DATA MONITORING SOFTWARE FOR SSFDR

A PROJECT WORK DONE AT

Centre for Air Borne Systems, Bangalore - 37.

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

MASTER OF COMPUTER APPLICATIONS

OF BAHARATHIAR UNIVERSITY, COIMBATORE.

SUBMITTED BY

R.GOVINDARAJA

Reg.No: 9838M0504

P-577

Under the guidance of

External Guide

Mr. S.Ramanathan, Sc 'E',
Centre for Air Borne Systems,
Defence R & D Organization,
Bangalore – 560 037.

Internal Guide

Dr. S.Thangasamy, Ph.D.,
Head of the Department,
Dept of Computer Science & Engg,
Kumaraguru College of Technology,
Coimbatore-641006.

Department of Computer Science & Engineering

Kumaraguru College of Technology

Coimbatore, Tamil Nadu - 641006.

भारत सरकार - रक्षा मंत्रालय
Government of India - Ministry of Defence
रक्षा अनुसंधान तथा विकास संगठन
Defence Research & Development Organisation
वायुवाहित प्रणाली केन्द्र
Centre for Air Borne Systems
बेलूर, एमलूर तपाल
Belur, Yemlur P.O.
बेंगलूर - ५६० ०३७.
Bangalore - 560 037.

20th Apr 2001

CABS/4422/Trainee/Adm

CERTIFICATE

This is to certify that the project work entitled "**Data Monitoring Software for SSFDR**" submitted by **Mr. R.GOVINDARAJA, Reg.No. 9838M0504,** student of the final year M.C.A of Kumaraguru College of Technology, Coimbatore, affiliated to Bharathiar University, Coimbatore is a bonafide work carried out by him under my guidance, from December 2000 to April 2001 at Centre for Air Borne Systems (**CABS**), Defence Research and Development Organization (**DRDO**), **Ministry of Defence**, Bangalore-37. This is required for the fulfillment of the degree of Master of Computer Applications of **Bharathiar University** for year 2000 –2001.

Centre for Air Borne Systems
Bangalore

[illegible]

(Shri S.Ramanathan)
Scientist 'E'
CABS

S. Ramanathan
Scientist 'E'
Centre for Airborne Systems
Defence. R & D Organisation
Bangalore - 560 037

CERTIFICATE

This is to certify that the project work entitled

Data Monitoring Software for SSFDR

Submitted to the

Department of Computer Science and Engineering

Kumaraguru College of Technology

in partial fulfillment of the requirements for the award of the degree of Master of Computer applications is a record of original work done by **Mr.R.GOVINDARAJA, Reg.No. 9838M0504** during his period of study in the Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore under my supervision and this project work has not formed the basis of award of any Degree/Diploma Associateship /Fellowship or similar title to any candidate of any University.

[illegible] 26/4/01
Professor and Head

[illegible]
Staff-in-charge

Submitted to University Examination held on [illegible]

[illegible] 14/5/01
Internal Examiner

[illegible]
External Examiner

DECLARATION

I here by declare that the project work entitled

"""Data Monitoring Software for SSFDR"
done at
CENTRE FOR AIR BORNE SYSTEMS
(Min. of Defence)
BANGALORE

submitted in partial fulfillment of the requirements for the award of Degree of

Master of Computer Applications

is a report of original work done by me during the period of study in

KUMARAGURU COLLEGE OF TECHNOLOGY

(Affiliated to Bharathiar University)

Coimbatore-641 006

Under the Supervision of **Mr. S.Ramanathan, Sc 'E'**, Centre for Air Borne Systems, Bangalore and **Dr. S.Thangasamy, Ph.D.,** Head of the Department, Kumaraguru College of Technology, Coimbatore.

Name of the Candidate	Register Number	Signature
R.Govindaraja	**9838M0504**	[illegible]

Place : Coimbatore

Date : 26 · 04 · 2001

ACKNOWLEDGEMENT

I express my sincere thanks to ***Dr.K.Ramchand,*** *Director, CABS and* ***Mr.K.Tamil Mani,*** *Additional Director, CABS, Bangalore for providing me the opportunity to do the project in this sophisticated establishment.*

I express my profound gratitude and thanks to my external guide ***Mr.S.Ramanathan, Sc'E'*** *of CABS for all the blessings and guidance.*

I am bound to express my gratitude to ***Dr.K.Padmanaban,*** *Principal, Kumaraguru College of Technology (Affiliated to Bharathiar University), Coimbatore, TamilNadu for his constant encouragement throughout my course.*

I wish to express my grateful thanks to my internal Guide ***Prof. S.Thangaswamy****, Head, Department of Computer Science & Engineering,, Kumaraguru College of Technology (Affiliated to Bharathiar University), Coimbatore, TamilNadu for constantly encouraging me to pursue new goals and ideas and had given his tremendous guidance and suggestions throughout my project.*

My Special thanks to ***Mr.MRT Devendran****,* ***Mr.G.PrabhuSankar*** *and* ***Ms.N.Prathima*** *for their excellent guidance, encouragement and valuable time spend during the project period.*

I am proud of my ***Family*** *for encouraging me whenever I was depressed, to face the challenges in the file and made the MCA degree possible.*

I convey my thanks to all the ***Friends,*** *and others those who helped me directly or indirectly to complete my carrier.*

R.GOVINDARAJA

SYNOPSIS

A Project work entitled “Data Monitoring Software for SSFDR“ for Centre for Air Borne Systems (CABS), Bangalore has been carried out to replay and monitor flight recorded data.

CABS is a DRDO organisation, primarily involved in Research & Development of avionics systems for services. This centre maintain a medium transport aircraft for airborne applications.

Aviation safety continues to improve worldwide. An airplane is equipped with one or more flight data recorders that utilize analog or digital method of recording and storing them on board in form of magnetic tapes and disks. Accurate information on aircraft performance during flight is recorded in test storage media. This data helps to learn and improve performance, safety and reliability.

Solid State Flight Data Recorder (SSFDR) is one among them, is fitted on the medium transport aircraft (AVRO HS-748) at CABS to record critical parameters from aircraft sensor/transducers. After the flight, a portable interface unit is used to download these critical parameters to a PC for further analysis.

The aim of this project is to develop a suitable software to convert the data available from the portable unit and display the aircraft parameters on a computer monitor for visual display in the form of graph, engineering unit and as aircraft instrument (synthetic) for further analysis.

The software is written, such a way that, in the form of selectable aircraft data will be displayed graphically with respect to time and instrument display as in aircraft cockpit. Parallely, along with the display, the data is also stored in a file in engineering units. This data file also gives values of the selected parameters and they could be further displayed to indicate the actual values as in an aircraft cockpit.

The software is written using JAVA, graphics and Swing. The configuration of the project is such that, it could be expanded for further upgrading, modification etc.

This work is carried out as a first step in the field of data collection and display of actual data recorder. CABS propose to further carry out upgradation of this work for a total solution.

CONTENTS

1. INTRODUCTION

1.1 ORGANIZATION PROFILE

Back Ground

The Centre for Airborne Systems (**CABS**) was set up on February 01 1991 with the responsibility of guiding the development of technology for force multipliers using available expertise and infrastructure within the country. The Centre for Air Borne Systems is a part of the **D**efence **R**esearch & **D**evelopment **O**rganization (DRDO), **M**inistry of **D**efence.

CABS initially focuses attention on airborne surveillance and early warning systems. It has also applied the technology available into allied areas to develop, especially GPS, the lightning test facility for military, civil aircraft and software packages for Radar Data processing.

The Centre

The Centre is located at Belur. It is located on the southern side of Bangalore Airport on the banks of Bellondur Tank. Being essentially a system house, there is an integration lab cum hangar with all the facilities, which can accommodate large aircraft such as IL 76,Boeing 707 etc.

CABS is equipped with the following facilities

Lightning Test Facility (LTF)

A test facility for studies of lightning on aircraft and its interaction has been established for conducting high voltage and high current tests as per airworthiness specification. The facility is suitable to test a full fighter aircraft as well as small samples. In India, this is the only place which has Lightning Test Facility exclusively for Aircraft.

Software Development Facility

The Software Development Facility (SDF) comprising of state-of-art computers on Local Area Network (LAN) and a variety of software and CASE tools is being used by various groups of CABS for development of software packages. It has a strong software team and has developed a number of software packages like

- Multi – Sensor Data Processor
- Scenario Generator & Plot Simulator
- Antenna Aperture Synthesis & Numerical pattern Evaluation
- Clutter Data Analysis Software
- Performance Evaluation of Airborne Pulse Doppler Radar
- Hybrid Navigation System Simulator

SDF has nearly 30 Pentium based PC's and also has two Sun Ultra Space systems. The PC's are working on window's environment and real time multitasking operating systems. The Sun system's environment is Solaris with Motif GUI.

Technical Information Centre (TIC)

The Technical Information Centre (TIC) at CABS has a good collection of journals and books covering various disciplines of science, engineering, aeronautics aerodynamics, structures and materials, computer science and technology. It has online access to nearly 400 international database through "Dialog". It also has other audio-visual facilities such as Microfiche, 35mm slide projector, overhead projector, video / audio cassettes etc.

A dark room facility has been established at this Centre. With the expertise available in-house, the immediate need of 35mm slide on project activities are made. This is proposed to be expanded further.

Management Information System (MIS)

The Management Information System (MIS) group caters to all the information-processing needs of the centre. This is done through a set of computerized data processing procedures integrated with necessary manuals for the purpose of providing timely and effective information. Information details and reports on various aspects of projects, training etc.

Data processing, word processing, graphical and design software are used to assist in developing visual aids for presenting in seminars. The MIS also meets the DTP functions. The group has helped in organizing a number of seminars and workshops.

With the approval of DOE, CABS has established itself as a node of ERNET. MIS has taken up the administration of e-mail on this ernet network.

Hangar

CABS also have an integration lab cum hangar with ground power, ground cooling units and ground handling equipment, which can accommodate large aircraft.

A sophisticated system / integration and ground checkout facility for large-scale avionics system is also available. A high technology, well-instrumented avionics, "**Flight Test Laboratory**" for flight development and testing avionics systems is available at CABS.

Basic Researches conducted at CABS:

Mechanical / Aeronautical

- Design and Development of smart antenna demo model
- Collaborative researches program in Thermal Engineering and Electronic
- Stability of compressible boundary layer flows

Software

- Threat assessment and threat prioritization for Airborne Surveillance applications
- Software reliability models for avionics system Radar

Radar

- Design of broad band slotted wave guide and micro strip patch arrays
- Adaptive filtering for airborne radar detection in non-homogeneous clutter

CABS Quality Policy

The **Quality Policy** of CABS is to create, innovate, design, develop and integrate **Airborne Force Multipliers in a Total Quality System** in cost effective ways to meet user requirements.

1.2 PROBLEM DEFINITION

The **Solid State Flight Data Recorder** is used to record the flight parameters and preserve the same for investigation in the event of a crash or an incident. Flight records are analyzed at the end of every sortie and preserved for further analysis.

This "Data Monitoring Software for SSFDR" has to perform the following functions.

1. Convert the data into Engineering values
2. Display the values in engineering units.
3. Instrument display of selected parameters.
4. Time history plot of selected parameters
5. Documentation

Carried out the above work in a modular way such that upgrading is feasible.

2.SYSTEM STUDY AND ANALYSIS

2.1 EXISTING SYSTEM

The entire process involves, downloading the information from the Solid State **F**light **D**ata **R**ecorder (SSFDR) and converting them into engineering values. This process already exists in the organization, which performs only the basic operation of downloading and storing the data into hard disk. No other functionality or analysis is done on the data that is downloaded.

The steps involved in these procedures: -

- The data available on the Solid State Flight Data Recorder is in a form of decimal values.
- Portable Analysis Unit makes use of this data and downloads it into the local computer
- Portable Analysis Unit also has the ability to convert the existing data into engineering units by making use of the conversion factors.

The existing system, which includes the PAU, is run under the OS/2 operating system. The existing system is not flexible to make further study or analysis of the data and is not user friendly. Hence it takes longer time and larger effort.

2.2 LIMITATIONS OF EXISTING SYSTEM

Drawbacks in the existing system are

- The main drawback is that, it is developed under the OS/2 operating system.
- Due to the other embedding software's that run under different operating system may not be able to run along with this.
- The existing system does not have the advanced feature like instrument Display as in the aircraft cockpit.

Even though the interaction with the aircraft device (SSFDR) can be made with the software package GS2 (Ground Station ver 2.0), the graphs plotting and the analog meters are not shown.

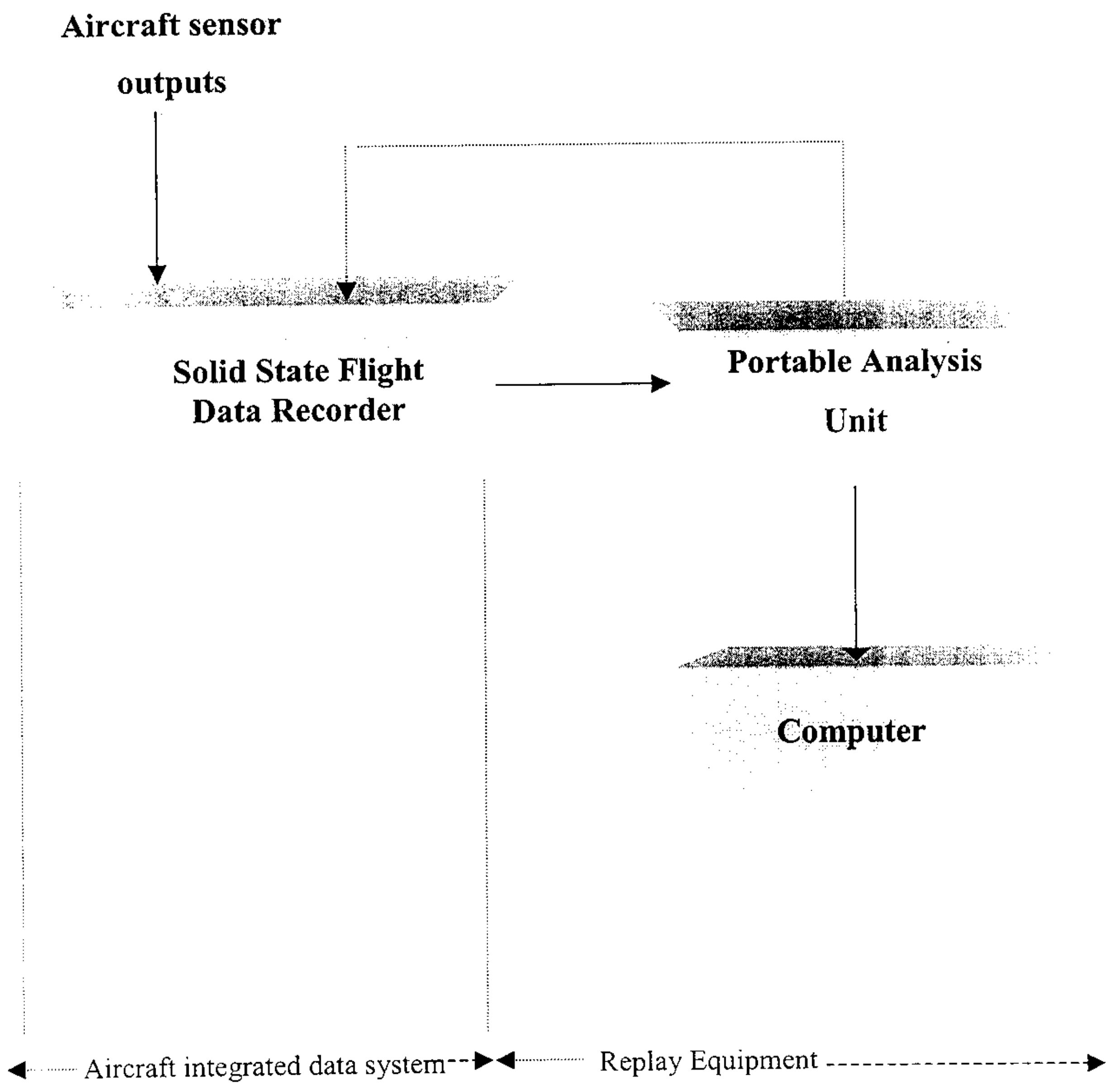
2.3 SOLID STATE FLIGHT DATA RECORDER

Every aircraft must contain a data recording system to record flight parameters and preserve the same for investigation in the event of a crash or an incident. Preflight inspection of the flight data recorder is mandatory in every aircraft to ensure satisfactory functioning. Flight records are analyzed at the end of every sortie and are preserved. It helps in maintenance of aircraft and also for future flight research. The Loral Fairchild Model F1000 Solid State Flight Data Recorder (SSFDR) is used to record selected aircraft parameters, including audio onto a Solid State Non-Volatile Memory. The Recording is protected to survive stipulated crash conditions. It could be replayed with suitable Ground Replay Equipment (GRE), following an incident.

Facilities exist within the SSFDR for aircraft in-situ data retrieval. The SSFDR records for a minimum of atleast 25 hours of aircraft data. The SSFDR is designed to operate in conjunction with a Flight Data Acquisition Unit (FDAU). With a Portable Analysis Unit (PAU) the data could be downloaded in-situ on a OS/2 based PC.

Aircraft data from SSFDR is downloaded on a portable test unit. At the time of data analysis in the laboratory the aircraft data from the Portable Test Unit is downloaded on to a local PC using RS232C communication. The down loaded data is preserved for further analysis.

The data from PAU is transferred to a local PC in the form of a data file.


Aircraft sensor outputs
Solid State Flight Data Recorder
Portable Analysis Unit
Computer
Aircraft integrated data system
Replay Equipment

Environmental Characteristics

Operating Temperature	:	-55^0 to + 70^0 C
Operating Altitude	:	-1000 to + 55000 feet
Operating Vibration	:	D0 – 160C
Humidity	:	100 %

Location

The model F1000 flight recorder system should be installed in the aircraft fuselage location, provided by the airframe manufacturer, as far aft practicable.

Automatic FDR Delayed Shutdown Feature

Internal to the SSFDR is circuitry, which will disable the recording function after a period of 8 to 10 minutes, upon the condition that powered flight is no longer possible.

Testing

The Model f1000 contains an internal self-test circuitary known as follows: After applying power to the flight recorder, the "FLIGHT RECORDER OFF" indicator, or the "FLIGHT RECORDER FAULT" indicator light should go out within 5 seconds and remain out.

Functional Verification

Using the Portable Analysis Unit (PAU), Farichild P/N 17TES0010, real-time data or previously recorded data can be extracted for complete verification of recorder function in the installation. Plugged into the front

Automatic Test Equipment (ATE) connector of the F1000, the PAU can display and print data without removing the SSFDR from the aircraft. Data can be printed out on an accessory printer, P/N 17TES0020. Selection of aircraft parameters can be varied, and the PAU can display in raw form. This raw form may be octal, decimal or hexadecimal.

The sample data that can be downloaded from the SSFDR are listed below in the "raw decimal" format:

```
,
Subframe:, 1,01:00:53,
0[illegible]33,0000,0000,1830,2533,2071,0000,1777,0000,1189,3981,1808,2038,2015,0000,3088,
0000,1143,0009,1813,2542,2066,1284,4095,0000,0003,0000,1817,2037,3687,3762,1602,
0000,1200,0000,1848,2544,2078,0000,1777,0000,0007,0000,1835,2039,2015,0000,0000,
0000,0000,0001,1818,2555,2066,1280,4095,0624,0015,0000,1843,2038,0000,0000,1141
,
Subframe:, 2,01:00:54,
1[illegible]54,0000,0000,1834,2551,2074,0000,1773,0000,1181,3981,1832,2037,2018,0000,0010,
0000,1153,0009,1802,2544,2075,1266,4095,0000,0003,0000,1813,2037,3703,3750,1730,
0000,1204,0000,1844,2547,2069,0000,1775,0000,0009,0000,1831,2039,2016,0000,0000,
0000,0000,0001,1848,2550,2065,1278,4095,0625,0015,0000,1844,2041,0000,0000,1145
,
Subframe:, 3,01:00:55,
2[illegible]31,0000,0000,1816,2559,2062,0000,1773,0000,1122,3981,1854,2041,2017,0000,0000,
0000,1190,0009,1841,2548,2067,1289,4095,0000,0003,0000,1841,2042,3704,3761,1145,
0000,1200,0000,1837,2539,2066,0000,1770,0000,0012,0000,1834,2041,2021,0000,0009,
[illegible]0,0000,0001,1831,2547,2057,1295,4095,0626,0015,0000,1821,2040,0000,0000,1143
,
Subframe:, 4,01:00:56,
3[illegible]12,0000,0000,1845,2553,2043,0000,1770,0000,1120,3981,1841,2042,2015,0000,3072,
0000,1170,0009,1825,2534,2027,1303,4095,0000,0003,0000,1821,2044,3700,3772,1211,
0000,1200,0000,1816,2537,2043,0000,1777,0000,0016,0000,1823,2043,2017,0000,0000,
0000,0000,0001,1821,2548,2067,1307,4095,0627,0015,0000,1853,2041,0000,0000,1142
```

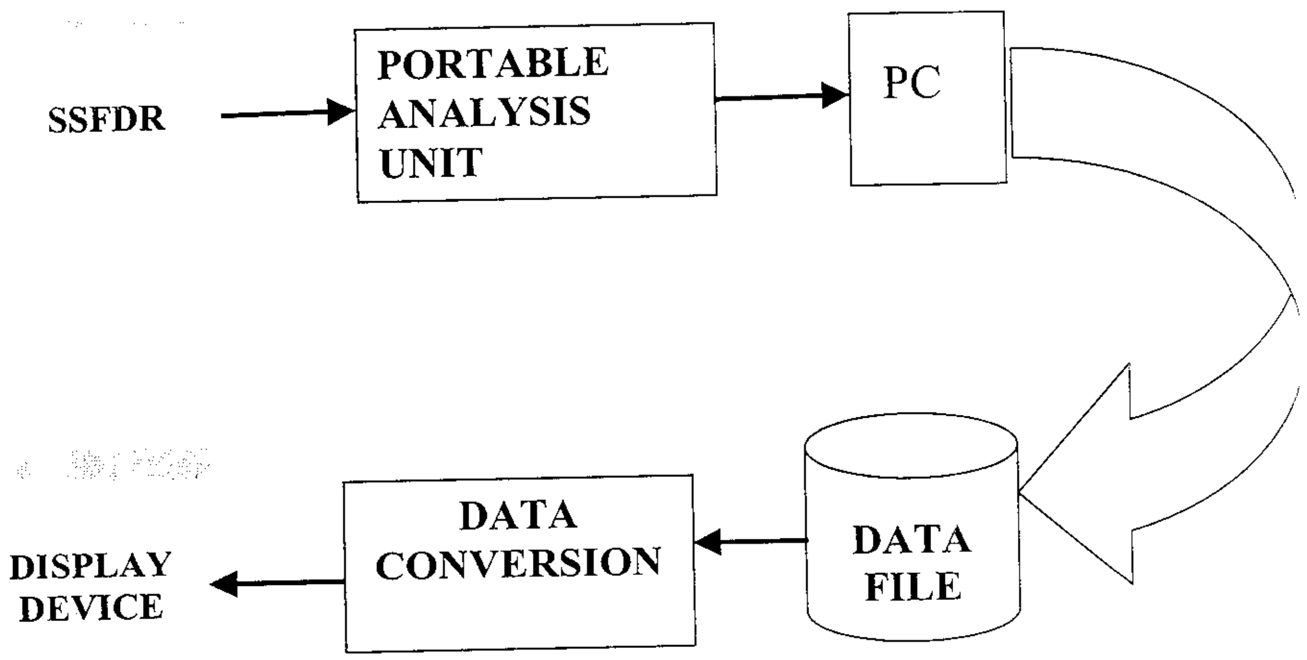
2.4 PROPOSED SYSTEM

The drawbacks of the existing system are avoided in the proposed system. Even though the new system is designed to avoid the existing drawbacks, it has to use the existing system for downloading. The data down loaded from the existing system will be used by the proposed system and will be presented in the following form for further analysis.

1.Engineering values

2.Graphical Display

3.Instrument Display



This project work is aimed at designing and developing a "Data Monitoring Software for SSFDR". The main purpose of this software is to display and store the flight parameters such as altitude, angle of attack, engine RPM and aircraft control parameters. The software also aims to provide separate analog display for all the aircraft parameters. A graphical view is also provided to display data graphically.

The proposed system is divided into three modules. Each module performs well-defined tasks.

- Engineering values display
- Graphical display
- Instrument display

Engineering Values

Data file from the SSFDR, in the form of raw decimal is the input for this module. An interactive menu is used to select the path for the data file. Provisions for editing the calibrated files and also change of path of these files are possible in this software at run time. After selecting the input and calibrated file, the engineering values are calculated using the menu option. The results are displayed on text area.

Graphical Display

This module is designed to provide graphical view for the downloaded flight parameters. The main aim of plotting graph is to analyze the flight data by comparing various flight parameters. The range of Y-axis differs according to flight parameter. For all the graphs, X-axis represents the time in seconds and Y-axis represent the flight parameter. It is designed to plot a minimum of one and a maximum of 3 parameters at a time. In graphical mode, the vertical and horizontal scrollbars are used to adjust the width and height of the graph correspondingly.

Instrument Display

The Computer simulated analog meters were show for the corresponding engineering units. There are two types of meters, which are designed to display the flight parameters.

- Analog meter display
- Vertical bar meter display

Initially the user can select the parameter for display. Analog meters are used to display the roll angle, elevator position, engine rpm etc. The vertical bar meter is used to display temperature like turbine gas temperature.

Analog meters are designed, inside the meter scale marks and values are displayed. The scale of the meters changes according to the parameter. The meter consists a needle to indicate the incoming data. The incoming data also displayed in decimal form.

Vertical bar meter a bar with marks on left side. The bar is filled with suitable color depending on the incoming data.

Slider control is used to adjust the rate of data change that means the speed of data changing. One more slider is used to indicate the length of the data. The user can start the process from middle using this slider.

2.5 REQUIREMENTS ON NEW SYSTEM

The sortie data downloaded from the SSFDR will gives the flying duration of the Aircraft in raw octal, decimal, hexa values. Now let us consider the raw decimal value.

The engineers should have to calculate the engineering units and plot the graph manually. The entire process will take time. These are the drawback in the existing system. The proposed system will overcome these drawbacks.

The following are the requirements over software:

1. The 64 parameters that have been recorded in SSFDR should be acquired from the aircraft and displayed. These parameters include the altitude, angle of attack, angle of sideslip, Air speed, Turbine Gas Temperature, and etc.

2. The data acquired from the portable analysis unit is stored in readable text format the data are used for

    - Displaying Engineering values
    - Design analog meters
    - Plotting Graphs
    - Data storage

3. The type of display should be listed in a menu and user should be chosen them in real time.

4. Once the user has chosen a configuration of parameters it should be saved. The configuration that can be loaded as and when needed.

The software has the following modules:

- Calculating engineering unit
- Displaying engineering unit
- Plotting graphs
- Displaying analog meters

Calculating engineering unit

The engineering unit is calculated by using the Lagrange's interpolation formula:

$$Y = \frac{(x - x_1)(x - x_2) \ldots\ldots\ldots\ldots (x - x_n)}{(x_0 - x_1)(x_0 - x_2) \ldots\ldots\ldots\ldots (x_0 - x_n)} y_0$$

$$+ \frac{(x - x_0)(x - x_2) \ldots\ldots\ldots\ldots (x - x_n)}{(x_1 - x_0)(x_1 - x_2) \ldots\ldots\ldots\ldots (x_1 - x_n)} y_1$$

$$+ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$+ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$+ \frac{(x - x_0)(x - x_1) \ldots\ldots\ldots\ldots (x - x_{n-1})}{(x_n - x_0)(x_n - x_1) \ldots\ldots\ldots\ldots (x_n - x_{n-1})} y_n$$

where,

$x_1, x_2, x_3 \ldots\ldots\ldots\ldots x_n \longrightarrow$ Raw decimal value

$y_1, y_2, y_3 \ldots\ldots\ldots\ldots y_n \longrightarrow$ Calibrated value

$Y \longrightarrow$ Calculated Engineering unit

Displaying Engineering unit

The engineering unit is displayed in the table format for all the parameters. The engineering unit for the complete data is listed in the table

Plotting graph

The graphs for various parameters are shown with one graph and three graph. The one graph displays the particular parameter like pitch angle and the three graphs displays the related parameter like lateral, longitudinal and vertical accelerations.

Displaying analog meters

The Computer simulated analog meters were show for the calculated engineering units. The meter is designed with a line moving like a needle pointing the parameter reading according to the engineering value.

2.6 USER CHARACTERISTICS

The software "Data Monitoring Software for SSFDR", was used by the Scientists who are involved in the research oriented areas. They are highly qualified with the Aircraft parameters. By displaying the engineering value they can judge the graphical display for the engineering units.

The software is very helpful to the engineers in the organization for the clearance and verification of parameters.

3. PROGRAMMING ENVIRONMENT

3.1 SYSTEM SPECIFICATION

Hardware Requirements

Processor	:	Pentium III
Processor Speed	:	600MHz
RAM	:	64 MB
HDD	:	20 GB
Floppy Disk	:	1.44 MB
Monitor	:	Samtron
Keyboard	:	Acer Multimedia Keyboard
Mouse	:	Logitech
DVD	:	Pioneer DVD-ROM Drive

Software Requirements

Operating System	:	Windows 9x/2K
Software	:	Java-1.2, Swing and Graphics
Editing Tool	:	Notepad

3.2 OPERATING SYSTEM

Windows95

Multi task to run more than one application at a time. Windows 95 adds several features to the windows graphical application and significantly improves it has the ability to run dos with in windows allowing user to gain the benefits of windows.

Choice of the language

Each language has its own ability of covering a various fields like business, science, network and etc., In this project we given preference for Java language.

Because of in order to develop the Graphical user interface component, automatic source code for the components and opening previous work sheet are the main objective of our project. Also compiling and run directly through windows under DOS.

3.3 SOFTWARE

JAVA Language

Java is a technology that makes it easy to build distributed application across the network. Java allows the user to do the following

1.Writing robust and reliable programs.

2.Build on application on almost any platform and run the application on any other platform without recompiling your code.

3.Distribute your application over a net work in a secure fashion.

There are six features that make Java as a power tool for Internet applications,

1. Security
2. The core API
3. Open standard
4. Distributed and Dynamic
5. Object oriented
6. Multithread

Security

Security is one of the main problems for Internet application developers. So the users must be afraid of two things.

One is that the confidential information will be compromised and the next is that their computer systems are vulnerable to corruption or destruction by hackers.

Java's security model has three primary components

1. Class loader
2. Byte code verifier
3. Security manager

Class loader

The class loader retrieves classes from the network and it separate the server classes from the local class.

Byte code verifier

The byte code verifier ensures that the Java that the Java programs have been compiled correctly.

Security manager

The security manager implements a security policy for the JVM which activities that the JVM is allowed to perform.

The core API

API stands for application programming interface. The core API provides a common set functions on all platforms. The API is divided into packages, which are group of class that perform related function.

Open Standards

The most existing aspect of Java's cross platform capability is that Java class file do not need to be compiled for each platform in advance.

The same compiled Java program will work on the PC and every platform that runs a JAVA. We can write a JAVA application on your system and it should run on every supported platform.

Distributed and dynamic

In windows operating systems parts of programs can be placed into DLL (Dynamic Link Library) so they can be shared and load dynamically.

The JVM class loader fetches class files from the network, as well as from the disk, making the JAVA application distributed as well as dynamic.

Object oriented

Object oriented programming is a way to write software that is reusable extensible and maintainable JAVA is an object oriented programming incorporated in to the language.

Multithread

A multithread application can have several threads of execution running independently and simultaneously. These threads may communicate and co-operate, and to the user will appear to be single program to following function

1. Maintain user interface responsiveness
2. Multitasking
3. Multiuser applications
4. Multiprocessing

A Platform-Independent Solution

Java has been molded to fit many different project since it was first created in the early 1990s. It has gone from a programming language for personal digital assistants to a programming language or interactive TV set-top boxes, and then to its final incarnation as a Web programming language. This transformation should give you some indication of Java's flexibility and potability; it is designed to be machine independent and function within different operating systems. Portability is one of Java's principal goals. Java code is designed to be platform-independent and has several features designed to help it achieve that goal.

When you write a Java application, you are writing a program that is designed to run on a very special computer: the Java Virtual Machine. The Java Virtual Machine is the first step towards a platform-independent solution. When you are developing software in a language like C++ ,you generally program for a specific platform, such as a Windows machine or a Macintosh. The programming language will use a variety of functions that are very specific to whatever processor is user by the machine you are programming for .

Because the language uses machine –specific instructions, you have to modify your program for a new processor if you want to run your program on another achieve. This task can be very time-consuming and resource-intensive. Java code is not written for any type of physical computer. Instead it is written for a special computer called the Virtual Machine, which is really another piece of software.

The Virtual Machine then interprets and runs the Java program you have written. The Virtual Machine is programmed for specific machines, so there is a Windows 95 Virtual Machine, a Sun Virtual Machine, and so on.

There is even a copy of the Virtual Machine build into Netscape, allowing the browser to run Java programs. by porting the Virtual Machine from platform to platform, instead of the Java programs themselves, any Java program can be used on any machine running a version of the Virtual Machine.

This feature is the reason why the same Java applet can run on UNIX workstations as well as on Windows machines. Sun has gone to great lengths to port the Virtual Machine to nearly every major type of machine on the market .By doing so, Sun ensures the portability of Java The benefit for you is the ability to write your code once and then use it on many machines.

The Future of Java

Because Java is a relatively new addition to the Internet, you may be wondering what the future of Java might have in store. Now that you have an understanding of what Java can and cannot do, the following section provide a glimpse at what the future has in store for Java.

Support provided by Java

Java mainly supports two type of programming, such as

1. Applet program
2. Application program

Application program

A JAVA application runs in the simplest possible environment. It only gets the input from the command line and displays the output in the console.

Applet Programming

The Java applet is a JAVA program that is launched from a web documents that is the class file is embeded in an html file. The JAVA applet programs are windows based application that is displayed in the windows or frames or dialogs.

Stream Tokenizer

This class extracts identifiable sub strings and punctuations for man input stream according to user defined rules. This process is called tokenizing because tokens reduce the stream. Token typically represents keyword, variable names, string literls and etc.,

Random Access Files

This class encapsulates full read and full write access files. This class can be derived from either input class or output class.

The methods implements by random access files are

1. Reading of primitive types and byte arrays.
2. Writing of primitive types and byte arrays.
3. Positioning of the file pointer.

Swing

Swing is the major component of JFC which is the result of a large collaborative effort of SUN, IBM and other companies.

Swing provides the capability to quickly and easily changes the look and feel (L&F) of a single component or a group of component. This capability is known as pluggable look and feel (PL&F) is important features of swing .The swing PL&F architecture makes it easy to customize both the appearance and the behavior of any particular swing control or any swing component.

Swing extends abstract window tool by supplying many more types of these graphical user interface components provides 100% pure Java implementation of these and allowing the appearance and behavior of these components to be easily handled .

The Swing component does not depend on the native windows implementation to support them.

Swing Features and Concepts

In this some of the major concepts we need to know to build Swing GUIs the containment hierarchy, layout management, event handling, painting, and threads. Along the way, we touched upon related topics, such as borders. This section tells about some of the Swing features that we haven't yet discussed:

- Features `JComponent`
- Icons
- Actions
- Pluggable Look & Feel support
- Support for assistive technologies
- Separate data and state models

Features `JComponent`

Except for the top-level containers, all components that begin with `J` inherit from the `JComponent` class. They get many features from `JComponent`, such as the ability to have borders, tool tips, and a configurable look and feel. They also inherit many convenient methods.

Icon support

The swing can support you to add the icon image to the button, label, and menus .The new components included in swing or tabbed panels, progress bar, root pan, split pan , tool tip , tree, view port and fancy borders for spinners and splenders.

Actions

With `Action` objects, the Swing API provides special support for sharing data and state between two or more components tha can generate action events. For example, if we have a button and a menu item that perform the same function, consider using an `Action` object to co-ordinate the text, icon, and enabled state for the two components.

Pluggable Look & Feel

A single program can have any one of several looks and feels. we can let the user determine the look and feel, or we can specify the look and feel programatically.

Support for Assistive Technologies

Assistive technologies such as screen readers can use the Accessibility API to get information from Swing components. Because support for the Accessibility API is built into the Swing components, Swing program will probably work just fine with assistive technologies, even if we do nothing special. With very little extra effort, however, we can make program function even more smoothly with assistive technologies, which might well expand its market.

Separate Data and State Models

Most noncontainer Swing components have models. A button (`JButton`), for example, has a model (`ButtonModel`) that stores the button's state -- what its keyboard mnemonic is, whether it's enabled, selected, or pressed, and so on. Some components have multiple models. A list (`JList`), for example, uses a `ListModel` to hold the list's contents, and a `ListSelectionModel` to track the list's current selection.

We don't usually need to know about the models that a component uses. For example, almost all programs that use buttons deal directly with the `JButton` object, and don't deal at all with the `ButtonModel` object. Why then do separate models exist? Because they give the ability to work with components more efficiently and to easily share data and state between components. One common case is when a component, such as a list or table, contains lots of data. It can be much faster to manage the data by directly working with a data model than by having the component forward each data request to the model. We can use the component's default model, or we can implement our own.

Painting

When a Swing GUI needs to paint itself -- whether for the first time, in response to becoming unhidden, or because it needs to reflect a change in the program's state -- it starts with the highest component that needs to be repainted and works its way down the containment hierarchy. This process is orchestrated by the AWT painting system, and made more efficient and smooth by the Swing repaint manager and double-buffering code.

Swing components generally repaint themselves whenever necessary. When you invoke the `setText` method on a component, for example, the component should automatically repaint itself and, if appropriate, resize itself. If it doesn't, it's a bug. The workaround is to invoke the **`repaint`** method on the component to request that the component be scheduled for painting. If the component's size or position needs to change but doesn't do so automatically, you should invoke **`revalidate`** upon the component before invoking `repaint`.

Like event-handling code, painting code executes on the event-dispatching thread. While an event is being handled, no painting will occur.

Similarly, if a painting operation takes a long time, no events will be handled during that time.

Java 2d

This API provides comprehensive support for two dimensional drawing, image processing, graphics rendering color management and painting. It consist of an imaging model that support line art, text images spatial and color transformations and image compositing

Displaying Graphics with Graphics2D

In this shows how to use `Graphics2D` to display graphics with fancy outline and fill styles, transform graphics when they are rendered, constrain rendering to a particular area, and generally control the way graphics look when they are rendered. You'll also learn how to create complex `Shape` objects by combining simple ones and how to detect when the user clicks on a displayed graphics primitive. These topics are discussed in the following:

- Stroking and Filling Graphics Primitives
- Transforming Shapes, Text, and Images
- Clipping the Drawing Region
- Compositing Graphics
- Controlling Rendering Quality
- Constructing Complex Shapes from Geometry Primitives
- Supporting User Interaction

Stroking and Filling Graphics Primitives

By changing the stroke and paint attributes in the Graphics2D context before rendering, you can easily apply fancy line styles and fill patterns to graphics primitives. For example, you can draw a dashed line by creating an appropriate Stroke object and calling setStroke to add it to the Graphics2D context before you render the line. Similarly, you can apply a gradient fill to a Shape by creating a GradientPaint object and adding it to the Graphics2D context by calling setPaint before you render the Shape.

Drawing Curves

The Cubic and Quad applets demonstrate how to create cubic and quadratic curves using CubicCurve2D and QuadCurve2D respectively. These applets also demonstrate how the curves are drawn with respect to the positioning of the control points by allowing you to interactively move both the control points and the end points.

Line Styles

Line styles are defined by the stroke attribute in the Graphics2D rendering context. To set the stroke attribute, you create a BasicStroke object and pass it into the Graphics2D setStroke method.

A BasicStroke object holds information about the line width, join style, end-cap style, and dash style. This information is used when a Shape is rendered with the draw method.

The line width is the thickness of the line-measured perpendicular to its trajectory. The line width is specified as a `float` value in user coordinate units, which are roughly equivalent to 1/72 inch when the default transform is used.

The join style is the decoration that is applied where two line segments meet. `BasicStroke` supports three join styles:

- `JOIN_BEVEL`
- `JOIN_MITER`
- `JOIN_ROUND`

The end-cap style is the decoration that is applied where a line segment ends.
`BasicStroke` supports three end-cap styles:

- CAP_BUTT
- CAP_ROUND
- CAP_SQUARE

The dash style defines the pattern of opaque and transparent sections applied along the length of the line. The dash style is defined by a dash array and a dash phase. The dash array defines the dash pattern. Alternating elements in the array represent the dash length and the length of the space between dashes in user coordinate units. Element 0 represents the first dash, element 1 the first space, and so on. The dash phase is an offset into the dash pattern, also specified in user coordinate units. The dash phase indicates what part of the dash pattern is applied to the beginning of the line.

Fill Patterns

Fill patterns are defined by the paint attribute in the `Graphics2D` rendering context. To set the paint attribute, you create an instance of an object that implements the `Paint` interface and pass it into the `Graphics2D` `setPaint` method.

Transforming Shapes, Text, and Images

We can modify the transform attribute in the `Graphics2D` context to move, rotate, scale, and shear graphics primitives when they are rendered. The transform attribute is defined by an instance of `AffineTransform`. (An affine transform is a transformation such as translate, rotate, scale, or shear in which parallel lines remain parallel even after being transformed.)

`Graphics2D` provides several methods for changing the transform attribute. We can construct a new `AffineTransform` and change the `Graphics2D` transform attribute by calling `transform`.

`AffineTransform` defines the following factory methods to make it easier to construct new transforms:

- `getRotateInstance`
- `getScaleInstance`
- `getShearInstance`
- `getTranslateInstance`

Alternatively we can use one of the `Graphics2D` transformation methods to modify the current transform. When we call one of these convenience methods, the resulting transform is concatenated with the current transform and is applied during rendering:

- `rotate`--to specify an angle of rotation in radians
- `scale`--to specify a scaling factor in the x and y directions
- `shear`--to specify a shearing factor in the x and y directions
- `translate`--to specify a translation offset in the x and y directions

We can also construct an `AffineTransform` directly and concatenate it with the current transform by calling the `transform` method.

The `drawImage` method is also overloaded to allow specifying an `AffineTransform` that is applied to the image as it is rendered. Specifying a transform when we call `drawImage` does not affect the `Graphics2D` transform attribute.

Clipping the Drawing Region

Any `Shape` can be used as a clipping path that restricts the portion of the drawing area that will render. The clipping path is part of the `Graphics2D` context; to set the clip attribute, we call `Graphics2D.setClip` and pass in the `Shape` that defines the clipping path we want to use. We can shrink the clipping path by calling the `clip` method and passing in another `Shape`; the clip is set to the intersection of the current clip and the specified `Shape`.

Overview of Printing in Java

The system controls the overall printing process, just like it controls when and how a program can draw. Your application provides information about the document to be printed, and the printing system determines when each page needs to be rendered.

This callback printing model enables printing to be supported on a wide range of printer and systems. It even allows users to print to a bitmap printer from a computer that doesn't have enough memory or disk space to hold the bitmap of an entire page. In this situation the printing system will ask your application to render the page repeatedly so that it can be printed as a series of smaller images. (These smaller images are typically referred to as bands, and this process is commonly called banded printing.)

To support printing, an application needs to perform two tasks:

- Job control--managing the print job
- Imaging--rendering the pages to be printed

Job Control

Although the system controls the overall printing process, your application has to get the ball rolling by setting up a `PrinterJob`. The `PrinterJob`, the key point of control for the printing process, stores the print job properties, controls the display of print dialogs, and is used to initiate printing.

To steer the `PrinterJob` through the printing process, your application needs to

- Get a `PrinterJob` by calling `PrinterJob.getPrinterJob`.
- Tell the `PrinterJob` where the rendering code is by calling `setPrintable` or `setPageable`.
- If desired, display the Page Setup and Print dialogs by calling `pageDialog` and `printDialog`.
- Initiate printing by calling `print`.

The rendering of pages is controlled by the printing system through calls to the application's imaging code.

Imaging

Application must be able to render any page when the printing system requests it. This rendering code is contained in the `print` method of a page painter a class that implements the `Printable` interface. You implement `print` to render page contents by using a `Graphics` or a `Graphics2D` rendering context. You can use either one page painter to render all of the pages in a print job or different page painters for different types of pages. When the printing system needs to render a page, it calls `print` on the appropriate page painter.

When you use a single page painter, the print job is called a printable job. Using a printable job is the simplest way to support printing. More complex printing operations that use multiple page painters are referred to as pageable jobs. In a pageable job an instance of a class that implements the `Pageable` interface is used to manage the page painters.

Printable Jobs

In a printable job all pages use the same page painter and `PageFormat`, which defines the size and orientation of the page to be printed. The page painter is asked to render each page in indexed order, starting the page at index 0. The page painter might be asked to render a page multiple times before the next page is requested, but no pages are skipped. For example, if a user prints pages 2 and 3 of a document, the page painter is asked to render the pages at indices 0, 1, and 2 even though the page at index 0 will not be printed.

If a print dialog is presented, it will not display the number of pages, because that information is not available to the printing system. The page painter informs the printing system when the end of the document is reached.

Pageable Jobs

Pageable jobs are useful if your application builds an explicit representation of a document, page by page. In a pageable job different pages can use different page painters and `PageFormats`. The printing system can ask the page painters to render pages in any order, and pages can be skipped. For example, if a user prints pages 2 and 3 of a document, the page painter will be asked to render only the pages at indices 1 and 2.

The multiple page painters in a pageable job are coordinated by a class that implements the `Pageable` interface, such as `Book`. A `Book` represents a collection of pages that can use different page painters and that can vary in size and orientation. You can also use your own implementation of the `Pageable` interface if `Book` does not meet your application's needs.

4. SYSTEM DESIGN & DEVELOPMENT

4.1 INPUT DESIGN

The basic input for the system is provided through the data file that is downloaded from the SSFDR. This data file contains the decimal values noted from the aircraft sensors. This file stores 17 parameters of aircraft that are described by 64 words. This downloading of the data file is actually done using the Portable Analysis Unit. Only the data file is processed as input for the proposed system.

Graph

- Once the engineering values are calculated, they are produced as input to this module.
- Initially the type of the graph whether it is single or triple graph is determined.
- The next step is to specify the parameters to which the graph is to be plotted

Instrument Display

This module is to simulate the reading in the aircraft into the graphical meter. To do so, following inputs are

- Calculated engineering units from the previous module.
- Select the parameter to which instrument to be displayed
- Slider control to adjust the speed of data change is also provided for the user

4.2 OUTPUT DESIGN

The calculated engineering values are provided as text display. This text value may also be taken as printout. The engineering values are formatted with specified column and row and fractional part also controlled.

The performance of the aircraft is graphically displayed. This display is more useful for the scientist and pilots to easily identify the performance of the aircraft. Multiple graphs also helps in making comparison with two parameters. One more facility in graph is to adjust the scrollbars corresponding width and height of the graph is changed.

Instrument display is simulation of the analog meters in the aircraft. Temperature is indicated by thermometer type instrument. Each and every meter has different engineering units.

4.3 FILE DESIGN

The SSFDR sends the Aircraft data in the form of ARINC file format. The files used for manipulating may include

- Raw Decimal file
- Calibrated file
- Engineering value file

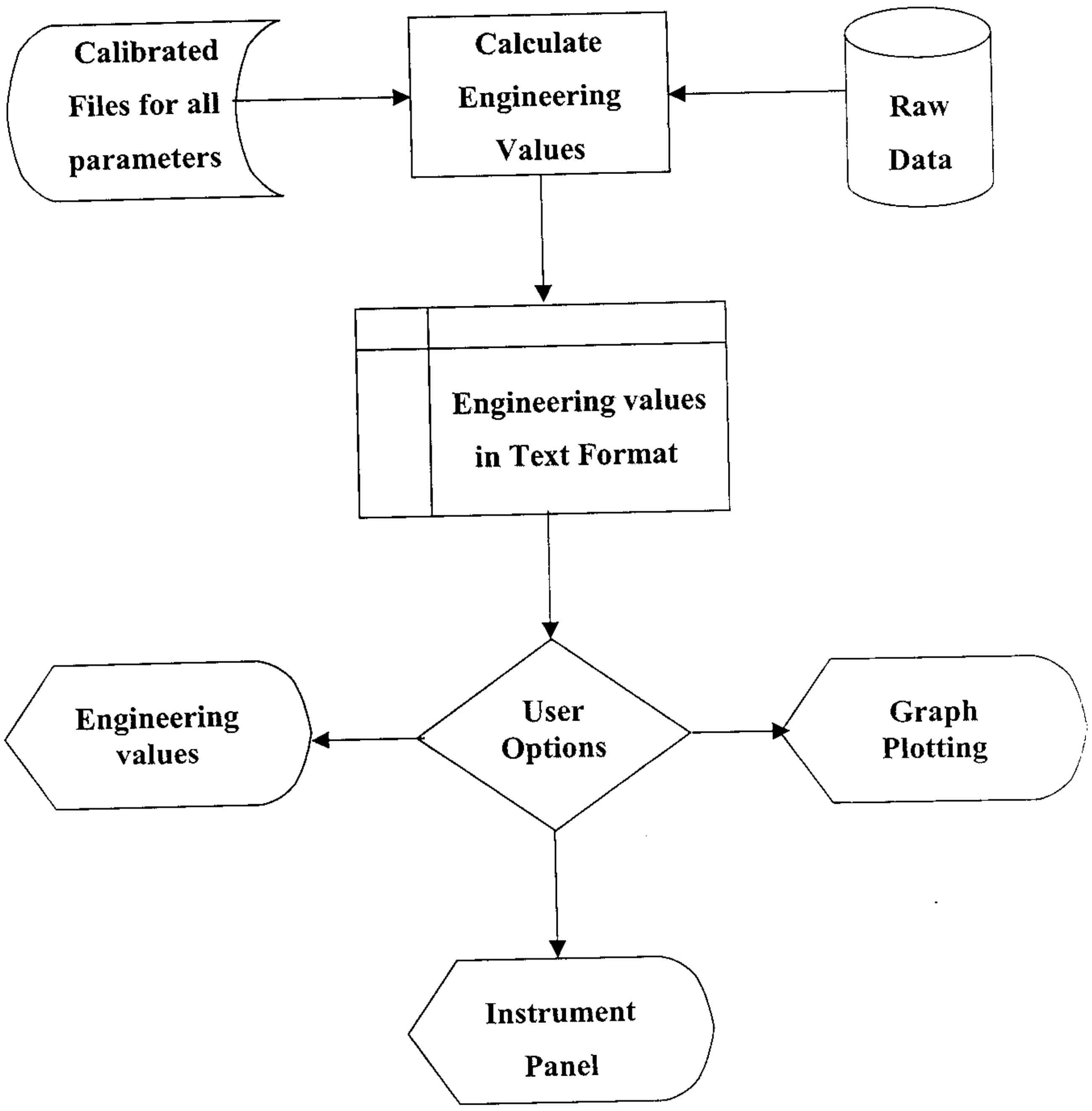
The calibration is taken from the Aircraft and the data are stored in the file for all the available parameters. These files are called calibrated file. The calibrated file may corresponds to the engineering unit, are the reference file for converting the raw decimal value into engineering unit.

The raw decimal file (ARINC format), from the SSFDR is the source file that will be given as input.

The engineering value file is the destination file. The engineering units are stored in this file which is essential for displaying the engineering units, graphs and analog meters.

4.4 PROCESS DESIGN

The downloaded file from the SSFDR is given as the input and the engineering unit is stored as the output. This process will continue for each and every sortie for the Aircraft. The following diagram gives pictorial view:



Calibrated Files for all parameters
Calculate Engineering Values
Raw Data
Engineering values in Text Format
Engineering values
User Options
Graph Plotting
Instrument Panel

5. SYSTEM TESTING & IMPLEMENTATION

5.1 TESTING

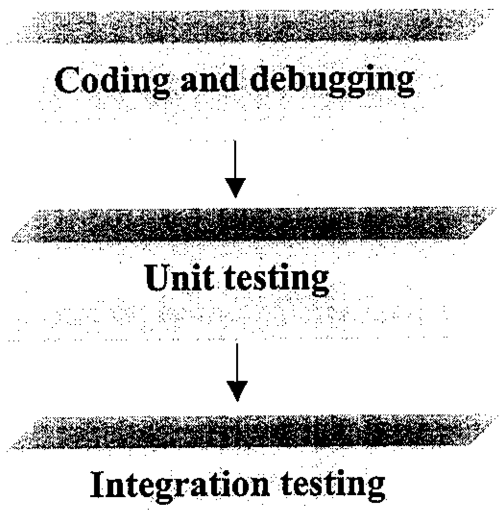
The goals of verification and validation activities are to assess and improve the quality of the work products generated during development and modification of software. Quality attributes of interest include correctness, completeness, consistency, reliability, usefulness, usability, conformance to standard, and overall cost effectiveness.

Testing is an important phase without which the system cannot be released to the users. Testing is vital for the success of any system. It is aimed at ensuring that all process function accurately according to the specification. The logical and physical designs are continually examined to ensure that the system will work perfectly when implemented. Each module is tested individually using test data and verified for correction and accuracy. System testing can be broadly classified as Unit testing and Integration testing.

Unit Testing

Unit testing comprises the set of tests performed by an individual programmer prior to integration of the unit into a larger system. A program unit denotes a routine or a collection of routines implemented by and individual programmer.

Unit testing has the goal of discovering errors in the individual modules of the system. Modules are tested in isolation form one another in an artificial environment known as an " test harness " which consists of the driver programs and data necessary to exercise the modules. Unit testing should be an exhaustive as possible to ensure that each representative case handled by each module has been tested. Unit testing is eased by a system structure that is composed of small, loosely coupled modules.
The situation is illustrated as follows:



There are three types of tests that the source code must satisfy

Functional tests

Specify typical operation conditions, typical input values, and typical expected results e.g., testing a real - valued square root routine with small positive numbers, zero and negative numbers.

Performance tests

They are designed to verify response time under varying loads percent of execution time spent in various segments of the program, throughput, and primary and secondary memory utilization.

Stress tests

They are designed to overload a system in various ways.

System testing was carried out by testing the modules separately, followed by testing modules clustered as a unit. Errors encountered during testing were corrected.

Program testing is nothing but testing a number of programs that form a cluster achieve a certain goal. During program testing two kinds of errors are likely to occur. They are syntax errors and logical errors. Syntax errors occur when a program statement violates one or more rules of the language in which the program is written. These errors have to be corrected before the program is executed.

Running a syntax free program always does not mean that it would produce correct outputs. There is a possibility that logical errors could occur. Logical errors may occur due to incorrect handling of data, improper sequence of program statements etc. when a program is free of logical errors, it may be expected to produce the desired outputs correctly. Comparing the actual outputs with the expected outputs tests this.

Integration Testing

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover error associated with interfacing. The objective is to take unit-tested modules and built a program structure that has been dictated by design. There are two approaches for the integration testing as

- Top-down integration
- Bottom-up integration

The integration testing was found to be the most important task for the entire process of this project. As this project is implemented as a link between different modules there was a lot area where the modules individually functioned but as integrated software failed. All type of integration errors was eliminated and the modules were found to be working to their need.

Validation test

At the culmination of integration testing, software is completed assembled as package, interfacing errors have been uncovered and corrected, and a final series software test- validation testing is needed. According to albeit harsh validation succeeds when the software functions in a manner that can be reasonably expected by the customer. Reasonable expectations are defined in the software requirement specification.

The validation testing was done to this project by making this software run by different individuals not associated with this project and their reactions and feedback were taken. According to their views the entire operations were changed to the maximum possible extent. This project though being user friendly, it was found that the software needs certain amount of knowledge for executing this.

5.2 IMPLEMENTATION

This project being merely a development of subsystem does not replace any other system. The implementation of this is much easier as it involves very little to modify in the existing. The implementation of the project involves the following steps.

- Only the data file that is being downloaded through Portable Analysis Unit is being used in the proposed one.
- This project being developed independently and can be implemented directly.

Apart from these, there are some basic features required to implement.

- This software can be implemented on 32-bit platform and it requires the JRE (Java Runtime Environment) for running this project.
- This software requirement requires minimum of 32MB memory space for its operations. Memory capacity less than 32MB may degrade its performance.

6. CONCLUSION

This software developed for CABS is a versatile one and could be adopted for any type of airborne data recorders. Since it is a user configured one, the display could be of user's configurable one. This could be further expanded to cover the other aspects such as simulation of six degree of freedom model using the data as inputs. It could retrace the aircraft trajectory. Pilots and test engineers can utilize this software effectively to analyze any incident.

This system has been developed according to the specification given by the Flight Test Laboratory. Since the duration of the project is of shorter period, this was executed at the first level. But it will give the feel of the aircraft history and more than sufficient in meeting the specifications. The system could be further modified/upgraded to future requirements and specifications.

7. SCOPE FOR FUTURE DEVELOPMENT

The system has been developed according to the specification give by the Flight Test Laboratory. The software works when the data is in the raw decimal format.

Apart from these features that have been incorporated in the system, there is also scope for future enhancements. This system proposed merely acts as a sub system by incurring data from other module. This downloading of data file may also developed within this system. By doing so, this software becomes a complete package and therefore may also replace the existing system.

In future, creating an image for the Aircraft that shows the various positions for all the parameters and animation can also extend the software for viewer's assumption.

BIBLIOGRAPHY

1. The Complete Reference Java – 2

    - Patrick Naughton
    - Herbert Schildt

2. Pure JFC 2D Graphics and Imaging

    - Satyaraj Pantham. Ph.D.

3. Java Swing - O'Reilly

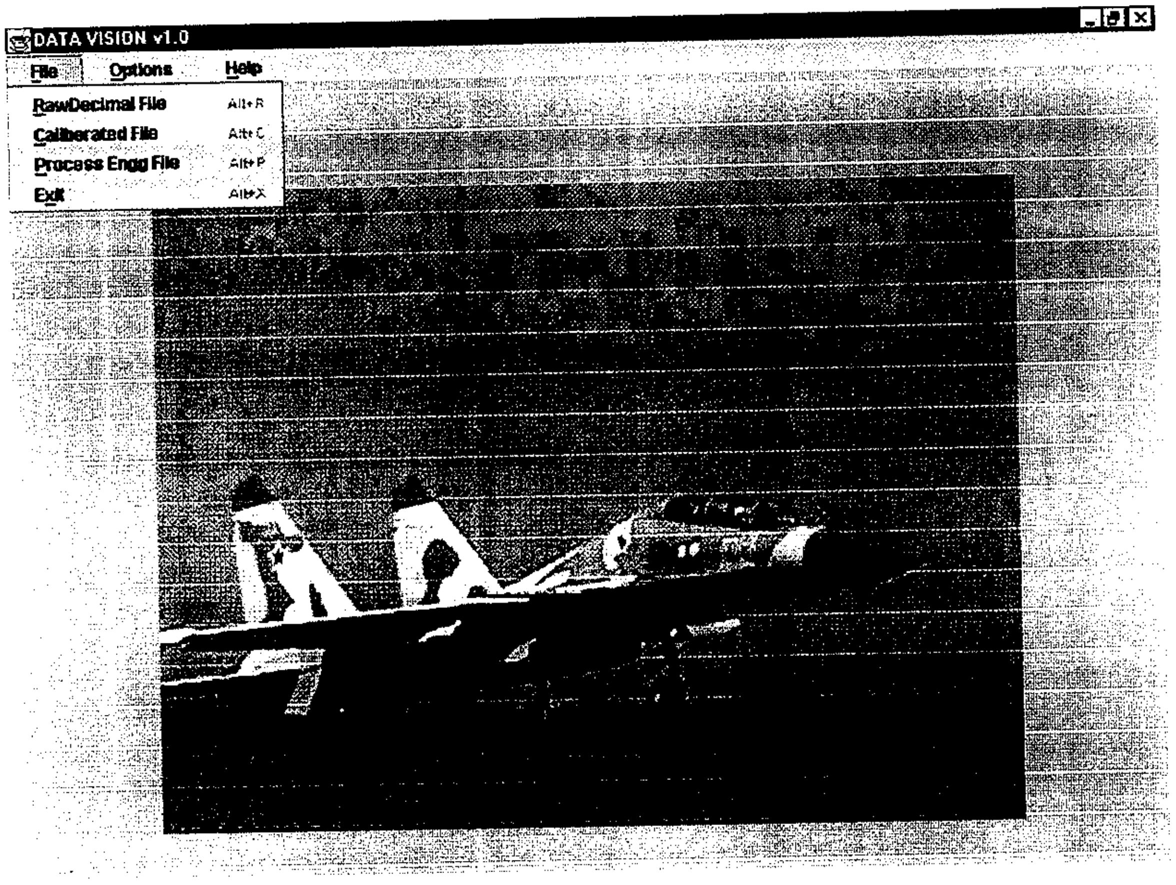
    - Robert Echstein
    - Marc Loy
    - Dave Wood

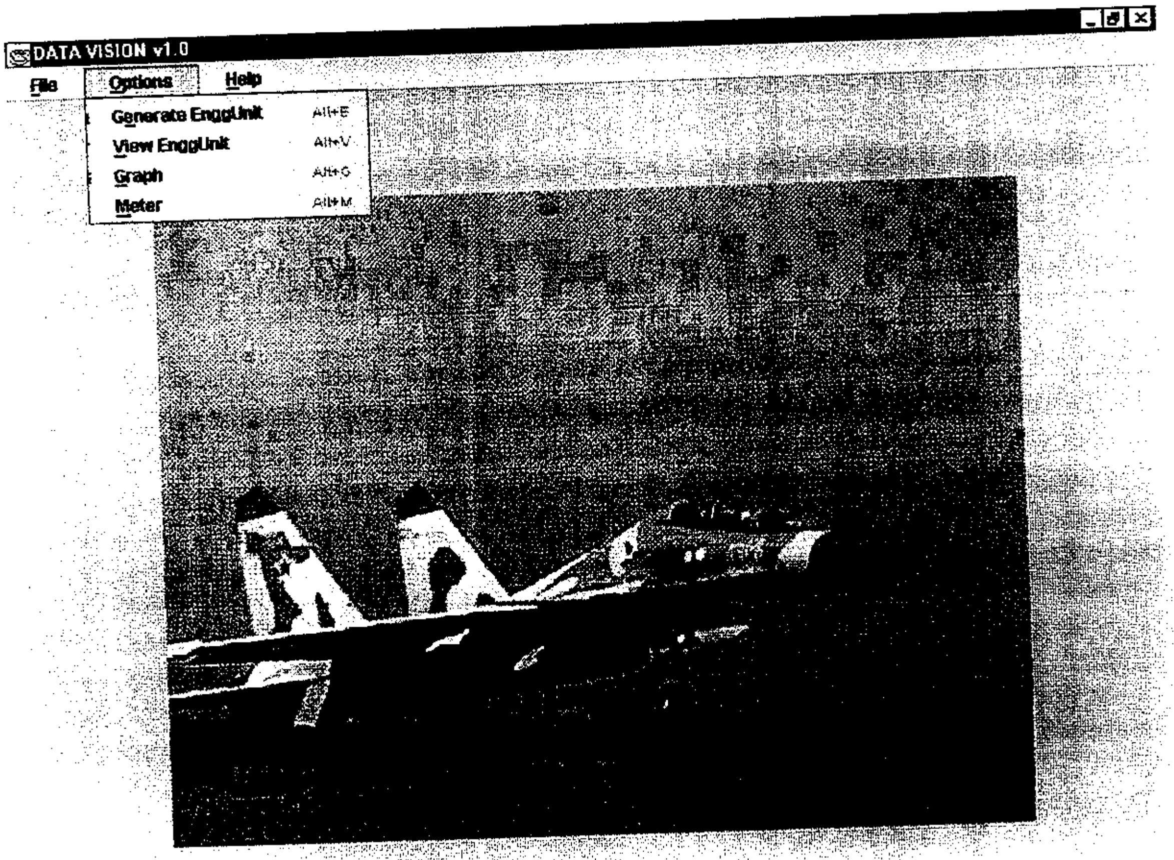
4. www.java.sun.com
5. www.codeguru.com/java
6. www.objectplanet.com
7. www.sourcecode.com

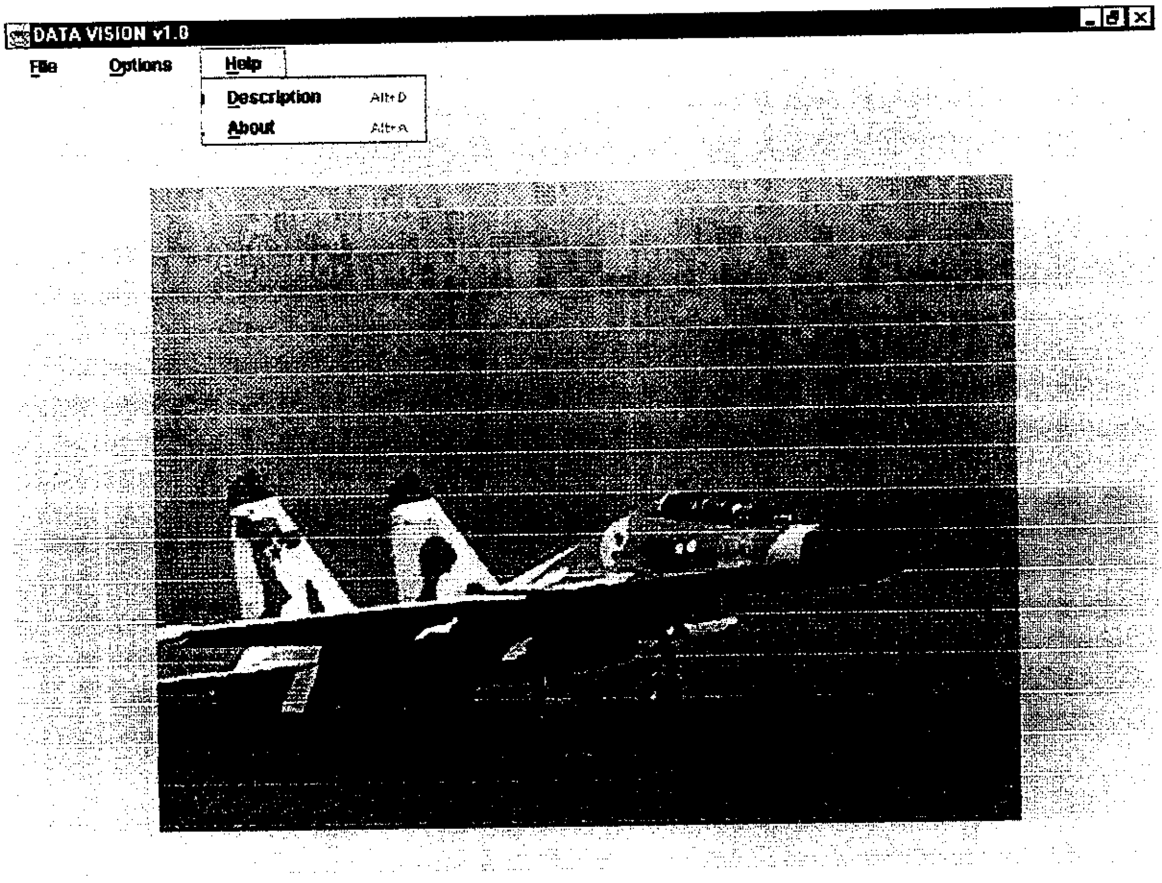
LIST OF PARAMETERS

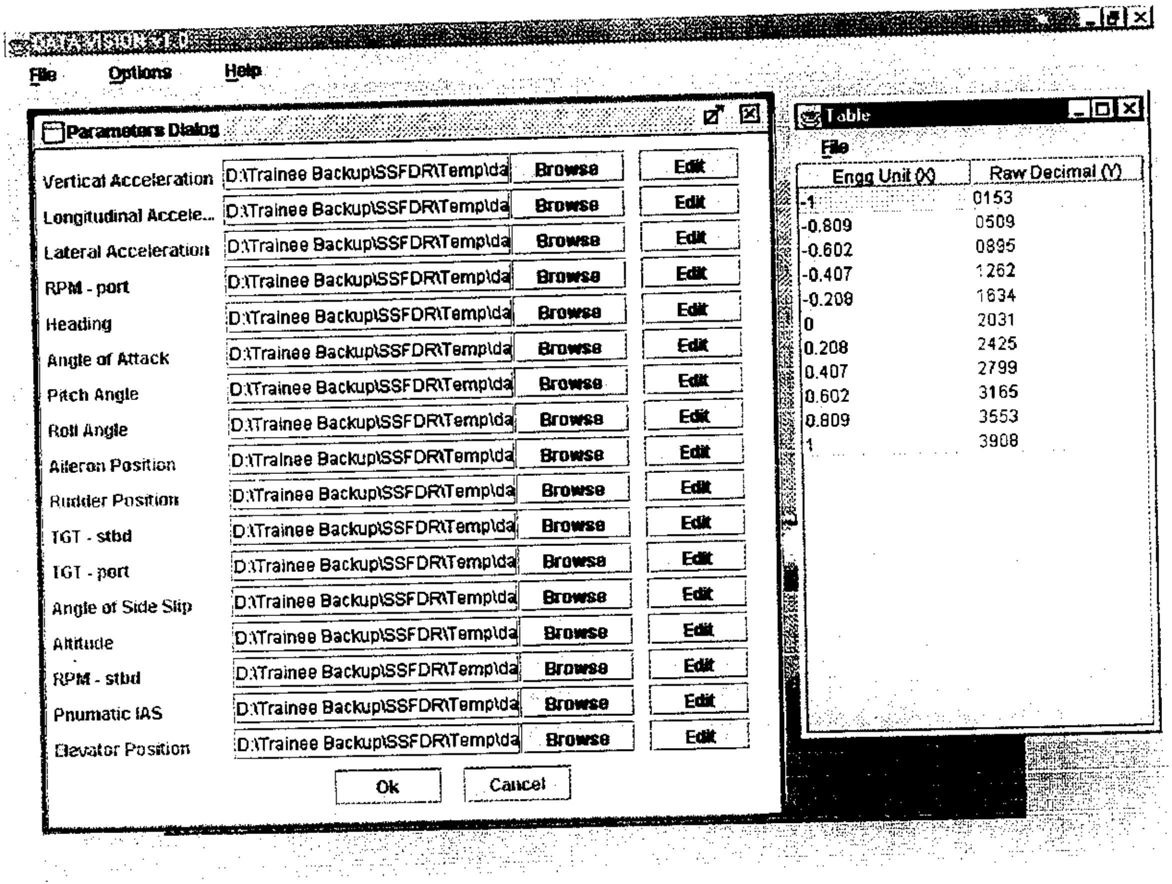
	Name of Parameters	Word Nos.
1.	Normal Acceleration	4,12,20,28,36,44,52,60
2.	Longitudinal Acceleration	5,21,37,53
3.	Lateral Acceleration	6,22,38,54
4.	Engine RPM port	8
5.	Heading	9
6.	Angle of Attack	10
7.	Pitch Angle	13,29,45,61
8.	Roll Angle	14,46
9.	Aileron Position	18
10.	Rudder Position	23,55
11.	Pneumatic Altitude	26,34
12.	Turbine Gas Temperature (TGT) stbd	30
13.	Turbine Gas Temperature (TGT) Port	31
14.	Engine RPM stbd	40
15.	Pneumatic (Indicator Air Speed) IAS	42
16.	Angle of Side slip	32
17.	Frame Count	57
18.	Elevator Position	64

SCREENS


DATA VISION v1.0
File
Options
Help
RawDecimal File Alt+R
Caliberated File Alt+C
Process Engg File Alt+P
Exit Alt+X

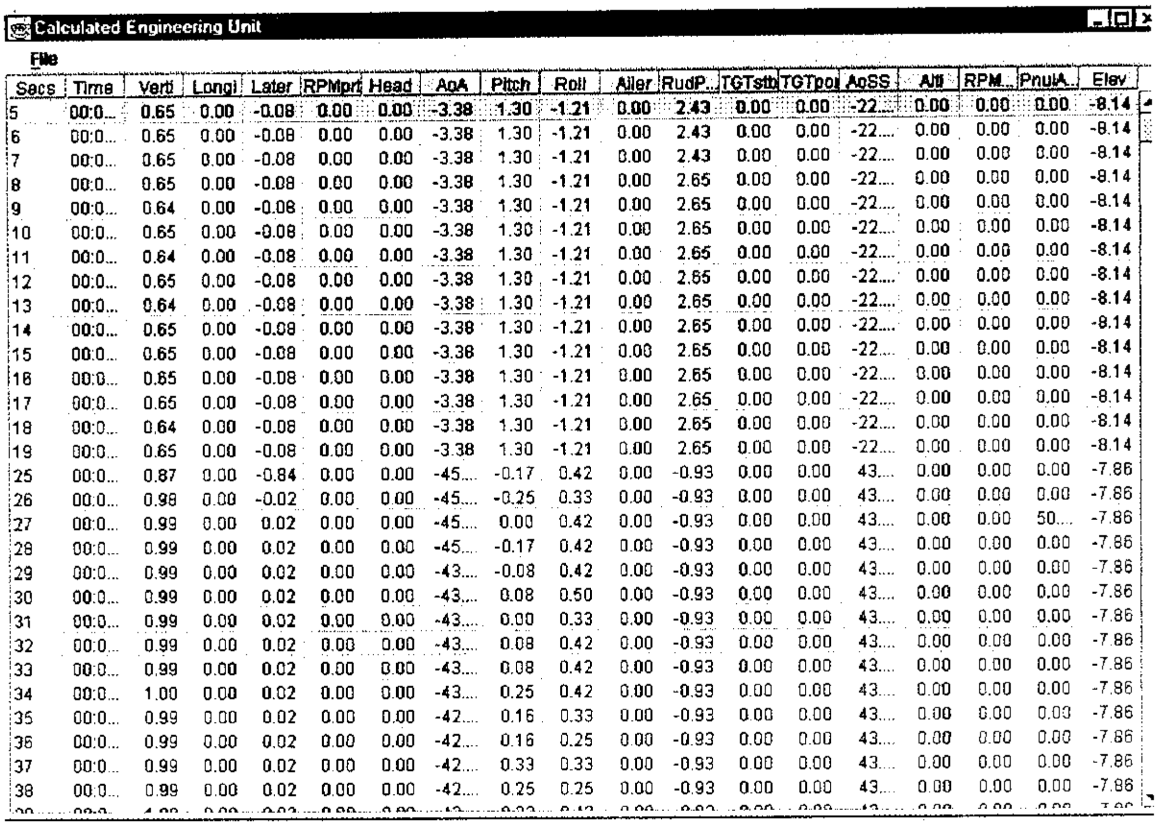

DATA VISION v1.0
File
Options
Help
Generate EnggUnit Alt+E
View EnggUnit Alt+V
Graph Alt+G
Meter Alt+M


DATA VISION v1.0
File
Options
Help
Description Alt+D
About Alt+A



Vertical Acceleration	D:\Trainee Backup\SSFDR\Temp\da	Browse	Edit
Longitudinal Accele...	D:\Trainee Backup\SSFDR\Temp\da	Browse	Edit
Lateral Acceleration	D:\Trainee Backup\SSFDR\Temp\da	Browse	Edit
RPM - port	D:\Trainee Backup\SSFDR\Temp\da	Browse	Edit
Heading	D:\Trainee Backup\SSFDR\Temp\da	Browse	Edit
Angle of Attack	D:\Trainee Backup\SSFDR\Temp\da	Browse	Edit
Pitch Angle	D:\Trainee Backup\SSFDR\Temp\da	Browse	Edit
Roll Angle	D:\Trainee Backup\SSFDR\Temp\da	Browse	Edit
Aileron Position	D:\Trainee Backup\SSFDR\Temp\da	Browse	Edit
Rudder Position	D:\Trainee Backup\SSFDR\Temp\da	Browse	Edit
TGT - stbd	D:\Trainee Backup\SSFDR\Temp\da	Browse	Edit
TGT - port	D:\Trainee Backup\SSFDR\Temp\da	Browse	Edit
Angle of Side Slip	D:\Trainee Backup\SSFDR\Temp\da	Browse	Edit
Altitude	D:\Trainee Backup\SSFDR\Temp\da	Browse	Edit
RPM - stbd	D:\Trainee Backup\SSFDR\Temp\da	Browse	Edit
Pnumatic IAS	D:\Trainee Backup\SSFDR\Temp\da	Browse	Edit
Elevator Position	D:\Trainee Backup\SSFDR\Temp\da	Browse	Edit

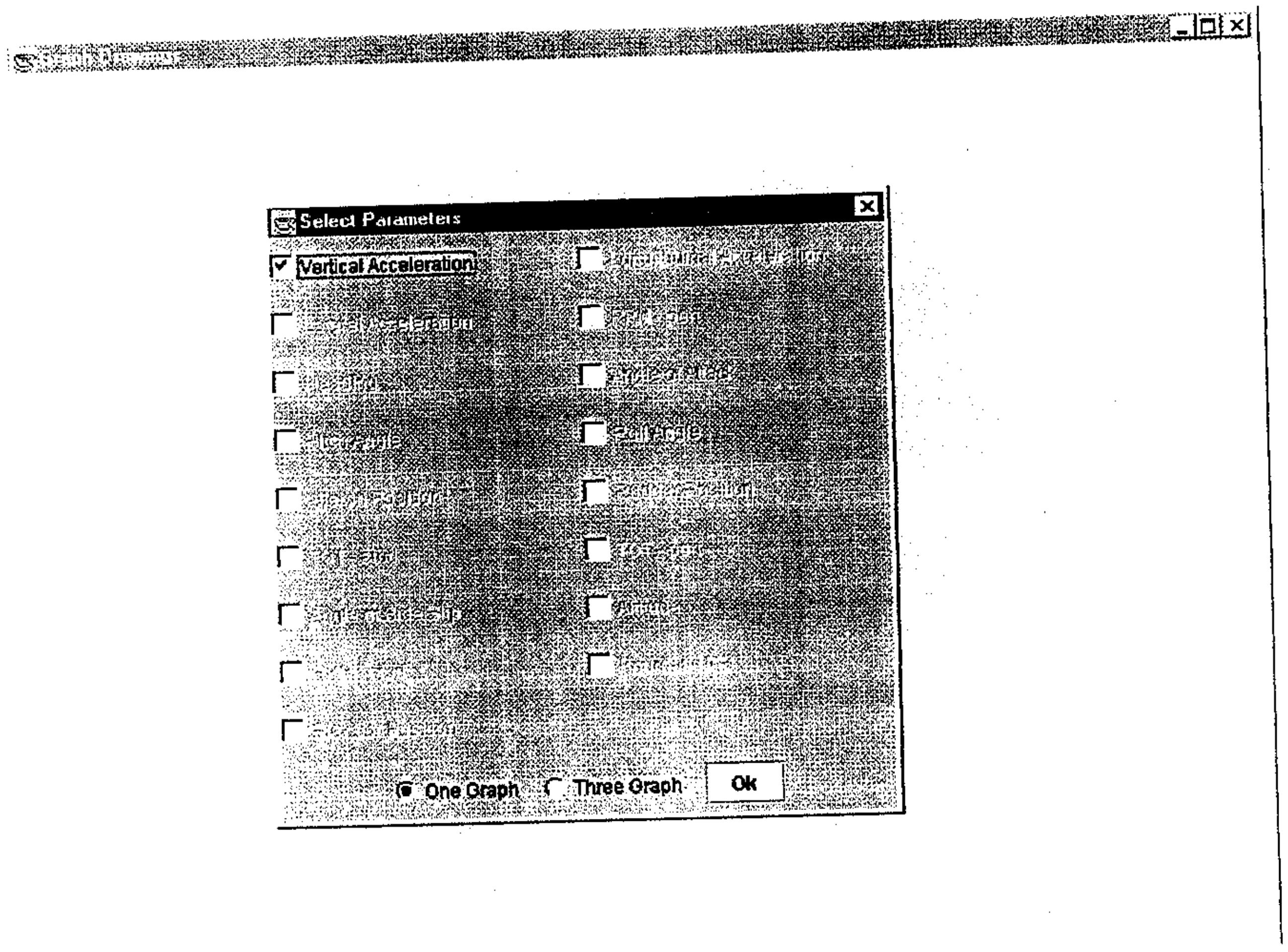
Engg Unit (X)	Raw Decimal (Y)
-1	0153
-0.809	0509
-0.602	0895
-0.407	1262
-0.208	1634
0	2031
0.208	2425
0.407	2799
0.602	3165
0.809	3553
1	3908



Calculated Engineering Unit

File

Secs	Time	Verti	Longi	Later	RPMprt	Head	AoA	Pitch	Roll	Ailer	RudP...	TGTstb	TGTpor	AoSS	Alti	RPM...	PnulA...	Elev
5	00:0...	0.65	0.00	-0.08	0.00	0.00	-3.38	1.30	-1.21	0.00	2.43	0.00	0.00	-22....	0.00	0.00	0.00	-8.14
6	00:0...	0.65	0.00	-0.08	0.00	0.00	-3.38	1.30	-1.21	0.00	2.43	0.00	0.00	-22....	0.00	0.00	0.00	-8.14
7	00:0...	0.65	0.00	-0.08	0.00	0.00	-3.38	1.30	-1.21	0.00	2.43	0.00	0.00	-22....	0.00	0.00	0.00	-8.14
8	00:0...	0.65	0.00	-0.08	0.00	0.00	-3.38	1.30	-1.21	0.00	2.65	0.00	0.00	-22....	0.00	0.00	0.00	-8.14
9	00:0...	0.64	0.00	-0.08	0.00	0.00	-3.38	1.30	-1.21	0.00	2.65	0.00	0.00	-22....	0.00	0.00	0.00	-8.14
10	00:0...	0.65	0.00	-0.08	0.00	0.00	-3.38	1.30	-1.21	0.00	2.65	0.00	0.00	-22....	0.00	0.00	0.00	-8.14
11	00:0...	0.64	0.00	-0.08	0.00	0.00	-3.38	1.30	-1.21	0.00	2.65	0.00	0.00	-22....	0.00	0.00	0.00	-8.14
12	00:0...	0.65	0.00	-0.08	0.00	0.00	-3.38	1.30	-1.21	0.00	2.65	0.00	0.00	-22....	0.00	0.00	0.00	-8.14
13	00:0...	0.64	0.00	-0.08	0.00	0.00	-3.38	1.30	-1.21	0.00	2.65	0.00	0.00	-22....	0.00	0.00	0.00	-8.14
14	00:0...	0.65	0.00	-0.08	0.00	0.00	-3.38	1.30	-1.21	0.00	2.65	0.00	0.00	-22....	0.00	0.00	0.00	-8.14
15	00:0...	0.65	0.00	-0.08	0.00	0.00	-3.38	1.30	-1.21	0.00	2.65	0.00	0.00	-22....	0.00	0.00	0.00	-8.14
16	00:0...	0.65	0.00	-0.08	0.00	0.00	-3.38	1.30	-1.21	0.00	2.65	0.00	0.00	-22....	0.00	0.00	0.00	-8.14
17	00:0...	0.65	0.00	-0.08	0.00	0.00	-3.38	1.30	-1.21	0.00	2.65	0.00	0.00	-22....	0.00	0.00	0.00	-8.14
18	00:0...	0.64	0.00	-0.08	0.00	0.00	-3.38	1.30	-1.21	0.00	2.65	0.00	0.00	-22....	0.00	0.00	0.00	-8.14
19	00:0...	0.65	0.00	-0.08	0.00	0.00	-3.38	1.30	-1.21	0.00	2.65	0.00	0.00	-22....	0.00	0.00	0.00	-8.14
25	00:0...	0.87	0.00	-0.84	0.00	0.00	-45....	-0.17	0.42	0.00	-0.93	0.00	0.00	43....	0.00	0.00	0.00	-7.86
26	00:0...	0.98	0.00	-0.02	0.00	0.00	-45....	-0.25	0.33	0.00	-0.93	0.00	0.00	43....	0.00	0.00	0.00	-7.86
27	00:0...	0.99	0.00	0.02	0.00	0.00	-45....	0.00	0.42	0.00	-0.93	0.00	0.00	43....	0.00	0.00	50....	-7.86
28	00:0...	0.99	0.00	0.02	0.00	0.00	-45....	-0.17	0.42	0.00	-0.93	0.00	0.00	43....	0.00	0.00	0.00	-7.86
29	00:0...	0.99	0.00	0.02	0.00	0.00	-43....	-0.08	0.42	0.00	-0.93	0.00	0.00	43....	0.00	0.00	0.00	-7.86
30	00:0...	0.99	0.00	0.02	0.00	0.00	-43....	0.08	0.50	0.00	-0.93	0.00	0.00	43....	0.00	0.00	0.00	-7.86
31	00:0...	0.99	0.00	0.02	0.00	0.00	-43....	0.00	0.33	0.00	-0.93	0.00	0.00	43....	0.00	0.00	0.00	-7.86
32	00:0...	0.99	0.00	0.02	0.00	0.00	-43....	0.08	0.42	0.00	-0.93	0.00	0.00	43....	0.00	0.00	0.00	-7.86
33	00:0...	0.99	0.00	0.02	0.00	0.00	-43....	0.08	0.42	0.00	-0.93	0.00	0.00	43....	0.00	0.00	0.00	-7.86
34	00:0...	1.00	0.00	0.02	0.00	0.00	-43....	0.25	0.42	0.00	-0.93	0.00	0.00	43....	0.00	0.00	0.00	-7.86
35	00:0...	0.99	0.00	0.02	0.00	0.00	-42....	0.16	0.33	0.00	-0.93	0.00	0.00	43....	0.00	0.00	0.00	-7.86
36	00:0...	0.99	0.00	0.02	0.00	0.00	-42....	0.16	0.25	0.00	-0.93	0.00	0.00	43....	0.00	0.00	0.00	-7.86
37	00:0...	0.99	0.00	0.02	0.00	0.00	-42....	0.33	0.33	0.00	-0.93	0.00	0.00	43....	0.00	0.00	0.00	-7.86
38	00:0...	0.99	0.00	0.02	0.00	0.00	-42....	0.25	0.25	0.00	-0.93	0.00	0.00	43....	0.00	0.00	0.00	-7.86
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]



Select Parameters

[✓ Vertical Acceleration]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

Roll Angle

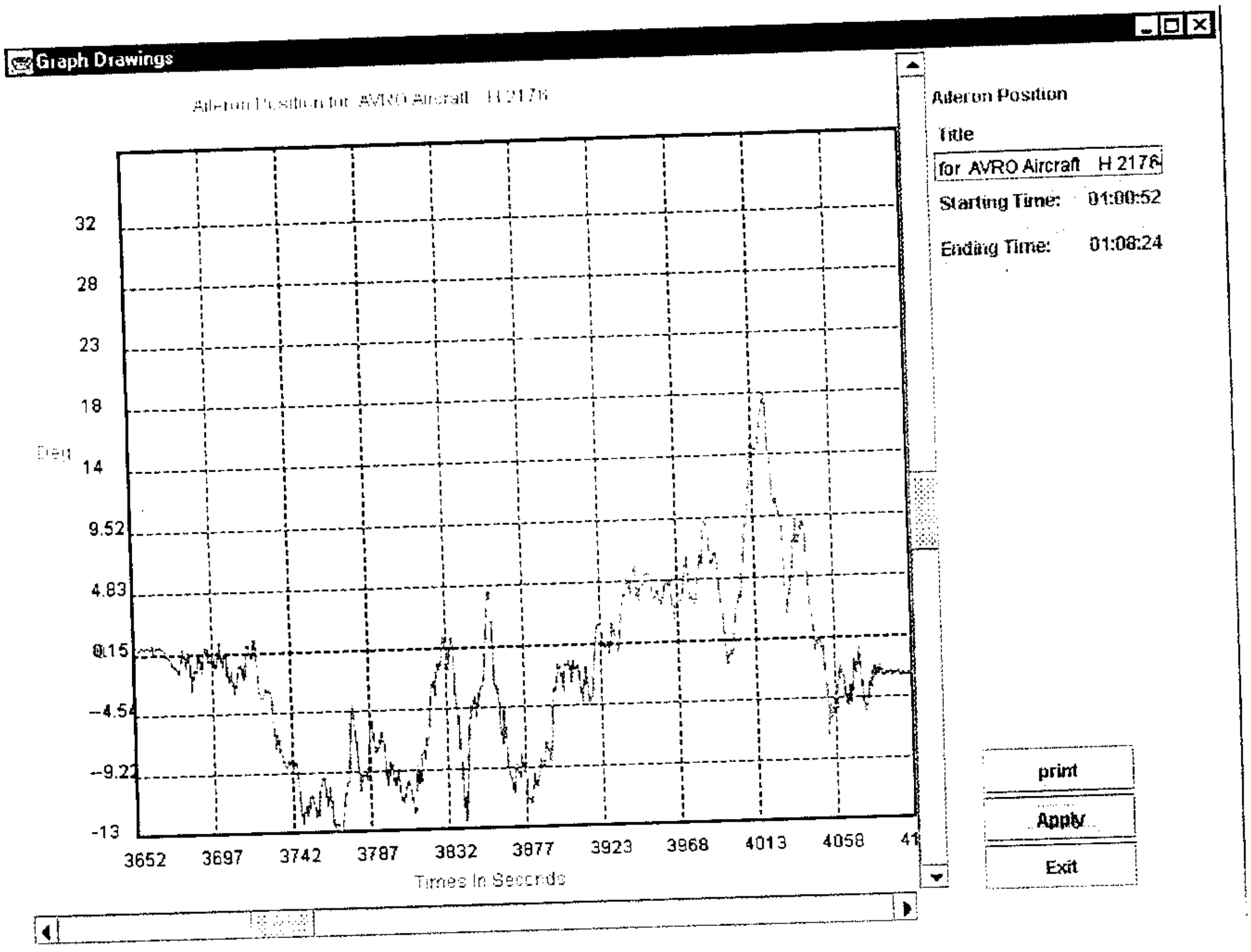
[illegible]

[illegible]

Altitude

[illegible]

[✓ One Graph] Three Graph



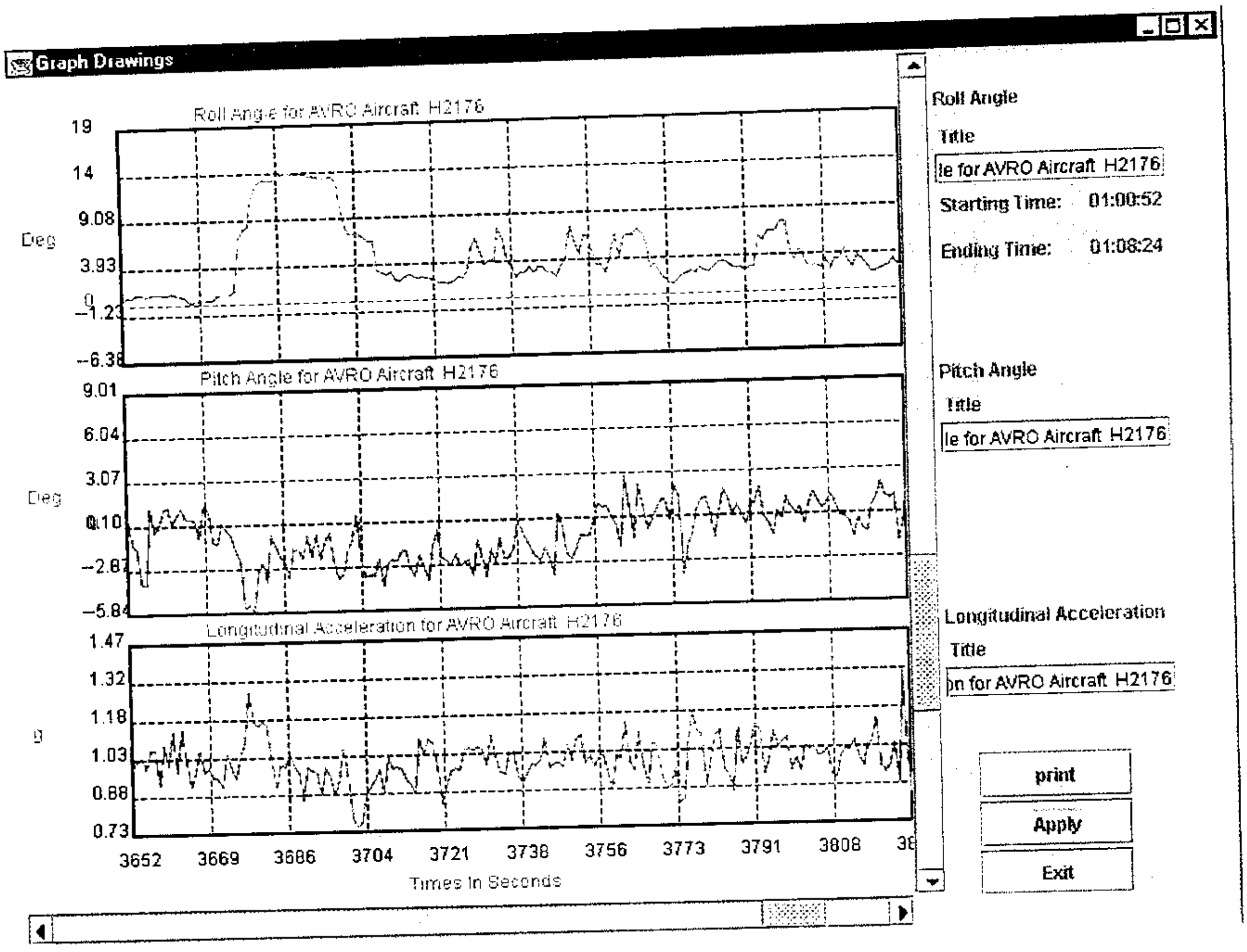
Aileron Position

Title

for AVRO Aircraft H 2176

Starting Time: 01:00:52

Ending Time: 01:08:24



Roll Angle

Title: le for AVRO Aircraft H2176

Starting Time: 01:00:52

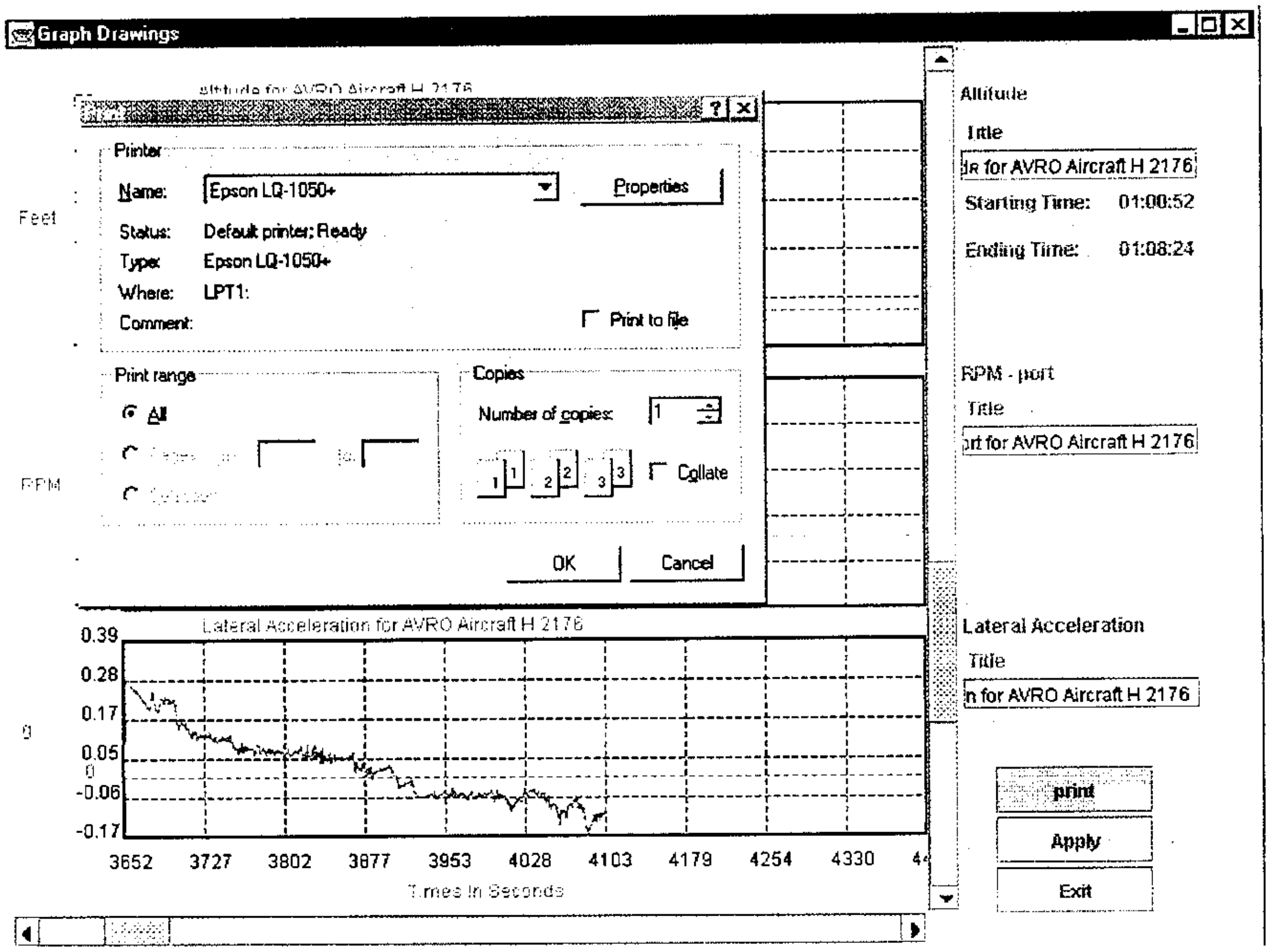
Ending Time: 01:08:24

Pitch Angle

Title: le for AVRO Aircraft H2176

Longitudinal Acceleration

Title: on for AVRO Aircraft H2176



Printer

Name: Epson LQ-1050+

Status: Default printer; Ready

Type: Epson LQ-1050+

Where: LPT1:

Comment:

Print to file

Print range

All

Copies

Number of copies: 1

Collate

Altitude

Title

de for AVRO Aircraft H 2176

Starting Time: 01:00:52

Ending Time: 01:08:24

RPM - port

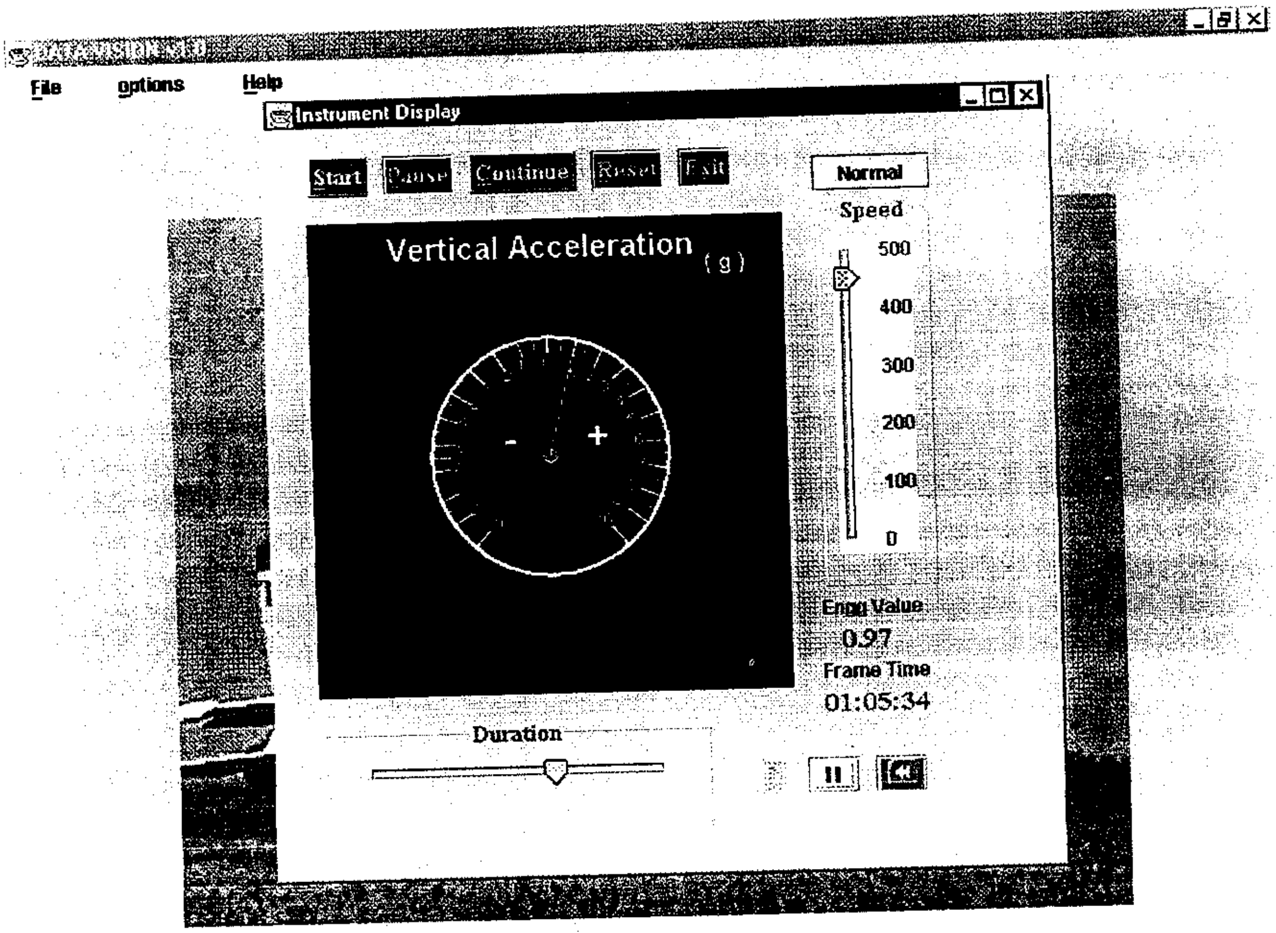
Title

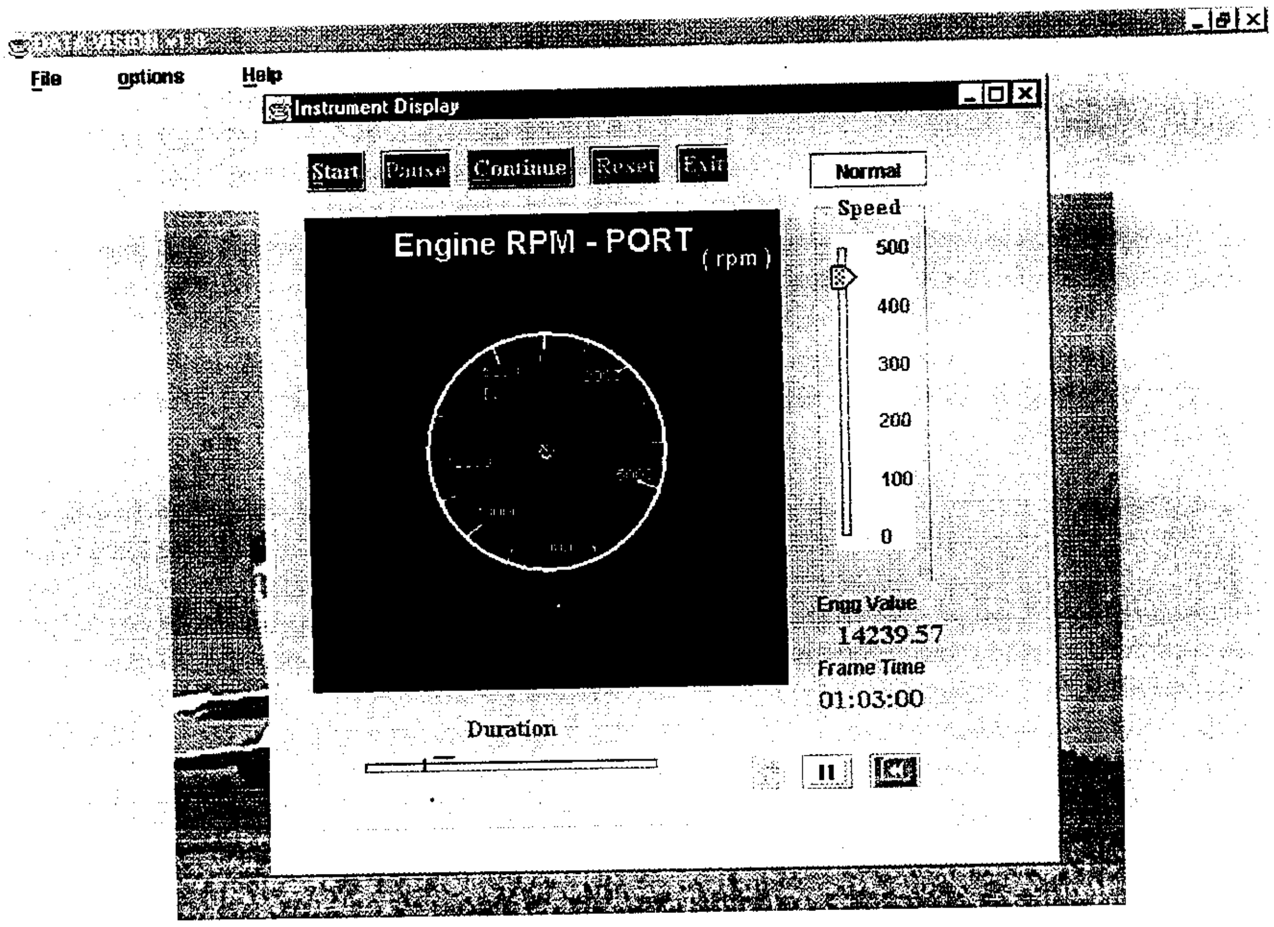
rt for AVRO Aircraft H 2176

Lateral Acceleration

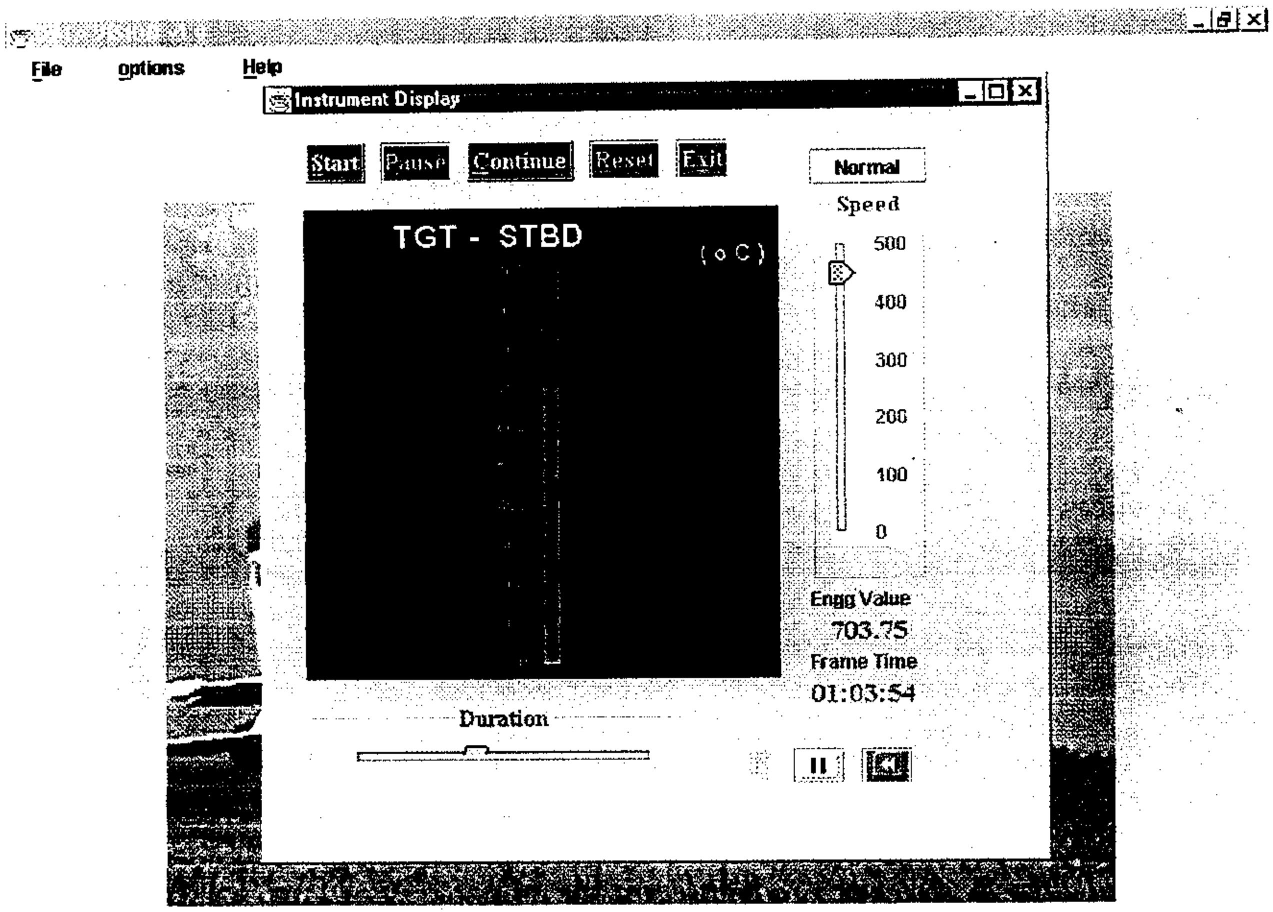
Title

n for AVRO Aircraft H 2176


DATA VISION v1.0
File
options
Help
Instrument Display
Start
Pause
Continue
Reset
Exit
Normal
Speed
500
400
300
200
100
0
Vertical Acceleration (g)
-
+
Engg Value
0.97
Frame Time
01:05:34
Duration



Engg Value
14239.57
Frame Time
01:03:00


File
options
Help
Instrument Display
Start
Pause
Continue
Reset
Exit
Normal
Speed
500
400
300
200
100
0
TGT - STBD
(o C)
Engg Value
703.75
Frame Time
01:03:54
Duration