table 9. Similar to the gauge instrument, the strip has minimum and maximums and increments between graduations on the face to make it easier to interpolate values similar to the gauge instruments. However, it also has the additional “ScaleRange” value. If this is greater than the interval from the “minV” to the “maxV”, then the entire scale is shown as a moving-pointer/fixed-scale type. If it is less than the full-scale value, then the strip type is moving-scale/fixed-pointer.

All testing of the instruments was done using the simulation mode of the software and controlled as described in the next section.

Control Input

The data processing system is controlled via a few basic keyboard inputs and mouse movements. The program “Hot Keys” are shown in Table 10.

Table 9. Strip Instrument Parameters

Parameter	Description
minV, maxV	The minimum and maximum values shown on the instrument
Increment	The interval between graduations
ScaleRange	The amount of the scale to actually display at once
ShowPointer	Boolean value that indicates whether to display the pointer and digital value or not

Table 10. Software Command Functions

Key	Function
D	Show the latest <u>D</u> ata from the serial comm port
E	Show the last <u>E</u> rror information from the serial comm port
I	Send commands to <u>I</u> nitalize the IMU
R	Toggle <u>R</u> ecorder mode on or off
S	Toggle <u>S</u> imulation mode on or off
Z	<u>Z</u> ero out latitude and longitude error to begin practice approach
Up/Down	Add or subtract 2 knots of airspeed in simulation mode

The “D”, “E”, and “I” commands are described in the next section on IMU communication. The “R” key turns on recording mode. When the recorder mode is on, aircraft pitch, roll, heading, airspeed, altitude, vertical speed, latitude, longitude, course deviation and glideslope deviation are all recorded to a file on the computer hard disk at a 5 Hz rate. A header line is included that shows the date and time of the test and time-tags each line of data. The simulation mode can also be recorded. A sample portion of a data file is shown in Appendix E.

The “S” key controls whether the software is in sim mode or is “live”, meaning that it is displaying information from the IMU. In the simulation mode, a full six-degree of freedom flight model to replicate aircraft characteristics would have provided the most accurate simulation, but project time constraints led to the decision to implement a more modest solution. Aircraft characteristics such as the spiral and Dutch roll modes and the phugoid were determined to be unnecessary to simulate. Due to project cost constraints, the input methods available using the laptop computer selected were used for the simulation mode.

In the simulation mode, the sideways movement of the mouse controls roll and fore-aft movement controls pitch (forward for nose down pitch). The arrow keys control aircraft airspeed directly (not power setting), and gravity is simulated with a very basic model (airspeed decays when nose-up and builds when nose-down). This solution provided enough fidelity to practice the basic instrument maneuvers and simulated approaches shown in the test matrix in Appendix D.

The "Z" key re-sets the practice airfield to one nautical mile north and seven nautical miles east of the current position to start a new practice approach. Once again, to control project software complexity, the final approach heading was fixed at 090 degrees magnetic, although glideslope angle and field elevation were variable. Another compromise was to show course and glideslope deviation with two needles on the gyro display (a vertical needle to show course deviation and a horizontal needle to show glideslope deviation). The needles did not function as "flight director" needles that command a certain aircraft attitude to eliminate error, although that functionality could be added later. Additional styles of deviation indicators (error pointers on the sides of the display or pitch and bank command bars) could also be implemented later.

When the code for the instruments, recorder, and simulator was sufficiently mature, concentration was shifted toward developing the IMU communication interface.

IMU Communication Interface

The communication with the IMU was conducted using the MSComm Visual Basic common control. This is a pre-packaged ActiveX control for Visual Basic that includes the functions for sending and receiving data from a serial interface. The output

from the computer consisted of commands to initialize the IMU. The input consisted of the text string containing the IMU data.

The data string is an ASCII string of 9 parameters separated by spaces. The data rate from the IMU is approximately 20 Hz, although the program only takes data at a 5 Hz rate. A sample string is shown in Table 11 with a description of the data contained therein. The first 6 parameters are described thoroughly in the table and the last three are explained in the section on system debugging and troubleshooting.

Data Processing

This data stream had to be processed and placed into an array for access by the individual instruments. This array consisted of the information that a pilot would require to perform the assigned tasks and fields to hold aircraft position and deviations. The fields of the array are described in Table 12.

Table 11: IMU Data String

I -002.7 +14.1 285.5 -0.00 -1.03 +3.72 +4.69 +188.8		
Param	Value	Description
1	I	IMU data quality- Capital I = good data, Lowercase i = poor data
2	-002.7	IMU roll data- -179.9 to +180.0 degrees (positive values indicate right wing down)
3	+14.1	IMU pitch data- -90.0 to +90.0 degrees (positive values indicate nose up)
4	285.5	IMU heading data - 000.0 to 359.9 degrees magnetic
5	-0.00	IMU horizontal acceleration - -2.00 to +2.00 gs (negative values indicate deceleration)
6	-1.03	IMU vertical acceleration - -2.00 to +2.00 gs (negative values indicate upward accel, therefore, "unaccelerated" straight-and-level flight is -1.00)
7	+3.72	Voltage of data channel 1 - 0.00 to +9.99 (This is the output of the atmospheric pressure sensor for altitude)
8	+4.69	Voltage of data channel 2 - 0.00 to +9.99 (This is the output of the velocity calibration circuit)
9	+188.8	Velocity of IMU - 000.0 to 399.9 kilometers per hour from cal circuit

Table 12. Data Array

Field	Value	Range
0	Pitch (degrees)	-90.0 to +90.0
1	Roll (degrees)	-180.0 to +180.0
2	Yaw (degrees) – Not currently used	N/A
3	Heading (degrees magnetic)	0.0 to 359.9
4	Airspeed (knots)	0.0 to >100,000.0
5	Altitude (feet)	<-10,000.0 to >100,000.0
6	VSI (feet per minute)	<-10,000 to >+10,000
7	Not used - Future growth	
8	Latitude (miles from Greenwich Meridian)	0.0 to >10,000.0
9	Longitude (miles from Equator)	0.0 to >10,000.0
10	Not used - Future growth	
11	Course Deviation (degrees) – Error from final course to simulated runway	<-10.0 to >+10.0
12	Glideslope Deviation (degrees) – Error from glidepath to simulated runway	<-10.0 to >+10.0

Some of the limitations on data range are only because of the data type used, which is vastly greater than any values that would be experienced in real life. This is why some values only say “greater than” or “less than” a certain value, which is always more than sufficient.

The discrepancies between the information provided in the IMU data stream and the information required in the data array were resolved by the control computer before the data was placed into the array. The roll, pitch, and heading data needed no additional processing before use. The other parameters were derived as described in the following paragraphs.

Since the IMU only provides accelerations in the vertical and horizontal axes, the accelerations must be integrated for rates and integrated again for position. Horizontal acceleration (provided in g) was converted to feet per second and then integrated to get

horizontal velocity in feet per second, which was then converted to knots and inserted into the array.

Vertical acceleration (in g) was converted to feet per second and then integrated to get vertical velocity. This was converted to feet per minute and inserted into the array. Then the vertical velocity was integrated again to get altitude and this was inserted into the array.

While this technique initially seemed to be an acceptable method, two effects caused error to accumulate in these values over time. The first was the calibration error of the IMU on alignment. Even when unaccelerated, it would sometimes output a small acceleration. Inserting a calibration constant into the software easily compensated for this. The second, more serious, error involved the natural tendency for an integral to accrue error over time. The techniques used to overcome this problem were significant and deserve a more in-depth discussion.

Design Loop

Most of the problems encountered to this point in the project did not involve a major cycle back to the start of the SEP. Some minor changes had been accommodated through the Requirements loop and Functional Analysis, and the hardware interfaces as initially designed worked remarkably well. Solutions to the IMU alignment and altitude determination issues would only come from revisiting the Functional Analysis and a new transformation of the re-worked functional architecture to physical architecture. This is the essence of the design loop in the SEP.

Alignment

The first problem to be analyzed was IMU alignment. Although the IMU would obtain a good alignment quickly (< 10 sec) when it was not moving, when it was moving and being shaken during alignment (such as in a GA aircraft in turbulence) it would not get a good alignment. Usually this meant that the roll axis would begin to precess during the first roll maneuver. Improving the shock mounting did little to improve the performance.

After consultation with the engineers at Watson Industries, it was determined that to get the best alignment possible, the IMU required a velocity input. This velocity did not have to be exceptionally accurate, even an approximation would help performance.

This new step in the functional block diagram could be transformed into one of two physical architectures. A sensor could be attached to the aircraft pitot-static system to measure airspeed and feed the result to the IMU, or a circuit could be designed that would allow the test conductor to manually set a particular "target speed" for a test maneuver. Since portability of the system was a concern and an interface to the aircraft pitot-static system would greatly increase system complexity and reduce portability, the "target speed" circuit option was chosen.

This circuit was constructed using a 50K fixed resistor and 100K variable resistor across the 12V power supply to the IMU. This allowed input of 0 to 8.0V DC to the IMU, which set the input at 0 to 320 kilometers per hour (0 to 173 knots). A switch was also added to enable or disable this circuit as desired by the tester. In use, the target speed was set with the variable resistor and the IMU re-aligned immediately before the test point.

The second purpose for this circuit was to prevent velocity from developing too much error over time. The velocity computed from the integrated IMU output was compared to the "target speed" and the error between the two served as a correction to the IMU velocity. A variable weight was assigned to this error (controlled by the test conductor) in order to prevent velocity from generating gross errors through the time period of the particular test point. Though this technique worked, it involved an additional in-flight calibration step, which cost valuable flight time. Therefore, the proof-of-concept tests were conducted using a fixed value for the airspeed instrument.

Altitude

Aircraft altitude was determined by integrating the vertical axis acceleration once for vertical velocity and again for altitude. Integration generates significant error over time, and integrating twice made that error unacceptable for this project. Even after compensating for alignment error with a calibration constant that could be adjusted by the test conductor, the longest time that could be achieved before there was 100 feet of altitude error in the system was 90 seconds. This best-case error condition was with the IMU stationary on a desktop! Since this error was unacceptable, the SEP design loop was entered again to re-evaluate the functional architecture and options.

Using the output of a constant voltage circuit to set one particular speed solved the airspeed issue, but the test points chosen for this project were relatively constant airspeed. The project test altitudes would be changing significantly, so a constant-altitude circuit would not be sufficient. Altitude would have to be measured directly.

Three options for the physical hardware to determine aircraft altitude were analyzed. The first involved a pressure sensor to measure atmospheric pressure and

convert that to an altitude (just as GA aircraft altimeters do). The second option involved measuring altitude with an active signal (such as radar or laser) directed down from the aircraft and a reflection sensed. The third option involved using a GPS receiver to measure altitude.

GPS added another major component that would require a new interface to the data processing computer and a radar or laser altimeter would require aircraft modifications. Either option would be expensive. Therefore, a pressure sensor seemed to be the best option.

The Motorola MPX4115A pressure sensor was selected for its small size, low cost, and on-chip temperature compensation. It measures altitude from sea level to 40,000 feet with analog output and uses a 5V DC power supply (Motorola, 2000).

First, the power supply was constructed using a National Semiconductor LM 117 operational amplifier. This op-amp can take a 12V DC input and provide a well-regulated 5V DC output with very little additional circuitry. The IMU had four channels for analog input that it would convert to digital and insert into the data stream, so this was selected for the A-D conversion and communication to the control computer.

Unfortunately, this was not the end of the problem. The first time the sensor was used (driving up and down a hill in a car) the output of the pressure sensor circuit only indicated a change in altitude about every 65 feet. While a pilot does not need an altimeter accurate to a foot to conduct instrument flight, an altimeter only accurate to 65 feet is unacceptable. A more in-depth analysis of the problem was necessary.

While the pressure sensor had plenty of range (-3,000 feet pressure altitude to over 40,000 feet) and sensitivity (published as 0.1 mV and 45.9 mV/kPa), the analog-to-

digital conversion resulted in a dramatic reduction in sensitivity. An analysis of the transfer function that governed the pressure measurement process yielded a function of this form:

$$\frac{V}{45.9mV/kPa * 0.0034kPa/ft} = h$$

Where V is the voltage in millivolts and h is the pressure altitude in feet. The leftmost value in the denominator is the relationship between pressure and voltage for the sensor, taken from Motorola documentation, and the rightmost value is the linear approximation of the relationship between pressure and altitude from 0 to 20,000 ft MSL.

The IMU A-D circuit only provided voltage precision to 10 mV. Substituting this minimum change in voltage showed that the minimum change in altitude required for a change in the digital elevation data was 64.1 feet.

$$\frac{10mV}{45.9mV/kPa * 0.0034kPa/ft} = 64.1ft$$

An indicated change in altitude only every 64 feet is obviously unsatisfactory for conducting instrument flight. Another iteration through the design loop yielded two possible solutions: 1) Using an separate A-D converter with greater sensitivity, or 2) Using the integrated vertical acceleration from the IMU, and correcting it with the output of the pressure sensor. To keep system cost and complexity low, the second option was selected.

Parameter 7 in the IMU data string (as shown in Table 11) was the output voltage of the pressure sensor, converted to a digital readout. This was converted to an altitude with the following transfer function:

$$Alt = 769.69V^2 - 12219V + 37544$$

Where Alt is the pressure altitude in feet and V is the output voltage. This transfer function was computed by graphing the published output versus pressure from Motorola documents and standard-day pressure versus altitude, then computing a second-order polynomial curve fit for the region from 0 to 20,000 feet (the region of interest).

The system altitude was then computed by using this measured altitude as input for a Kalman filter that the IMU computed altitude data was filtered through. The pressure sensor altitude and IMU integrated altitude were compared and the difference (times a variable weighting factor that could be controlled by the test conductor) was used to correct the IMU altitude, which was then displayed. This allowed some variability in IMU performance to be corrected real-time during the test and proved to work very well after some initial in flight experimentation with different weighting factors.

When using a weighting factor somewhere around 0.3, the altimeter tracked in sync with the aircraft altimeter during all maneuvers when the IMU had a good alignment. This corrected altitude was also used to compute a correction to the vertical velocity derived from the IMU data.

Final System Architecture

The end result of this development was the system shown in Figure 4. Once it was constructed and preliminary testing with sim mode was complete, initial operational trials were conducted in an automobile. Testing with an automobile provided a low-cost method of analyzing system performance in a two-dimensional environment, building up to full flight test. The system worked reasonably well and additional valuable

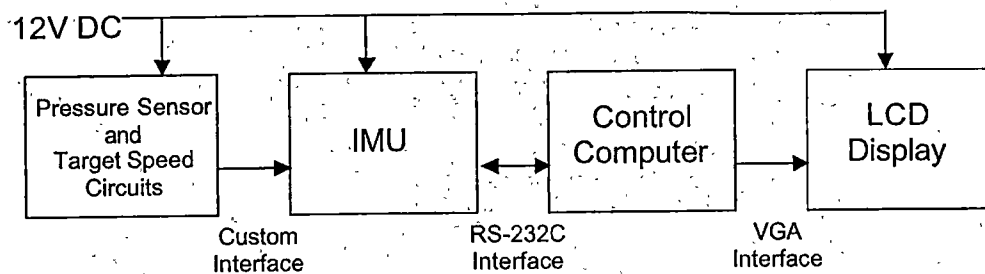


Figure 4. Top-Level Functional Architecture

information was collected on IMU alignment techniques and calibration constant settings with this technique.

The pressure sensor and target speed circuit schematics are shown in Figure 5. Detail photographs of these circuits and their connections to the IMU are shown in Appendix B.

When installed in the aircraft, the components were arranged as shown in Figure 6. The IMU and control circuitry were placed on the back seat, while the control computer was placed on the test conductors laptop sitting in the right seat of the aircraft. Although a mounting system using a hinged arm that could be screwed into the aircraft instrument panel was purchased, it was not used and the test conductor held the display in place on top of the instrument glare shield. This arrangement allowed the test conductor to quickly change display configurations and closely observe the pilot and display operation during test maneuvers.

The total cost and weight of the system is shown in Table 13. Although the cost is substantial, it may primarily be attributed to the cost of the IMU. A lower-cost data collection system would not meet the minimum requirements shown in Table 1.

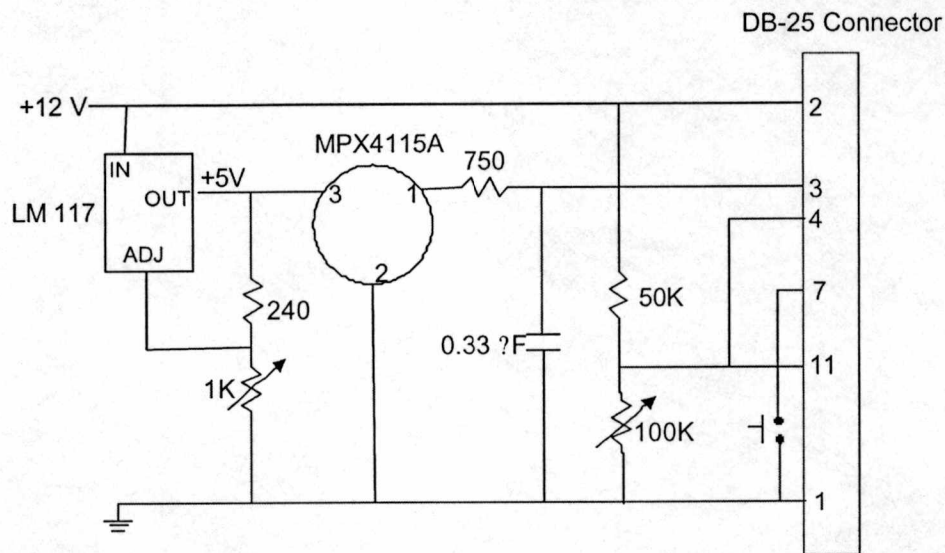


Figure 5. Pressure Sensor and Target Speed Circuit Schematics

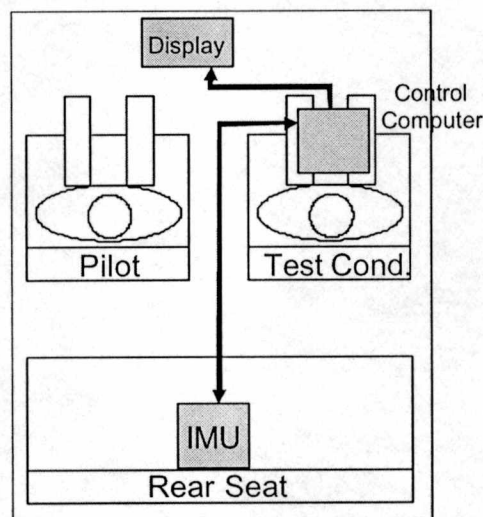


Figure 6. Physical Architecture

Table 13. System Cost and Weight

System	Cost (US \$)	Weight (lb)
Watson IMU	13,141	4.0
Laptop Computer	1,500	6.0
Earth Tech LCD Display	895	1.3
Motorola MPX4115A Pressure Sensor	29	0.02
LM 117 Operational Amplifier	11	0.02
Miscellaneous integration hardware	58	1.2
Total	\$15,634	12.5 lb

V. System Verification

The final step in the SEP loop is verification of the system to ensure that the final product meets the modified requirements. For this, it was necessary to construct a few sample displays and test them with different pilots to measure their performance and gather their opinions. The displays chosen for analysis are shown and described in Appendix C. This section describes the data gathered and the results.

The initial test plan was to have pilots fly the S-3 and precision instrument approach maneuvers listed in Appendix D and their performance would be recorded with the system software. Each pilot was scheduled to perform the maneuvers a couple of times with each display in sim mode on the ground, and then in flight while being recorded.

Overview

Three flights were conducted, totaling 4.7 hours of flight time. Two flights were flown in a Cessna 152 (N1846C) and one in a Cherokee 180 (N8486W). The Cherokee was flown from Henderson airport just south of Las Vegas and the 152 from Boulder City airport, approximately 10 miles southeast of Las Vegas. Flights were from 3000 ft MSL to 8000 ft MSL over the desert valleys to the south and southwest of Las Vegas. Two flights were flown in daylight and one at night.

IMU alignment problems resulted in scaling back the test plan. The S-3 maneuvers were not performed, as the poor IMU performance was exacerbated in sustained turns. Limited test time also prevented one pilot from getting ground experience with the system before flight.

Three pilots evaluated the system. The pilots were all very experienced, but came from various backgrounds and brought a different point-of-view to the test. Pilot data is summarized in Table 14.

Display evaluation consisted of both pilot comments on the display and quantitative measurements of pilot deviation when performing a specified task. Needles were projected on the gyro instrument of each display showing the deviation as the pilots attempted to eliminate any errors.

Because of the IMU alignment problems in turbulent environments noted previously, it was necessary to get the aircraft set up on an intercept heading to the final course, align the IMU, then immediately start the task. Usually by the time the aircraft was approaching the simulated touchdown point the IMU was beginning to develop unacceptable errors and approximately 20% of the time a run had to be aborted and set up again because of IMU error. The most common IMU problem was divergence in the roll channel.

Data was recorded as the pilots performed the task and post-flight the data was analyzed to determine average deviation. The time period used for data commenced when the pilot maneuvered the aircraft to within 3 degrees of the inbound course and did not subsequently deviate beyond these limits for 2 minutes. This method reduced the influence of pilot overshoots when acquiring the inbound course, which were sometimes

Table 14. Pilot Experience

Pilot	Total Hours	Background
Pilot 1	2200	Navy Test Pilot, Primarily FA-18
Pilot 2	3500	Air Force Test Pilot, Primarily F-15 & F-16
Pilot 3	1500	Civilian Flight Test Engineer, Commercial Pilot, CFII

unpredictable. The two-minute period allowed for sufficient data collection before getting down to the point where the course and glideslope windows were too small for the pilot to track accurately (near the simulated touchdown point).

Display layouts were not presented to each pilot in the same order, but the data collection always consisted of one approach using each of the three displays, followed by a fourth approach using the first display again. Results for the first and fourth passes were averaged. This method was chosen to attempt to reduce the effect of pilots getting accustomed to the system through repeated exposure, even though the display layouts were different.

Data Analysis

The data files (as shown in Appendix E) were recorded on each approach and post-flight were imported into Microsoft Excel for analysis. Pilot comments were recorded during and immediately after each simulated approach.

Figure E-1 shows plots of the data from one test point. The four plots are the pitch, roll, heading, and altitude of the aircraft. At the left side of the graph, the course capture can be noted as the pilot reverses the aircraft roll and heading begins to stabilize at approximately 090 degrees magnetic.

Figure E-2 shows the course and glideslope deviation for this approach over the same time period. Approximately halfway across the graph, the course and glideslope both begin to deviate up to just under one degree of error. At the same time on the altitude graph, a leveling off can be noted, and assigned as the cause of the error. These graphs can serve to pinpoint the source of pilot error, and which instruments were not

effective at transmitting information to the pilot when attempting to perform a particular task.

The glideslope deviation graph also exposed two previously unnoticed bugs in the software. The first was the way the graph occasionally appeared to "backtrack", having two glideslope error values at one time, obviously impossible in a linear system. The source of this problem was isolated to the individual record time-stamp assigned to each line of data. The minutes and seconds were recorded directly, but the software development package used (Visual Basic) had no provision for tracking tenths of seconds. Therefore, the computer operating system had to be accessed directly and the tenths of seconds appended to the time value.

An error in programming caused ten-tenths of seconds to be appended as "10", rather than as "0" and incrementing the seconds appropriately. An example is that "14:02" and 10/10ths was recorded as "14:02.10", not "14:03.0". When imported into Excel for analysis, the value was converted to "14:02.1", but maintained its proper position in the data stream. Therefore, the plot appears to "backtrack" on itself to go back to the error value at the phantom time "14:02.1" rather than plotting it at "14:03.0".

The second error was evidenced in the periodic sawtooth pattern in the data, clearly noticeable toward the right side of the graph in Figure E-2. Obviously the aircraft could not make deviations that rapidly, and the altitude, latitude, and longitude traces did not evidence the same sawtooth pattern, so the aircraft data collection system was functioning properly. The glideslope error computation step was analyzed next and at one stage an improper conversion to an integer value was performed. This conversion from a floating point value to an integer value created the sawtooth error shown. After

fixing this error, other test points were flown with sim mode and the graph did not have the sawtooth discontinuities.

The average pilot deviation was computed as the absolute value of the errors in course and glideslope over the time frame in question summed and divided by the number of samples. The course and glideslope errors derived this way were then summed to produce the numbers shown in Table 15. This is obviously a simplistic measure of error, but sufficient for this limited system evaluation. Some interesting observations can be made from the qualitative and quantitative data.

The “Basic” display represented a standard GA “T-shaped” instrument layout with airspeed and altitude gauges flanking a gyro on the top row and heading and VSI below (see Figure C-1). Overall, the three pilots exhibited the greatest amount of error with this display, suggesting that significant improvements can be made in cockpit layout. Pilot comments with this display included, “VSI not in [my] crosscheck—dark color makes it disappear”, and “ILS needles tend to ‘override’ bank angle on [gyro]”. Although pilot 3 (with the most GA experience) commented that he, “Liked [the] steam gauges with digital readout.”

The “Big” display is shown in Figure C-2. The large error of pilot 3 using the “Basic” display is anomalous, but it is worth noting that he did the best using the “Big”

Table 15. Average Pilot Deviation

Pilot	Basic Display	Big Display	Green Display
Pilot 1	1.75	1.23	0.83
Pilot 2	0.75	0.71	0.81
Pilot 3	2.10	0.59	1.67

display--both in comparison to the other displays he flew and in comparison to the other pilots. This may reflect his greater experience with "conventional" GA cockpit displays in instrument flight and led to his comment, "That display was pretty good!" Pilot 2 really disliked the display, commenting, "[It] almost made me sick to look at" and pilot 1 noted that, "VSI too sensitive [to use]."

The "Green" display is shown in Figure C-3. From the experience data on pilots 1 and 2 it is obvious that most of their instrument flight experience is with military aircraft HUDs while pilot 3 is almost exclusively a GA pilot. Correspondingly, the deviation of pilots 1 and 2 using the "Green" display (which most closely represents an F-16 HUD) was half that of pilot 3. Indeed, one of the comments of pilot 2 was, "Easy. Obviously I'm used to flying approaches on something that looks like this." This is not to say that there were no complaints, pilot 1 commented negatively on the lack of a VSI with this display and pilot 2 mentioned that getting a feel for bank angle was difficult without a bank angle pointer and scale.

VI. Project Summary

Conclusions

As aviation accidents attributed to mechanical failures gradually decline in frequency, human factors causes of accidents become more and more significant. Improvements in pilot-vehicle interfaces can help reduce the human errors that lead to mishaps. Cockpit display improvements can be a large part of these improvements.

The system constructed and tested in this project is step toward developing tools for performing this type of testing on a low-cost basis. Using this system, researchers can gather data on the aircraft flight parameters, process that data, and display the results in an infinite variety of formats to the pilot. Furthermore, not only can the data be used instantly in the processing of the display, but it can also be saved and reviewed later. This means the system can not only be used for display testing, but can function as a low-cost data collection system to measure aircraft flight parameters for flying qualities and performance testing or as a tool in a flight training program.

However, as used in this test, the system had some significant weaknesses. The IMU used to collect aircraft flight data was unreliable. Despite several attempts to compensate for the deficiencies, it was barely serviceable for the test and some test plan points had to be eliminated and others aborted. The software allowed for basic instruments, but the appearance of those instruments was very simple and some standard aircraft instruments (most noticeably the HSI) were not included.

Although the test techniques and display layouts used in the system verification were not sufficiently rigorous for a dedicated display evaluation, they did prove that the

system met the majority of the input requirements. The data collected was similar to the expected results for the displays and pilots tested.

Recommendations

Although the final system architecture was suitable for the limited display testing conducted, there are many improvements that can be implemented. Most of these improvements add little additional cost and would greatly improve performance.

Hardware

The first major area of improvement should be to obtain a more reliable data collection system. The difficulties mentioned with the IMU used would not be a problem with a system based on the Global Positioning System (GPS). These systems use an antenna array to measure aircraft attitude using GPS signals, instead of mechanical gyroscopes or solid-state accelerometers, eliminating the problems caused by gyro drift. They can also supply accurate ground speed, altitude, and vertical velocity, removing the requirement for the altitude sensor and velocity calibration circuit.

GPS based attitude systems are available from several manufacturers. Trimble manufactures a unit called the TANS Vector, and Ashtech has a unit called the ADU-2. Both systems use a four-antenna array for attitude measurement.

The advertised system accuracy for the ADU-2 is exceptional at ± 0.3 deg pitch and roll, using a 1-meter baseline antenna arrangement. The output is also an RS-232C serial data stream and would interface very easily with the existing software. The system is slightly more expensive at \$14,000, but the most significant disadvantage is that it cannot easily be moved between aircraft as the antenna array must be permanently installed and calibrated for that aircraft.

In this project the A-D converter built into the IMU was used to convert the signal from the altitude pressure sensor to a digital form. If the ADU-2 were used, pressure altitude could still be measured for comparison with the GPS derived altitude, but an independent A-D converter would be required. The A-D PCMCIA card from Omega Industries mentioned earlier could be coupled with the same pressure sensor circuit used in this project to reduce altitude resolution from 64 feet to 6 feet or better.

Once the A-D converter card is integrated, a pressure sensor for aircraft pitot measurement could be included. The data processing computer could use forecast or estimated winds to compute an approximate airspeed from the GPS-supplied ground speed, but a pitot sensor would directly provide the indicated airspeed. The Motorola MPX5010D would be an outstanding choice for this purpose and allow airspeed measurement down to 1 knot.

Software

There are many upgrades that can be carried out in software; only a few possibilities will be listed here.

- Eliminating the requirement for all instruments to be rectangular would be a useful upgrade (see the black box incorporating the VSI in the "Big Gyro" display in appendix B). This can be done by making the controls invisible on the form, and having them do their own "hit tests" to determine when they are selected. This would allow better integration of the different instrument types and provide more options for "stacking" instruments on each other.
- Additional instruments should be added including: Horizontal Situation Indicators, Turn and Slip Indicators and horizontal as well as vertical strip instruments. An HSI

and Turn and Slip indicator are widely-used instruments that would be popular test instruments. Other additional instruments may be used for advanced display concept testing.

- Bank angle pointers should be added to gyros as it is hard to precisely target an angle of bank using the current gyro instruments and capturing and maintaining a desired angle of bank is a popular test maneuver.
- Different pointer styles should be added to gauges, for pointers with different colors, sizes and shapes, as well as gauges with multiple pointers (such as 3-pointer altimeters) to test the pilots' ability to interpret readings on gauges of different styles.
- Major and minor graduations should be added to gauges and strip instruments to assist pilots with rapidly interpreting gauge and strip readings.
- Colored "zones" to indicate certain conditions on gauges and strips should be added to help with the identification of "out-of-limit" situations.
- Different types of course and glideslope deviation indicators should be added, such as "flight director" needles, and pitch and bank command bars

Although in practice it did not prove to be too much of a distraction, a modification to the program to eliminate the flicker in some instruments with complicated features and complex colors (primarily gyro instruments) would eliminate a possible source of error in pilot performance.

At the same time that more thorough quantitative data is gathered, a formal pilot rating system should be used to improve the qualitative data. The Cooper-Harper Workload Rating scale or Bedford Workload Rating scale are both excellent options. Not

only should displays be evaluated as a whole when performing certain tasks, but the critical instruments could also be rated individually.

Future Projects

The next step should be to take the results of this test and re-enter the SEP at the top. Several critical lessons have been learned and if the recommendations mentioned previously are used as requirements and functional blocks for the next iteration, the resulting system should be a very valuable test asset. Using this system, future research should then conduct further research into display symbology. Even the simple design verification process used here provided some tantalizing insights into how improvements in cockpit instrumentation could improve performance and safety in GA aircraft.

More thorough test plans with test points that focus on specific flight parameters should be prepared and the flights conducted using displays that are more carefully constructed and more homogenous. This way the capacity of a specific instrument to transmit information to the pilot could be isolated and analyzed. Perhaps then there will be a revolution in GA aircraft flight instrument displays similar to that experienced in military and commercial aircraft.

References

References

1. Barrows, A., et al. (1996). "GPS-Based Attitude and Guidance Displays for General Aviation", in IEEE Emerging Technologies.
2. Cooper, G., and Harper, R. (1969). *The use of pilot ratings in the evaluation of aircraft handling qualities*, NASA Tech. Rept. TN D-5153.
3. Earth Computer Technologies (1997). *EarthVision LCD Controller Manual*, Earth Computer Tech.
4. Edwards, E. (1972). "Man and Machine: Systems for Safety", in Proceedings of BALPA Technical Symposium, British Airline Pilots Association.
5. FAA and NASA (1990). *Memorandum of Understanding*, March 15, 1990
6. Hawkins, F. H. (1987). *Human Factors in Flight* 2nd Ed., Ashgate Publishing.
7. Hayward, R. C., et al. (1997). "Inertially Aided GPS Based Attitude Heading Reference System (AHRS) for General Aviation Aircraft", in Proceedings of ION GPS 97.
8. Leonard, J. (1999). *Systems Engineering Fundamentals*, Defense Systems Management College Press.
9. Microsoft (1997). *Visual Basic 5.0 Reference Library*, Microsoft Press.
10. Motorola (2000). *Semiconductor Technical Data-MPX4115A/D* Rev 3, Motorola Inc.
11. Motorola (2000). *Semiconductor Technical Data-MPX5010* Rev 2, Motorola Inc.
12. Sanders, M. S. and McCormick, E. J. (1993). *Human Factors in Engineering and Design* 7th Ed., McGraw-Hill, Inc.
13. Trimble Navigation Ltd. (1998). "TANS Vector GPS Attitude System", Trimble Nav.
14. Wiener, E. and Nagel, D. (1988). *Human Factors in Aviation*, Academic Press
15. Wierwille, W. and Casali, J. (1983). "A validated rating scale for global mental workload measurement application" in 27th Proceedings of the Human Factors Society.
16. Watson Industries (2000). *Owner's Manual IMU-E604* Rev. G., Watson Ind.

Appendices

Appendix A. List of Abbreviations

A-D	Analog-Digital conversion	SEP	Systems Engineering Process
AHRS	Attitude/Heading Reference Sys	VSI	Vertical Speed Indicator
D-A	Digital-Analog conversion		
FFBD	Functional Flow Block Diagram		
FOG	Fiber Optic Gyro		
FPS	Feet Per Second		
GA	General Aviation		
GPS	Global Positioning System		
GS	Glideslope		
HUD	Heads Up Display		
HDD	Heads Down Display		
Hz	Hertz (cycles per second)		
ILS	Instrument Landing System		
IMU	Inertial Measurement Unit		
KIAS	Knots Indicated Airspeed		
KPM	Knots Per Minute (acceleration)		
LCD	Liquid Crystal Display		
MFD	Multi-Function Display		
MOE	Measures Of Effectiveness		
PVI	Pilot-Vehicle Interface		
SA	Situational Awareness		

Appendix B. System Detail Photos

This appendix shows detail photographs of the pressure altitude and velocity calibration circuits and their connections to the IMU.

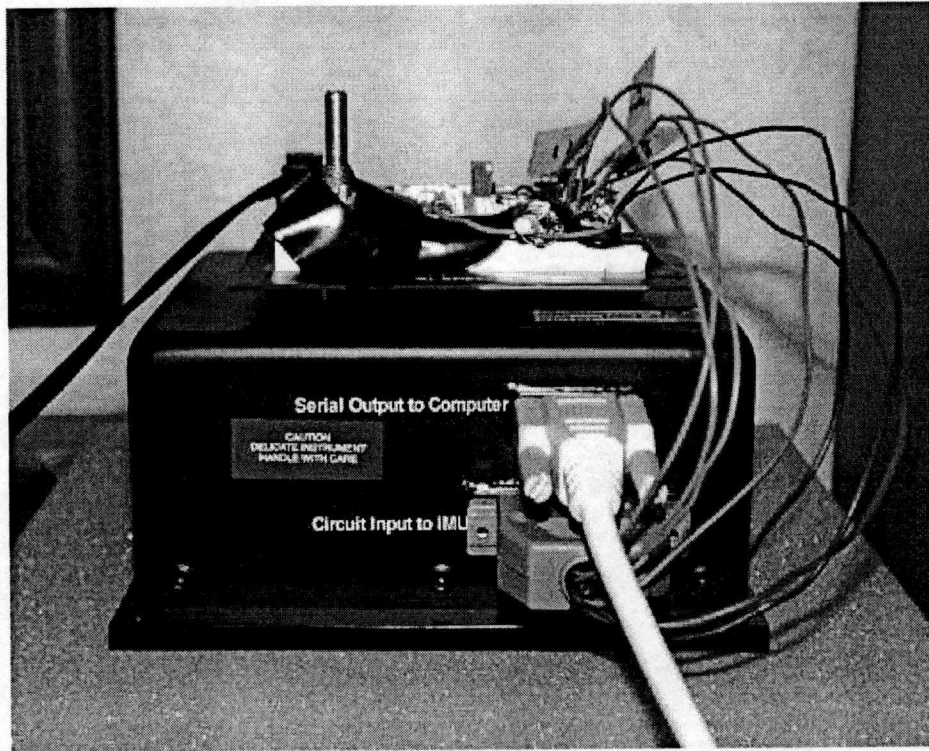


Figure B-1. Front View of IMU and Circuits

Figure B-1 shows a front view of the IMU with the pressure sensor and velocity calibration circuits and the connections between those circuits. The black cable leading out of the frame to the left is the 12V DC power supply cable. The connections between the circuits and IMU and IMU and computer are labeled.

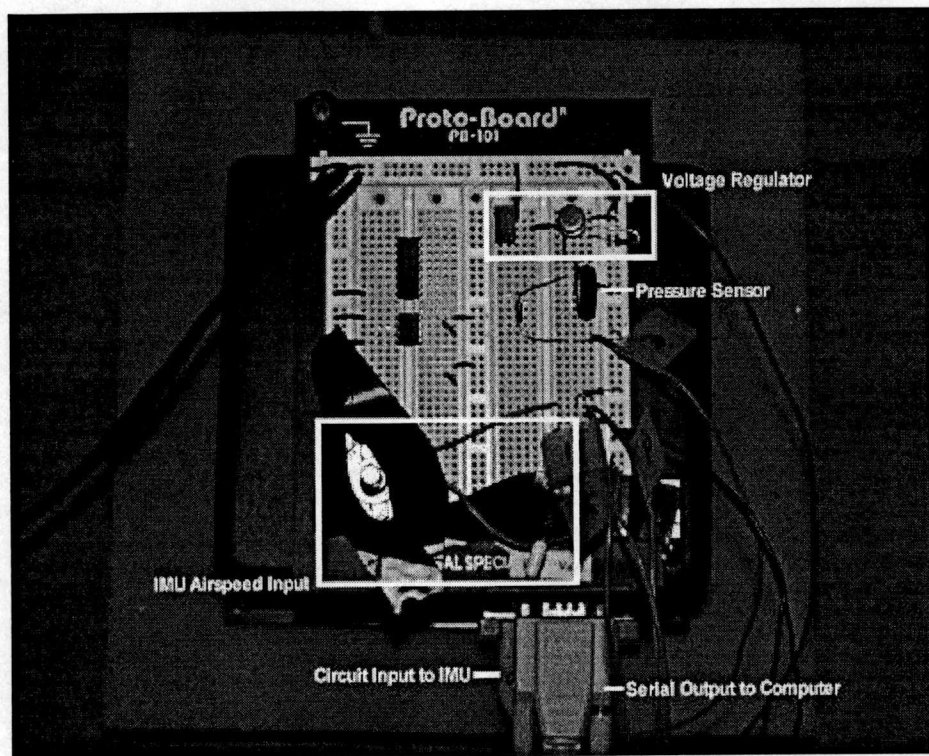


Figure B-2. Top View of IMU and Circuits

Figure B-2 shows details of the pressure sensor and velocity calibration circuits and how they are laid out on the Proto-Board. The voltage regulator section in the upper right converts the 12V DC power supply to 5.1V DC for the pressure sensor (immediately below the regulator).

The IMU velocity calibration circuit is along the lower portion of the image. The connections between the circuit and IMU and IMU and computer are at the very bottom of the image. The black cable leading out of the frame to the left is the 12V DC power supply cable.

Appendix C. Tested Display Layouts

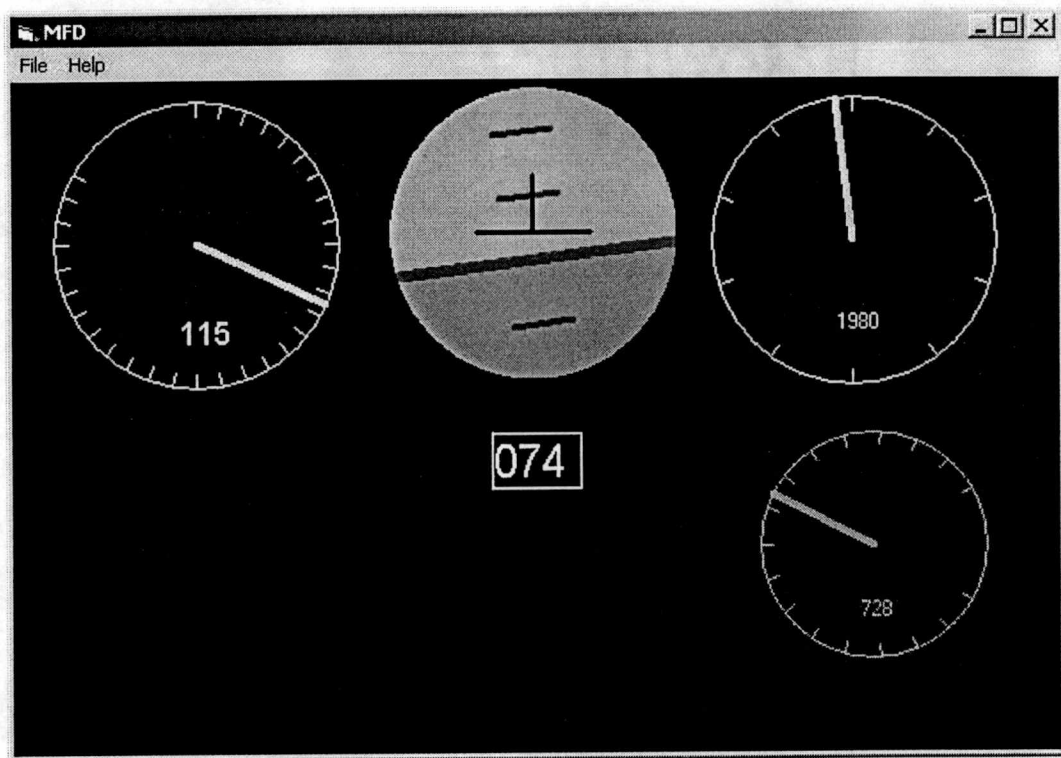


Figure C-1. Basic Display

This display is designed to replicate a traditional GA aircraft instrument panel. It incorporates a black background with a high-contrast white circular gauge airspeed indicator on the upper left and a white circular gauge altimeter on the upper right. There is a circular gyroscope display in the upper center with a blue hemisphere to indicate the sky and a brown hemisphere to indicate the earth. There is a black, inverted-T shape plane symbol centered in the middle of the gyro and dark green pitch graduations every 10 degrees.

Below the gyro is a white digital readout of aircraft heading, and to the right of that (below the altimeter) is a vertical speed indicator in lower-contrast teal color. The

VSI needle points directly left (horizontal) to indicate no vertical speed, and moves up or down as required, with the max positive vertical speed at the 2 o'clock position and max negative vertical speed at the 4 o'clock position.

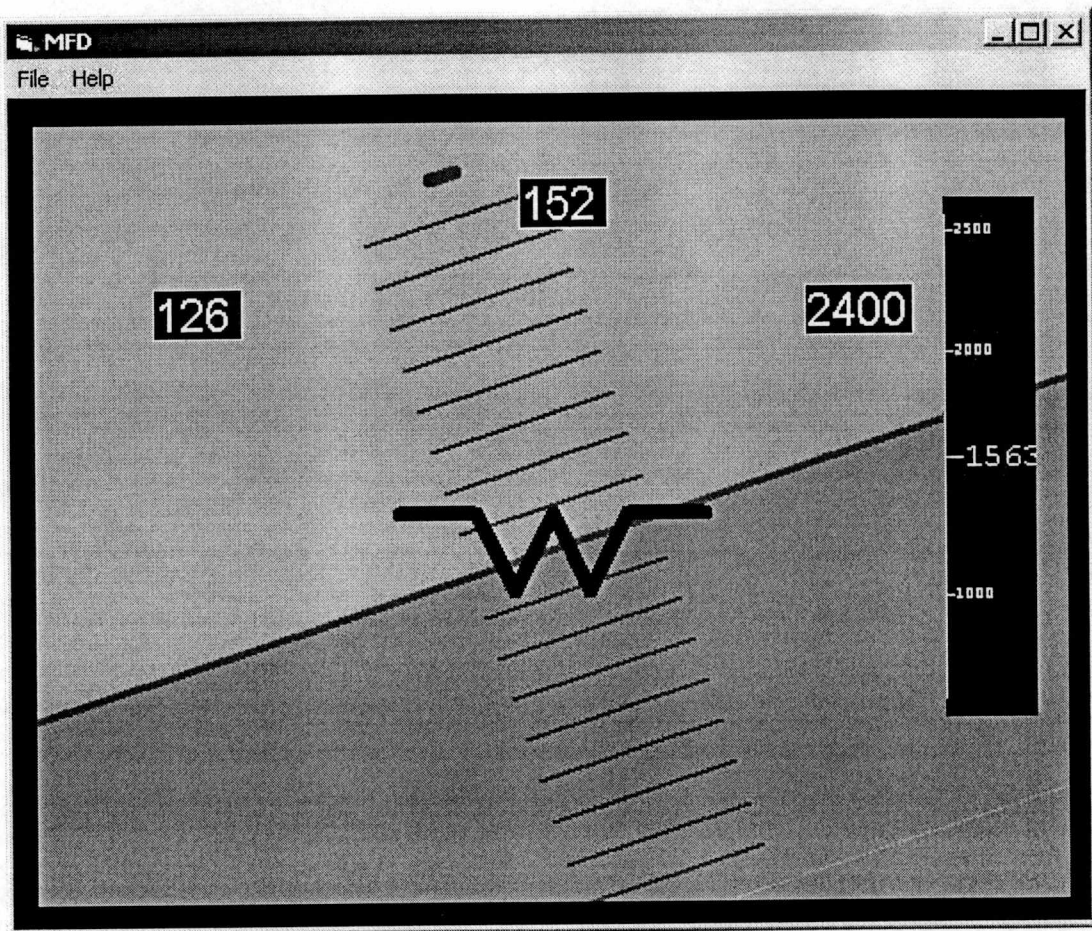


Figure C-2. Big Gyro Display

This background of this display is a gyro instrument, filling the available area of the display. The gyro has blue and brown hemispheres to indicate the sky and ground respectively, with a “W” shaped aircraft pointer in the middle and dark green graduations every ten degrees of pitch with shorter, thicker green spots at ± 90 degrees.

A digital readout of airspeed is in the upper left area of the display, a digital heading indicator is in the upper center, and altitude is in the upper right of the display. A vertical strip instrument runs up and down the right side of the display to indicate vertical speed. All these instruments are in high-contrast white on a black background.

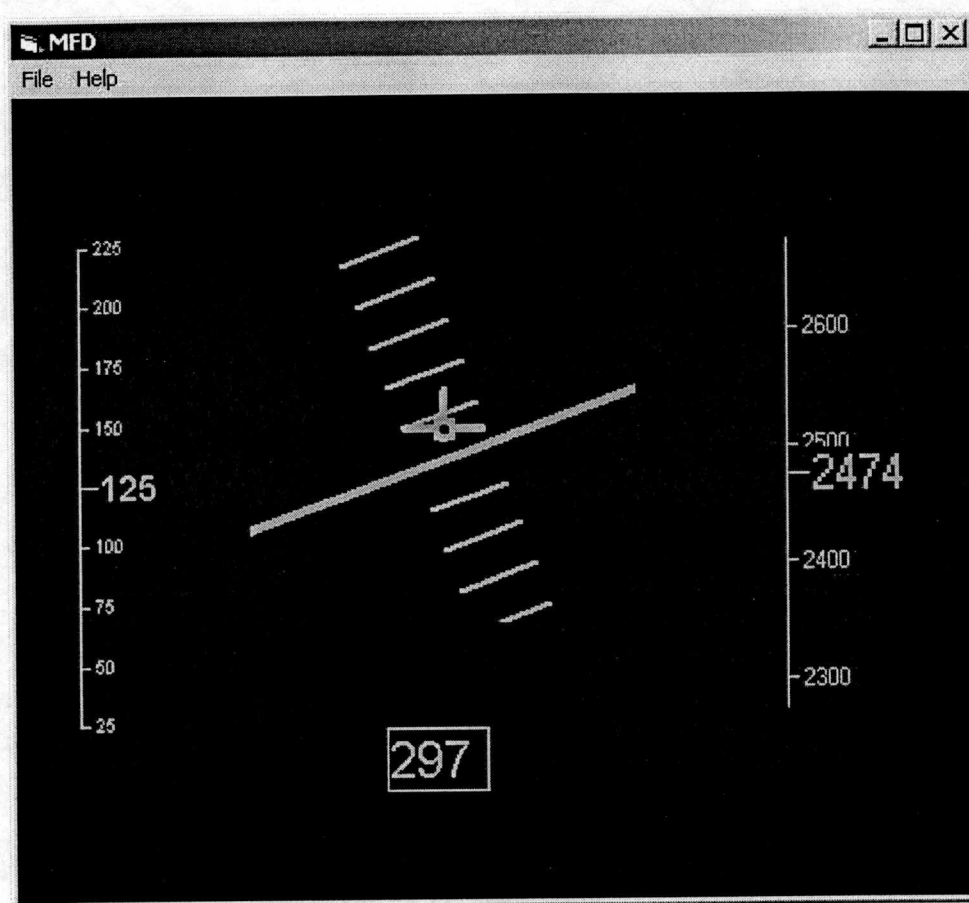


Figure C-3. Green Display

This display is designed to approximate an F-16 HUD display. The center is filled with a "pitch ladder" gyro instrument with a "velocity vector" style pointer. This pointer does not actually indicate aircraft flight path, but is just another style of waterline indicator available with the software. The pitch ladder has graduations every 10 degrees of pitch, with a thicker, longer line indicating the horizon.

Along the left side of the display is a vertical strip instrument for airspeed and along the right side is a vertical strip instrument for altitude. Both of these are "fixed-pointer, moving scale" strips. In the bottom center is a digital readout of aircraft heading. All symbology on this display is bright green on a black background.

Appendix D. Display Test Plan

The test plan presented here is a limited test plan created to prove the system concept using the displays shown in Appendix C. It can also serve as a starting point for researchers intending to probe further into ergonomics testing with GA aircraft.

Table D-1. Test Tasks

Task	Technique	Critical Param.
Altitude Capture	Establish 1500 fpm rate of climb/descent at constant airspeed. Select altitude to capture and aggressively maneuver aircraft to capture and maintain that altitude while keeping heading and airspeed constant.	Altitude, Airspeed
Heading Capture	Establish 45 deg angle of bank on constant altitude and airspeed. Choose heading and aggressively maneuver to capture and maintain that heading.	Heading, Altitude, Airspeed
30-30 Rolls	Establish 30 deg angle of bank on constant altitude and airspeed. Aggressively roll aircraft to capture and maintain 30 deg bank in opposite direction.	Bank Angle, Altitude, Airspeed
S-1	500 fpm climb for 1 minute, immediately transitioning to a 500 fpm descent for 1 minute, finishing on original altitude. Airspeed must remain constant.	Airspeed, VSI
S-3	500 fpm climb with a 1.5 deg/sec turn (standard rate) for 1 minute transitioning to a 500 fpm descent while maintaining the turn for another minute (total of 180 degrees of turn). Then reversing turn direction and completing another climb/descent cycle over the next two minutes. Finish on original heading and altitude, airspeed is constant.	Airspeed, VSI, Turn Rate
Precision Approach	Start 1 nm south of final course and 7 nm west of desired touchdown point. Capture final course of 090 deg magnetic and track course and glideslope down to 200 feet above touchdown point.	Course deviation, Glideslope deviation, Airspeed

Appendix E. Sample Data

This appendix shows an example of the data recorded by the software and the types of graphs that can be generated. In the data file, the first line is a header that shows the date and time the data was captured, and the headings of the data columns. The subsequent lines begin with the minute, second, and tenths of seconds that line of data was captured. This comma-delimited data file can be imported directly in another program for analysis (in this project, Microsoft Excel was used).

```
"10/29/00 7:41:06 AM", "Pitch", "Roll", "Heading", "Airspeed", "Altitude", "VSI", "Latitude", "Longitude", "Course Dev", "G/S Dev"  
"41:6.4", "-0.8", "-0.8", "68.2", "103.52", "6046.344", "-11.579", "0.026", "-0.064", "8.05", "1.1"  
"41:6.7", "-0.8", "-0.5", "68.2", "103.458", "6045.824", "-12.113", "0.028", "-0.069", "8.04", "1.1"  
"41:6.9", "-0.6", "-1.1", "68.3", "103.458", "6048.259", "-0.122", "0.03", "-0.075", "8.03", "1.1"  
"41:7.2", "-0.8", "-1", "68.3", "103.52", "6047.68", "-6.958", "0.032", "-0.08", "8.01", "1.1"  
"41:7.4", "-0.9", "-1.2", "68.3", "103.458", "6047.101", "-13.725", "0.034", "-0.085", "8", "1.1"  
"41:7.6", "-0.9", "-1.5", "68.2", "103.582", "6046.498", "-28.603", "0.036", "-0.091", "7.99", "1.1"  
"41:7.8", "-0.7", "-1.6", "68.1", "103.52", "6048.84", "-20.644", "0.038", "-0.096", "7.98", "1.1"  
"41:8.0", "-0.7", "-1.7", "68", "103.458", "6051.1", "-16.746", "0.041", "-0.101", "7.97", "1.1"  
"41:8.3", "-0.8", "-1.7", "67.9", "103.52", "6050.374", "-17.228", "0.043", "-0.107", "7.96", "1.1"  
"41:8.5", "-0.8", "-0.9", "67.9", "103.582", "6049.662", "-21.758", "0.045", "-0.112", "7.94", "1.1"
```

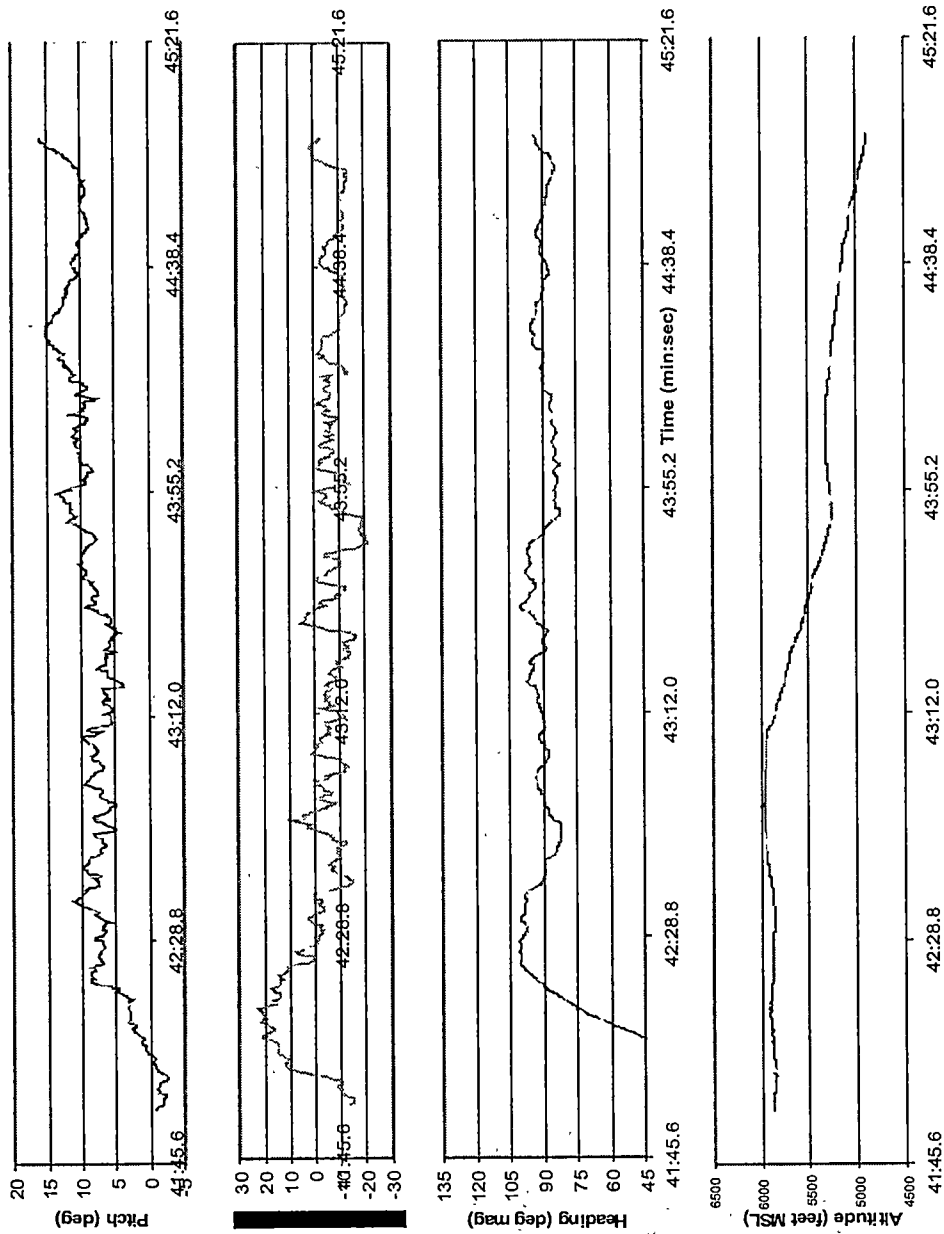


Figure E-1. Pitch, Roll, Heading, Altitude Trace

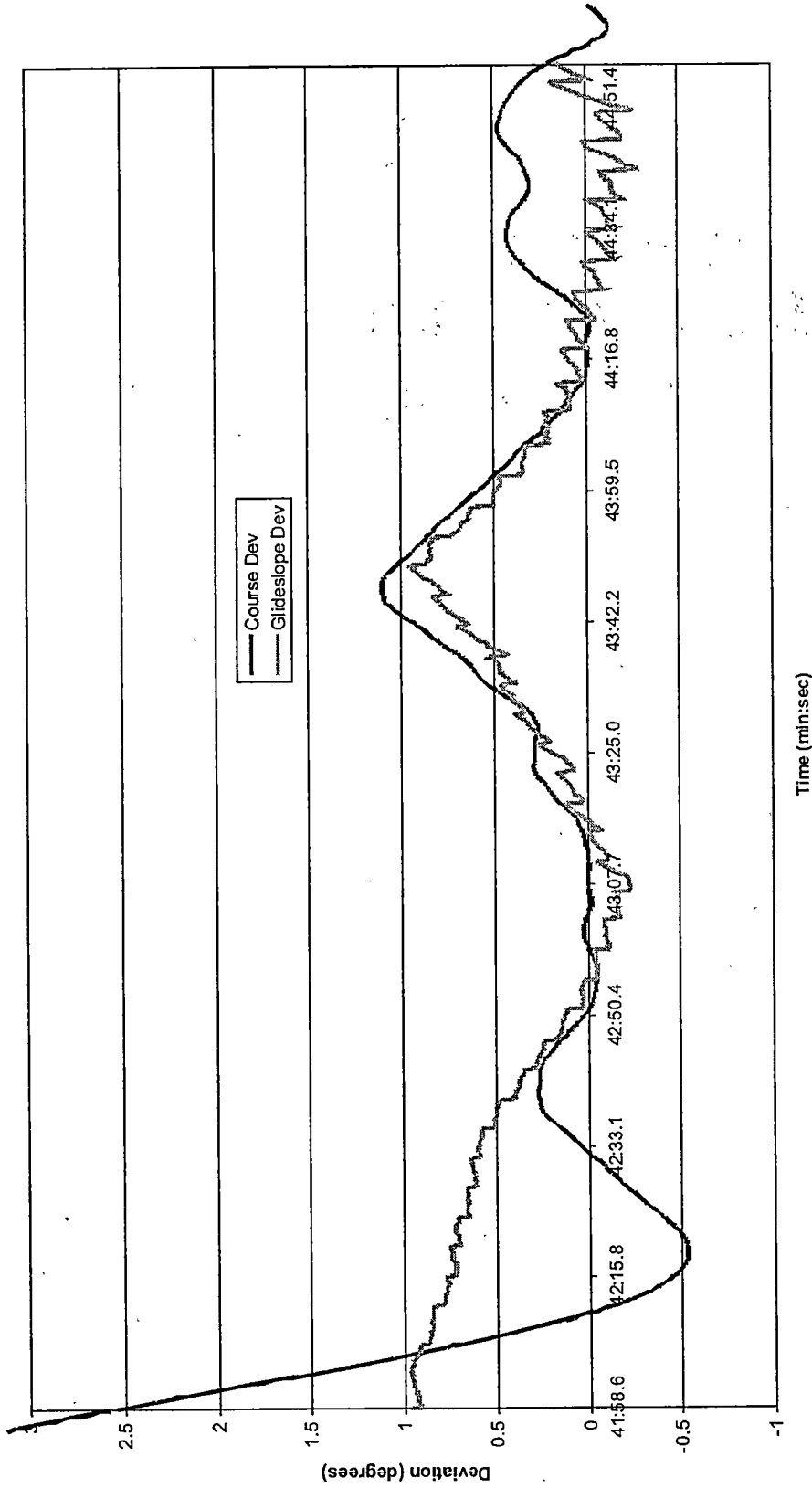


Figure E-2. Course/Glideslope Deviation Plot

Appendix F. Program Code

Visual Basic was used to produce the software for this project because of the rapid prototyping features and ease-of-creation. Although major portions of the program functionality is tied up in the layout of the forms and the settings of their properties, this code serves as an example of the coding for the instruments. The code for the Main form is included first, which handles initialization and input/output. The gauge instrument code follows and is representative of that for the other instrument types. All code is © Copyright 2000, Mark Johnson.

Code for Main Form

```
Private Declare Function OSWinHelp% Lib "user32" Alias "WinHelpA" (ByVal hwnd&, ByVal  
HelpFile$, ByVal wCommand%, dwData As Any)  
Private Declare Function Sleep% Lib "kernel32" (ByVal dwMillisec%)  
Private Declare Function GetSystemTime% Lib "kernel32" (ByRef lpSystemTime As  
LPSYSTEMTIME)
```

Private Type LPSYSTEMTIME

```
wYear As Byte  
wYear2 As Byte  
wMonth As Byte  
wMonth2 As Byte  
wDayOfWeek As Byte  
wDayOfWeek2 As Byte  
wDay As Byte  
wDay2 As Byte  
wHour As Byte  
wHour2 As Byte  
wMinute As Byte  
wMinute2 As Byte  
wSecond As Byte  
wSecond2 As Byte  
wMilliseconds As Byte  
wMilliseconds2 As Byte
```

End Type

```
Const pi = 3.141592656  
Dim dCalRoll As Double  
Dim dCalPitch As Double
```

```
Private SimMode As Boolean  
Private Recorder As Boolean  
Private nFileNumber As Integer
```

```
Dim oldX As Integer, oldY As Integer  
Dim test As Integer
```

```

Dim nPitch As Double, nPitchRate As Double, nRoll As Double, nHdg As Double
Dim nAS As Double, nAlt As Double, nVSI As Double
Dim dCalVel As Double 'Velocity input to IMU
Dim dCalAlt As Double
Dim dCalNz As Double 'Cal for vert. accel
Dim dCalNx As Double 'Cal for horiz. accel
Dim dCalZeroVSI As Double 'Rate to zero out VSI
Dim dCalAltCorr As Double 'Rate to zero out alt error
Dim lSimRWHeight As Long 'Height of sim Runway

```

```

Private dCalVSIA(50) As Double

```

```

Private dData(15) As Double 'The data from the gyro
'0 = Pitch (deg)
'1 = Roll (deg)
'2 = Yaw (deg)
'3 = Heading (deg)
'4 = Airspeed (knots)
'5 = Altitude (feet)
'6 = VSI (feet per minute)
'8 = Latitude (miles from GM)
'9 = Long (miles from equator)
'11 = Course deviation (deg)
'12 = Glideslope deviation (deg)

```

```

Private ctlInst(100) As Control 'The controls on the display
Private cCtls As Integer 'Number of controls on the display

```

```

Public Function GetData(Index As Integer) As Double
    GetData = dData(Index)
End Function

```

```

Public Sub Form_Draw()
    Dim i As Integer
    Dim latErr As Double, longErr As Double
    Dim X As Double, Z As Double
    Dim DME As Double

    Line (100, 100)-(4000, 600), BackColor, BF
    CurrentX = 100
    CurrentY = 100
    latErr = dData(8) - 1
    longErr = dData(9) + 7
    If longErr = 0 Then longErr = 0.000001
    DME = CInt(Sqr(latErr ^ 2 + longErr ^ 2) * 10) / 10
    X = latErr / longErr 'Angular error of course
    Z = (dData(5) - lSimRWHeight) / 5280 / DME 'Angular error of altitude
    'Arcsin(X) = Atn(X / Sqr(X ^ 2 + 1))
    'But for exceptionally small angles Arcsin(X) ~ X
    dData(11) = CLng((X * 180 / pi) * 100) / -100
    dData(12) = 3.5 - CLng((Z * 180 / pi) * 100) / 100

    For i = 0 To cCtls - 1
        ctlInst(i).Update
    Next i
End Sub

```

Public Sub KD(KeyCode As Integer, Shift As Integer)

Form_KeyDown KeyCode, Shift

End Sub

Private Sub Form_DblClick()

SetOptions

End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)

Dim Buffer As Variant

Dim strData As String

Select Case KeyCode

Case vbKeyUp

nAS = nAS + 2

Case vbKeyDown

nAS = nAS - 2

Case vbKeyEscape

mnuFileExit_Click

Case vbKeyD 'Show CommPort data

MsgBox ctlComm.Input, , "Data"

Case vbKeyE

MsgBox "Error=" & CStr(ctlComm.CommEvent)

Case vbKeyI 're-Initialize IMU

dCalPitch = 0

MsgBox "Initializing"

Buffer = ctlComm.Input

ctlComm.Output = "!"

Sleep (12000) 'Wait 12 sec for initialization

Buffer = ctlComm.Input

Do

Sleep (2000)

Buffer = ctlComm.Input

strData = CStr(Buffer)

Loop Until (Len(strData) > 40)

ParseIMUString strData

dCalPitch = dData(0) 'Re-calibrate pitch angle

dData(6) = 0 'Zero out VSI

dData(4) = dCalVel 'Set velocity to cal speed

If dCalAlt > 0 Then dData(5) = dCalAlt

Case vbKeyR 'Toggle data Recorder

If Recorder = True Then

Recorder = False

Close #nFileNumber

Else

Recorder = True

nFileNumber = FreeFile

Open "C:\Data" + CStr(nFileNumber) + ".mfd" For Output As nFileNumber

Write #nFileNumber, CStr(Now); "Pitch"; "Roll"; "Heading"; "Airspeed"; "Altitude"; "VSI";
"Latitude"; "Longitude"; "Course Dev"; "G/S Dev"

End If

```

Case vbKeyS      'Toggle Sim mode
    If SimMode = True Then
        SimMode = False
    Else
        SimMode = True
    End If
    dData(6) = 0      'Zero out VSI
    dData(4) = dCalVel 'Set velocity to cal speed
    If dCalAlt > 0 Then dData(5) = dCalAlt

Case vbKeyZ      'Zero out lat/long for sim approach
    dData(8) = 0
    dData(9) = 0
    nAlt = 4500     'Set altitude

Case Else
    'Send any other keystrokes to the IMU
    'Buffer = Chr$(KeyCode)
    'ctlComm.Output = Chr$(KeyCode)
    'Sleep (200)
    'MsgBox ctlComm.Input, , "Data"
End Select
End Sub

Private Sub Form_Load()
    Dim sFile As String
    Dim strRet As String * 256

    ctlComm.CommPort = 1
    ctlComm.Settings = "9600,N,8,1"
    ctlComm.InputLen = 0
    ctlComm.PortOpen = True

    SimMode = True
    sFile = "defaults.mfd"

    'Read-in defaults
    dCalNz = GetSetting(App.Title, "General", "CalNz", 1)
    dCalNx = GetSetting(App.Title, "General", "CalNx", 0)
    dCalZeroVSI = GetSetting(App.Title, "General", "CalZeroVSI", 0)
    dCalAltCorr = GetSetting(App.Title, "General", "CalAltCorr", 0)
    lSimRWHeight = GetSetting(App.Title, "General", "SimRWHeight", 4000)
    sFile = GetSetting(App.Title, "General", "DisplayFileName", "general.mfd")
    dlgCmn.FileName = sFile
    'Read in display
    Err = GPPS("General", "NumberControls", "0", strRet, 256, sFile)
    If CInt(strRet) = 0 Then
        LoadDefault
        Exit Sub
    End If

    LoadFile (sFile)
End Sub

Private Sub LoadDefault()

```

```
Me.Left = GetSetting(App.Title, "Settings", "MainLeft", 1000)
Me.Top = GetSetting(App.Title, "Settings", "MainTop", 1000)
Me.Width = GetSetting(App.Title, "Settings", "MainWidth", 6500)
Me.Height = GetSetting(App.Title, "Settings", "MainHeight", 6500)
```

```
BackColor = 0
ForeColor = QBColor(2)
```

```
Move 100, 100, 12800, 9600
nPitch = nPitchRate = nRoll = 0
nHdg = 0
nAS = 0
nAlt = 2000
test = oldX = oldY = 0
```

```
Set ctlInst(0) = Me.Controls.Add("MFD.Gyro", "ctlAtt")
Set ctlInst(1) = Me.Controls.Add("MFD.Strip", "ctlAS")
Set ctlInst(2) = Me.Controls.Add("MFD.Gauge", "ctlAlt")
Set ctlInst(3) = Me.Controls.Add("MFD.Gauge", "ctlVSI")
Set ctlInst(4) = Me.Controls.Add("MFD.Text", "ctlHdg")
cCtls = 5
```

```
With ctlInst(0)
    .Visible = True
    .Left = 3000
    .Top = 1000
    .Width = 2000
    .Height = 2000
    .Square = False
    .ZOrder 1
    .nDataPitch = 0
    .nDataRoll = 1
    .nSymbol = 0
    .nSymSize = 40
    .nPitchScale = 2
    .ILS = True
End With
```

```
With ctlInst(1)
    .minV = 0
    .maxV = 300
    .Increment = 25
    .Visible = True
    .Left = 500
    .Top = 1000
    .Width = 1000
    .Height = 4000
    .nDataField = 4
    .SetFont 16, "Arial", False, False
End With
```

```
With ctlInst(2)
    .stAng = 90
    .endAng = 360
    .TickInt = 100
    .maxV = 1000
```

```

        .Continuous = True
        .Visible = True
        .Left = 5500
        .Top = 1000
        .Width = 2000
        .nDataField = 5
    End With

    With ctrlInst(3)
        .stAng = -30
        .endAng = 300
        .minV = -5000
        .maxV = 5000
        .TickInt = 500
        .SetColor 1, QBColor(3)
        .Visible = True
        .Left = 5500
        .Top = 4000
        .Width = 2000
        .nDataField = 6
    End With

    With ctrlInst(4)
        .Visible = True
        .Left = 3500
        .Top = 4000
        .Width = 800
        .Height = 500
        .nDataField = 3
        .nMinDigits = 3
        .SetFont 20, "Arial", False, False
    End With
End Sub

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If X < oldX Then nRoll = nRoll - 1
    If X > oldX Then nRoll = nRoll + 1
    If nRoll > 180 Then nRoll = -179
    If nRoll < -180 Then nRoll = 179

    If Y > oldY Then nPitchRate = nPitchRate + 1
    If Y < oldY Then nPitchRate = nPitchRate - 1

    oldX = X
    oldY = Y
End Sub

Private Sub Form_Unload(Cancel As Integer)
    Dim i As Integer

    'close all sub forms
    For i = Forms.Count - 1 To 1 Step -1
        Unload Forms(i)
    Next

```

```

If Me.WindowState <> vbMinimized Then
    SaveSetting App.Title, "Settings", "MainLeft", Me.Left
    SaveSetting App.Title, "Settings", "MainTop", Me.Top
    SaveSetting App.Title, "Settings", "MainWidth", Me.Width
    SaveSetting App.Title, "Settings", "MainHeight", Me.Height
End If

'Save default info
SaveSetting App.Title, "General", "CalNz", dCalNz
SaveSetting App.Title, "General", "CalNx", dCalNx
SaveSetting App.Title, "General", "CalZeroVSI", dCalZeroVSI
SaveSetting App.Title, "General", "CalAltCorr", dCalAltCorr
SaveSetting App.Title, "General", "SimRWHeight", lSimRWHeight
If Len(dlgCmn.FileName) > 1 Then
    SaveSetting App.Title, "General", "DisplayFileName", dlgCmn.FileName
End If
ctlComm.PortOpen = False
End Sub

Private Sub mnuViewOptions_Click()
    frmOptions.Show vbModal, Me
End Sub

Private Sub mnuFileExit_Click()
    Unload Me
End Sub

Private Sub mnuFileSaveAs_Click()
    Dim i As Integer

    dlgCmn.Flags = cdIOFNHideReadOnly
    dlgCmn.Filter = "MFD Files (*.mfd)|*.mfd|All Files (*.*)|*.*"
    dlgCmn.FilterIndex = 1
    dlgCmn.ShowSave
    FileSaveData
End Sub

Private Sub mnuFileSave_Click()
    If Len(dlgCmn.FileName) = 0 Then
        mnuFileSaveAs_Click
        Exit Sub
    End If
    FileSaveData
End Sub

Private Sub FileSaveData()
    Dim strSect As String
    Dim fn As String
    Dim Size As Integer
    Dim Name As String
    Dim Bold As Boolean, Italic As Boolean

    fn = dlgCmn.FileName

    Err = WPPS("General", "NumberControls", CStr(cCtls), fn)
    Err = WPPS("General", "Top", CStr(Me.Top), fn)

```

```

Err = WPPS("General", "Left", CStr(Me.Left), fn)
Err = WPPS("General", "Height", CStr(Me.Height), fn)
Err = WPPS("General", "Width", CStr(Me.Width), fn)

For i = 0 To cCtrls - 1
    strSect = "Control" & CStr(i)
    'Save the Control-unique data
    ctlInst(i).Save (i)

    'Save the general control data
    Err = WPPS(strSect, "Top", CStr(ctlInst(i).Top), fn)
    Err = WPPS(strSect, "Left", CStr(ctlInst(i).Left), fn)
    Err = WPPS(strSect, "Height", CStr(ctlInst(i).Height), fn)
    Err = WPPS(strSect, "Width", CStr(ctlInst(i).Width), fn)
    Err = WPPS(strSect, "DrawWidth", CStr(ctlInst(i).GetDrawWidth), fn)
    Err = WPPS(strSect, "ForeColor", CStr(ctlInst(i).GetColor(1)), fn)
    Err = WPPS(strSect, "BackColor", CStr(ctlInst(i).GetColor(0)), fn)

    ctlInst(i).GetFont Size, Name, Bold, Italic
    Err = WPPS(strSect, "FontName", CStr(Name), fn)
    Err = WPPS(strSect, "FontSize", CStr(Size), fn)
    Err = WPPS(strSect, "FontBold", CStr(Bold), fn)
    Err = WPPS(strSect, "FontItalic", CStr(Italic), fn)
Next i
End Sub

```

Private Sub mnuFileOpen_Click()

```

Dim sFile As String

With dlgCmn
    .DialogTitle = "Open"
    .CancelError = False
    'ToDo: set the flags and attributes of the common dialog control
    .Filter = "MFD Files (*.mfd)|*.mfd|All Files (*.*)|*.*"
    .ShowOpen
    If Len(.FileName) = 0 Then
        Exit Sub
    End If
    sFile = .FileName
End With
LoadFile (sFile)
End Sub

```

Private Sub LoadFile(sFile As String)

```

Dim strRet As String * 256
Dim strR As String

Dim sSect As String
Dim Size As Integer
Dim Name As String
Dim Bold As Boolean, Italic As Boolean

On Error Resume Next
'Remove all current controls
For i = Me.Controls.Count To 1 Step -1
    Me.Controls.Remove i

```

Next i

Err = GPPS("General", "NumberControls", "0", strRet, 256, sFile)
cCtls = CInt(strRet)

For i = 0 To cCtls - 1

sSect = "Control" & CStr(i)

Err = GPPS(sSect, "Type", "None", strRet, 256, sFile)

strR = Left(strRet, InStr(strRet, Chr(0)) - 1)

Select Case strR

Case "Gauge"

Set ctlInst(i) = Me.Controls.Add("MFD.Gauge", "ctlGauge" & CStr(i))

MsgBox "Created Gauge"

Case "Gyro"

Set ctlInst(i) = Me.Controls.Add("MFD.Gyro", "ctlGyro" & CStr(i))

Case "TextBox"

Set ctlInst(i) = Me.Controls.Add("MFD.Text", "ctlText" & CStr(i))

MsgBox "Created TextBox"

Case "Strip"

Set ctlInst(i) = Me.Controls.Add("MFD.Strip", "ctlStrip" & CStr(i))

MsgBox "Created Strip"

Case Else

MsgBox "No match!!"

End Select

'Load the instrument-unique data

ctlInst(i).Load (i)

'Load data common to all instruments

Err = GPPS(sSect, "Top", "0", strRet, 256, sFile)

ctlInst(i).Top = CLng(strRet)

Err = GPPS(sSect, "Left", "0", strRet, 256, sFile)

ctlInst(i).Left = CLng(strRet)

Err = GPPS(sSect, "Height", "0", strRet, 256, sFile)

ctlInst(i).Height = CLng(strRet)

Err = GPPS(sSect, "Width", "0", strRet, 256, sFile)

ctlInst(i).Width = CLng(strRet)

Err = GPPS(sSect, "DrawWidth", "2", strRet, 256, sFile)

ctlInst(i).SetDrawWidth CLng(strRet)

Err = GPPS(sSect, "ForeColor", "256", strRet, 256, sFile)

ctlInst(i).SetColor 1, CLng(strRet)

Err = GPPS(sSect, "BackColor", "0", strRet, 256, sFile)

ctlInst(i).SetColor 0, CLng(strRet)

Err = GPPS(sSect, "FontName", "Arial", strRet, 256, sFile)

Name = strRet

Err = GPPS(sSect, "FontSize", "12", strRet, 256, sFile)

Size = CLng(strRet)

Err = GPPS(sSect, "FontBold", "False", strRet, 256, sFile)

Bold = CBool(strRet)

Err = GPPS(sSect, "FontItalic", "False", strRet, 256, sFile)

Italic = CBool(strRet)

ctlInst(i).SetFont Size, Name, Bold, Italic

ctlInst(i).Visible = True

Next i

End Sub

Private Sub Timer1_Timer()

Dim lpBuffer As LPSYSTEMTIME

Dim IMSec As Long

'First check for recording

If Recorder = True Then

GetSystemTime lpBuffer

IMSec = lpBuffer.wMilliseconds + lpBuffer.wMilliseconds2 * 256

IMSec = CLng(IMSec / 100)

Write #nFileNumber, CStr(lpBuffer.wMinute) + ":" + CStr(lpBuffer.wSecond) + "." +
CStr(IMSec); _

CStr(CLng(dData(0) * 1000) / 1000); _

CStr(CLng(dData(1) * 1000) / 1000); _

CStr(CLng(dData(3) * 1000) / 1000); _

CStr(CLng(dData(4) * 1000) / 1000); _

CStr(CLng(dData(5) * 1000) / 1000); _

CStr(CLng(dData(6) * 1000) / 1000); _

CStr(CLng(dData(8) * 1000) / 1000); _

CStr(CLng(dData(9) * 1000) / 1000); _

CStr(CLng(dData(11) * 1000) / 1000); _

CStr(CLng(dData(12) * 1000) / 1000)

End If

If SimMode = True Then

'Update Pitch Angle

nPitch = Fix((nPitch + nPitchRate / 10 * Cos(nRoll * pi / 180)) * 100) / 100

If nPitch > 90 Then

nPitch = 180 - nPitch

nHdg = nHdg + 180

If nRoll > 0 Then

nRoll = nRoll - 180

Else

nRoll = nRoll + 180

End If

End If

If nPitch < -90 Then

nPitch = -180 - nPitch

nHdg = nHdg + 180

If nRoll > 0 Then

nRoll = nRoll - 180

Else

nRoll = nRoll + 180

End If

End If

If nHdg > 360 Then nHdg = nHdg - 360

' Update VSI and subsequently A/S and alt

nVSI = Sin(nPitch * pi / 180) * nAS * (5280 / 60)

nAS = nAS - nVSI / 40000

If nAS < 0 Then nAS = 0

nAlt = nAlt + nVSI * (Timer1.Interval / 60000)

'Update heading

nHdg = Fix((nHdg + Sin(nRoll * pi / 180) * 2) * 100) / 100

```

If nHdg > 360 Then nHdg = nHdg - 360
If Int(nHdg) < 1 Then nHdg = nHdg + 360

dData(0) = nPitch
dData(1) = nRoll
' dData(2) = nYaw
dData(3) = nHdg
dData(4) = nAS
dData(5) = nAlt
dData(6) = nVSI
GoTo EndSub
End If 'Sim Mode

'Get data from Comm port
Dim strIn As String
If ctlComm.InBufferCount > 0 Then
    strIn = ctlComm.Input
    MsgBox strIn, , "Port Data"
    ParseMUString strIn
Else
    If ctlComm.CommEvent > 0 Then
        MsgBox "Error=" + CStr(ctlComm.CommEvent)
    End If
End If

EndSub:
    UpdateLatLong
    Form_Draw
End Sub

Private Sub ParseMUString(strData As String)
    Dim i As Integer
    Dim dTemp As Double
    Dim stIndex As Integer
    Dim endIndex As Integer

    For i = 1 To Len(strData)
        If (Mid(strData, i, 1) = "l") Or (Mid(strData, i, 1) = "i") Then
            stIndex = i
            i = Len(strData)
        End If
    Next i

    For i = stIndex To Len(strData)
        If Mid(strData, i, 1) = Chr$(13) Then
            endIndex = i
            i = Len(strData)
        End If
    Next i

    strData = Mid(strData, stIndex, endIndex - stIndex)
    MsgBox strData, , "Port Data"
    dData(1) = CDBl(Mid(strData, 3, 6)) - dCalRoll 'Roll
    dData(0) = CDBl(Mid(strData, 10, 5)) - dCalPitch 'Pitch
    dData(3) = CDBl(Mid(strData, 16, 5)) 'Heading

```

```

'Integrate horizontal accel to get airspeed
'      oldAS +      nx      + cal * Int/1000 * 32 f/s^2 * 3600 sec/hour / 5280 ft/mile
'dData(4) = dData(4) + (CDBl(Mid(strData, 22, 5)) + dCalNx) * Timer1.Interval * 0.02181818
dCalVel = CDBl(Mid(strData, 46, 6)) * 0.62137 'Convert KPH to mph
dData(4) = dCalVel

' *****
' Altitude and VSI
' *****

dTemp = CInt(CDBl(Mid(strData, 28, 5)) * 50) / 50
'Integrate vertical accel (nz) to get VSI (fpm)
'      oldVSI -      nz      + cal * Int/1000 * 32 f/s^2 * 60 sec/min
dData(6) = dData(6) - (dTemp + dCalNz) * Timer1.Interval / 1.92

'Integrate vertical speed to get altitude
'      oldAlt + VSI * Interval / 60 sec/min
dData(5) = dData(5) + (dData(6) * Timer1.Interval / 60000)

'Convert User data 1 voltage to altitude
dTemp = CDBl(Mid(strData, 34, 5))
dCalAlt = 769.69 * dTemp ^ 2 - 12219 * dTemp + 37544

'Correct altitude for error
dData(5) = dData(5) - (dData(5) - dCalAlt) * dCalAltCorr

'Store this new corrected altitude
For i = 49 To 0 Step -1
    dCalVSIA(i + 1) = dCalVSIA(i)
Next i
If dCalVSIA(50) = 0 Then dCalVSIA(50) = dCalAlt
dCalVSIA(0) = dCalAlt 'dData(5) = dCalAlt
'Use altitude change over time to smooth VSI
'Diff VSI = (CurrAlt - PastAlt) / TimeInt
dTemp = (dCalAlt - dCalVSIA(50)) / (Timer1.Interval * 0.05)
' VSI = VSI - ( VSI - DiffVSI) * 0.1
dData(6) = dData(6) - (dData(6) - dTemp) * dCalZeroVSI
End Sub

Private Sub UpdateLatLong()
'Latitude = Latitude + sin(Heading) * Airspeed * Time
dData(8) = dData(8) + Cos(dData(3) * pi / 180) * dData(4) * Timer1.Interval / 1000 / 3600

'Longitude = Long + Cos(Heading) * Airspeed * Time
dData(9) = dData(9) + Sin(dData(3) * pi / 180) * dData(4) * Timer1.Interval / 1000 / -3600
End Sub

Public Sub Repos(X As Single, Y As Single)
If IsNull(Me.ActiveControl) Then MsgBox "No active control"
Me.ActiveControl.Left = Me.ActiveControl.Left + X
Me.ActiveControl.Top = Me.ActiveControl.Top + Y
End Sub

Sub SetOptions()
Dim Size As Integer
Dim Name As String

```

Dim Bold As Boolean, Italic As Boolean

Dim i As Integer, j As Integer

Load frmOptions

'Set all the default values

With frmOptions

 ' First Tab

 .txtDataField = frmMain.ActiveControl.nDataField

 .txtLeft = frmMain.ActiveControl.Left

 .txtTop = frmMain.ActiveControl.Top

 .txtWidth = frmMain.ActiveControl.Width

 .txtHeight = frmMain.ActiveControl.Height

 .txtDrawWidth = frmMain.ActiveControl.GetDrawWidth

 .txtForeColor = frmMain.ActiveControl.GetColor(1)

 .txtForeColor.ForeColor = frmMain.ActiveControl.GetColor(1)

 .txtBackColor = frmMain.ActiveControl.GetColor(0)

 .txtBackColor.BackColor = frmMain.ActiveControl.GetColor(0)

 frmMain.ActiveControl.GetFont Size, Name, Bold, Italic

 .cmdlg.FontName = Name

 .cmdlg.FontSize = Size

 .cmdlg.FontBold = Bold

 .cmdlg.FontItalic = Italic

 'Gauge Tab

 .txtGaMin = 0

 .txtGaMax = 100

 .txtGaStartA = 0

 .txtGaEndA = 360

 .txtGaTickInt = 10

 .txtGaPointWidth = 4

 .chkGACont = 0

 'Gyro Tab

 .txtDataPitch = 0

 .txtDataRoll = 1

 .txtSkyColor = QBColor(11)

 .txtGroundColor = RGB(187, 132, 91)

 .txtLineColor = QBColor(2)

 .chkSquare = 0

 .optSymbol.Item(0) = True

 .txtGySymSize = 40

 .chkGyILS = 0

 .txtGyPitchScale = 1

 'Text Tab

 .txtTxMinDig = 0

 'Strip Tab

 .txtStMinVal = 0

 .txtStMaxVal = 200

 .txtStIncr = 10

 .txtStScaleRng = 200

 .chkStShowPointer = 1

```

'Calibrations Tab
.txtCalNx = dCalNx
.txtCalNz = dCalNz
.txtZeroVSI = dCalZeroVSI
.txtZeroAlt = dCalAltCorr
.txtCalSimRWH = ISimRWHeight

End With

'Override default values with actual instrument values
Select Case frmMain.ActiveControl.GetType
Case "Gyro"
    With frmOptions
        .optInst.Item(0) = True
        .txtDataPitch = frmMain.ActiveControl.nDataPitch
        .txtDataRoll = frmMain.ActiveControl.nDataRoll
        .txtSkyColor = frmMain.ActiveControl.nSkyColor
        .txtGroundColor = frmMain.ActiveControl.nGroundColor
        .txtLineColor = frmMain.ActiveControl.nLineColor
        .txtGySymSize = CStr(frmMain.ActiveControl.nSymSize)
        .optSymbol.Item(frmMain.ActiveControl.nSymbol) = True
        .chkSquare = If(frmMain.ActiveControl.Square, 1, 0)
        .chkGyILS = If(frmMain.ActiveControl.ILS, 1, 0)
        .txtGyPitchScale = frmMain.ActiveControl.nPitchScale
    End With
Case "Gauge"
    With frmOptions
        .optInst.Item(1) = True
        .txtGaMin = frmMain.ActiveControl.minV
        .txtGaMax = frmMain.ActiveControl.maxV
        .txtGaStartA = frmMain.ActiveControl.stAng
        .txtGaEndA = frmMain.ActiveControl.endAng
        .txtGaTickInt = frmMain.ActiveControl.TickInt
        .txtGaPointWidth = frmMain.ActiveControl.nPointerWidth
        .chkGACont = If(frmMain.ActiveControl.Continuous, 1, 0)
    End With
Case "Text"
    With frmOptions
        .optInst.Item(2) = True
        .txtTxMinDig = frmMain.ActiveControl.nMinDigits
    End With
Case "Strip"
    With frmOptions
        .optInst.Item(3) = True
        .txtStMinVal = frmMain.ActiveControl.minV
        .txtStMaxVal = frmMain.ActiveControl.maxV
        .txtStIncr = frmMain.ActiveControl.Increment
        .txtStScaleRng = frmMain.ActiveControl.ScaleRange
        .chkStShowPointer = If(frmMain.ActiveControl.ShowPointer, 1, 0)
    End With
Case Else
'Clicked in empty space of display
End Select

frmOptions.Show vbModal, Me

```

```

If frmOptions.Delete Then
    For i = 0 To cCtls - 1
        If ctlInst(i) Is frmMain.ActiveControl Then
            j = i
            i = cCtls
        End If
    Next i

    'Remove control from form
    For i = 0 To frmMain.Controls.Count - 1
        If ctlInst(j) Is frmMain.Controls(i) Then
            frmMain.Controls.Remove i
            i = frmMain.Controls.Count
        End If
    Next i

    'Remove reference from my collection
    For i = j To cCtls - 2
        Set ctlInst(i) = ctlInst(i + 1)
    Next i
    cCtls = cCtls - 1
    Unload frmOptions
    Exit Sub
End If

If frmOptions.Add Then
    For i = 0 To 3
        If frmOptions.optInst.Item(i) = True Then
            j = i
            i = 4
        End If
    Next i

    Select Case j
    Case 0
        Set ctlInst(cCtls) = Me.Controls.Add("MFD.Gyro", "ctlGyro" & cCtls)
    Case 1
        Set ctlInst(cCtls) = Me.Controls.Add("MFD.Gauge", "ctlGauge" & cCtls)
    Case 2
        Set ctlInst(cCtls) = Me.Controls.Add("MFD.Text", "ctlTexta" & cCtls)
    Case 3
        Set ctlInst(cCtls) = Me.Controls.Add("MFD.Strip", "ctlStrip" & cCtls)
    End Select

    ctlInst(cCtls).Visible = True
    ctlInst(cCtls).SetFocus 'Make this the active control
    cCtls = cCtls + 1
    frmOptions.Accept = True
End If

If Not (frmOptions.Accept) Then
    Unload frmOptions
    Exit Sub
End If

dCalNz = frmOptions.txtCalNz

```

```

dCalNx = frmOptions.txtCalNx
dCalZeroVSI = frmOptions.txtZeroVSI
dCalAltCorr = frmOptions.txtZeroAlt
ISimRWHeight = frmOptions.txtCalSimRWH

With frmMain.ActiveControl
    'Tab 1 settings
    .Move CLng(frmOptions.txtLeft), CLng(frmOptions.txtTop), CLng(frmOptions.txtWidth),
CLng(frmOptions.txtHeight)
    .Size = CLng(frmOptions.txtWidth)
    .SetColor 1, CLng(frmOptions.txtForeColor)
    .SetColor 0, CLng(frmOptions.txtBackColor)
    .SetDrawWidth CInt(frmOptions.txtDrawWidth)
    .SetFont frmOptions.comdlg.FontSize, frmOptions.comdlg.FontName,
frmOptions.comdlg.FontBold, frmOptions.comdlg.FontItalic
    .nDataField = frmOptions.txtDataField

    Select Case .GetType
    Case "Gyro"
        'Gyro settings
        .nSkyColor = CLng(frmOptions.txtSkyColor)
        .nGroundColor = CLng(frmOptions.txtGroundColor)
        .nLineColor = CLng(frmOptions.txtLineColor)
        .nPitchScale = CInt(frmOptions.txtGyPitchScale)
        .Square = CBool(frmOptions.chkSquare)
        For i = 0 To 3
            If frmOptions.optSymbol.Item(i).Value = True Then
                frmMain.ActiveControl.nSymbol = i
                i = 5
            End If
        Next i
        .nSymSize = CInt(frmOptions.txtGySymSize)
        .ILS = If(frmOptions.chkGylLS, True, False)

    Case "Gauge"
        'Gauge settings
        .minV = CLng(frmOptions.txtGaMin)
        .maxV = CLng(frmOptions.txtGaMax)
        .stAng = CInt(frmOptions.txtGaStartA)
        .endAng = CInt(frmOptions.txtGaEndA)
        .TickInt = CLng(frmOptions.txtGaTickInt)
        .nPointerWidth = CInt(frmOptions.txtGaPointWidth)
        .Continuous = CBool(frmOptions.chkGACont)

    Case "Text"
        'Text settings
        .nMinDigits = CLng(frmOptions.txtTxMinDig)

    Case "Strip"
        'Strip settings
        .minV = CLng(frmOptions.txtStMinVal)
        .maxV = CLng(frmOptions.txtStMaxVal)
        .Increment = CLng(frmOptions.txtStIncr)
        .ScaleRange = CLng(frmOptions.txtStScaleRng)
        .ShowPointer = CBool(frmOptions.chkStShowPointer)
    Case Else

```

```
End Select
End With
On Error GoTo 0

Unload frmOptions
End Sub
```

Gauge Instrument Code

```
Public minV, maxV As Long
Public stAng, endAng As Integer
Public nPointerWidth As Integer
Public TickInt As Long
Public Continuous As Boolean
Public nDataField As Integer 'This is the index in the data array the info is from
```

```
Private Repos As Boolean
Private oldX As Single, oldY As Single
```

```
Private Value As Long
Const pi = 3.141592656
```

Private Sub UserControl_Initialize()

```
    minV = 0
    maxV = 100
    TickInt = 10
    Continuous = False
    Repos = False
```

```
    Value = 0
    'Size = 2000
    stAng = 240
    endAng = 300
    nPointerWidth = 4
```

End Sub

Private Sub UserControl_KeyDown(KeyCode As Integer, Shift As Integer)

```
    frmMain.KD KeyCode, Shift
```

End Sub

Private Sub UserControl_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)

```
    Repos = True
    oldX = X
    oldY = Y
```

End Sub

Private Sub UserControl_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)

```
    If Repos = False Then Exit Sub
    frmMain.Repos X - oldX, Y - oldY
```

End Sub

Private Sub UserControl_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)

```
    Repos = False
```

End Sub

Private Sub UserControl_Paint()

```
    Dim i As Integer
    Dim strVal As String
    Dim Radius As Long
```

```

Dim Angle As Double
Dim tsX As Long, tsY As Long, endX As Long, endY As Long
Dim cX As Long, cY As Long

'Ensure sizing
Height = Width
Radius = Width / 2 - (DrawWidth * Screen.TwipsPerPixelX)

cX = Width / 2
cY = Height / 2

'Erase gauge
FillStyle = 0
FillColor = BackColor
Circle (cX, cY), Radius, BackColor

'Draw new gauge
FillStyle = 1
Circle (cX, cY), Radius

For i = minV To maxV Step TickInt
    Angle = (stAng * pi / 180 - (endAng * pi / 180 * ((i - minV) / (maxV - minV))))
    tsX = cX + Cos(Angle) * Radius * 0.9
    tsY = cY - Sin(Angle) * Radius * 0.9
    endX = cX + Cos(Angle) * Radius
    endY = cY - Sin(Angle) * Radius
    Line (tsX, tsY)-(endX, endY)
Next i

'Write text value also
strVal = CStr(Value)
CurrentX = Radius - TextWidth(strVal) / 2
CurrentY = 3 * Radius / 2
Print Value

Angle = (stAng * pi / 180 - (endAng * pi / 180 * ((Value - minV) / (maxV - minV))))
endX = cX + Cos(Angle) * Radius
endY = cY - Sin(Angle) * Radius
DrawPointer cX, cY, endX, endY, Color
End Sub

Private Sub UserControl_DbClick()
    frmMain.SetOptions
End Sub

Public Function GetType()
    GetType = "Gauge"
End Function

Private Sub DrawPointer(X, Y, endX, endY, Color)
    Dim oldDW As Variant
    oldDW = DrawWidth
    DrawWidth = nPointerWidth
    Line (X, Y)-(endX, endY)
    DrawWidth = oldDW
End Sub

```

Public Sub SetColor(id, C)

 If id = 0 Then BackColor = C

 If id = 1 Then ForeColor = C

End Sub

Public Function GetColor(id As Integer) As Long

 If id = 0 Then GetColor = BackColor

 If id = 1 Then GetColor = ForeColor

End Function

Public Sub SetFont(Size, Name, Bold, Italic)

 Font.Size = Size

 Font.Name = Name

 Font.Bold = False

 Font.Italic = False

 If Bold Then Font.Bold = True

 If Italic Then Font.Italic = True

End Sub

Public Sub GetFont(Size As Integer, Name As String, Bold As Boolean, Italic As Boolean)

 Size = Font.Size

 Name = Font.Name

 Bold = Font.Bold

 Italic = Font.Italic

End Sub

Public Function GetDrawWidth() As Integer

 GetDrawWidth = DrawWidth

End Function

Public Sub SetDrawWidth(Value As Integer)

 DrawWidth = Value

End Sub

Public Sub Update()

 Dim Val As Double

 Val = frmMain.GetData(nDataField)

 Value = Val

 If Continuous = False Then

 If Val > maxV Then Value = maxV

 If Val < minV Then Value = minV

 End If

 Refresh

End Sub

Public Sub Save(Index As Integer)

 Dim strSect As String

 Dim fn As String

 strSect = "Control" & CStr(Index)

 fn = frmMain.dlgCmn.FileName

 Err = WPPS(strSect, "Type", "Gauge", fn)

 Err = WPPS(strSect, "DataField", CStr(nDataField), fn)

```

Err = WPPS(strSect, "MinVal", CStr(minV), fn)
Err = WPPS(strSect, "MaxVal", CStr(maxV), fn)
Err = WPPS(strSect, "StartAngle", CStr(stAng), fn)
Err = WPPS(strSect, "EndAngle", CStr(endAng), fn)
Err = WPPS(strSect, "TickInt", CStr(TickInt), fn)
Err = WPPS(strSect, "PointerWidth", CStr(nPointerWidth), fn)
Err = WPPS(strSect, "Continuous", CStr(Continuous), fn)
End Sub

```

Public Sub Load(Index As Integer)

```

Dim strSect As String
Dim fn As String
Dim strRet As String * 256

strSect = "Control" & CStr(Index)
fn = frmMain.dlgCmn.FileName

'MsgBox "Loading from " & fn & ", Section " & strSect

Err = GPPS(strSect, "DataField", "0", strRet, 256, fn)
nDataField = CInt(strRet)
Err = GPPS(strSect, "MinVal", "0", strRet, 256, fn)
minV = CLng(strRet)
Err = GPPS(strSect, "MaxVal", "100", strRet, 256, fn)
maxV = CLng(strRet)
Err = GPPS(strSect, "StartAngle", "0", strRet, 256, fn)
stAng = CInt(strRet)
Err = GPPS(strSect, "EndAngle", "360", strRet, 256, fn)
endAng = CInt(strRet)
Err = GPPS(strSect, "TickInt", "10", strRet, 256, fn)
TickInt = CLng(strRet)
Err = GPPS(strSect, "Continuous", "False", strRet, 256, fn)
Continuous = CBool(strRet)
End Sub

```

Vita

A self-taught computer programmer, Mark Johnson was raised throughout the United States and graduated from the University of California at Davis in 1989 with a Bachelor of Science in Electrical Engineering. He was immediately commissioned in the Navy and designated as a Naval Aviator in 1991.

After two operational deployments flying the FA-18 Hornet, he was selected for the US Navy Test Pilot School, graduating in December 1997. Subsequently, he was assigned to fly operational test missions in the FA-18 and F-16 aircraft. During this time he was one of the three founding members of Automated Profile Management, LLC, which developed software for managing user profiles on Windows-based networks.

Mark has accumulated over 2200 flight hours in more than 40 different aircraft types and over 300 carrier-arrested landings. He is proficient in several computer-programming languages, including C++ and Basic.